On urgency in asynchronous timed session types*

Maurizio Murgia
School of Computing
University of Kent.
Canterbury, Uk

M.Murgia@kent.ac.uk

We study an urgent semantics of asynchronous timed session types, where input actions happen as soon as possible. We show that with this semantics we can recover to the timed setting an appealing property of untimed session types: namely, progress is preserved when passing from synchronous to asynchronous communication.

1 Introduction

Session types are abstractions of communication protocols [13], used to statically or dynamically check that distributed programs interact correctly. The original binary synchronous theory has subsequently been extended in several directions: explicit support for multiparty protocols and choreographies [10], asynchronous communication through FIFO buffers [10], time [7, 3], and others [9].

In [5], among other things, it has been proved that session types interacting correctly with synchronous communication still behave correctly when messages are buffered. As reasoning about synchronous systems is easier (synchronous progress is decidibile, asynchronous one is not), concurrent applications can be designed and verified with synchronous communication in mind, and then run on top of real-world asynchronous mediums (e.g., TCP) while preserving correctness. We refer to this practice as *design synchronous/deploy asynchronous* methodology.

In the timed setting, as noted in [3], this property is lost, at least with the asynchronous semantics of [7]. In this work, we propose an alternative semantics of asynchronous session types, that forbids delays when reading actions are possible, similar to an urgent semantics of Communicating Timed Automata [2]. As noted in [2], this semantics, that we call *input urgent* asynchronous semantics, better captures common reading primitives of programming languages/APIs, that returns as soon as a message is available. Our semantics makes therefore session types abstract models of *programs*. The semantics of [7], instead, being more general (allows more behaviour), seems preferable for modelling *protocols* at a higher level of abstraction.

The main contribution of this paper is that the preservation result of [5] can be lifted to the timed setting, when using the input urgent semantics. As timed synchronous progress is decidable [3], and, as discussed above, input urgent semantics models programs, our result paves the way for the application of the design synchronous/deploy asynchronous methodology to time-sensitive distributed software.

*This work has been partially sponsored by EPSRC EP/N035372/1.

2 Timed session types

We now introduce timed session types, both with synchronous and asynchronous communication, their semantics and the associated notions of progress. The synchronous theory is taken from [3], with minor variations.

Preliminaries. 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 not-negative real numbers, and $d, d', \ldots$ to range over the set of natural numbers $\mathbb{N}$. Let $\mathbb{C}$ be a set of *clocks*, 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. The syntax of guards (ranged over by $g, g', \ldots$) is:

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

We give meaning to guards in terms of *clock valuations*, namely functions of type $\mathbb{C} \to \mathbb{R}_{\geq 0}$ which associate each clock with its value. We denote with $\mathbb{V} = \mathbb{C} \to \mathbb{R}_{\geq 0}$ the set of clock valuations (ranged over by $\nu, \eta, \ldots$), and with ν_0 the valuation mapping each clock to zero. We use $\mathscr{K}, \mathscr{K}', \ldots$ to range over sets of clock valuations. We write $\nu + \delta$ for the valuation which increases ν by δ, i.e., $(\nu + \delta)(t) = \nu(t) + \delta$. For a set $R \subseteq \mathbb{C}$, we write $\nu[R]$ for the *reset* of the clocks in R, i.e.,

$$\nu[R](t) = \begin{cases} 0 & \text{if } t \in R \\ \nu(t) & \text{otherwise} \end{cases}$$

Definition 1 (**Semantics of guards**)**.** *For all guards g, we define the set of clock valuations $[\![g]\!]$ inductively as follows, where $\circ \in \{<, \leq, =, \geq, >\}$:*

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

Definition 2 (**Past**)**.** *For all sets of clock valuations $\mathscr{K}$, we define its past $\downarrow\mathscr{K}$ as follows:*

$$\downarrow\mathscr{K} = \{\nu \mid \exists \delta \geq 0 : \nu + \delta \in \mathscr{K}\}$$

To model messages in transit we use queues. Queues are terms of the following grammar:

$$\rho \ ::= \ \emptyset \ \mid \ \mathtt{a};\rho$$

We use ρ, σ to range over queues and we omit trailing occurrences of $\emptyset$. We write $|\rho|$ for the number of messages in the queue ρ (we omit the straigthforward definition).

Syntax. A TST p models the behaviour of a single participant involved in an interaction. Roughly, in an internal choice $\bigoplus_i !\mathtt{a}_i\{g_i, R_i\} \,.\, p_i$ a participant has to perform one of the outputs $!\mathtt{a}_i$ in a time window where g_i is true. Conversely, in an external choice $\sum_i ?\mathtt{a}_i\{g_i, R_i\} \,.\, q_i$ the participant is available to receive each message $\mathtt{a}_i$ in *any instant* within the time window defined by g_i.

Definition 3. Timed session types *$p, q, \ldots$ are terms of the following grammar:*

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

where (i) $I \neq \emptyset$ and finite, (ii) actions in internal/external choices are pairwise distinct, (iii) recursion is guarded. We omit true guards, empty resets, and trailing occurrences of $\mathbf{1}$.

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

$$(p,\ \texttt{a},\ \nu) \xrightarrow{\texttt{!a}}_{\text{s}} (p,\ \emptyset,\ \nu) \qquad [\texttt{!}]$$

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

$$(p,\ \emptyset,\ \nu) \xrightarrow{\delta}_{\text{s}} (p,\ \emptyset,\ \nu+\delta) \qquad \text{if } \nu+\delta \in \texttt{rdy}(p) \qquad [\textsc{Del}]$$

$$\frac{(p\{^{p}/_{X}\},\ \rho,\ \nu) \xrightarrow{\alpha}_{\text{s}} (p',\ \rho',\ \nu')}{(\text{rec}\,X\,.\,p,\ \rho,\ \nu) \xrightarrow{\alpha}_{\text{s}} (p',\ \rho',\ \nu')} \qquad [\textsc{Rec}]$$

$$\frac{(p,\rho,\nu) \xrightarrow{\tau}_{\text{s}} (p',\rho',\nu')}{(p,\rho,\nu) \mid (q,\sigma,\eta) \xrightarrow{\tau}_{\text{s}} (p',\rho',\nu') \mid (q,\sigma,\eta)} \qquad [\text{S-}\oplus]$$

$$\frac{(p,\rho,\nu) \xrightarrow{\delta}_{\text{s}} (p,\rho,\nu') \qquad (q,\sigma,\eta) \xrightarrow{\delta}_{\text{s}} (q,\sigma,\eta')}{(p,\rho,\nu) \mid (q,\sigma,\eta) \xrightarrow{\delta}_{\text{s}} (p,\rho,\nu') \mid (q,\sigma,\eta')} \qquad [\text{S-}\textsc{Del}]$$

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

$$\texttt{rdy}(\textstyle\bigoplus \texttt{!a}_i\{g_i,R_i\}\,.\,p_i) = \downarrow \bigcup \llbracket g_i \rrbracket \qquad \texttt{rdy}(\sum \cdots) = \texttt{rdy}(\mathbf{1}) = \mathbb{V}$$

$$\texttt{rdy}(\text{rec}\,X\,.\,p) = \texttt{rdy}(p\{^{p}/_{X}\})$$

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

Synchronous semantics. Semantics of TSTs is given in terms of a timed labelled transition relation between configurations (defined below). Labels (ranged over by $\alpha, \alpha', \ldots$) are either silent actions τ, delays $\delta > 0$, or branch labels. Labels δ model elapse of time. Branch labels and τ model discrete actions, and take no time.

Definition 4. (Configurations) *A* configuration *is a term of the form* $(p,\rho,\nu) \mid (q,\sigma,\eta)$. *A configuration is synchronous where* $|\rho|, |\sigma| \leq 1$.

We are now ready to define the synchronous semantics of TSTs. Unlike [3], that uses committed choices, we use queues in synchronous configurations to simplify the comparison of synchronous and asynchronous semantics. The two semantics are equivalent.

Definition 5. (Synchronous semantics of TSTs) *The semantics of TSTs is defined as the smallest labelled relation between synchronous configurations closed under the rules in fig. 1. As usual, we denote with* $\rightarrow_{\text{s}}{}^{*}$ *the reflexive and transitive closure of the relation* $\rightarrow_{\text{s}}$.

Rule $[\oplus]$ allows to commit to the branch $\texttt{a}$ of an internal choice, when the corresponding guard is satisfied in the clock valuation ν and the queue is empty. This results in the configuration $(p,\ \texttt{a},\ \nu)$ which can only fire $\texttt{!a}$ ([!]). Rule [?] allows an external choice to fire any of its enabled input actions. Note that the queue must be empty. This is what makes the semantics synchronous. Indeed, without the emptyness requirement, we would have obtained a 1-bounded asynchronous semantics. Rule [Del] allows time to pass; this is always possible for external choices and success term, while for an internal choice we require, through the function `rdy`, that some guard remains satisfiable. Note that also here we require empty queues: this guarantees that message are read at the same time of writing, i.e. communication is synchronous. The other rules are almost standard.

Example 1. *Let* $p = \texttt{!a} \oplus \texttt{!b}\{t > 2\}$, $q = \texttt{?b}\{t > 5\}$, *and consider the following:*

$$\begin{aligned}
(p, \emptyset, \nu_0) \mid (q, \emptyset, \eta_0) &\xrightarrow{7}_{\mathrm{s}} \xrightarrow{\tau}_{\mathrm{s}} (\mathbf{1}, \texttt{b}, \nu_0 + 7) \mid (q, \emptyset, \eta_0 + 7) \\
&\xrightarrow{\tau}_{\mathrm{s}} (\mathbf{1}, \emptyset, \nu_0 + 7) \mid (\mathbf{1}, \emptyset, \eta_0 + 7) \qquad (1) \\
(p, \emptyset, \nu_0) \mid (q, \emptyset, \eta_0) &\xrightarrow{\delta}_{\mathrm{s}} \xrightarrow{\tau}_{\mathrm{s}} (\mathbf{1}, \texttt{a}, \nu_0 + \delta) \mid (q, \emptyset, \eta_0 + \delta) \qquad (2) \\
(p, \emptyset, \nu_0) \mid (q, \emptyset, \eta_0) &\xrightarrow{3}_{\mathrm{s}} \xrightarrow{\tau}_{\mathrm{s}} (\mathbf{1}, \texttt{b}, \nu_0 + 3) \mid (q, \emptyset, \eta_0 + 3) \qquad (3)
\end{aligned}$$

The computation in (1) *reaches success. In* (2), p *commits to the choice* $\texttt{!a}$ *after some delay* δ*; at this point, time cannot pass, and no synchronisation is possible. In* (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.*

Synchronous progress. We recall the progress based notion of compliance of [3], that we refer here as synchronous compliance. TSTs p and q are synchronous compliant when their composition never reaches a deadlock state.

Definition 6 (Synchronous compliance)**.** *We say that* $(p, \rho, \nu) \mid (q, \sigma, \eta)$ *is* success *whenever* $p = \mathbf{1} = q$ *and* $\rho = \emptyset = \sigma$. *We say that* $(p, \rho, \nu) \mid (q, \sigma, \eta)$ *is* s-deadlock *whenever* (*i*) $(p, \rho, \nu) \mid (q, \sigma, \eta)$ *not success, and* (*ii*) *there is no* δ *such that* $(p, \rho, \nu + \delta) \mid (q, \sigma, \eta + \delta) \xrightarrow{\tau}_{\mathrm{s}}$. *We then write* $(p, \nu) \bowtie_s (q, \eta)$ *whenever:*

$$(p, \emptyset, \nu) \mid (q, \emptyset, \eta) \rightarrow_{\mathrm{s}}^{*} (p', \rho, \nu') \mid (q', \sigma, \eta') \quad \textit{implies} \quad (p', \rho, \nu') \mid (q', \sigma, \eta') \textit{ not s-deadlock}$$

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

Example 2. *Let* $p = \texttt{?a}\{t \leq 3\}.\texttt{!b}\{t \leq 3\}$. *We have that* p *is compliant with* $q = \texttt{!a}\{t \leq 2\}.\texttt{?b}\{t \leq 3\}$, *but it is not compliant with* $q' = \texttt{!a}\{t \leq 4\}.\texttt{?b}\{t \leq 4\}$.

Input urgent asynchronous semantics. We now introduce the input urgent semantics of TSTs. Note that here we use configurations (definition 4), i.e. queues can be unbounded.

Definition 7. (Input urgent asynchronous semantics of TSTs) *The input urgent asynchronous semantics of TSTs is defined as the smallest labelled relation between configuration closed under the rules in fig. 1. As usual, we denote with* $\rightarrow_{\mathrm{a}}^{*}$ *the reflexive and transitive closure of the relation* $\rightarrow_{\mathrm{a}}$.

Rule [$\oplus$] allows to append the message $\texttt{a}$ to the queue, when the corresponding guard is satisfied in the clock valuation ν. Rule [!] just says that the message in the head of the queue can be consumed by the communication partner. Rule [?] allows an external choice to fire any of its enabled input actions. Rule [DEL] allows time to pass; this is always possible for external choices and success term, while for an internal choice we require, through the function `rdy`, that some guard remains satisfiable. Rule [S-DEL] allows time to pass for composite systems. While the first two premises are standard, the third one is what makes the semantics urgent: we require, through the predicate δ-sync, that elapsing of time does not prevent nor delay any possible communication. The other rules are almost standard.

Example 3. *Let* $p = \texttt{!a}\{t \leq 2\}.\texttt{!b}\{t \leq 3\}$, $q = \texttt{?a}\{t \geq 4\}.\texttt{?b}\{t \geq 5\}$, *a possible execution of the system is:*

$$(p, \emptyset, \nu_0) \mid (q, \emptyset, \eta_0) \xrightarrow{\texttt{!a}}_{\mathrm{a}} \xrightarrow{\texttt{!b}}_{\mathrm{a}} (\mathbf{1}, \texttt{a};\texttt{b}, \nu_0) \mid (q, \emptyset, \eta_0) \xrightarrow{4}_{\mathrm{a}} \xrightarrow{\texttt{?a}}_{\mathrm{a}} \xrightarrow{1}_{\mathrm{a}} \xrightarrow{\texttt{?b}}_{\mathrm{a}}$$

Note that urgency prevents the transition $(\mathbf{1}, \texttt{a};\texttt{b}, \nu_0) \mid (q, \emptyset, \eta_0) \xrightarrow{\delta}_{\mathrm{a}}$ *if* $\delta > 4$.

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

$$(p,\ \texttt{a};\rho,\ \nu) \xrightarrow{\texttt{!a}}_{\mathrm{a}} (p,\ \rho,\ \nu) \qquad [\texttt{!}]$$

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

$$(p,\ \rho,\ \nu) \xrightarrow{\delta}_{\mathrm{a}} (p,\ \rho,\ \nu+\delta) \qquad \text{if } \nu+\delta \in \mathtt{rdy}(p) \qquad [\textsc{Del}]$$

$$\frac{(p\{p/x\},\ \rho,\ \nu) \xrightarrow{\alpha}_{\mathrm{a}} (p',\ \rho',\ \nu')}{(\mathrm{rec}\,X\,.\,p,\ \rho,\ \nu) \xrightarrow{\alpha}_{\mathrm{a}} (p',\ \rho',\ \nu')} \qquad [\textsc{Rec}]$$

$$\frac{(p,\rho,\nu) \xrightarrow{\tau}_{\mathrm{a}} (p',\rho',\nu')}{(p,\rho,\nu) \mid (q,\sigma,\eta) \xrightarrow{\tau}_{\mathrm{a}} (p',\rho',\nu') \mid (q,\sigma,\eta)} \qquad [\text{S-}\oplus]$$

$$\frac{(p,\rho,\nu) \xrightarrow{\delta}_{\mathrm{a}} (p,\rho,\nu') \quad (q,\sigma,\eta) \xrightarrow{\delta}_{\mathrm{a}} (q,\sigma,\eta') \quad \forall \delta' < \delta : (p,\rho,\nu) \mid (q,\sigma,\eta) \text{ not } \delta'-\text{sync}}{(p,\rho,\nu) \mid (q,\sigma,\eta) \xrightarrow{\delta}_{\mathrm{a}} (p,\rho,\nu') \mid (q,\sigma,\eta')} \qquad [\text{S-}\textsc{Del}]$$

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

$$(p,\rho,\nu) \mid (q,\sigma,\eta)\ \delta-\text{sync} \iff \exists \texttt{a} : \begin{cases} (p,\rho,\nu+\delta) \xrightarrow{\texttt{!a}}_{\mathrm{a}} \wedge (q,\sigma,\eta+\delta) \xrightarrow{\texttt{?a}}_{\mathrm{a}} & \text{or} \\ (p,\rho,\nu+\delta) \xrightarrow{\texttt{?a}}_{\mathrm{a}} \wedge (q,\sigma,\eta+\delta) \xrightarrow{\texttt{!a}}_{\mathrm{a}} \end{cases}$$

Figure 2: Urgent semantics of asynchronous timed session types (symmetric rules omitted).

Below, we sketch a proof of some relations between synchronous and asynchronous semantics. Namely, asynchronous semantics simulates the synchronous one. Furthermore, when queues are empty, asynchronous delays are mimicked by the synchronous semantics.

Lemma 1. *Let $(p,\rho,\nu) \mid (q,\sigma,\eta)$ be a synchronous configuration. Then:*

$$(p,\rho,\nu) \mid (q,\sigma,\eta) \xrightarrow{\alpha}_{\mathrm{s}} (p',\rho',\nu') \mid (q',\sigma',\eta') \implies (p,\rho,\nu) \mid (q,\sigma,\eta) \xrightarrow{\alpha}_{\mathrm{a}} (p',\rho',\nu') \mid (q',\sigma',\eta')$$

Furthermore:

$$(p,\emptyset,\nu) \mid (q,\emptyset,\eta) \xrightarrow{\delta}_{\mathrm{a}} (p',\rho',\nu') \mid (q',\sigma',\eta') \implies (p,\emptyset,\nu) \mid (q,\emptyset,\eta) \xrightarrow{\delta}_{\mathrm{s}} (p',\rho',\nu') \mid (q',\sigma',\eta')$$

Proof. First note the following facts (can be easily proved by rule induction):

$$\forall (p,\rho,\nu),(p',\rho',\nu') : \ (p,\rho,\nu) \xrightarrow{\alpha}_{\mathrm{s}} (p',\rho',\nu') \implies (p,\rho,\nu) \xrightarrow{\alpha}_{\mathrm{s}} (p',\rho',\nu') \tag{4}$$

$$\forall (p,\rho,\nu) : \ (p,\rho,\nu) \xrightarrow{\delta}_{\mathrm{s}} \implies \rho = \emptyset \tag{5}$$

$$\forall p,\nu,p',\nu' : \ (p,\emptyset,\nu) \xrightarrow{\delta}_{\mathrm{a}} (p',\emptyset,\nu') \implies (p,\emptyset,\nu) \xrightarrow{\delta}_{\mathrm{s}} (p',\emptyset,\nu') \tag{6}$$

Back to the main statement, the first part can be proved by cases on the rule used in the derivation of $(p,\rho,\nu) \mid (q,\sigma,\eta) \xrightarrow{\alpha}_{\mathrm{s}} (p',\rho',\nu') \mid (q',\sigma',\eta')$. We only show the more complicated case, namely rule [S-DEL]. Suppose:

$$\frac{(p,\rho,\nu) \xrightarrow{\delta}_{\mathrm{s}} (p,\rho,\nu') \quad (q,\sigma,\eta) \xrightarrow{\delta}_{\mathrm{s}} (q,\sigma,\eta')}{(p,\rho,\nu) \mid (q,\sigma,\eta) \xrightarrow{\delta}_{\mathrm{s}} (p,\rho,\nu') \mid (q,\sigma,\eta')}$$

By eq. (5): $\sigma = \emptyset = \rho$. Therefore, by an inspection of the rules in fig. 2, we can conclude that, for all δ' and for all $\mathtt{a}$, both $(p, \rho, \nu) \not\xrightarrow{!\mathtt{a}}_{\mathrm{a}}$ and $(q, \sigma, \eta) \not\xrightarrow{!\mathtt{a}}_{\mathrm{a}}$. So, $(p, \rho, \nu) \mid (q, \sigma, \eta)$ not δ'–sync for any δ'. Therefore, thanks to eq. (4), we can use rule [S-DEL]:

$$\frac{(p, \rho, \nu) \xrightarrow{\delta}_{\mathrm{a}} (p, \rho, \nu') \quad (q, \sigma, \eta) \xrightarrow{\delta}_{\mathrm{a}} (q, \sigma, \eta') \quad \forall \delta' < \delta : (p, \rho, \nu) \mid (q, \sigma, \eta) \text{ not } \delta' - \text{sync}}{(p, \rho, \nu) \mid (q, \sigma, \eta) \xrightarrow{\delta}_{\mathrm{a}} (p, \rho, \nu') \mid (q, \sigma, \eta')}$$

For the furthermore case, the only possibility is:

$$\frac{(p, \emptyset, \nu) \xrightarrow{\delta}_{\mathrm{a}} (p, \emptyset, \nu') \quad (q, \emptyset, \eta) \xrightarrow{\delta}_{\mathrm{a}} (q, \emptyset, \eta') \quad \forall \delta' < \delta : (p, \emptyset, \nu) \mid (q, \emptyset, \eta) \text{ not } \delta' - \text{sync}}{(p, \emptyset, \nu) \mid (q, \emptyset, \eta) \xrightarrow{\delta}_{\mathrm{a}} (p, \emptyset, \nu') \mid (q, \emptyset, \eta')}\,[\text{S-DEL}]$$

By eq. (6):

$$\frac{(p, \emptyset, \nu) \xrightarrow{\delta}_{\mathrm{s}} (p, \emptyset, \nu') \quad (q, \emptyset, \eta) \xrightarrow{\delta}_{\mathrm{s}} (q, \emptyset, \eta')}{(p, \emptyset, \nu) \mid (q, \emptyset, \eta) \xrightarrow{\delta}_{\mathrm{s}} (p, \emptyset, \nu') \mid (q, \emptyset, \eta')}\,[\text{S-DEL}]$$

□

Asynchronous progress. We extend the notion of progress to the asynchronous setting. It differs from definition 6 only in that it uses the asynchronous semantics.

Definition 8 (Asynchronous compliance)**.** *We say that* $(p, \rho, \nu) \mid (q, \sigma, \eta)$ *is* a-deadlock *whenever* (i) $(p, \rho, \nu) \mid (q, \sigma, \eta)$ *not success, and* (ii) *there is no* δ *such that* $(p, \rho, \nu + \delta) \mid (q, \sigma, \eta + \delta) \xrightarrow{\tau}_{\mathrm{a}}$. *We then write* $(p, \nu) \bowtie_a (q, \eta)$ *whenever:*

$$(p, \emptyset, \nu) \mid (q, \emptyset, \eta) \rightarrow_{\mathrm{a}}^{*} (p', \rho, \nu') \mid (q', \sigma, \eta') \quad \textit{implies} \quad (p', \rho, \nu') \mid (q', \sigma, \eta') \textit{ not a-deadlock}$$

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

3 Results

In this section we sketch a proof of the main result of the paper theorem 1, namely that synchronous progress implies asynchronous progress. The proof is quite standard: we introduce a property (being the composition of asynchronous coinductive compliant TSTs, definition 9) that is enjoyed by all the asynchronous reducts of $(p, \emptyset, \nu_0) \mid (q, \emptyset, \eta_0)$, provided $p \bowtie_s q$ (proposition 1). We then show that all such configurations are not deadlock (proposition 2).

Asynchronous coinductive compliance relations, basically require that in configurations at most one queue is not empty, if all queues are empty then the associated TSTs are synchronous compliant, and that the communication partner is always ready to read the message at the head of the queue.

Definition 9. *We say that* $\mathscr{R}$ *is an* asyncronous coinductive compliance *iff* $(p, \rho, \nu) \mathscr{R} (q, \sigma, \eta)$ *implies:*

1. $\rho = \emptyset \vee \sigma = \emptyset$.
2. $\rho = \emptyset = \sigma \implies (p, \nu) \bowtie_s (q, \eta)$.
3. $(p, \rho, \nu) \xrightarrow{!\mathtt{a}}_{\mathrm{a}} (p', \rho', \nu') \implies \exists (q', \sigma', \eta') : (q, \sigma, \eta) \xrightarrow{?\mathtt{a}}_{\mathrm{a}} (q', \sigma', \eta') \wedge (p', \rho', \nu') \mathscr{R} (q', \sigma', \eta')$
4. $(q, \sigma, \eta) \xrightarrow{!\mathtt{a}}_{\mathrm{a}} (q', \sigma', \eta') \implies \exists (p', \rho', \nu') : (p, \rho, \nu) \xrightarrow{?\mathtt{a}}_{\mathrm{a}} (p', \rho', \nu') \wedge (p', \rho', \nu') \mathscr{R} (q', \sigma', \eta')$

5. $(p,\rho,\nu) \xrightarrow{\tau}_{\mathrm{a}} (p',\rho',\nu') \implies (p',\rho',\nu')\,\mathscr{R}\,(q,\sigma,\eta)$
6. $(q,\sigma,\eta) \xrightarrow{\tau}_{\mathrm{a}} (q',\sigma',\eta') \implies (p,\rho,\nu)\,\mathscr{R}\,(q',\sigma',\eta')$

The following lemma is instrumental to the proof of proposition 1.

Lemma 2. *Let $\mathscr{R}$ be an asynchronous coinductive compliance, and let $(p,\rho,\nu)\,\mathscr{R}\,(q,\sigma,\eta)$. Then:*

$$(p,\rho,\nu) \mid (q,\sigma,\eta) \xrightarrow{\delta}_{\mathrm{a}} \implies \rho = \emptyset = \sigma$$

Proof. Suppose $(p,\rho,\nu) \mid (q,\sigma,\eta) \xrightarrow{\delta}_{\mathrm{a}}$. We have to show $\rho = \emptyset = \sigma$. Suppose, by contradiction, this is not the case, and assume that, say, $\rho = \mathtt{a};\rho''$ for some $\mathtt{a},\rho''$. Then, by rule [!], $(p,\rho,\nu) \xrightarrow{!\mathtt{a}}_{\mathrm{a}}$. By item 3 of definition 9, it must be $(q,\sigma,\eta) \xrightarrow{?\mathtt{a}}_{\mathrm{a}}$. Then, $(p,\rho,\nu) \mid (q,\sigma,\eta)$ not 0−synch. Since the only rule for deriving delays is [S-Del], there is ν',η' such that:

$$\frac{(p,\rho,\nu) \xrightarrow{\delta}_{\mathrm{a}} (p,\rho,\nu') \quad (q,\sigma,\eta) \xrightarrow{\delta}_{\mathrm{a}} (q,\sigma,\eta') \quad \forall\delta' < \delta : (p,\rho,\nu) \mid (q,\sigma,\eta) \text{ not } \delta'-\text{sync}}{(p,\rho,\nu) \mid (q,\sigma,\eta) \xrightarrow{\delta}_{\mathrm{a}} (p,\rho,\nu') \mid (q,\sigma,\eta')}$$

Since $\delta > 0$, we get a contradiction. The case $\sigma \neq \emptyset$ is similar. □

Proposition 1. *Let $(p,\nu) \bowtie_s (q,\eta)$. Then, for some asynchronous coinductive compliance $\mathscr{R}$:*

$$(p,\emptyset,\nu) \mid (q,\emptyset,\eta) \rightarrow_{\mathrm{a}}^{*} (p',\rho,\nu') \mid (q',\sigma,\eta') \implies (p',\rho,\nu')\,\mathscr{R}\,(q',\sigma,\eta')$$

Proof. Suppose $(p,\nu) \bowtie_s (q,\eta)$. Let:

$$\mathscr{R} = \{((p',\rho,\nu'),(q',\sigma,\eta')) \mid (p,\emptyset,\nu) \mid (q,\emptyset,\eta) \rightarrow_{\mathrm{a}}^{*} (p',\rho,\nu') \mid (q',\sigma,\eta')\}$$

It suffice to prove that $\mathscr{R}$ is an asynchronous coinductive compliance. So, let $(p',\rho,\nu')\,\mathscr{R}\,(q',\sigma,\eta')$. We proceed by induction on the lenght n of the reduction $(p,\emptyset,\nu) \mid (q,\emptyset,\eta) \rightarrow_{\mathrm{a}}^{*} (p',\rho,\nu') \mid (q',\sigma,\eta')$. For $n = 0$, it must be $(p',\rho,\nu') \mid (q',\sigma,\eta') = (p,\emptyset,\nu) \mid (q,\emptyset,\eta)$. Items 1 and 2 of definition 9 follows immediately. Items 3 and 4 hold trivially: for all $\mathtt{a}$, we have that $(p,\emptyset,\nu) \not\xrightarrow{!\mathtt{a}}_{\mathrm{a}}$ and $(q,\emptyset,\eta) \not\xrightarrow{!\mathtt{a}}_{\mathrm{a}}$ (by an inspection of the rules in fig. 2, exploiting queue emptiness). For item 5, suppose $(p,\emptyset,\nu) \xrightarrow{\tau}_{\mathrm{a}} (p'',\rho',\nu'')$. By rule [S-$\oplus$], $(p,\emptyset,\nu) \mid (q,\emptyset,\eta) \rightarrow_{\mathrm{a}} (p'',\rho',\nu'') \mid (q,\emptyset,\eta)$. Therefore $(p'',\rho',\nu'')\,\mathscr{R}\,(q,\emptyset,\eta)$. Similarly for item 6. For the induction step, let

$$(p,\emptyset,\nu) \mid (q,\emptyset,\eta) \rightarrow_{\mathrm{a}}^{n} (p'',\rho',\nu'') \mid (q'',\sigma',\eta'') \rightarrow_{\mathrm{a}} (p',\rho,\nu') \mid (q',\sigma,\eta')$$

We proceed by cases on the rule used for deriving $(p'',\rho',\nu'') \mid (q'',\sigma',\eta'') \rightarrow_{\mathrm{a}} (p',\rho,\nu') \mid (q',\sigma,\eta')$.

- [S-$\oplus$].

$$\frac{(p'',\rho',\nu'') \xrightarrow{\tau}_{\mathrm{a}} (p',\rho,\nu')}{(p'',\rho',\nu'') \mid (q'',\sigma',\eta'') \xrightarrow{\tau}_{\mathrm{a}} (p',\rho',\nu') \mid (q'',\sigma',\eta'')}$$

  Note that it must be $(q'',\sigma',\eta'') = (q',\sigma,\eta')$. By the induction hypothesis:

$$(p'',\rho',\nu'')\,\mathscr{R}\,(q',\sigma,\eta')$$

  Therefore, by item 5:

$$(p',\rho,\nu') \mid (q',\sigma,\eta')$$

- [S-DEL].

$$\frac{(p'',\rho',\nu'') \xrightarrow{\delta}_{\mathrm{a}} (p'',\rho',\nu') \quad (q'',\sigma',\eta'') \xrightarrow{\delta}_{\mathrm{a}} (q'',\sigma',\eta') \quad \forall \delta' < \delta : (p'',\rho',\nu') \mid (q'',\sigma',\eta'') \text{ not } \delta' - \text{sync}}{(p'',\rho',\nu'') \mid (q'',\sigma',\eta'') \xrightarrow{\delta}_{\mathrm{a}} (p'',\rho',\nu') \mid (q'',\sigma',\eta') = (p',\rho,\nu') \mid (q',\sigma,\eta')}$$

First note that $p'' = p'$, $q'' = q'$, $\rho' = \rho$ and $\sigma' = \sigma$. By lemma 2 it follows $\rho = \emptyset = \sigma$. By the induction hypothesis and item 2 it follows:

$$(p'',\nu'') \bowtie_s (q'',\eta'')$$

By lemma 1:

$$(p'',\rho',\nu'') \mid (q'',\sigma',\eta'') \xrightarrow{\delta}_{\mathrm{s}} (p',\rho,\nu') \mid (q',\sigma,\eta')$$

Since all queues mentioned above are empty, by lemmas A.2 and 3.6 of [3](modulo minor notational differences):

$$(p',\nu') \bowtie_s (q',\eta')$$

The thesis then follows by the same argument used in the base case of this proof.

- [S-τ]

$$\frac{(p'',\rho',\nu'') \xrightarrow{!\mathtt{a}}_{\mathrm{a}} (p',\rho,\nu') \quad (q'',\sigma',\eta'') \xrightarrow{?\mathtt{a}}_{\mathrm{a}} (q',\sigma,\eta')}{(p'',\rho',\nu'') \mid (q'',\sigma',\eta'') \xrightarrow{\tau}_{\mathrm{s}} (p',\rho,\nu') \mid (q',\sigma,\eta')}$$

The thesis follows by the induction hypothesis, exploiting item 3 and the trivial fact that semantics is deterministic.

□

Proposition 2. *Let $\mathscr{R}$ be an asynchronous coinductive compliance, and let $(p,\rho,\nu)\,\mathscr{R}\,(q,\sigma,\eta)$. Then $(p,\rho,\nu) \mid (q,\sigma,\eta)$ is not a-deadlock.*

Proof. Suppose $(p,\rho,\nu)\,\mathscr{R}\,(q,\sigma,\eta)$ for some asynchronous coinductive compliance $\mathscr{R}$. By item 1 of definition 9, at most one queue among ρ,σ is not empty. If $\rho = \mathtt{a};\rho'$ for some $\mathtt{a},\rho'$, by rule [!]:

$$(p,\ \mathtt{a};\rho',\ \nu) \xrightarrow{!\mathtt{a}}_{\mathrm{a}} (p,\ \rho',\ \nu)$$

Then, by item 3, $(q,\sigma,\eta) \xrightarrow{?\mathtt{a}}_{\mathrm{a}} (q',\sigma',\eta')$, for some (q',σ',η'). Therefore, by rule [S-τ]:

$$\frac{(p,\rho,\nu) \xrightarrow{!\mathtt{a}}_{\mathrm{a}} (p,\rho',\nu) \quad (q,\sigma,\eta) \xrightarrow{?\mathtt{a}}_{\mathrm{a}} (q',\sigma',\eta')}{(p,\rho,\nu) \mid (q,\sigma,\eta) \xrightarrow{\tau}_{\mathrm{s}} (p,\rho',\nu) \mid (q',\sigma',\eta')}$$

The case where σ is not empty is similar. It remains the case $\rho = \emptyset = \sigma$. By item 2, it must be $(p,\nu) \bowtie_s (q,\eta)$. If $(p,\emptyset,\nu)\,\mathscr{R}\,(q,\emptyset,\eta)$ is success, we are done. If not, we know that there is δ such that:

$$(p,\emptyset,\nu+\delta) \mid (q,\emptyset,\eta+\delta) \xrightarrow{\tau}_{\mathrm{s}}$$

Note that it is a reduction of the synchronous semantics. But then, by lemma 1:

$$(p,\emptyset,\nu+\delta) \mid (q,\emptyset,\eta+\delta) \xrightarrow{\tau}_{\mathrm{a}}$$

□

The main result follows.

Theorem 1. *For all p,q:*

$$p \bowtie_s q \implies p \bowtie_a q$$

Proof. Composition of propositions 1 and 2. □

4 Conclusions and related work

Following [3], we pursued a line of research aimed at lifting key properties of session types to the timed setting. We have shown that the interesting property of preservation of untimed synchronous progress when passing to asynchronous semantics (discovered in [5]), can be recovered using a certain urgent asynchronous semantics, that closely models realistic programming primitives.

Timed session types have been introduced in [7], in the multiparty asynchronous version, where they have been used to statically type check a timed π-calculus. Their theory has subsequently been extended to dynamic verification [12]. [3] introduced the binary and synchronous theory, subsequently applied in a contract-oriented middleware [4] and the companion verification tool-chain [1]. [6] studies progress in the context of Communicating Timed Automata [11]. Several works study urgency in timed systems, here we mention [8]. As far as we know, urgency in the context of asynchronous communication has only been studied in [2], where the idea of modelling input primitives with input urgency originates.

References

[1] Nicola Atzei & Massimo Bartoletti (2016): *Developing Honest Java Programs with Diogenes*. In: *FORTE 2016*, *LNCS* 9688, Springer, pp. 52–61.

[2] Massimo Bartoletti, Laura Bocchi & Maurizio Murgia (2017): *Compositional Asynchronous Timed Refinement*. Technical Report. Available at `https://www.cs.kent.ac.uk/people/staff/lb514/tr2.pdf`.

[3] Massimo Bartoletti, Tiziana Cimoli & Maurizio Murgia (2017): *Timed Session Types*. *Logical Methods in Computer Science* 13(4).

[4] Massimo Bartoletti, Tiziana Cimoli, Maurizio Murgia, Alessandro Sebastian Podda & Livio Pompianu (2015): *A contract-oriented middleware*. In: *FACS*, *LNCS* 9539, Springer, pp. 86–104.

[5] Massimo Bartoletti, Alceste Scalas & Roberto Zunino (2014): *A Semantic Deconstruction of Session Types*. In: *Proc. CONCUR*, *LNCS* 8704, Springer, pp. 402–418.

[6] Laura Bocchi, Julien Lange & Nobuko Yoshida (2015): *Meeting Deadlines Together*. In: *CONCUR*, *LIPIcs* 42, Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, pp. 283–296.

[7] Laura Bocchi, Weizhen Yang & Nobuko Yoshida (2014): *Timed Multiparty Session Types*. In: *CONCUR*, *LNCS* 8704, Springer, pp. 419–434.

[8] Sébastien Bornot, Joseph Sifakis & Stavros Tripakis (1997): *Modeling Urgency in Timed Systems*. In: *COMPOS*, *LNCS* 1536, Springer, pp. 103–129.

[9] Mariangiola Dezani-Ciancaglini & Ugo de'Liguoro (2009): *Sessions and Session Types: An Overview*. In: *WS-FM*, *LNCS* 6194, Springer, pp. 1–28.

[10] Kohei Honda, Nobuko Yoshida & Marco Carbone (2016): *Multiparty Asynchronous Session Types*. *J. ACM* 63(1), pp. 9:1–9:67.

[11] Pavel Krcál & Wang Yi (2006): *Communicating Timed Automata: The More Synchronous, the More Difficult to Verify*. In: *CAV*, *LNCS* 4144, Springer, pp. 249–262.

[12] Rumyana Neykova, Laura Bocchi & Nobuko Yoshida (2017): *Timed runtime monitoring for multiparty conversations*. *Formal Asp. Comput.* 29(5), pp. 877–910.

[13] Kaku Takeuchi, Kohei Honda & Makoto Kubo (1994): *An Interaction-based Language and its Typing System*. In: *PARLE*, *LNCS* 817, Springer, pp. 398–413.