Timed Interfaces*

Luca de Alfaro[1], Thomas A. Henzinger[2], and Mariëlle Stoelinga[1]

[1] Computer Engineering, University of California, Santa Cruz
{luca,marielle}@soe.ucsc.edu
[2] EECS, University of California, Berkeley
tah@eecs.berkeley.edu

Abstract. We present a theory of *timed interfaces*, which is capable of specifying both the timing of the inputs a component expects from the environment, and the timing of the outputs it can produce. Two timed interfaces are *compatible* if there is a way to use them together such that their timing expectations are met. Our theory provides algorithms for checking the compatibility between two interfaces and for deriving the composite interface; the theory can thus be viewed as a type system for real-time interaction. Technically, a timed interface is encoded as a timed game between two players, representing the inputs and outputs of the component. The algorithms for compatibility checking and interface composition are thus derived from algorithms for solving timed games.

1 Introduction

A formal notion of component interfaces provides a way to describe the interaction between components, and to verify the compatibility between components automatically. Traditional type systems capture only the *data* dimension of interfaces ("what are the value constraints on data communicated between components?"). We have developed an approach, called *interface theories* [dAH01a,dAH01b], which can be viewed as a behavioral type system that also captures the *protocol* dimension of interfaces ("what are the temporal ordering constraints on communication events between components?"). This paper extends this formalism to capture, in addition, the *timing* dimension of interfaces ("what are the real-time constraints on communication events between components?"). This permits, for example, the specification and compatibility check of component interactions based on time-outs. *Timed interfaces* support the component-based design of real-time systems in the following ways:

Component interface specification. A component is an open system that expects inputs and provides outputs to the environment. An interface specifies how a component interacts with its environment, by describing both the assumptions made by the component on the inputs and the guarantees provided by the

* This research was supported in part by the NSF CAREER award CCR-0132780, the NSF grant CCR-9988172 the AFOSR MURI grant F49620-00-1-0327, the DARPA PCES grant F33615-00-C-1693, the MARCO GSRC grant 98-DT-660, and the ONR grant N00014-02-1-0671.

component on the outputs. Timed interfaces can refer to the timing of the input and output events. An interface is *well-formed* as long as there is *some* environment that satisfies the input assumptions made by the component; otherwise, the component would not be usable in any design.

Interface compatibility checking. When two components are composed, we can check that the components satisfy each other's assumptions. Since the result of composition is generally still an open system, it may depend on the environment whether or not this is the case. Two interfaces are *compatible* if their composition is well-formed, i.e., if there exists an environment that makes them work together. The composition yields a new interface for the composite system that specifies the derived input assumptions required to make the original components working together, as well as the resulting output guarantees.

An (untimed or timed) interface is naturally modeled as a *game* between two players, Output and Input. Player Output represents the component: the moves of Output represent the possible outputs that the component may generate (the output guarantees); player Input represents the environment: the moves of Input represent the inputs that the system accepts from the environment (the input assumptions). For instance, a functional type $D \mapsto D'$ in a programming language is an interface that corresponds to a one-shot game. The Input player provides inputs that can be accepted (values in D) and the Output player provides outputs that can be generated (values in D'). If the sets of legal inputs and possible outputs can change dynamically over time, then the interface is naturally modeled as an iterative game played on a set of states [Abr96], where the moves —i.e., the acceptable inputs and possible outputs— may depend on the state of the system. Such an interface is *well-formed* if the Input player has a winning strategy in the game, i.e., the environment can meet all input assumptions. For *timed* interfaces, we need the additional well-formedness condition that a player must not achieve its goal by blocking time forever.

The game-theoretic view of interfaces becomes most apparent in their composition. When two interfaces are composed, the combined interface may contain *error states*. These occur when one component interface can generate an output that is not a legal input for the other component interface, showing that the first component violates an input assumption of the second. In addition, our timed games give rise to *time errors*, where one of the players cannot let time pass. Two interfaces are *compatible* if there is a way for the Input player, who chooses the inputs of the composite interface, to avoid all errors. If so, then there exists an environment of the combined system which makes both components satisfy each other's input assumptions.

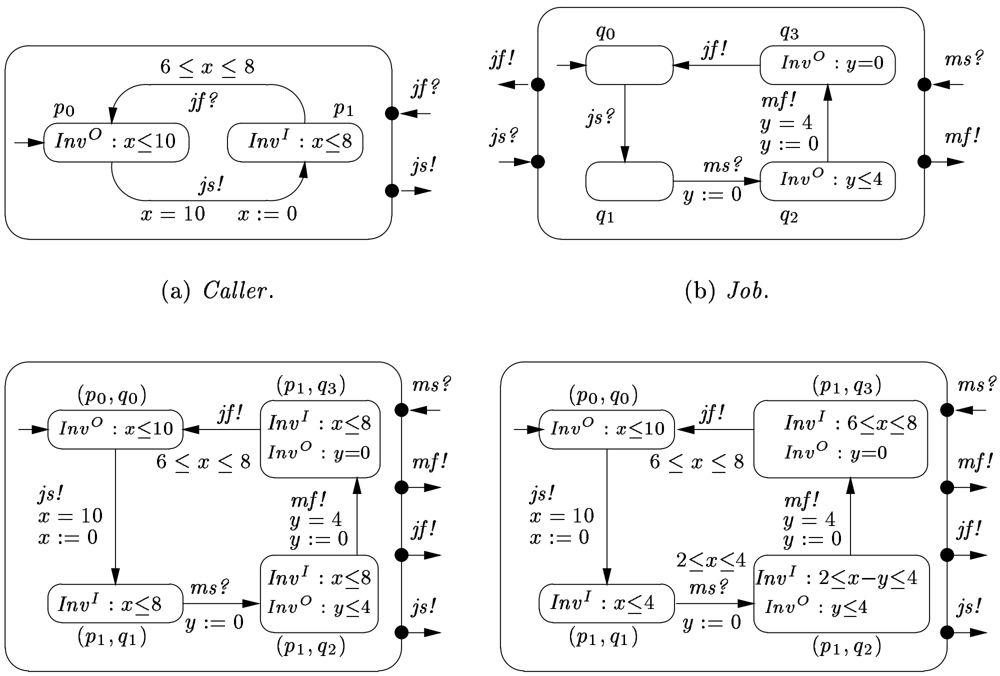
Different theories of timed interfaces arise depending on the details of how timed games are defined. For example, communication can be through actions or shared variables, and composition can be synchronous or asynchronous. The main contribution of this paper does not lie in such details of the formalism (which may be changed), but in the notion of interface of a real-time component as a timed game. This notion sets our interfaces apart from the type systems for protocols of [RR01,CRR02], from message-sequence charts [RGG96], and from traditional models for timed systems such as timed automata [AD94,MMT91]. Many models are unable to express input assumptions and postulate that a component must work in all environments; this is the *input-enabled* approach of, e.g., [AH97,SGSAL98]. Models that can encode input assumptions, such as process algebras, often phrase the compatibility question as a *graph* (rather than *game*) question, in which input and output play the same role: two components are considered compatible if they cannot reach a deadlock [RR01]. In our game-based approach, input and output play dual roles: two components are compatible if there is *some* input behavior such that, for *all* output behaviors, no incompatibility arises. This notion captures the idea that an interface can be useful as long as it can be used in some design. In this, interfaces are close to types in programming languages, to trace theory [Dil88], and to the semantics of interaction [Abr96].

2 Timed Interfaces as Timed Games: Preview

In a timed interface, the Input and Output players have two kinds of moves: *immediate moves,* which represent events sent or received by the interface, and *timed moves,* that consist in an amount of time that the players propose to spend idle. We assume a time domain $\mathbb{T}$; suitable choices for $\mathbb{T}$ are the nonnegative reals $\mathbb{R}_{\geq 0}$, or the nonnegative integers $\mathbb{N}$. The successor state is determined as follows. When Input chooses $t_I \in \mathbb{T}$ and Output chooses $t_O \in \mathbb{T}$, the global time will advance by $\min\{t_I, t_O\}$; if a player chooses an immediate move and the other a timed move, the immediate move prevails; if both players choose immediate moves, one of them occurs nondeterministically [MPS95,AMPS98].

Only game outcomes along which global time diverges are physically meaningful. Obviously, each player is capable of blocking the progress of time by playing a sequence of timed moves whose summation converges (so-called Zeno behavior). To rule out such behavior, we require a *well-formedness* criterion, which states that at all reachable states, each player can ensure that time progresses unless the other player blocks its progress. The composition of timed interfaces may give rise to two kinds of error states: *immediate error states,* where one interface emits an output that cannot be accepted by the other, and *time error states,* where the well-formedness criterion is violated. Two interfaces are compatible if there is a strategy for Input in the combined interface that avoids all errors. The composite interface is obtained by restricting the input moves (i.e., the accepted environments) such that the error states are not entered.

We illustrate these concepts through a simple example from scheduling. The interfaces are modeled as *timed interface automata.* This is a syntax derived from timed automata [AD94]. However, the syntax is interpreted as a (timed) game, rather than as a (timed) transition system. In particular, timed automata use invariants for specifying an upper bound for the advancement of time at a location. Timed interface automata have two kinds of invariants: *input invariants*, which specify upper bounds for the timed moves played by Input, and *output invariants*, which specify upper bounds for the timed moves played by Output.



(a) *Caller*. (b) *Job*.

(c) $Caller \otimes Job$. (d) $Caller \| Job$.

Fig. 1: Timed interface automata for periodic job scheduling.

Example: scheduling a periodic job. The timed interface automaton *Caller* shown in Figure 1(a) represents an application that activates a periodic job every 10 seconds. Each time the job is activated, it must terminate between 6 and 8 seconds. The clock x measures the time elapsed since the last activation of the job. At the initial location p_0, the Output player can play any timed move Δ such that $\Delta + x \le 10$, that is, it can advance time only up to the point where $x = 10$. Once $x = 10$, in order not to block progress of time, Output must eventually play the move *js* (job start); i.e., the output invariant $x \le 10$ expresses a deadline for the occurrence of the *js* action. Since the input invariant associated with p_0 is always true, the Input player can play any $\Delta \in \mathbb{T}$ (to reduce clutter, invariants that are always true are omitted from the figures). The move *js* resets x to 0, ensuring that x counts the time from the start of a periodic job activation.

At location p_1, Output can play arbitrary timed moves, since the output invariant is true. The input invariant $x \le 8$ prevents the Input player from advancing time beyond $x = 8$. So, to let time progress, Input must play the move *jf* (job finished) somewhere between $6 \le x \le 8$. In particular, $x \le 8$ is a deadline for the reception of action *jf*.

Composition. The timed interface automaton *Caller* can be composed with the timed interface automaton *Job*, which models the interface of the job to be executed. The job starts when the input move *js* (job start) is received; the job then waits for the input move *ms* (machine start) from a scheduler. Once *ms* is received, the job executes for 4 seconds (clock y keeps track of the execution time), after which it first indicates to the scheduler that the machine is no longer used (output move *mf*, machine finished), and then finishes (output move *jf*, job finished). Note how, at state q_2, the input invariant is true, indicating that Input can play any timed move, while the output invariant is $y \leq 4$, indicating that Output can let time progress only until $y = 4$. Again, to avoid blocking the progress of time, Output must eventually play the move *mf*. Similarly, Output is forced to play *jf* at state q_3.

To compute the composition $\mathit{Caller} \parallel \mathit{Job}$, we first compute their product $\mathit{Caller} \otimes \mathit{Job}$, shown in Figure 1(c), which represents the joint behavior arising from *Caller* and *Job*. Moves synchronize in multi-way, broadcast fashion: the synchronization between corresponding input and output moves gives rise to an output move. To compute the composition $\mathit{Caller} \| \mathit{Job}$ from $\mathit{Caller} \otimes \mathit{Job}$, note that the product contains error states: at location (p_1, q_3), if $err_3 : \neg(6 \leq x \leq 8)$ holds, then interface *Job* can generate the output *jf*, which cannot be accepted by *Caller*. However, the Input player can avoid the error states by choosing the timing of the input *ms*. To see this, note that at location (p_1, q_2) the set of *uncontrollable states* from which Input cannot avoid reaching the error states is described by the predicate $uncontr_2 : \exists \Delta \in \mathbb{T}.(y + \Delta = 4 \wedge \neg(6 \leq x + \Delta \leq 8))$, or after simplification, $uncontr_2 : \neg(2 \leq x - y \leq 4)$. Hence, we conjoin $2 \leq x - y \leq 4$ to the input invariant of (p_1, q_1), which yields $2 \leq x - y \leq 4 \wedge x \leq 8$ as the input invariant in $\mathit{Caller} \| \mathit{Job}$ (see Figure 1(d)). We simplify this to $2 \leq x - y \leq 4$, because the states where $x > 8$ cannot be reached: from $y \leq 4$ (the output invariant) and $x - y \leq 4$ (a portion of the input invariant) follows $x \leq 8$.

Now, to avoid entering uncontrollable states at location (p_1, q_2) we have to restrict the time during which the input move *ms* can be received: the most liberal restriction consists in requiring $2 \leq x \leq 4$, which is added to enabling condition of *ms*. Finally, consider the location (p_1, q_1), whose input invariant is $x \leq 8$. To ensure progress of time beyond $x = 8$, Input must eventually play the move *ms*. However, this move is available only when $2 \leq x \leq 4$. Thus, if $x > 4$, the Input player does not have a strategy that ensures that time diverges unless blocked by the Output player. Therefore, $x > 4$ indicates time error states, which are ruled out by strengthening the input invariant to $x \leq 4$. Notice how the composition of timed interfaces effects the composition of timing requirements: the requirement of *Caller* that the job must be completed between 6 and 8 seconds gives rise to the requirement for the scheduler that the machine is started between 2 and 4 seconds after the job start.

3 Timed Interfaces as Timed Games: Definitions

We model interfaces as timed games between two players, Input and Output, abbreviated I and O.

Definition 1 (timed interfaces) A *timed interface* is a tuple $\mathcal{P} = (S_\mathcal{P}, s_\mathcal{P}^{init}, Acts_\mathcal{P}^I, Acts_\mathcal{P}^O, \rho_\mathcal{P}^I, \rho_\mathcal{P}^O)$ consisting of the following components.

- $S_\mathcal{P}$ is a set of *states*.
- $s_\mathcal{P}^{init} \in S_\mathcal{P}$ is the *initial state*.
- $Acts_\mathcal{P}^I$ and $Acts_\mathcal{P}^O$ are sets of *immediate input* and *output* actions, respectively. These sets must be disjoint from each other, and disjoint from the time domain $\mathbb{T}$. We denote by $Acts_\mathcal{P} = Acts_\mathcal{P}^I \cup Acts_\mathcal{P}^O$ the set of all immediate actions, by $\Gamma_\mathcal{P}^I = Acts_\mathcal{P}^I \cup \mathbb{T}$ the set of all input actions, and by $\Gamma_\mathcal{P}^O = Acts_\mathcal{P}^O \cup \mathbb{T}$ the set of all output actions. The elements in $\mathbb{T}$ are *timed actions*.
- $\rho_\mathcal{P}^I \subseteq S_\mathcal{P} \times \Gamma_\mathcal{P}^I \times S_\mathcal{P}$ is the *input transition relation*, and $\rho_\mathcal{P}^O \subseteq S_\mathcal{P} \times \Gamma_\mathcal{P}^O \times S_\mathcal{P}$ is the *output transition relation*. We often write an element (s, α, s') of a transition relation as $s \xrightarrow{\alpha} s'$, and call it a *step*. Given a state $s \in S_\mathcal{P}$ and a player $\gamma \in \{I, O\}$, the set of *moves* of player γ at s is $\Gamma_\mathcal{P}^\gamma(s) = \{\alpha \in \Gamma_\mathcal{P}^\gamma \mid \exists s' \in S_\mathcal{P}.(s \xrightarrow{\alpha} s') \in \rho_\mathcal{P}^\gamma\}$.

We require the transition relations to be deterministic: for $\gamma \in \{I, O\}$, if $(s, a, s') \in \rho_\mathcal{P}^\gamma$ and $(s, a, s'') \in \rho_\mathcal{P}^\gamma$, then $s' = s''$. Furthermore, we require time determinism for time steps over both relations: for all $\Delta \in \mathbb{T}$, if $(s, \Delta, s') \in \rho_\mathcal{P}^I$ and $(s, \Delta, s'') \in \rho_\mathcal{P}^O$, then $s' = s''$. Time steps of duration 0 do not leave the state: for $\gamma \in \{I, O\}$, if $(s, 0, s') \in \rho_\mathcal{P}^\gamma$, then $s = s'$. We also require Wang's Axiom [Yi90]: for all $\Delta, \Delta' \in \mathbb{T}$ with $\Delta' \leq \Delta$, we have $(s, \Delta, s'') \in \rho_\mathcal{P}^\gamma$ iff there is a state s' such that both $(s, \Delta', s') \in \rho_\mathcal{P}^\gamma$ and $(s', \Delta - \Delta', s'') \in \rho_\mathcal{P}^\gamma$. Finally, if there are any immediate actions available to a player in a state, then also the timed action 0 is available: for $\gamma \in \{I, O\}$, if $\Gamma_\mathcal{P}^\gamma(s) \neq \emptyset$, then $(s, 0, s) \in \rho_\mathcal{P}^\gamma$. ∎

The game proceeds as follows. At a state $s \in S_\mathcal{P}$, Input chooses a move from $\Gamma_\mathcal{P}^I(s)$, and Output chooses a move from $\Gamma_\mathcal{P}^O(s)$. If no moves are available for a player, that player will automatically lose the game. If two moves are played, then these determine both the successor state and the player bl that is *blamed* for having played first; assigning the blame is important in establishing whether a player is blocking the progress of time [AH97]. The following definition is asymmetric: when Input and Output play the same timed move, the Output player is blamed. As we will illustrate later, this asymmetry is necessary to capture the cause-effect relationship between outputs and inputs.

Definition 2 (move outcomes) For all states $s \in S_\mathcal{P}$ and moves $\alpha_I \in \Gamma_\mathcal{P}^I(s)$ and $\alpha_O \in \Gamma_\mathcal{P}^O(s)$, the *outcome* $\delta_\mathcal{P}(s, \alpha_I, \alpha_O)$ of α_I and α_O at s is the set of triples (α, s', bl) such that $(s \xrightarrow{\alpha} s') \in \rho_\mathcal{P}^I \cup \rho_\mathcal{P}^O$, and $\alpha \in \Gamma_\mathcal{P}^I \cup \Gamma_\mathcal{P}^O$ and $bl \in \{I, O\}$ are obtained as follows.

- If $\alpha_I, \alpha_O \in \mathbb{T}$, then $\alpha = \min\{\alpha_I, \alpha_O\}$. Moreover, $bl = I$ if $\alpha_I < \alpha_O$, and $bl = O$ otherwise.
- If $\alpha_I \in Acts_\mathcal{P}$ and $\alpha_O \in \mathbb{T}$, then $\alpha = \alpha_I$ and $bl = I$.
- If $\alpha_O \in Acts_\mathcal{P}$ and $\alpha_I \in \mathbb{T}$, then $\alpha = \alpha_O$ and $bl = O$.
- If $\alpha_I, \alpha_O \in Acts_\mathcal{P}$, then either $\alpha = \alpha_I$ and $bl = I$, or $\alpha = \alpha_O$ and $bl = O$. ∎

As usual, the players choose their moves according to strategies that may depend on the history of the game. Our strategies are partial functions, rather than total ones, because the sets of moves available to the players at a state can be empty. Furthermore, if both players choose immediate actions, then the outcome is nondeterministic. Consequently, the possible outcomes of two strategies form a set of finite and infinite sequences. A state is reachable in the game if it can be reached by some outcome of some input and output strategies.

Definition 3 (strategy outcomes) A *strategy* for player $\gamma \in \{I, O\}$ is a partial function $\pi^\gamma : S^*_{\mathcal{P}} \hookrightarrow \Gamma^\gamma_{\mathcal{P}}$ that associates, with every finite sequence of states $\overline{s} \in S^*_{\mathcal{P}}$ whose final state is s, a move $\pi^\gamma(\overline{s}) \in \Gamma^\gamma_{\mathcal{P}}(s)$ provided that $\Gamma^\gamma_{\mathcal{P}}(s) \neq \emptyset$; otherwise $\pi^\gamma(\overline{s})$ is undefined. Let $\Pi^I_{\mathcal{P}}$ be the set of strategies for player I, and let $\Pi^O_{\mathcal{P}}$ be the set of strategies for player O. Given a state $s \in S_{\mathcal{P}}$, an input strategy $\pi^I \in \Pi^I_{\mathcal{P}}$, and an output strategy $\pi^O \in \Pi^O_{\mathcal{P}}$, the set of *outcomes* $\widehat{\delta}_{\mathcal{P}}(s, \pi^I, \pi^O)$ of π^I and π^O from s consists of all finite and infinite sequences $\sigma = (s_0, bl_0), \alpha_1, (s_1, bl_1), \alpha_2, (s_2, bl_2), \ldots$ such that (1) $s_0 = s$; (2) $bl_0 \in \{I, O\}$; (3) if $|\sigma| < \infty$, then σ ends in a pair (s_k, bl_k) such that $\Gamma^I_{\mathcal{P}}(s_k) = \emptyset$ or $\Gamma^O_{\mathcal{P}}(s_k) = \emptyset$; and (4) for all $n < |\sigma|$, we have $(\alpha_{n+1}, s_{n+1}, bl_{n+1}) \in \delta_{\mathcal{P}}(s_n, \pi^I(\sigma_{0:n}), \pi^O(\sigma_{0:n}))$, where $\sigma_{0:n}$ denotes the prefix $(s_0, bl_0), \alpha_1, \ldots, (s_n, bl_n)$ of σ with length n.

A state $s \in S_{\mathcal{P}}$ is *reachable* in $\mathcal{P}$ if there are two strategies $\pi^I \in \Pi^I_{\mathcal{P}}$ and $\pi^O \in \Pi^O_{\mathcal{P}}$, and $k \geq 0$, such that $s = s_k$ for some outcome $(s_0, bl_0), a_1, (s_1, bl_1), a_2, (s_2, bl_2), \ldots \in \widehat{\delta}_{\mathcal{P}}(s^{init}_{\mathcal{P}}, \pi^I, \pi^O)$. ∎

3.1 Well-formedness of timed interfaces

A timed interface is well-formed if from every reachable state (i) Input has a strategy such that either time diverges, or Output is always to blame beyond some point; and (ii) symmetrically, Output has a strategy such that either time diverges, or Input is always to blame beyond some point. To give the precise definitions, let $time(\alpha) = \alpha$ for $\alpha \in \mathbb{T}$, and $time(\alpha) = 0$ otherwise.

Definition 4 (time divergence and time blocking) Let $\sigma = (s_0, bl_0), \alpha_1, (s_1, bl_1), \alpha_2, (s_2, bl_2), \ldots$ be an outcome of a game in $\mathcal{P}$. We define $\sigma \models t\_div$ if the accumulated time in σ is infinite, that is, $\sum_{k=1}^{|\sigma|} time(\alpha_k) = \infty$. For $\gamma \in \{I, O\}$, we define $\sigma \models blame^\gamma$ if either σ is finite and $\Gamma^\gamma_{\mathcal{P}}(s) = \emptyset$ for σ's last state s, or σ is infinite and there is a $k \geq 0$ such that $bl_i = \gamma$ for all $i > k$. For a set $U \subseteq S_{\mathcal{P}}$ of states, we define $\sigma \models \Box U$ if $s_k \in U$ for all $k \geq 0$. ∎

A state of a timed interface $\mathcal{P}$ is live in a set U of states if both players have strategies to stay forever in U and let time advance, unless the other player can be blamed for blocking the progress of time. Again, the game is not played symmetrically: Input can choose its strategy after Output, which shows that the game is turn-based (first Output chooses its move, then Input does).

Definition 5 (live states and well-formedness) Let $U \subseteq S_{\mathcal{P}}$ be a set of states. A state $s \in S_{\mathcal{P}}$ is *I-live in U* if Input can win the game with goal $(t\_div \vee$

$blame^O) \wedge \Box U$; that is, if for all strategies $\pi^O \in \Pi^O_{\mathcal{P}}$ there is a strategy $\pi^I \in \Pi^I_{\mathcal{P}}$ such that $\sigma \models (t\_div \vee blame^O) \wedge \Box U$ for all outcomes $\sigma \in \widehat{\delta}_{\mathcal{P}}(s, \pi^I, \pi^O)$. A state $s \in S_{\mathcal{P}}$ is *O-live in* U if Output can win the game with goal $(t\_div \vee blame^I) \wedge \Box U$; that is, if there is a strategy $\pi^O \in \Pi^O_{\mathcal{P}}$ such that for all strategies $\pi^I \in \Pi^I_{\mathcal{P}}$ and outcomes $\sigma \in \widehat{\delta}_{\mathcal{P}}(s, \pi^I, \pi^O)$, we have $\sigma \models (t\_div \vee blame^I) \wedge \Box U$. A state $s \in S_{\mathcal{P}}$ is *live in* U if it is both I-live and O-live in U.

The timed interface $\mathcal{P}$ is *well-formed in* U if all reachable states of $\mathcal{P}$ are live in U. The timed interface $\mathcal{P}$ is *well-formed* if it is well-formed in $S_{\mathcal{P}}$. ∎

In particular, for all reachable states s of a well-formed timed interface $\mathcal{P}$, both $\Gamma^I_{\mathcal{P}}(s) \neq \emptyset$ and $\Gamma^O_{\mathcal{P}}(s) \neq \emptyset$. Only well-formed timed interfaces represent valid interface specifications. For this reason, we will only define the composition of well-formed timed interfaces.

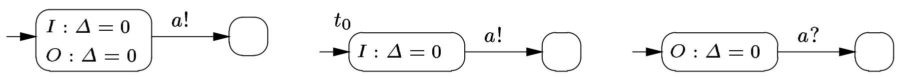
3.2 Product and composition of timed interfaces

Two timed interfaces $\mathcal{P}$ and $\mathcal{Q}$ are *composable* if $Acts^O_{\mathcal{P}} \cap Acts^O_{\mathcal{Q}} = \emptyset$. The *shared actions* of $\mathcal{P}$ and $\mathcal{Q}$ are given by $shared(\mathcal{P}, \mathcal{Q}) = Acts_{\mathcal{P}} \cap Acts_{\mathcal{Q}}$. For two composable timed interfaces $\mathcal{P}$ and $\mathcal{Q}$, their composition $\mathcal{P} \| \mathcal{Q}$ is computed in two steps: first, we form the *product* $\mathcal{P} \otimes \mathcal{Q}$ together with the set $i\text{-}errors(\mathcal{P}, \mathcal{Q})$ of immediate error states; then, $\mathcal{P} \| \mathcal{Q}$ is obtained by strengthening the input invariants of $\mathcal{P} \otimes \mathcal{Q}$ to make it well-formed in $S_{\mathcal{P} \otimes \mathcal{Q}} \setminus i\text{-}errors(\mathcal{P}, \mathcal{Q})$. The product $\mathcal{P} \otimes \mathcal{Q}$ represents the joint behavior of $\mathcal{P}$ and $\mathcal{Q}$, in which $\mathcal{P}$ and $\mathcal{Q}$ synchronize on the input timed moves, on the output timed moves, and on the shared actions, and behave independently otherwise.

Definition 6 (product) Given two composable timed interfaces $\mathcal{P}_1$ and $\mathcal{P}_2$, the *product* $\mathcal{P}_1 \otimes \mathcal{P}_2$ is the timed interface that consists of the following components.

- $S_{\mathcal{P}_1 \otimes \mathcal{P}_2} = S_{\mathcal{P}_1} \times S_{\mathcal{P}_2}$, and $s^{init}_{\mathcal{P}_1 \otimes \mathcal{P}_2} = (s^{init}_{\mathcal{P}_1}, s^{init}_{\mathcal{P}_2})$.
- $Acts^I_{\mathcal{P}_1 \otimes \mathcal{P}_2} = Acts^I_{\mathcal{P}_1} \cup Acts^I_{\mathcal{P}_2} \setminus shared(\mathcal{P}_1, \mathcal{P}_2)$, and $Acts^O_{\mathcal{P}_1 \otimes \mathcal{P}_2} = Acts^O_{\mathcal{P}_1} \cup Acts^O_{\mathcal{P}_2}$.
- $\rho^I_{\mathcal{P}_1 \otimes \mathcal{P}_2}$ is the set of transitions $(s_1, s_2) \xrightarrow{\alpha} (s'_1, s'_2)$ such that for $i = 1, 2$: if $\alpha \in \Gamma^I_{\mathcal{P}_i}$, then $(s_i \xrightarrow{\alpha} s'_i) \in \rho^I_{\mathcal{P}_i}$; otherwise $(s_i \xrightarrow{0} s'_i) \in \rho^I_{\mathcal{P}_i}$.
- $\rho^O_{\mathcal{P}_1 \otimes \mathcal{P}_2}$ is the set of transitions $(s_1, s_2) \xrightarrow{\alpha} (s'_1, s'_2)$ such that for $i = 1, 2$: if $\alpha \in \Gamma^O_{\mathcal{P}_i}$, then $(s_i \xrightarrow{\alpha} s'_i) \in \rho^O_{\mathcal{P}_i}$; if $\alpha \in Acts^I_{\mathcal{P}_i}$, then $(s_i \xrightarrow{\alpha} s'_i) \in \rho^I_{\mathcal{P}_i}$; and otherwise $(s_i \xrightarrow{0} s'_i) \in \rho^I_{\mathcal{P}_i}$. ∎

A state of the product is an *immediate error state* if one of the interfaces can produce an output that the other one cannot accept. A state of the product is a *time error state* if it is not live once we remove the immediate error states. The composition of two timed interfaces is obtained by pruning from the product all input transitions that start from or lead to a time error state.



(a) $\mathcal{P}_1 \otimes \mathcal{P}_2$ (b) $\mathcal{P}_1$ (c) $\mathcal{P}_2$

Fig. 2: Timed interfaces illustrating why well-formedness is needed.

Definition 7 (error states) Let $\mathcal{P}$ and $\mathcal{Q}$ be two well-formed and composable timed interfaces.

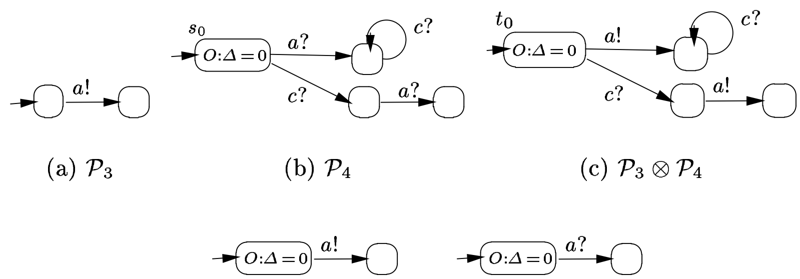
- *Immediate error states.* We say that a state $(s, t) \in S_{\mathcal{P}\otimes\mathcal{Q}}$ is an *immediate error state* if there is an action $\alpha \in \mathit{shared}(\mathcal{P}, \mathcal{Q})$ such that $(s \xrightarrow{\alpha} s') \in \rho^O_{\mathcal{P}}$ for some state s', but $(t \xrightarrow{\alpha} t') \notin \rho^I_{\mathcal{Q}}$ for all states t', or such that $(t \xrightarrow{\alpha} t') \in \rho^O_{\mathcal{Q}}$ for some t', but $(s \xrightarrow{\alpha} s') \notin \rho^I_{\mathcal{P}}$ for all s'. We denote by $\mathit{i\text{-}errors}(\mathcal{P}, \mathcal{Q}) \subseteq S_{\mathcal{P}\otimes\mathcal{Q}}$ the set of immediate error states.
- *Time error states.* We say that a state $(s, t) \in S_{\mathcal{P}\otimes\mathcal{Q}}$ is a *time error state* if (s, t) is reachable in $\mathcal{P} \otimes \mathcal{Q}$, but it is not I-live in $S_{\mathcal{P}\otimes\mathcal{Q}} \setminus \mathit{i\text{-}errors}(\mathcal{P}, \mathcal{Q})$. We denote by $\mathit{t\text{-}errors}(\mathcal{P}, \mathcal{Q}) \subseteq S_{\mathcal{P}\otimes\mathcal{Q}}$ the set of time error states. ∎

Note that the reachable immediate error states are a subset of the time error states. The composition $\mathcal{P}\|\mathcal{Q}$ is obtained by restricting the input behavior of $\mathcal{P} \otimes \mathcal{Q}$ to avoid all time error states. We restrict the input behavior only, leaving the output behavior unchanged, because when composing interfaces we can strengthen their input assumptions to ensure that no incompatibility arises, but we cannot modify their output behavior.

Definition 8 (compatibility and composition) Let $\mathcal{P}$ and $\mathcal{Q}$ be two well-formed and composable timed interfaces. The interfaces $\mathcal{P}$ and $\mathcal{Q}$ are *compatible* if $(s^{init}_{\mathcal{P}}, s^{init}_{\mathcal{Q}}) \notin \mathit{t\text{-}errors}(\mathcal{P}, \mathcal{Q})$. If $\mathcal{P}$ and $\mathcal{Q}$ are compatible, then the composition $\mathcal{P}\|\mathcal{Q}$ is defined by restricting the input transition relation, so that no error states are entered. Formally, abbreviating $U = S_{\mathcal{P}\otimes\mathcal{Q}} \setminus \mathit{t\text{-}errors}(\mathcal{P}, \mathcal{Q})$, we define $\rho^I_{\mathcal{P}\|\mathcal{Q}} = \rho^I_{\mathcal{P}\otimes\mathcal{Q}} \cap (U \times \mathit{Acts}^I_{\mathcal{P}\otimes\mathcal{Q}} \times U)$; all other components of $\mathcal{P}\|\mathcal{Q}$ are defined as in $\mathcal{P} \otimes \mathcal{Q}$. ∎

3.3 Discussion

Well-formedness. The composition of timed interfaces that are not well-formed may yield undesirable results, and hence is not defined in our theory. To illustrate this point, consider the interfaces in Figure 2. The time steps are represented as follows: for player $\gamma \in \{I, O\}$, if state s has label $\gamma : \Delta = 0$, then only the time step $(s, 0, s)$ is available for γ; if s has no γ-label, then all time steps (s, Δ, s) with $\Delta \in \mathbb{T}$ are available for γ at s. In interface $\mathcal{P}_1$, there is no deadline associated with the immediate move a: Output can play it at any time, or not at all. Similarly, $\mathcal{P}_2$ does not associate a deadline to a: Input can play it at any



(a) $\mathcal{P}_3$ (b) $\mathcal{P}_4$ (c) $\mathcal{P}_3 \otimes \mathcal{P}_4$

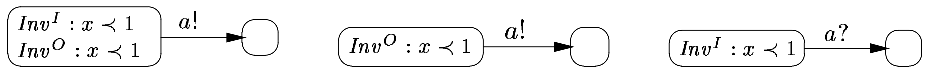
(d) $(\mathcal{P}_3 \parallel \mathcal{P}_4) \parallel \mathcal{Q}$ (e) $\mathcal{P}_3 \parallel \mathcal{Q}$

Fig. 3: Timed interfaces illustrating why two transition relations are needed.

time, or not at all. Note that $\mathcal{P}_2$ is not well-formed. This is because Output can only play the timed move 0 at state t_0, so t_0 is not O-live: if Input also plays the timed move 0, Output can neither let time diverge, nor blame Input. Consider now the product $\mathcal{P}_1 \otimes \mathcal{P}_2$. The product specifies that the output move a *must* be played at time 0, even though no such deadline for a was present at $\mathcal{P}_1$ or $\mathcal{P}_2$. The problem, intuitively, is that the deadline of 0 for Output in $\mathcal{P}_2$ does not apply to any output move. When an unrelated output move becomes present, such as a in $\mathcal{P}_1 \otimes \mathcal{P}_2$, the deadline is improperly transferred to the move. The well-formness condition requires that a player has a deadline only if it also has an action that can satisfy the deadline, thus preventing such "deadline transfers."

Two transition relations. If we use a single transition relation, rather than one for Input and one for Output, then both players would share the same timed moves at a state. The following example shows that, if we do so, the composition of interfaces may again introduce timing requirements that are not present in the component interfaces. In particular, composition may not be associative. Consider the timed interfaces $\mathcal{P}_3$ and $\mathcal{P}_4$ in Figure 3, where the timing constraints apply to both Input and Output moves. The interfaces $\mathcal{P}_3$ and $\mathcal{P}_4$ are compatible. In state s_0, the interface $\mathcal{P}_4$ has to take an input action at time 0. This input need not be provided by $\mathcal{P}_3$, because $\mathcal{P}_3$ is not forced to take its $a!$ action. Nevertheless, $\mathcal{P}_3$ and $\mathcal{P}_4$ work together in an environment that provides a $c!$ action at time 0. Now consider the product $\mathcal{P}_3 \otimes \mathcal{P}_4$ in Figure 3(c); note that $\mathcal{P}_3 \otimes \mathcal{P}_4 = \mathcal{P}_3 \parallel \mathcal{P}_4$, because the product is well-formed. The composition $\mathcal{P}_3 \parallel \mathcal{P}_4$ specifies that, if c is not received at 0, then output a is produced at time 0. As a result, $\mathcal{P}_3 \parallel \mathcal{P}_4$ also works in environments that never provide a c input. More formally, let $\mathcal{Q}$ be the interface that has an output move c, which can never be taken. Then $(\mathcal{P}_3 \parallel \mathcal{P}_4)$ and $\mathcal{Q}$ are compatible. However, $\mathcal{P}_4 \parallel \mathcal{Q}$ (Figure 3(e)) and $\mathcal{P}_3$ are not compatible. Hence, the compatibility of $\mathcal{P}_3$, $\mathcal{P}_4$, and $\mathcal{Q}$ depends on the order in which they are composed.

Game asymmetry. This example uses the timed interface automata notation as in Section 2, to be introduced formally in the next section. Consider the



(a) $\mathcal{A}$ (b) $\mathcal{A}_1$ (c) $\mathcal{A}_2$

Fig. 4: Timed interfaces illustrating why O is blamed and why I plays second.

automata $\mathcal{A}$, $\mathcal{A}_1$, and $\mathcal{A}_2$ in Figure 4, where $\prec \in \{<, \leq\}$. Note that $\mathcal{A} = \mathcal{A} \otimes \mathcal{A}_2$. Since $\mathcal{A}_1$ provides an output a within the deadline required by $\mathcal{A}_2$, the automata $\mathcal{A}_1$ and $\mathcal{A}_2$ should be compatible, and $\mathcal{A}$ should be well-formed. Consider first the case when $\prec$ is $\leq$. If Input is blamed when it plays the same time move as Output, then $\mathcal{A}$ is not well-formed. In fact, Output can play the strategy of advancing time until $x = 1$, and thereafter always play timed move 0. Input cannot let time progress, nor can it blame Output. Hence, any state (s_0, x) with $x \leq 1$ is not I-live, and $\mathcal{A}$ is not well-formed. Consider now the case when $\prec$ is $<$. If Input cannot play second, then for each strategy π^I of Input, there is a strategy π^O for Output that plays later, i.e., a strategy π^O such that $\pi^I(\overline{s}) < \pi^O(\overline{s}) < 1$ for all histories $\overline{s}$. Then the outcome of this strategy neither is time divergent nor does it blame O. If, on the other hand, I plays second, then it can win the game by playing later than O.

4 Timed Interface Automata

Timed interfaces provide a finite representation for timed games and serve as a basis on which the algorithms for compatibility checking and composition operate. Their syntax recalls that of timed automata [AD94]. In particular, timed interface automata use *clock variables* in order to keep track of the amount of time elapsed. The value of these variables can be reset to 0 when immediate actions occur, and otherwise increase with unit rate. Let $\mathcal{X}$ be a set of variables over the time domain $\mathbb{T}$. A *clock condition* over $\mathcal{X}$ is a boolean combination of formulas of the form $x \prec c$ or $x - y \prec c$, where c is an integer, $x, y \in \mathcal{X}$, and $\prec$ is either of $<$ or $\leq$. We denote the set of all clock conditions over $\mathcal{X}$ by $\Xi[\mathcal{X}]$.

Definition 9 (timed interface automata) A *timed interface automaton* (or TIA) is a tuple $\mathcal{A} = (Q_\mathcal{A}, q^{init}_\mathcal{A}, \mathcal{X}_\mathcal{A}, Acts^I_\mathcal{A}, Acts^O_\mathcal{A}, Inv^I_\mathcal{A}, Inv^O_\mathcal{A}, \rho_\mathcal{A})$ consisting of the following components.

- $Q_\mathcal{A}$ is a finite set of *locations.*
- $q^{init}_\mathcal{A} \in Q_\mathcal{A}$ is the *initial location.*
- $\mathcal{X}_\mathcal{A}$ is a finite set of *clocks.*
- $Acts^I_\mathcal{A}$ and $Acts^O_\mathcal{A}$ are finite and disjoint sets of *input* and *output actions,* respectively. Let $Acts_\mathcal{A} = Acts^I_\mathcal{A} \cup Acts^O_\mathcal{A}$ denote the set of all actions of $\mathcal{A}$.
- $Inv^I_\mathcal{A}$: $Q_\mathcal{A} \mapsto \Xi[\mathcal{X}_\mathcal{A}]$ maps each location of $\mathcal{A}$ to its *input invariant.*
- $Inv^O_\mathcal{A}$: $Q_\mathcal{A} \mapsto \Xi[\mathcal{X}_\mathcal{A}]$ maps each location of $\mathcal{A}$ to its *output invariant.*

- $\rho_{\mathcal{A}} \subseteq Q_{\mathcal{A}} \times \Xi[\mathcal{X}_{\mathcal{A}}] \times \mathit{Acts}_{\mathcal{A}} \times 2^{\mathcal{X}_{\mathcal{A}}} \times Q_{\mathcal{A}}$ is the *transition relation*. For $(q, g, a, r, q') \in \rho_{\mathcal{A}}$, the locations q and q' are the source and destination of the transition, $g \in \Xi[\mathcal{X}_{\mathcal{A}}]$ is a guard on the clock values that specifies when the transition can be taken, $a \in \mathit{Acts}_{\mathcal{A}}$ is an action labeling the transition, and $r \subseteq \mathcal{X}_{\mathcal{A}}$ is a set of clocks that are reset by the transition. We require the transition relation to be deterministic: for all $q \in Q_{\mathcal{A}}$ and $a \in \mathit{Acts}_{\mathcal{A}}$, there is at most one tuple of the form (q, g, a, r, q') with $(q, g, a, r, q') \in \rho_{\mathcal{A}}$. ∎

A *valuation* over a set $\mathcal{X}$ of clock variables is a function $v \colon \mathcal{X} \mapsto \mathbb{T}$. We write $\mathbf{0}_{\mathcal{X}}$ for the valuation that assigns 0 to all clocks in $\mathcal{X}$, and $\mathcal{V}(\mathcal{X})$ for the set of all valuations over $\mathcal{X}$. Given a valuation $v \in \mathcal{V}(\mathcal{X})$, we write $v + \Delta$ for the valuation defined by $(v + \Delta)(x) = v(x) + \Delta$ for all $x \in \mathcal{X}$. Given a set $r \subseteq \mathcal{X}$ of clocks, we write $v[r := 0]$ for the valuation that maps x to 0 if $x \in r$, and otherwise to $v(x)$. Given a clock condition $\varphi \in \Xi[\mathcal{X}]$, we write $v \models \varphi$ if φ is true under the valuation v. For $r \subseteq \mathcal{X}$, we write $\varphi[r := 0]$ for the condition obtained from φ by replacing every $x \in r$ by 0; obviously, $v[r := 0] \models \varphi$ iff $v \models \varphi[r := 0]$.

Definition 10 (timed interfaces induced by TIA) The TIA $\mathcal{A}$ is *nonempty* if $\mathbf{0}_{\mathcal{X}_{\mathcal{A}}} \models \mathit{Inv}^I_{\mathcal{A}}(q^{init}_{\mathcal{A}}) \wedge \mathit{Inv}^O_{\mathcal{A}}(q^{init}_{\mathcal{A}})$. A nonempty TIA $\mathcal{A}$ induces a timed interface $\mathcal{P} = [\![\mathcal{A}]\!]$ that has the state set $S_{\mathcal{P}} = \{\langle p, v\rangle \mid p \in Q_{\mathcal{A}},\ v \in \mathcal{V}(\mathcal{X}_{\mathcal{A}})\}$ and the initial state $s^{init}_{\mathcal{P}} = \langle q^{init}_{\mathcal{A}}, \mathbf{0}_{\mathcal{X}_{\mathcal{A}}}\rangle$. The actions are $\mathit{Acts}^I_{\mathcal{P}} = \mathit{Acts}^I_{\mathcal{A}}$ and $\mathit{Acts}^O_{\mathcal{P}} = \mathit{Acts}^O_{\mathcal{A}}$. For $\gamma \in \{I, O\}$ the transition relations of $\mathcal{P}$ are defined by $(\langle p, v\rangle \xrightarrow{\alpha} \langle p', v'\rangle) \in \rho^{\gamma}_{\mathcal{P}}$ if either (1) $\alpha \in \mathbb{T}$, $p = p'$, $v' = v + \alpha$, and for all $0 \le \Delta' \le \alpha$, we have $v + \Delta' \models \mathit{Inv}^{\gamma}_{\mathcal{A}}(p)$; or (2) $\alpha \in \mathit{Acts}^{\gamma}_{\mathcal{A}}$, and there is a tuple $(p, g, \alpha, r, p') \in \rho_{\mathcal{A}}$ with $v \models \mathit{Inv}^{\gamma}_{\mathcal{A}}(p) \wedge g$, $v' = v[r := 0]$, and $v' \models \mathit{Inv}^{\gamma}_{\mathcal{A}}(p')$.

The TIA $\mathcal{A}$ is *well-formed* if it is nonempty and the corresponding timed interface $[\![\mathcal{A}]\!]$ is well-formed. ∎

4.1 Product and composition of timed interface automata

Two TIAs $\mathcal{A}$ and $\mathcal{B}$ are *composable* if $\mathit{Acts}^O_{\mathcal{A}} \cap \mathit{Acts}^O_{\mathcal{B}} = \emptyset$ and $\mathcal{X}_{\mathcal{A}} \cap \mathcal{X}_{\mathcal{B}} = \emptyset$; their *shared actions* are $\mathit{shared}(\mathcal{A}, \mathcal{B}) = \mathit{Acts}_{\mathcal{A}} \cap \mathit{Acts}_{\mathcal{B}}$.

Definition 11 (product) For two composable TIAs $\mathcal{A}_1$ and $\mathcal{A}_2$, the *product* $\mathcal{A}_1 \otimes \mathcal{A}_2$ is the TIA that consists of the following components.

- $Q_{\mathcal{A}_1 \otimes \mathcal{A}_2} = Q_{\mathcal{A}_1} \times Q_{\mathcal{A}_2}$, and $q^{init}_{\mathcal{A}_1 \otimes \mathcal{A}_2} = (q^{init}_{\mathcal{A}_1}, q^{init}_{\mathcal{A}_2})$.
- $\mathcal{X}_{\mathcal{A}_1 \otimes \mathcal{A}_2} = \mathcal{X}_{\mathcal{A}_1} \cup \mathcal{X}_{\mathcal{A}_2}$.
- $\mathit{Acts}^I_{\mathcal{A}_1 \otimes \mathcal{A}_2} = \mathit{Acts}^I_{\mathcal{A}_1} \cup \mathit{Acts}^I_{\mathcal{A}_2} \setminus \mathit{shared}(\mathcal{A}_1, \mathcal{A}_2)$, and $\mathit{Acts}^O_{\mathcal{A}_1 \otimes \mathcal{A}_2} = \mathit{Acts}^O_{\mathcal{A}_1} \cup \mathit{Acts}^O_{\mathcal{A}_2}$.
- $\mathit{Inv}^I_{\mathcal{A}_1 \otimes \mathcal{A}_2}(p, q) = \mathit{Inv}^I_{\mathcal{A}_1}(p) \wedge \mathit{Inv}^I_{\mathcal{A}_2}(q)$ and $\mathit{Inv}^O_{\mathcal{A}_1 \otimes \mathcal{A}_2}(p, q) = \mathit{Inv}^O_{\mathcal{A}_1}(p) \wedge \mathit{Inv}^O_{\mathcal{A}_2}(q)$.
- $\rho_{\mathcal{A}_1 \otimes \mathcal{A}_2}$ is the set of transitions $((q_1, q_2), g_1 \wedge g_2, a, r_1 \cup r_2, (q'_1, q'_2))$ such that, for $i = 1, 2$: if $a \in \mathit{Acts}_{\mathcal{A}_i}$, then (q_i, g_i, a, r_i, q'_i) is a transition in $\rho_{\mathcal{A}_i}$; otherwise $q_i = q'_i$, $g_i = \mathit{true}$, and $r_i = \emptyset$. ∎

Theorem 1 *For nonempty and composable TIAs $\mathcal{A}$ and $\mathcal{B}$, we have $[\![\mathcal{A} \otimes \mathcal{B}]\!] = [\![\mathcal{A}]\!] \otimes [\![\mathcal{B}]\!]$.*

A *location labeling* for a TIA $\mathcal{A}$ is a function $\xi \colon Q_{\mathcal{A}} \mapsto \Xi[\mathcal{X}_{\mathcal{A}}]$ that associates with each location p of $\mathcal{A}$ a condition $\xi(p)$ over the clocks in $\mathcal{X}_{\mathcal{A}}$. The location labeling ξ defines the state set $[\![\xi]\!]_{\mathcal{A}} = \{\langle p, v\rangle \in S_{[\![\mathcal{A}]\!]} \mid v \models \xi(p)\}$ of the corresponding timed interface. We denote by $\mathit{I\text{-}live}_{\mathcal{A}}(\xi)$ the location labeling that defines the set $[\![\mathit{I\text{-}live}_{\mathcal{A}}(\xi)]\!]_{\mathcal{A}}$ of I-live states in $[\![\xi]\!]_{\mathcal{A}}$. By computing $[\![\mathit{I\text{-}live}_{\mathcal{A}}(\xi)]\!]_{\mathcal{A}}$ as the solution of a game on the region graph (see Section 4.2), we will see that the I-live states are indeed definable by clock conditions. To define the composition on TIAs, we also need the following enabling conditions. For $\gamma \in \{I, O\}$, a location $q \in Q_{\mathcal{A}}$, and an action $a \in \mathit{Acts}^{\gamma}_{\mathcal{A}}$, let $\mathit{enab}^{\gamma}_{\mathcal{A}}(q,a)$ be $\mathit{Inv}^{\gamma}_{\mathcal{A}}(q) \wedge g \wedge \mathit{Inv}^{\gamma}_{\mathcal{A}}(q')[r := 0]$ if there is a transition $(q, g, a, r, q') \in \rho_{\mathcal{A}}$; otherwise let $\mathit{enab}^{\gamma}_{\mathcal{A}}(q,a)$ be *false*. Given two composable TIAs $\mathcal{A}$ and $\mathcal{B}$, the states of $[\![\mathcal{A} \otimes \mathcal{B}]\!]$ that are not immediate error states can be defined by the location labeling $\mathit{ok}_{\mathcal{A}\otimes\mathcal{B}}$ that associates with each product location $(p,q) \in Q_{\mathcal{A}\otimes\mathcal{B}}$ the clock condition

$$\bigwedge_{a \in \mathit{Acts}^O_{\mathcal{A}} \cap \mathit{Acts}^I_{\mathcal{B}}} (\mathit{enab}^O_{\mathcal{A}}(p,a) \rightarrow \mathit{enab}^I_{\mathcal{B}}(q,a)) \wedge \bigwedge_{a \in \mathit{Acts}^O_{\mathcal{B}} \cap \mathit{Acts}^I_{\mathcal{A}}} (\mathit{enab}^O_{\mathcal{B}}(q,a) \rightarrow \mathit{enab}^I_{\mathcal{A}}(p,a)).$$

Definition 12 (compatibility and composition) Two well-formed and composable TIAs $\mathcal{A}$ and $\mathcal{B}$ are *compatible* if the corresponding timed interface $[\![\mathcal{A}]\!]$ and $[\![\mathcal{B}]\!]$ are compatible. The *composition* $\mathcal{A} \| \mathcal{B}$ is obtained from the product $\mathcal{A} \otimes \mathcal{B}$ by replacing the input invariants $\mathit{Inv}^I_{\mathcal{A}\otimes\mathcal{B}}$ with the location labeling $\mathit{I\text{-}live}_{\mathcal{A}\otimes\mathcal{B}}(\mathit{ok}_{\mathcal{A}\otimes\mathcal{B}})$. ∎

Theorem 2 *Two well-formed and composable TIAs $\mathcal{A}$ and $\mathcal{B}$ are compatible iff the composition $\mathcal{A} \| \mathcal{B}$ is nonempty. Moreover, if $\mathcal{A}$ and $\mathcal{B}$ are compatible, then $[\![\mathcal{A} \| \mathcal{B}]\!] = [\![\mathcal{A}]\!] \| [\![\mathcal{B}]\!]$, and $\mathcal{A} \| \mathcal{B}$ is well-formed.*

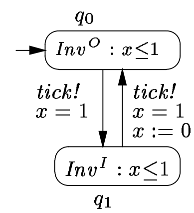
The following theorem states that composition is associative up to the equivalence $\equiv$, which for TIAs $\mathcal{A}$ and $\mathcal{B}$ is defined by $\mathcal{A} \equiv \mathcal{B}$ if $\rho^I_{[\![\mathcal{A}]\!]} \cap (\mathit{Reach}([\![\mathcal{A}]\!]) \times \mathit{Acts}^I_{[\![\mathcal{A}]\!]} \times S_{[\![\mathcal{A}]\!]}) = \rho^I_{[\![\mathcal{B}]\!]} \cap (\mathit{Reach}([\![\mathcal{B}]\!]) \times \mathit{Acts}^I_{[\![\mathcal{B}]\!]} \times S_{[\![\mathcal{B}]\!]})$ and all other components of $[\![\mathcal{A}]\!]$ and $[\![\mathcal{B}]\!]$ are the same. Here, $\mathit{Reach}(\mathcal{P})$ denotes the set of reachable states of the timed interface $\mathcal{P}$.

Theorem 3 *If $\mathcal{A}$, $\mathcal{B}$, and $\mathcal{C}$ are well-formed and pairwise composable TIAs, then $(\mathcal{A} \| \mathcal{B}) \| \mathcal{C} \equiv \mathcal{A} \| (\mathcal{B} \| \mathcal{C})$.*

4.2 Algorithms for composition and well-formedness checking

Live states. Before presenting the algorithms for checking well-formedness and computing composition, we show that, given a location labeling ξ, we can compute the labeling $\mathit{I\text{-}live}_{\mathcal{A}}(\xi)$ that defines the set of states in $\mathcal{A}$ where I can win the game on $\mathcal{A}$ with goal $(\mathit{t\_div} \vee \mathit{blame}^O) \wedge \Box[\![\xi]\!]_{\mathcal{A}}$.

To use existing algorithms, we first transform this game into an equivalent one with an ω-regular goal [Tho90]. Consider the TIA Tick^O shown on the right, where the action tick and the clock x are fresh. Thus, Tick^O observes the progress of time, and visits location q_1 every time unit. In particular, time diverges iff q_1 is visited infinitely often. Hence, Input can win in $[\![\mathcal{A}]\!]$ the game with goal $(t\_div \vee \mathit{blame}^O) \wedge \Box[\![\xi]\!]_{\mathcal{A}}$ iff Input can win in $[\![\mathcal{A} \otimes \mathit{Tick}^O]\!]$ the game with goal $(\mathit{blame}^O \vee \Box\Diamond q_1) \wedge \Box[\![\xi]\!]_{\mathcal{A}}$. On the enlarged state space $S_{[\![\mathcal{A}\otimes \mathit{Tick}^O]\!]} \times \{I, O\}$, where the states record the blame, this latter goal can be rewritten as the ω-regular goal φ_I: $(\Diamond\Box(bl = O) \vee \Box\Diamond q_1) \wedge \Box[\![\xi]\!]_{\mathcal{A}}$. The game with goal φ_I can be solved using the algorithms of [EJ91,dAHM01], which use a *controllable predecessor operator IPre*. Given a timed interface $\mathcal{P}$ and a set U of states, $\mathit{IPre}_{\mathcal{P}}(U)$ yields all states in which Input can in one move force the game into U. Formally, $\mathit{IPre}_{\mathcal{P}}(U)$ contains all states $s \in S_{\mathcal{P}}$ such that $\forall \alpha_O \in \Gamma^O_{\mathcal{P}}(s).\, \exists \alpha_I \in \Gamma^I_{\mathcal{P}}(s).\, \delta_{\mathcal{P}}(s, \alpha_I, \alpha_O) \subseteq U$. The set win_I of states in $S_{[\![\mathcal{A}\otimes \mathit{Tick}^O]\!]} \times \{I, O\}$ where Input can win the game with goal φ_I can be characterized by [EJ91]



$$\mathit{win}_I = \nu Z.\mu Y.\nu X.\Big[\xi \wedge \big[(q_0 \wedge bl{=}O \wedge \mathit{IPre}_{\mathcal{P}}(X)) \vee \\ (q_0 \wedge bl{=}I \wedge \mathit{IPre}_{\mathcal{P}}(Y)) \vee (q_1 \wedge \mathit{IPre}_{\mathcal{P}}(Z))\big]\Big].$$

for $\mathcal{P} = [\![\mathcal{A} \otimes \mathit{Tick}^O]\!]$. One can define the *region graph* of a TIA as for timed automata [AD94]. Since the operation *IPre* is computable on the region graph [MPS95], the expression above suggests a symbolic fixpoint algorithm. The result win_I can be expressed as a location labeling ζ on $\mathcal{A}$. We then obtain the desired labeling $I\text{-}\mathit{live}_{\mathcal{A}}(\xi)$ by letting $I\text{-}\mathit{live}_{\mathcal{A}}(\xi)(p) = \exists x.\exists bl.(\zeta(p, q_0) \vee \zeta(p, q_1))$.

Well-formedness. The following theorem shows that, to check the well-formedness of a TIA $\mathcal{A}$, we need to compute the location labelings $\mathit{Reach}(\mathcal{A})$, $I\text{-}\mathit{live}_{\mathcal{A}}(\mathit{True}_{\mathcal{A}})$, and $O\text{-}\mathit{live}_{\mathcal{A}}(\mathit{True}_{\mathcal{A}})$. Here, $\mathit{Reach}(\mathcal{A})$ is the location labeling that defines the set $[\![\mathit{Reach}(\mathcal{A})]\!]_{\mathcal{A}}$ of reachable states of $[\![\mathcal{A}]\!]$, and $O\text{-}\mathit{live}_{\mathcal{A}}(\xi)$ is the labeling that defines the set of O-live states in $[\![\xi]\!]_{\mathcal{A}}$, and $\mathit{True}_{\mathcal{A}}$ denotes the labeling that assigns *true* to each location. The set $[\![\mathit{Reach}(\mathcal{A})]\!]_{\mathcal{A}}$ is definable from clock conditions, because it can be computed on the region graph in the same way as the reachable states of a timed automaton can be computed [AD94]. Since $[\![O\text{-}\mathit{live}_{\mathcal{A}}(\xi)]\!]_{\mathcal{A}}$, for a location labeling ξ, can be computed similarly to $[\![I\text{-}\mathit{live}_{\mathcal{A}}(\xi)]\!]_{\mathcal{A}}$, it is also definable by clock conditions.

Theorem 4 *A TIA $\mathcal{A}$ is well-formed iff for all locations $p \in Q_{\mathcal{A}}$ the implication $\mathit{Reach}(\mathcal{A})(p) \rightarrow (I\text{-}\mathit{live}_{\mathcal{A}}(\mathit{True}_{\mathcal{A}})(p) \wedge O\text{-}\mathit{live}_{\mathcal{A}}(\mathit{True}_{\mathcal{A}})(p))$ is valid.*

Composition. The composition of two well-formed and composable TIAs $\mathcal{A}$ and $\mathcal{B}$ can be obtained from their product by replacing the input invariants $\mathit{Inv}^I_{\mathcal{A}\otimes\mathcal{B}}$ with the location labeling $I\text{-}\mathit{live}_{\mathcal{A}\otimes\mathcal{B}}(ok_{\mathcal{A}\otimes\mathcal{B}})$ (Definition 12). Then their compatibility can be decided by checking whether their composition is empty (Theorem 2).

References

[Abr96] S. Abramsky. Semantics of interaction. In *Trees in Algebra and Programming*, volume 1059 of *Lect. Notes in Comp. Sci.*, page 1. Springer, 1996.

[AD94] R. Alur and D.L. Dill. A theory of timed automata. *Theoretical Computer Science*, 126:183–235, 1994.

[AH97] R. Alur and T.A. Henzinger. Modularity for timed and hybrid systems. In *Concurrency Theory*, volume 1243 of *Lect. Notes in Comp. Sci.*, pages 74–88. Springer, 1997.

[AMPS98] E. Asarin, O. Maler, A. Pnueli, and J. Sifakis. Controller synthesis for timed automata. In *Proc. IFAC Symp. System Structure and Control*, pages 469–474. Elsevier, 1998.

[CRR02] S. Chaki, S.K. Rajamani, and J. Rehof. Types as models: Model checking message-passing programs. In *Proc. Symp. Principles of Programming Languages*, pages 45–57. ACM, 2002.

[dAH01a] L. de Alfaro and T.A. Henzinger. Interface automata. In *Proc. Symp. Foundations of Software Engineering*, pages 109–120. ACM, 2001.

[dAH01b] L. de Alfaro and T.A. Henzinger. Interface theories for component-based design. In *Embedded Software*, volume 2211 of *Lect. Notes in Comp. Sci.*, pages 148–165. Springer, 2001.

[dAHM01] L. de Alfaro, T.A. Henzinger, and R. Majumdar. Symbolic algorithms for infinite-state games. In *Concurrency Theory*, volume 2154 of *Lect. Notes in Comp. Sci.*, pages 536–550. Springer, 2001.

[Dil88] D.L. Dill. *Trace Theory for Automatic Hierarchical Verification of Speed-Independent Circuits*. MIT Press, 1988.

[EJ91] E.A. Emerson and C.S. Jutla. Tree automata, mu-calculus, and determinacy. In *Proc. Symp. Foundations of Computer Science*, pages 368–377. IEEE Computer Society, 1991.

[MMT91] M. Merritt, F. Modugno, and M. Tuttle. Time constrained automata. In *Concurrency Theory*, volume 527 of *Lect. Notes in Comp. Sci.*, pages 408–423. Springer, 1991.

[MPS95] O. Maler, A. Pnueli, and J. Sifakis. On the synthesis of discrete controllers for timed systems. In *Theoretical Aspects of Computer Science*, volume 900 of *Lect. Notes in Comp. Sci.*, pages 229–242. Springer, 1995.

[RGG96] E. Rudolph, P. Graubmann, and J. Gabowski. Tutorial on message sequence charts. *Computer Networks and ISDN Systems–SDL and MSC*, 28:1629–1641, 1996.

[RR01] S.K. Rajamani and J. Rehof. A behavioral module system for the pi-calculus. In *Static Analysis Symposium*, volume 2126 of *Lect. Notes in Comp. Sci.*, pages 375–394. Springer, 2001.

[SGSAL98] R. Segala, G. Gawlick, J. Søgaard-Andersen, and N. Lynch. Liveness in timed and untimed systems. *Information and Computation*, 141:119–171, 1998.

[Tho90] W. Thomas. Automata on infinite objects. In J. van Leeuwen, ed., *Handbook of Theoretical Computer Science*, volume B, pages 135–191. Elsevier, 1990.

[Yi90] Wang Yi. Real-time behaviour of asynchronous agents. In *Concurrency Theory*, volume 458 of *Lect. Notes in Comp. Sci.*, pages 502–520. Springer, 1990.