Specification Theories for Probabilistic and Real-Time Systems

Uli Fahrenberg, Axel Legay, and Louis-Marie Traonouez

Inria/IRISA, Rennes, France

Abstract. We survey extensions of modal transition systems to specification theories for probabilistic and timed systems.

1 Introduction

Many modern systems are big and complex assemblies of numerous components called implementations. The implementations are often designed by independent teams, working under a common agreement on what the specification of each implementation should be.

Over the past, one has agreed that any good specification theory should be equipped with a satisfaction relation (to decide whether an implementation satisfies a specification), a consistency check (to decide whether the specification admits an implementation), a refinement (to compare specifications in terms of inclusion of sets of implementations), logical composition (to compute the intersection of sets of implementations), and structural composition (to combine specifications).

The design of "good" specification theories has been the subject of intensive study, most of them for the case where implementations are represented by transition systems. In this paper, we survey two seminal works on extending specification theories to both probabilistic and timed systems.

Specification Theory for Probabilistic Systems. We consider implementations represented by *probabilistic automata* (PA). Probabilistic automata constitute a mathematical framework for the description and analysis of nondeterministic probabilistic systems. They have been developed by Segala [30] to model and analyze asynchronous, concurrent systems with discrete probabilistic choice in a formal and precise way. PA are akin to Markov decision processes (MDP). A detailed comparison with models such as MDP, as well as generative and reactive probabilistic transition systems is given in [29]. PA are recognized as an adequate formalism for randomized distributed algorithms and fault tolerant systems. They are used as semantic model for formalisms such as probabilistic process algebra [28] and a probabilistic variant of Harel's statecharts [20]. An input-output version of PA is the basis of PIOA and variants thereof [4, 7]. PA have been enriched with notions such as weak and strong (bi)simulations [30], decision algorithms for these notions [6] and a statistical testing theory [8].

S. Bensalem et al. (Eds.): FPS 2014 (Sifakis Festschrift), LNCS 8415, pp. 98–117, 2014.

In [14], we have introduced *abstract probabilistic automata* (APA) as a specification theory for PA. APA aims at model reduction by collapsing sets of concrete states to abstract states, *e.g.* by partitioning the concrete state space. This paper presents a three-valued abstraction of PA. The main design principle of our model is to abstract sets of distributions by constraint functions. This generalizes earlier work on interval-based abstraction of probabilistic systems [19,21,22]. To abstract from action transitions, we introduce *may*- and *must*-modalities in the spirit of modal transition systems [24, 26]. If all states in a partition p have a must-transition on action a to some state in partition p', the abstraction yields a must-transition between p and p'. If some of the p-states have no such transition while others do, it gives rise to a may-transition between p and p'. In this paper we will summarize main results on APA. We will also show how the model can be used as a specification theory for PA.

Specification Theory for Timed Systems. In [9,10], we represent both specifications and implementations by timed input/output transition systems [23], *i.e.* timed transitions systems whose sets of discrete transitions are split into Input and Output transitions. In contrast to [11] and [23], we distinguish between implementations and specifications by adding conditions on the models. This is done by assuming that the former have fixed timing behavior and they can always advance either by producing an output or delaying. In this paper, we summarize the specification theory for timed systems of [9,10]. We also show how a game-based methodology can be used to decide whether a specification is consistent, *i.e.* whether it has at least one implementation. The latter reduces to deciding existence of a strategy that despite the behavior of the environment will avoid states that cannot possibly satisfy the implementation requirements. Finally, we show that the approach extends to a robust theory for timed systems.

2 Abstract Probabilistic Automata

For any finite set S, $\mathsf{Dist}(S)$ denotes the set of all discrete probability distributions over S (*i.e.* all mappings $\mu : S \to [0,1]$ with $\sum_{s \in S} \mu(s) = 1$). $C(S)$ denotes a set of *probability constraints* together with a mapping $\mathsf{Sat} : C(S) \to 2^{\mathsf{Dist}(S)}$.

2.1 Abstract Probabilistic Automata

A *probabilistic automaton* (PA) [30] is a tuple (S, A, L, AP, V, s^0), where S is a finite set of states with the initial state $s^0 \in S$, A is a finite set of actions, $L : S \times A \times \mathsf{Dist}(S) \to \{\bot, \top\}$ is a transition relation, AP is a finite set of atomic propositions and $V : S \to 2^{AP}$ is a state-labeling function.

PA were introduced in [30] as a model suitable for systems which encompass both non-deterministic and stochastic behavior. Hence they generalize both LTS (non-determinism) and Markov chains (stochasticity). The notation $L : S \times A \times \mathsf{Dist}(S) \to \{\bot, \top\}$ instead of $L \subseteq S \times A \times \mathsf{Dist}(S)$ is traditional and will be convenient below. The left part of Fig. 1 shows an example of a PA.

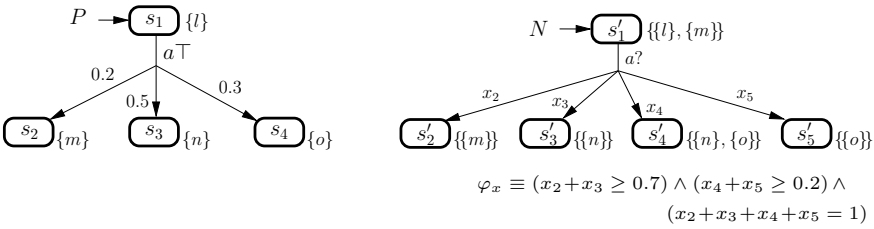


Fig. 1. Example PA P (left) and APA N with $P \models N$ (right)

As specifications of PA we use *abstract probabilistic automata* [14]. These can be seen as a common generalization of modal transition systems [16] and *constraint Markov chains* [3].

Definition 1. *An* abstract probabilistic automaton *(APA) [14] is a tuple* (S, A, L, AP, V, s^0)*, where* S *is a finite set of states,* $s^0 \in S$ *is the initial state,* A *is a finite set of actions, and* AP *is a finite set of atomic propositions.* $L : S \times A \times C(S) \to \{\bot, ?, \top\}$ *is a three-valued distribution-constraint function, and* $V : S \to 2^{2^{AP}}$ *maps each state in* S *to a set of admissible labelings.*

It is natural to think that distribution constraints should be *intervals* on transition probabilities as *e.g.* in *interval Markov chains* [19]. However, we will later see that natural constructions on APA such as conjunction or structural composition make it necessary to allow other, more expressive types of constraints.

The following notation will be convenient later: for $s, t \in S$ and $a \in A$, let $\mathsf{succ}_{s,a}(t) = \{s' \in S \mid V(s') = V(t), \exists \varphi \in C(S), \mu \in \mathsf{Sat}(\varphi) : L(s, a, \varphi) \neq \bot, \mu(s') > 0\}$ be the set of potential a-successors of s that have $V(t)$ as their valuation. Remark that when N is deterministic, we have $|\mathsf{succ}_{s,a}(v)| \leq 1$ for all s, t, a.

An APA is *deterministic* if (1) there is at most one outgoing transition for each action in all states and (2) two states with overlapping atomic propositions can never be reached with the same transition. An APA is in *single valuation normal form* (SVNF) if the valuation function V assigns at most one valuation to all states, *i.e.* $\forall s \in S, |V(s)| \leq 1$. From [14], we know that every APA can be turned into an APA in SVNF with the same set of implementations, and that this construction preserves determinism.

Note that every PA is an APA in SVNF where all constraints represent a single distribution. As a consequence, all the definitions we present for APA in the following can be directly extended to PA.

Let S and S' be non-empty sets and $\mu \in \mathsf{Dist}(S)$, $\mu' \in \mathsf{Dist}(S')$. We say that μ is *simulated* by μ' with respect to a relation $R \subseteq S \times S'$ and a *correspondence function* $\delta : S \to (S' \to [0, 1])$ if

1. for all $s \in S$ with $\mu(s) > 0$, $\delta(s)$ is a distribution on S',
2. for all $s' \in S'$, $\sum_{s \in S} \mu(s) \cdot \delta(s)(s') = \mu'(s')$, and
3. whenever $\delta(s)(s') > 0$, then $(s, s') \in R$.

We write $\mu \Subset_R^\delta \mu'$ if μ is simulated by μ' w.r.t. R and δ, $\mu \Subset_R \mu'$ if there exists δ with $\mu \Subset_R^\delta \mu'$, and $\mu \Subset^\delta \mu'$ for $\mu \Subset_{S\times S'}^\delta \mu'$.

For $\varphi \in C(S)$, $\varphi' \in C(S')$ and $R \subseteq S \times S'$, we write $\varphi \Subset_R \varphi'$ if $\forall \mu \in \mathsf{Sat}(\varphi) : \exists \mu' \in \mathsf{Sat}(\varphi') : \mu \Subset_R \mu'$.

Definition 2. *Let $N_1 = (S_1, A, L_1, AP, V_1, s_0^1)$ and $N_2 = (S_2, A, L_2, AP, V_2, s_0^2)$ be APA. A relation $R \subseteq S_1 \times S_2$ is a (weak)* modal refinement *if, for all $(s_1, s_2) \in R$, we have $V_1(s_1) \subseteq V_2(s_2)$ and*

1. *$\forall a \in A$, $\forall \varphi_2 \in C(S_2)$, if $L_2(s_2, a, \varphi_2) = \top$, then $\exists \varphi_1 \in C(S_1)$ such that $L_1(s_1, a, \varphi_1) = \top$ and $\varphi_1 \Subset_R \varphi_2$,*
2. *$\forall a \in A$, $\forall \varphi_1 \in C(S_1)$, if $L_1(s_1, a, \varphi_1) \neq \bot$, then $\exists \varphi_2 \in C(S_2)$ such that $L_2(s_2, a, \varphi_2) \neq \bot$ and $\varphi_1 \Subset_R \varphi_2$.*

We say that N_1 refines N_2 and write $N_1 \leq_{\mathsf{m}} N_2$, if there is a modal refinement relation $R \subseteq S_1 \times S_2$ with $(s_0^1, s_0^2) \in R$.

2.2 Conjunction

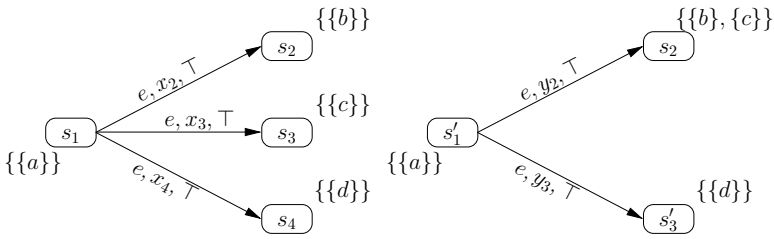
Definition 3. *Let $N = (S, A, L, AP, V, s_0)$, $N' = (S', A, L', AP, V', s_0')$ be deterministic APA which share actions and propositions. The* conjunction *of N and N' is the APA $N \wedge N' = (S \times S', A, \tilde{L}, AP, \tilde{V}, (s_0, s_0'))$, with $\tilde{V}((s, s')) = V(s) \cap V'(s')$ and $\tilde{L}$ defined as follows, for all $a \in A$ and $(s, s') \in S \times S'$:*

- *If there exists $\varphi \in C(S)$ such that $L(s, a, \varphi) = \top$ and for all $\varphi' \in C(S')$, we have $L'(s', a, \varphi') = \bot$, or if there exists $\varphi' \in C(S')$ such that $L'(s', a, \varphi') = \top$ and for all $\varphi \in C(S)$, we have $L(s, a, \varphi) = \bot$, then $\tilde{L}((s, s'), a, \mathit{false}) = \top$.*
- *Else, if either for all $\varphi \in C(S)$, we have $L(s, a, \varphi) = \bot$ or for all $\varphi' \in C(S')$, we have $L'(s', a, \varphi') = \bot$, then for all $\tilde{\varphi} \in C(S \times S')$, $\tilde{L}((s, s'), a, \tilde{\varphi}) = \bot$.*
- *Otherwise, for all $\varphi \in C(S)$ and $\varphi' \in C(S')$ such that $L(s, a, \varphi) \neq \bot$ and $L'(s', a, \varphi') \neq \bot$, define $\tilde{L}((s, s'), a, \tilde{\varphi}) = L(s, a, \varphi) \sqcup L'(s', a, \varphi')$ with $\tilde{\varphi}$ the constraint in $C(S \times S')$ such that $\tilde{\mu} \in \mathsf{Sat}(\tilde{\varphi})$ iff the distribution $t \to \sum_{t' \in S'} \tilde{\mu}((t, t')) \in \mathsf{Sat}(\varphi)$, and the distribution $t' \to \sum_{t \in S} \tilde{\mu}((t, t')) \in \mathsf{Sat}(\varphi')$.*
- *Finally, for all other $\tilde{\varphi}' \in C(S \times S')$, let $\tilde{L}((s, s'), a, \tilde{\varphi}') = \bot$.*

Observe that the conjunction of two deterministic APA is again deterministic. By the following theorem, conjunction is indeed the greatest lower bound.

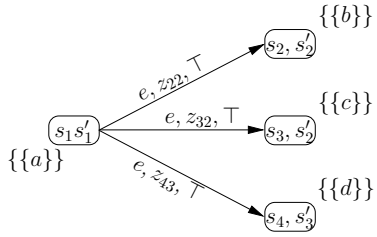
Theorem 1. *Let N_1, N_2, N_3 be deterministic APA. We have $N_1 \wedge N_2 \leq_{\mathsf{m}} N_1$ and $N_1 \wedge N_2 \leq_{\mathsf{m}} N_2$, and if $N_3 \leq_{\mathsf{m}} N_1$ and $N_3 \leq_{\mathsf{m}} N_2$, then also $N_3 \leq_{\mathsf{m}} N_1 \wedge N_2$.*

We finish this section with an example in which the conjunction of two APA with interval constraints is not an APA with interval constraints; hence interval constraints are not closed under conjunction. For the two APA N, N' in Fig. 2, which employ only interval constraints, the conjunction $N \wedge N'$ creates a constraint $0.4 \leq z_{22} + z_{32} \leq 0.8$ which is not an interval.



$\varphi_x : 0 \le x_2 \le \frac{1}{2} \wedge 0.2 \le x_3 \le 0.7 \wedge 0 \le x_4 \le \frac{1}{2}$

$\varphi_y : 0.4 \le y_2 \le 0.8 \wedge 0 \le y_3 \le 1$



$\varphi_z : 0 \le z_{22} \le 0.5 \wedge 0.2 \le z_{32} \le 0.7 \wedge 0 \le z_{43} \le 0.5 \wedge 0.4 \le z_{22} + z_{32} \le 0.8$

Fig. 2. Two APA with interval constraints (top) and their conjunction (bottom)

2.3 Structural Composition

Definition 4. *Let $N = (S, A, L, AP, V, s_0)$, $N' = (S', A, L', AP', V', s'_0)$ be APA with $AP \cap AP' = \emptyset$. The* structural composition *of N and N' is $N \| N' = (S \times S', A, \tilde{L}, AP \cup AP', \tilde{V}, (s_0, s'_0))$, with $\tilde{V}((s, s')) = \{B \cup B' \mid B \in V(s), B' \in V'(s')\}$ and $\tilde{L}$ defined as follows, for all $(s, s') \in S \times S'$ and $a \in A$:*

- *For all $\varphi \in C(S)$, $\varphi' \in C(S')$ for which $L(s, a, \varphi) \neq \bot$ and $L'(s', a, \varphi') \neq \bot$, let $\tilde{\varphi} \in C(S \times S')$ be a constraint for which $\tilde{\mu} \in \mathsf{Sat}(\tilde{\varphi})$ iff $\exists \mu \in \mathsf{Sat}(\varphi), \mu' \in \mathsf{Sat}(\varphi') : \forall t \in S, t' \in S' : \tilde{\mu}(t, t') = \mu(t)\mu(t')$. Now if $L(s, a, \varphi) = L'(s', a, \varphi') = \top$, let $\tilde{L}((s, s'), a, \tilde{\varphi}) = \top$, otherwise let $\tilde{L}((s, s'), a, \tilde{\varphi}) = ?$.*
- *If $L(s, a, \varphi) = \bot$ for all $\varphi \in C(S)$ or $L'(s', a, \varphi') = \bot$ for all $\varphi' \in C(S')$, let $\tilde{L}((s, s'), \alpha, \tilde{\varphi}) = \bot$ for all $\tilde{\varphi} \in C(S \times S')$.*

By the next theorem, structural composition respects refinement (or, in other words, refinement is a pre-congruence with respect to $\|$). This entails *independent implementability*: any composition of implementations of N_1 and N_2 is automatically an implementation of $N_1 \| N_2$.

Theorem 2. *For all APA N_1, N'_1, N_2, N'_2, $N_1 \le_{\mathsf{m}} N'_1$ and $N_2 \le_{\mathsf{m}} N'_2$ imply $N_1 \| N_2 \le_{\mathsf{m}} N'_1 \| N'_2$.*

It can be shown that structural composition of APA with interval constraints may yield APA with *polynomial* constraints, *e.g.* of the form $k_1 \le x_1 x_2 + x_2 x_3 \le$

k_2. APA with polynomial constraints are, however, closed under both structural composition and conjunction. The tool APAC [15] implements most of APA operations, for APA with polynomial constraints, and uses the Z3 solver [12] for algorithms on polynomial constraints.

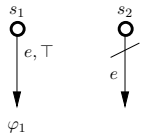
2.4 Over-Approximating Difference

We now turn to computing differences of APA. For APA N_1, N_2, we are interested in computing an APA representation of their implementation difference $[\![N_1]\!] \setminus [\![N_1]\!]$. This is based on work presented in [13].

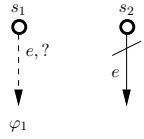
Let $N_1 = (S_1, A, L_1, AP, V_1, \{s_0^1\})$, $N_2 = (S_2, A, L_2, AP, V_2, \{s_0^2\})$ be deterministic APA in SVNF. Because N_1 and N_2 are deterministic, we know that the difference $[\![N_1]\!] \setminus [\![N_2]\!]$ is non-empty iff $N_1 \not\leq_{\mathsf{m}} N_2$. So let us assume that $N_1 \not\leq_{\mathsf{m}} N_2$, and let R be a maximal refinement relation between N_1 and N_2. Since $N_1 \not\leq_{\mathsf{m}} N_2$, we know that $(s_0^1, s_0^2) \notin R$. Given $(s_1, s_2) \in S_1 \times S_2$, we can distinguish between the following cases:

1. $(s_1, s_2) \in R$,
2. $V_1(s_1) \neq V_2(s_2)$, or
3. $(s_1, s_2) \notin R$ and $V_1(s_1) = V_2(s_2)$, and

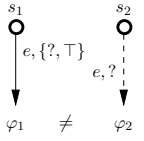
(a) there exists $e \in A$ and $\varphi_1 \in C(S_1)$ such that $L_1(s_1, e, \varphi_1) = \top$ and $\forall \varphi_2 \in C(S_2) : L_2(s_2, e, \varphi_2) = \bot$,



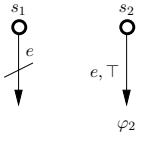
(b) there exists $e \in A$ and $\varphi_1 \in C(S_1)$ such that $L_1(s_1, e, \varphi_1) = ?$ and $\forall \varphi_2 \in C(S_2) : L_2(s_2, e, \varphi_2) = \bot$,



(c) there exists $e \in A$ and $\varphi_1 \in C(S_1)$ such that $L_1(s_1, e, \varphi_1) \geq ?$ and $\exists \varphi_2 \in C(S_2) : L_2(s_2, e, \varphi_2) = ?, \exists \mu \in \mathsf{Sat}(\varphi_1)$ such that $\forall \mu' \in Sat(\varphi_2) : \mu \not\in_R \mu'$,



(d) there exists $e \in A$ and $\varphi_2 \in C(S_2)$ such that $L_2(s_2, e, \varphi_2) = \top$ and $\forall \varphi_1 \in C(S_1) : L_1(s_1, e, \varphi_1) = \bot$,



(e) there exists $e \in A$ and $\varphi_2 \in C(S_2)$ such that $L_2(s_2, e, \varphi_2) = \top$ and $\exists \varphi_1 \in C(S_1) : L_1(s_1, e, \varphi_1) = ?$,

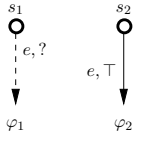
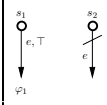
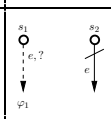
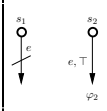
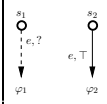
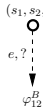
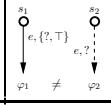
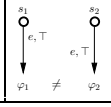
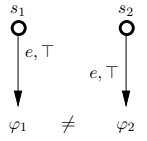


Table 1. Definition of the transition function L in $N_1 \setminus^* N_2$

$e \in$	N_1, N_2	$N_1 \setminus^* N_2$	Definition of L
$B_a(s_1, s_2)$			For all $a \neq e \in A$ and $\varphi \in C(S_1)$ such that $L_1(s_1, a, \varphi) \neq \bot$, let $L((s_1, s_2, e), a, \varphi^\bot) = L_1(s_1, a, \varphi)$. In addition, let $L((s_1, s_2, e), e, \varphi_1^\bot) = \top$. For all other $b \in A$ and $\varphi \in C(S)$, let $L((s_1, s_2, e), b, \varphi) = \bot$.
$B_b(s_1, s_2)$			
$B_d(s_1, s_2)$			For all $a \in A$ and $\varphi \in C(S_1)$ such that $L_1(s_1, a, \varphi) \neq \bot$, let $L((s_1, s_2, e), a, \varphi^\bot) = L_1(s_1, a, \varphi)$. For all other $b \in A$ and $\varphi \in C(S)$, let $L((s_1, s_2, e), b, \varphi) = \bot$.
$B_e(s_1, s_2)$			For all $a \neq e \in A$ and $\varphi \in C(S_1)$ such that $L_1(s_1, a, \varphi) \neq \bot$, let $L((s_1, s_2, e), a, \varphi^\bot) = L_1(s_1, a, \varphi)$. In addition, let $L((s_1, s_2, e), e, \varphi_{12}^B) = ?$. For all other $b \in A$ and $\varphi \in C(S)$, let $L((s_1, s_2, e), b, \varphi) = \bot$.
$B_c(s_1, s_2)$			For all $a \in A$ and $\varphi \in C(S_1)$ such that $L_1(s_1, a, \varphi) \neq \bot$ (including e and φ_1), let $L((s_1, s_2, e), a, \varphi^\bot) = L_1(s_1, a, \varphi)$. In addition, let $L((s_1, s_2, e), e, \varphi_{12}^B) = \top$. For all other $b \in A$ and $\varphi \in C(S)$, let $L((s_1, s_2, e), b, \varphi) = \bot$.
$B_f(s_1, s_2)$			

(f) there exists $e \in A$ and $\varphi_2 \in C(S_2)$ such that $L_2(s_2, e, \varphi_2) = \top$, $\exists \varphi_1 \in C(S_1) : L_1(s_1, e, \varphi_1) = \top$ and $\exists \mu \in \mathsf{Sat}(\varphi_1)$ such that $\forall \mu' \in \mathsf{Sat}(\varphi_2) : \mu \not\leqslant_R \mu'$.



Remark that due to determinism and SVNF, cases 1, 2 and 3 cannot happen at the same time. Moreover, although the cases in 3 can happen simultaneously, they cannot be triggered by the same action. We define the following sets.

Given a pair of states (s_1, s_2), let $B_a(s_1, s_2)$ be the set of actions in A such that case 3.a above holds. If there is no such action, then $B_a(s_1, s_2) = \emptyset$. Similarly, we define $B_b(s_1, s_2), B_c(s_1, s_2), B_d(s_1, s_2), B_e(s_1, s_2)$ and $B_f(s_1, s_2)$ to be the sets of actions such that case 3.b, c, d, e and 3.f holds, respectively. Given a set $X \subseteq \{a, b, c, d, e, f\}$, let $B_X(s_1, s_2) = \cup_{x \in X} B_x(s_1, s_2)$. In addition, let $B(s_1, s_2) = B_{\{a,b,c,d,e,f\}}(s_1, s_2)$.

Definition 5. *Let $N_1 = (S_1, A, L_1, AP, V_1, \{s_0^1\})$, $N_2 = (S_2, A, L_2, AP, V_2, \{s_0^2\})$ be deterministic APA in SVNF. If $N_1 \leq_{\mathsf{m}} N_2$, then $N_1 \setminus^* N_2$ is undefined; if $V_1(s_0^1) \neq V_2(s_0^2)$, we let $N_1 \setminus^* N_2 = N_1$. Otherwise, define $N_1 \setminus^* N_2 =$*

(S, A, L, AP, V, S_0), *where* $S = S_1 \times (S_2 \cup \{\bot\}) \times (A \cup \{\varepsilon\})$, $V(s_1, s_2, a) = V(s_1)$, *and* $S_0 = \{(s_0^1, s_0^2, f) \mid f \in B(s_0^1, s_0^2)\}$. L *is defined as follows:*

- *If* $s_2 = \bot$ *or* $e = \varepsilon$ *or* (s_1, s_2) *in case* 1 *or* 2, *then for all* $a \in A$ *and* $\varphi \in C(S_1)$ *such that* $L_1(s_1, a, \varphi) \neq \bot$, *let* $L((s_1, s_2, e), a, \varphi^\bot) = L_1(s_1, a, \varphi)$, *with* $\varphi^\bot$ *defined below. For all other* $b \in A$ *and* $\varphi \in C(S)$, *let* $L((s_1, s_2, e), b, \varphi) = \bot$.
- *Else, we have* (s_1, s_2) *in case* 3 *and* $B(s_1, s_2) \neq \emptyset$ *by construction. The definition of* L *is given in Table 1, with the constraints* $\varphi^\bot$ *and* φ_{12}^B *defined below.*

For $\varphi \in C(S_1)$, $\varphi^\bot \in C(S)$ *is defined as follows:* $\mu \in \mathsf{Sat}(\varphi^\bot)$ *iff* $\forall s_1 \in S_1$, $\forall s_2 \neq \bot$, $\forall b \neq \varepsilon$, $\mu(s_1, s_2, b) = 0$ *and the distribution* $s_1 \mapsto \mu(s_1, \bot, \varepsilon) \in \mathsf{Sat}(\varphi)$.

For a state $(s_1, s_2, e) \in S$ *with* $s_2 \neq \bot$, $e \neq \varepsilon$ *and two constraints* $\varphi_1 \in C(S_1)$, $\varphi_2 \in C(S_2)$ *such that* $L_1(s_1, e, \varphi_1) \neq \bot$ *and* $L_2(s_2, e, \varphi_2) \neq \bot$, *the constraint* $\varphi_{12}^B \in C(S)$ *is defined as follows:* $\mu \in \mathsf{Sat}(\varphi_{12}^B)$ *iff*

1. *for all* $(s_1', s_2', c) \in S$ *with* $\mu(s_1', s_2', c) > 0$, $c \in B(s_1', s_2') \cup \{\varepsilon\}$ *and either* $\mathsf{succ}_{s_2, e}(s_1') = \emptyset$ *and* $s_2' = \bot$, *or* $s_2' = \mathsf{succ}_{s_2, e}(s_1')$,
2. *the distribution* $s_1' \mapsto \sum_{c \in A \cup \{\varepsilon\}, s_2' \in S_2 \cup \{\bot\}} \mu(s_1', s_2', c) \in \mathsf{Sat}(\varphi_1)$, *and*
3. *either (a) there exists* $(s_1', \bot, c)$ *such that* $\mu(s_1', \bot, c) > 0$, *or (b) the distribution* $s_2' \mapsto \sum_{c \in A \cup \{\varepsilon\}, s_1' \in S_1} \mu(s_1', s_2', c) \notin \mathsf{Sat}(\varphi_2)$, *or (c) there exists* $s_1' \in S_1$, $s_2' \in S_2$ *and* $c \neq \varepsilon$ *such that* $\mu(s_1', s_2', c) > 0$.

Informally, distributions in φ_{12}^B must (1) follow the corresponding execution in N_1 and N_2 if possible, (2) satisfy φ_1 and (3) either (a) reach a state in N_1 that cannot be matched in N_2 or (b) break the constraint φ_2, or (c) report breaking the relation to at least one successor state.

The following theorem shows that the $*$-difference over-approximates the real difference.

Theorem 3. *For all deterministic APA* N_1 *and* N_2 *in SVNF such that* $N_1 \not\leq_{\mathsf{m}} N_2$, *we have* $[\![N_1]\!] \setminus [\![N_2]\!] \subseteq [\![N_1 \setminus^* N_2]\!]$.

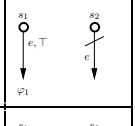
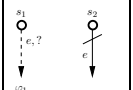
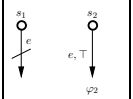
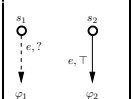
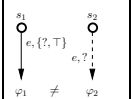
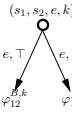
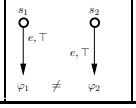
2.5 Under-Approximating Differences

Instead of the over-approximating difference $N_1 \setminus^* N_2$, we can also compute under-approximating differences. Intuitively, this is done by unfolding the APA N_1, N_2 up to some level K and then compute the difference of unfoldings:

Definition 6. *Let* $N_1 = (S_1, A, L_1, AP, V_1, \{s_0^1\})$, $N_2 = (S_2, A, L_2, AP, V_2, \{s_0^2\})$ *be deterministic APA in SVNF and* $K \in \mathbb{N}$. *If* $N_1 \leq_{\mathsf{m}} N_2$, *then* $N_1 \setminus^K N_2$ *is undefined; if* $V_1(s_0^1) \neq V_2(s_0^2)$, *we let* $N_1 \setminus^K N_2 = N_1$. *Otherwise, define* $N_1 \setminus^K N_2 = (S, A, L, AP, V, S_0^K)$, *where* $S = S_1 \times (S_2 \cup \{\bot\}) \times (A \cup \{\varepsilon\}) \times \{1, \ldots, K\}$, $V(s_1, s_2, a, k) = V(s_1)$, *and* $S_0^K = \{(s_0^1, s_0^2, f, K) \mid f \in B(s_0^1, s_0^2)\}$. L *is defined as follows:*

- *If* $s_2 = \bot$ *or* $e = \varepsilon$ *or* (s_1, s_2) *in case* 1 *or* 2, *then for all* $a \in A$ *and* $\varphi \in C(S_1)$ *such that* $L_1(s_1, a, \varphi) \neq \bot$, *let* $L((s_1, s_2, e, k), a, \varphi^\bot) = L_1(s_1, a, \varphi)$, *with* $\varphi^\bot$ *defined below. For all other* $b \in A$ *and* $\varphi \in C(S)$, *let* $L((s_1, s_2, e, k), b, \varphi) = \bot$.

Table 2. Definition of the transition function L in $N_1 \setminus^K N_2$

$e \in$	N_1, N_2	$N_1 \setminus^K N_2$	Definition of L
$B_a(s_1, s_2)$			For all $a \neq e \in A$ and $\varphi \in C(S_1)$ such that $L_1(s_1, a, \varphi) \neq \bot$, let $L((s_1, s_2, e, k), a, \varphi^\bot) = L_1(s_1, a, \varphi)$. In addition, let $L((s_1, s_2, e, k), e, \varphi_1^\bot) = \top$. For all other $b \in A$ and $\varphi \in C(S)$, let $L((s_1, s_2, e, k), b, \varphi) = \bot$.
$B_b(s_1, s_2)$			
$B_d(s_1, s_2)$			For all $a \in A$ and $\varphi \in C(S_1)$ such that $L_1(s_1, a, \varphi) \neq \bot$, let $L((s_1, s_2, e, k), a, \varphi^\bot) = L_1(s_1, a, \varphi)$. For all other $b \in A$ and $\varphi \in C(S)$, let $L((s_1, s_2, e, k), b, \varphi) = \bot$.
$B_e(s_1, s_2)$			For all $a \neq e \in A$ and $\varphi \in C(S_1)$ such that $L_1(s_1, a, \varphi) \neq \bot$, let $L((s_1, s_2, e, k), a, \varphi^\bot) = L_1(s_1, a, \varphi)$. In addition, let $L((s_1, s_2, e, k), e, \varphi_{12}^{B,k}) = ?$. For all other $b \in A$ and $\varphi \in C(S)$, let $L((s_1, s_2, e, k), b, \varphi) = \bot$.
$B_c(s_1, s_2)$			For all $a \in A$ and $\varphi \in C(S_1)$ such that $L_1(s_1, a, \varphi) \neq \bot$ (including e and φ_1), let $L((s_1, s_2, e, k), a, \varphi^\bot) = L_1(s_1, a, \varphi)$. In addition, let $L((s_1, s_2, e, k), e, \varphi_{12}^{B,k}) = \top$. For all other $b \in A$ and $\varphi \in C(S)$, let $L((s_1, s_2, e, k), b, \varphi) = \bot$.
$B_f(s_1, s_2)$			

– *Else we have (s_1, s_2) in case 3 and $B(s_1, s_2) \neq \emptyset$ by construction. The definition of L is given in Table 2, with the constraints $\varphi^\bot$ and $\varphi_{12}^{B,k}$ defined below.*

For $\varphi \in C(S_1)$, $\varphi^\bot \in C(S)$ is defined as follows: $\mu \in \mathsf{Sat}(\varphi^\bot)$ iff $\forall s_1 \in S_1$, $\forall s_2 \neq \bot$, $\forall b \neq \varepsilon$, $\forall k \neq 1, \mu(s_1, s_2, b, k) = 0$ and the distribution $s_1 \mapsto \mu(s_1, \bot, \varepsilon, 1) \in \mathsf{Sat}(\varphi)$.

For a state $(s_1, s_2, e, k) \in S$ with $s_2 \neq \bot$, $e \neq \varepsilon$ and two constraints $\varphi_1 \in C(S_1)$ and $\varphi_2 \in C(S_2)$ such that $L_1(s_1, e, \varphi_1) \neq \bot$ and $L_2(s_2, e, \varphi_2) \neq \bot$, the constraint $\varphi_{12}^{B,k} \in C(S)$ is defined as follows: $\mu \in \mathsf{Sat}(\varphi_{12}^{B,k})$ iff

1. *for all $(s_1', s_2', c, k') \in S$, if $\mu(s_1', s_2', c, k') > 0$, then $c \in B(s_1', s_2') \cup \{\varepsilon\}$ and either $\mathsf{succ}_{s_2,e}(s_1') = \emptyset$, $s_2' = \bot$ and $k' = 1$, or $s_2' = \mathsf{succ}_{s_2,e}(s_1')$,*
2. *the distribution $s_1' \mapsto \sum_{c \in A \cup \{\varepsilon\}, s_2' \in S_2 \cup \{\bot\}, k' \geq 1} \mu(s_1', s_2', c, k') \in \mathsf{Sat}(\varphi_1)$, and*
3. *either (a) there exists $(s_1', \bot, c, 1)$ such that $\mu(s_1', \bot, c, 1) > 0$, or (b) the distribution $s_2' \mapsto \sum_{c \in A \cup \{\varepsilon\}, s_1' \in S_1, k' \geq 1} \mu(s_1', s_2', c, k') \notin \mathsf{Sat}(\varphi_2)$, or (c) $k \neq 1$ and there exists $s_1' \in S_1$, $s_2' \in S_2$, $c \neq \varepsilon$ and $k' < k$ such that $\mu(s_1', s_2', c, k') > 0$.*

Theorem 4. *For all deterministic APA N_1, N_2 in SVNF such that $N_1 \not\leq_{\mathsf{m}} N_2$,*

1. *for all $K \in \mathbb{N}$, we have $N_1 \setminus^K N_2 \leq_{\mathsf{m}} N_1 \setminus^{K+1} N_2$,*
2. *for all $K \in \mathbb{N}$, $[\![N_1 \setminus^K N_2]\!] \subseteq [\![N_1]\!] \setminus [\![N_2]\!]$, and*
3. *for all PA $P \in [\![N_1]\!] \setminus [\![N_2]\!]$, there exists $K \in \mathbb{N}$ such that $P \in [\![N_1 \setminus^K N_2]\!]$.*

Note that item 3 implies that for all PA $P \in [\![N_1]\!] \setminus [\![N_2]\!]$, there is a finite specification capturing $[\![N_1]\!] \setminus [\![N_2]\!]$ "up to" P. Hence $\varinjlim [\![N_1 \setminus^K N_2]\!] = [\![N_1]\!] \setminus [\![N_2]\!]$, the direct limit.

2.6 Distances

In order to better assess how close the differences $N_1 \setminus^* N_2$ and $N_1 \setminus^K N_2$ approximate the real difference $[\![N_1]\!] \setminus [\![N_2]\!]$, we define distances on APA. These distances are based on work in [2,17,18]; see also [16].

Let $\lambda \in \mathbb{R}$ with $0 < \lambda < 1$ be a *discounting factor*.

Definition 7. *The* modal refinement distance *between the states of APA $N_1 = (S_1, A, L_1, AP, V_1, S_0^1)$, $N_2 = (S_2, A, L_2, AP, V_2, S_0^2)$ is defined to be the least fixed point to the equations*

$$d_{\mathsf{m}}(s_1, s_2) = \begin{cases} 1 & \text{if } V_1(s_1) \not\subseteq V_2(s_2), \\ \max \left\{ \begin{array}{ll} \sup\limits_{a,\varphi_1 : L_1(s_1,a,\varphi_1) \neq \bot} & \inf\limits_{\varphi_2 : L_2(s_2,a,\varphi_2) \neq \bot} D(\varphi_1, \varphi_2) \\ \sup\limits_{a,\varphi_2 : L_2(s_2,a,\varphi_2) = \top} & \inf\limits_{\varphi_1 : L_1(s_1,a,\varphi_1) = \top} D(\varphi_1, \varphi_2) \end{array} \right\} & \text{otherwise,} \end{cases}$$

where

$$D(\varphi_1, \varphi_2) = \sup_{\mu_1 \in \mathsf{Sat}(\varphi_1)} \inf_{\mu_2 \in \mathsf{Sat}(\varphi_2)} \inf_{\delta : \mu_1 \Subset^{\delta} \mu_2} \sum_{(s_1,s_2) \in S_1 \times S_2} \lambda\, \mu_1(s_1) \delta(s_1, s_2) d_{\mathsf{m}}(s_1, s_2).$$

We let $d_{\mathsf{m}}(N_1, N_2) = \max_{s_1^0 \in S_1^0} \min_{s_2^0 \in S_2^0} d_{\mathsf{m}}(s_1^0, s_2^0)$.

Note that $\sup \emptyset = 1$. The *through refinement distance* is

$$d_{\mathsf{t}}(N_1, N_2) = \sup_{P_1 \in [\![N_1]\!]} \inf_{P_2 \in [\![N_2]\!]} d_{\mathsf{m}}(P_1, P_2).$$

We need to extend this to general sets of PA; for $\mathcal{S}_1$, $\mathcal{S}_2$ sets of PA, we let $d_{\mathsf{t}}(\mathcal{S}_1, \mathcal{S}_2) = \sup_{P_1 \in \mathcal{S}_1} \inf_{P_2 \in \mathcal{S}_2} d_{\mathsf{m}}(P_1, P_2)$. The next proposition shows that our distances behave as expected, *cf.* [16].

Proposition 1. *For all APA N_1, N_2, $d_{\mathsf{t}}(N_1, N_2) \leq d_{\mathsf{m}}(N_1, N_2)$, and $N_1 \leq_{\mathsf{m}} N_2$ implies $d_{\mathsf{m}}(N_1, N_2) = 0$.*

Theorem 5. *Let N_1, N_2 be deterministic APA in SVNF such that $N_1 \not\leq_{\mathsf{m}} N_2$.*

1. *The sequence $(N_1 \setminus^K N_2)_{K \in \mathbb{N}}$ converges in the distance d_{m}, and $\lim\limits_{K \to \infty} d_{\mathsf{m}}(N_1 \setminus^* N_2, N_1 \setminus^K N_2) = 0$.*

2. *The sequence* $(\llbracket N_1 \setminus^K N_2 \rrbracket)_{K\in\mathbb{N}}$ *converges in the distance* $d_{\mathfrak{t}}$*, and* $\lim_{K\to\infty} d_{\mathfrak{t}}(\llbracket N_1 \rrbracket \setminus \llbracket N_2 \rrbracket, \llbracket N_1 \setminus^K N_2 \rrbracket) = 0$.
3. *The distance* $d_{\mathfrak{t}}(\llbracket N_1 \setminus^* N_2 \rrbracket, \llbracket N_1 \rrbracket \setminus \llbracket N_2 \rrbracket) = 0$.

Note that item 3 follows directly from items 1 and 2. It implies that even though $N_1 \setminus^* N_2$ is an over-approximation of the real difference, the two are infinitesimally close in the distance $d_{\mathfrak{t}}$. Similarly, the under-approximating differences $N_1 \setminus^K N_2$ come arbitrarily close to the real difference for sufficiently large K.

3 Real-Time Specifications

In this section we consider that $\Sigma = \Sigma^i \uplus \Sigma^o$ is a finite set of actions partitioned into inputs Σ^i and outputs Σ^o. We first define basic models for timed systems, namely TIOTS and TIOA.

A *timed I/O transition system* (TIOTS) is a tuple $(S, s^0, \Sigma, \longrightarrow)$, where S is an infinite set of states, $s_0 \in S$ is the initial state, and $\longrightarrow : S \times (\Sigma \cup \mathbb{R}_{\geq 0}) \times S$ is a transition relation. We assume that any TIOTS satisfies the following conditions:

1. Time determinism: $\forall s, s', s'' \in S. \forall d \in \mathbb{R}_{\geq 0}$, if $s \xrightarrow{d} s'$ and $s \xrightarrow{d} s''$, then $s' = s''$.
2. Time reflexivity: $\forall s \in S.\ s \xrightarrow{0} s$.
3. Time additivity: $\forall s, s'' \in S. \forall d, d' \in \mathbb{R}_{\geq 0}$, $s \xrightarrow{d+d'} s''$ iff $\exists s' \in S.\ s \xrightarrow{d} s'$ and $s' \xrightarrow{d'} s''$.

We now consider a finite set C of real-time *clocks*. A *clock valuation* u over C is a mapping $C \mapsto \mathbb{R}_{\geq 0}$. Let $d \in \mathbb{R}_{\geq 0}$, we denote $u + d$ the valuation such that $\forall x \in C.\ (u+d)(x) = u(x) + d$. Let $\lambda \subseteq C$, we denote $u[\lambda]$ the valuation agreeing with u on clocks in $C \setminus \lambda$, and assigning 0 on clocks in λ. Let $\mathcal{B}(C)$ denote all *clock constraints* φ generated by the grammar $\varphi ::= x \prec k \mid x - y \prec k \mid \varphi \wedge \varphi$, where $k \in \mathbb{Q}$, $x, y \in C$ and $\prec \in \{<, \leq, >, \geq\}$. By $\mathcal{U}(C) \subset \mathcal{B}(C)$, we denote the set of constraints restricted to upper bounds and without clock differences. We write $u \models \varphi$ if u satisfies φ. Let $Z \subseteq \mathbb{R}^C_{\geq 0}$, we write $Z \models \varphi$ if $\forall u \in Z.\, u \models \varphi$ and we denote $\llbracket \varphi \rrbracket = \{u \in \mathbb{R}^C_{\geq 0} \mid u \models \varphi\}$.

A *timed I/O automaton* is a tuple $\mathcal{A} = (L, l^0, C, E, \Sigma, I)$, where L is a finite set of *locations*, $l^0 \in L$ is the *initial location*, C is a finite set of real valued *clocks*, $E \subseteq L \times \Sigma \times \mathcal{B}(C) \times 2^C \times L$ is a set of *edges*, $I : L \mapsto \mathcal{U}(C)$ assigns an *invariant* to each location.

The semantics of a TIOA is a TIOTS $\langle\!\langle \mathcal{A} \rangle\!\rangle = (L \times \mathbb{R}^C_{\geq 0}, (l^0, \mathbf{0}), \Sigma, \longrightarrow)$, where $\mathbf{0}$ is the valuation mapping all clocks to zero, and $\longrightarrow$ is the largest transition relation generated by the following rules:

$$\frac{(l, a, \varphi, \lambda, l') \in E \quad u \models \varphi \quad u' = u[\lambda]}{(l, u) \xrightarrow{a} (l', u')} \qquad \frac{d \in \mathbb{R}_{\geq 0} \quad u + d \models I(l)}{(l, u) \xrightarrow{d} (l, u + d)}$$

Examples of TIOA are shown on Fig. 3. Edges with input actions are drawn with continuous lines, while edges with output actions are drawn with dashed lines.

3.1 Timed Specifications

A timed specification theory is introduced in [9, 10] using TIOA and TIOTS. Specifications and implementations models are defined with TIOA with additional requirements for their TIOTS semantics.

Definition 8. *A* specification $\mathcal{S}$ *is a TIOA whose semantics* $\langle\!\langle \mathcal{S} \rangle\!\rangle$ *satisfies the following conditions:*

1. *Action determinism:* $\forall s, s', s'' \in S. \forall a \in \Sigma \cup \mathbb{R}_{\geq 0}$, *if* $s \xrightarrow{a} s'$ *and* $s \xrightarrow{a} s''$, *then* $s' = s''$.
2. *Input-enabledness:* $\forall s \in S. \forall i? \in \Sigma^i. \exists s' \in S.\, s \xrightarrow{i?} s'$.

An implementation $\mathcal{I}$ *is a specification whose semantics* $\langle\!\langle \mathcal{I} \rangle\!\rangle$ *satisfies the additional conditions:*

3. *Output urgency:* $\forall s, s', s'' \in S$, *if* $\exists o! \in \Sigma^o.\, s \xrightarrow{o!} s'$ *and* $\exists d \in \mathbb{R}_{\geq 0}.\, s \xrightarrow{d} s''$, *then* $d = 0$.
4. *Independent progress:* $\forall s \in S$, *either* $\forall d \in \mathbb{R}_{\geq 0}. \exists s' \in S.\, s \xrightarrow{d} s'$, *or* $\exists d \in \mathbb{R}_{\geq 0}. \exists o! \in \Sigma^o. \exists s', s'' \in S.\, s \xrightarrow{d} s'$ *and* $s' \xrightarrow{o!} s''$.

An *alternating timed simulation* between two TIOTS $P_1 = (S_1, s_1^0, \Sigma, \longrightarrow_1)$ and $P_2 = (S_2, s_2^0, \Sigma, \longrightarrow_2)$ is a relation $R \subseteq S_1 \times S_2$ such that $\forall (s_1, s_2) \in R$,

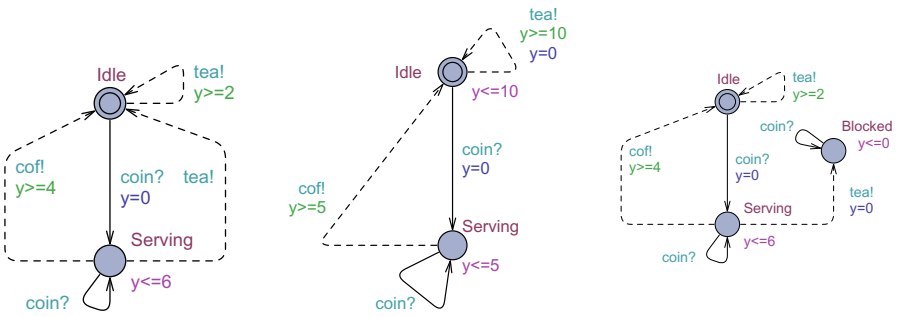
1. if $s_1 \xrightarrow{a}_1 t_1$ for some $a \in \Sigma^o \cup \mathbb{R}_{\geq 0}$, then $s_2 \xrightarrow{a}_2 t_2$ and $(t_1, t_2) \in R$.
2. if $s_2 \xrightarrow{a}_2 t_2$ for some $a \in \Sigma^i$, then $s_1 \xrightarrow{a}_1 t_1$ and $(t_1, t_2) \in R$.

We write $P_1 \leq P_2$ if there exits an alternating simulation $R \subseteq S_1 \times S_2$ with $(s_1^0, s_2^0) \in R$. For two specifications $\mathcal{S}_1$ and $\mathcal{S}_2$, we say that $\mathcal{S}_1$ *refines* $\mathcal{S}_2$, written $\mathcal{S}_1 \leq \mathcal{S}_2$, iff $\langle\!\langle \mathcal{S}_1 \rangle\!\rangle \leq \langle\!\langle \mathcal{S}_2 \rangle\!\rangle$.

An implementation $\mathcal{I}$ *satisfies* a specification $\mathcal{S}$, denoted $\mathcal{I} \models \mathcal{S}$, iff $\langle\!\langle \mathcal{I} \rangle\!\rangle \leq \langle\!\langle \mathcal{S} \rangle\!\rangle$. A specification $\mathcal{S}$ is *consistent* iff there exists an implementation $\mathcal{I}$ such that $\mathcal{I} \models \mathcal{S}$. We write $[\![\mathcal{S}]\!] = \{\mathcal{I} \mid \mathcal{I}$ is an implementation and $\mathcal{I} \models \mathcal{S}\}$ the set of all implementations of a specification.

It is shown in [10] that timed specifications also define timed games between two players: an input player that represents the environment and plays with input actions, and an output player that represents the component and plays with output actions. This timed game semantics is used to solve various decision problems, for instance consistency and refinement checking.

Consider the timed specification M of a coffee machine in Fig. 3a, and the implementation MI in Fig. 3b. We can check that this implementation satisfies the specification using a refinement game. The game proceeds as a turn-based game with two players: a spoiler starts by playing delays or output actions from the implementation, or input actions from the specification; then a replicator tries to copy the action on the other model. The spoiler wins whenever the replicator cannot mimic one of its move. Otherwise the replicator wins. For instance a strategy for the spoiler could start by delaying MI by 10 time units. Then the strategy of the replicator is two delay M by 10 time units. On the



(a) Specification M (b) Implementation MI (c) Partially inconsistent specification

Fig. 3. Specification and implementation of a coffee machine with TIOA

second move the spoiler plays action coin? on M and reaches location Serving. The replicator does the same on MI. On the third move the spoiler delays MI by 5 time units. This is allowed by the specification, so the replicator still has a winning strategy. Then the spoiler is forced to play action coff! on MI, due to the invariant in location Serving, and replicator does the same on M. The game has then returned to the initial state.

In this game a winning strategy for the replicator is necessarily infinite, as he will have to play as long as the spoiler is playing actions. However there exists symbolic techniques and algorithms for timed games [5] that restrict the game to memoryless state-based strategies on a finite number of symbolic states.

Similarly, consistency is solved using a safety game. The verifier controls the output actions of the specification, while the spoiler controls the input. The spoiler objectives is to reach an inconsistent state, that does not satisfy the independent progress condition (*i.e.* the verifier has no delay or output actions). Contrary to the refinement game, the game is concurrent: both players choose a couple delay and action at the same time, then the move that is performed is the one with the smaller delay. Consider for instance another specification of a coffee machine shown in Fig. 3c. The location Blocked is inconsistent, but the verifier can still play a strategy to avoid it (for instance by never playing action tea!). Therefore this specification is also consistent, and indeed one can check that the MI also satisfies this specification.

3.2 Robust Timed Specifications

We now introduce some perturbations in the timing constants of the models and check whether "good" properties are still satisfied. This is known as the robustness problem. Let $\varphi \in \mathcal{B}(C)$ be a guard over clocks C and let $\Delta \in \mathbb{Q}_{\geq 0}$. The *enlarged guard* $\lceil \varphi \rceil_\Delta$ is constructed according to the following rules:

- Any term $x \prec k$ of φ with $\prec \in \{<, \leq\}$ is replaced by $x \prec k + \Delta$
- Any term $x \succ k$ of φ with $\succ \in \{>, \geq\}$ is replaced by $x \succ k - \Delta$

Similarly, the *restricted guard* $\lfloor\varphi\rfloor_\Delta$ is constructed with the following rules:

- Any term $x \prec k$ of φ with $\prec \in \{<, \leq\}$ is replaced by $x \prec k - \Delta$
- Any term $x \succ k$ of φ with $\succ \in \{>, \geq\}$ is replaced by $x \succ k + \Delta$.

We lift the perturbation to implementations models. Given a jitter Δ, the perturbation means a Δ-enlargement of invariants and output edge guards, and on contrary a Δ-restriction of input edge guards:

Definition 9. *Let $\mathcal{I} = (L, l^0, C, E, \Sigma, I)$ be an implementation and $\Delta \in \mathbb{Q}_{\geq 0}$, the* Δ-perturbation *of $\mathcal{I}$ is the TIOA $\mathcal{I}_\Delta = (L \cup l^{\mathrm{u}}, l^0, C, E_\Delta, \Sigma, I_\Delta)$, where*

1. *Every edge $(l, o!, \varphi, \lambda, l') \in E$ is replaced by $(l, o!, \lceil\varphi\rceil_\Delta, \lambda, l') \in E_\Delta$.*
2. *Every edge $(l, i?, \varphi, \lambda, l') \in E$ is replaced by $(l, i?, \lfloor\varphi\rfloor_\Delta, \lambda, l') \in E_\Delta$.*
3. *$\forall l \in L.\, I_\Delta(l) = \lceil I(l)\rceil_\Delta$.*
4. *$\forall l \in L. \forall i? \in \Sigma^i$ there exists an edge $(l, i?, \varphi^{\mathrm{u}}, \emptyset, l^{\mathrm{u}}) \in E_\Delta$ with*

$$\varphi^{\mathrm{u}} = \neg\Big(\bigvee_{(l, i?, \varphi, \lambda, l') \in E} \lfloor\varphi\rfloor_\Delta\Big).$$

l^{u} is a universal location *such that, $\forall a \in \Sigma. \exists (l^{\mathrm{u}}, a, \top, \emptyset, l^{\mathrm{u}}) \in E$, where $\top$ is the clock constraints such that $[\![\top]\!] = \mathbb{R}^C_{\geq 0}$.*

An implementation $\mathcal{I}$ *robustly satisfies* a specification $\mathcal{S}$ for a given delay $\Delta \in \mathbb{Q}_{\geq 0}$, denoted $\mathcal{I} \models_\Delta \mathcal{S}$, if $\mathcal{I}_\Delta \leq \mathcal{S}$. A specification $\mathcal{S}$ is Δ-*robust consistent* iff there exists an implementation $\mathcal{I}$ such that $\mathcal{I} \models_\Delta \mathcal{S}$. We write $[\![\mathcal{S}]\!]_\Delta = \{\mathcal{I} \mid \mathcal{I}$ is an implementation and $\mathcal{I} \models_\Delta \mathcal{S}\}$ the set of all Δ-robust implementations a specification.

Refinement game is used to check robust satisfaction. Consider again the specification M and the implementation MI from 3. The Δ-perturbation of MI is presented on Fig.4. For $\Delta = 1$, we can check that $\mathsf{MI}_1 \leq \mathsf{M}$. For $\Delta = 2$ the spoiler has the following winning strategy: he plays coin? on M, then delays by 7 time units on MI_2. This cannot be mimicked by the replicator since he cannot delays more than 6 time units on the specification M. Indeed we can show then $\Delta = 1$ is the maximum value such that MI robustly satisfies M.

To solve robust consistency, the technique from [25] transforms the consistency game into a robust game. Then, the same game algorithms can be applied on this robust game. This transformation is illustrated in Fig.5. On the left, consider the specification of Fig.5a, of which we want to check the robust consistency for $\Delta = 1$. We transform the TIOA by splitting output edges, as shown on the right in Fig. 5b. In this game in location Serving, if the verifier plays its move at time $y = 5$, he must wait 1 time unit in location Serving_a and then reach location Serving_b at $y = 6$. Here the spoiler has a strategy to reach the location Bad and wins. Therefore the winning strategy for the verifier is to move to Serving_a at $y = 4$, then wait 1 time unit, and reach Serving_b at $y = 5$, where the spoiler is forced to return to location Idle. For $\Delta = 2$ this strategy fails, since location Serving_b is only reached after $y \geq 6$. This shows that the specification is 1-robust consistent.

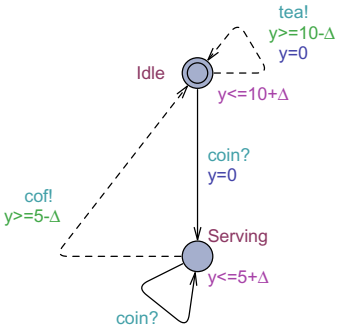
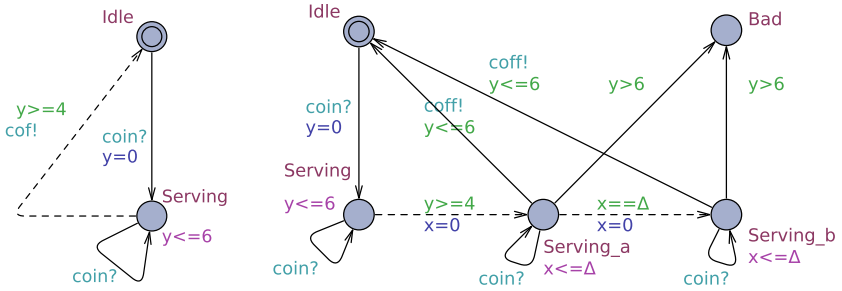


Fig. 4. Δ-perturbation of an implementation



(a) Specification (b) Robust consistency game

Fig. 5. Robust consistency game transformation for a timed specification

3.3 Conjunction

Definition 10. *The conjunction of two timed specifications* $\mathcal{S}_1 = (L_1, l_1^0, C_1, E_1, \Sigma, I_1)$, $\mathcal{S}_2 = (L_2, l_2^0, C_2, E_2, \Sigma, I_2)$ *is the TIOA* $\mathcal{S}_1 \wedge \mathcal{S}_2 = (L, l^0, C, E, \Sigma, I)$ *where* $L = L_1 \times L_2$, $l^0 = (l_1^0, l_2^0)$, $C = C_1 \uplus C_2$, $I((l_1, l_2)) = I_1(l_1) \wedge I_2(l_2)$, *and the set of edges is defined according to the following rule:*

$$((l_1, l_2), a, \varphi_1 \wedge \varphi_2, \lambda_1 \cup \lambda_2, (l_1', l_2')) \in E \text{ iff}$$
$$(l_1, a, \varphi_1, \lambda_1, l_1') \in E_1 \text{ and } (l_2, a, \varphi_2, \lambda_2, l_2') \in E_2$$

Theorem 6. *For any timed specification* $\mathcal{S}_1$, $\mathcal{S}_2$, *and* $\mathcal{T}$ *over the same alphabet:*

1. $\mathcal{S}_1 \wedge \mathcal{S}_2 \leq \mathcal{S}_2$ *and* $\mathcal{S}_1 \wedge \mathcal{S}_2 \leq \mathcal{S}_1$
2. $(\mathcal{T} \leq \mathcal{S}_1)$ *and* $(\mathcal{T} \leq \mathcal{S}_2)$ *implies* $\mathcal{T} \leq (\mathcal{S}_1 \wedge \mathcal{S}_2)$
3. $[\![\mathcal{S}_1 \wedge \mathcal{S}_2]\!] = [\![\mathcal{S}_1]\!] \cap [\![\mathcal{S}_2]\!]$
4. $[\![(\mathcal{S}_1 \wedge \mathcal{S}_2) \wedge \mathcal{T}]\!] = [\![\mathcal{S}_1 \wedge (\mathcal{S}_2 \wedge \mathcal{T})]\!]$

It turns out that this operator is robust, in the sense of precisely characterizing also the intersection of the sets of *robust* implementations. So not only conjunction is the greatest lower bound with respect to implementation semantics, but also with respect to the robust implementation semantics. More precisely:

Theorem 7. *For any timed specifications $\mathcal{S}_1$ and $\mathcal{S}_2$ over the same alphabet and $\Delta \in \mathbb{Q}_{\geq 0}$, $[\![\mathcal{S}_1 \wedge \mathcal{S}_2]\!]^\Delta = [\![\mathcal{S}_1]\!]^\Delta \cap [\![\mathcal{S}_2]\!]^\Delta$.*

3.4 Structural Composition

Two specifications $\mathcal{S}_1$, $\mathcal{S}_2$ can be composed iff $\Sigma_1^o \cap \Sigma_2^o = \emptyset$. Structural composition is obtained in by a product, where the inputs of one specification synchronize with the outputs of the other:

Definition 11. *The* structural composition *of two composable timed specifications $\mathcal{S}_1 = (L_1, l_1^0, C_1, E_1, \Sigma_1, I_1)$, $\mathcal{S}_2 = (L_2, l_2^0, C_2, E_2, \Sigma_2, I_2)$ is the TIOA $\mathcal{S}_1 \parallel \mathcal{S}_2 = (L, l^0, C, E, \Sigma, I)$, where $L = L_1 \times L_2$, $l^0 = (l_1^0, l_2^0)$, $C = C_1 \uplus C_2$, $\Sigma = \Sigma^o \cup \Sigma^i$ with $\Sigma^o = \Sigma_1^o \uplus \Sigma_2^o$ and $\Sigma^i = (\Sigma_1^i \setminus \Sigma_2^o) \cup (\Sigma_2^i \setminus \Sigma_1^o)$, $I((l_1, l_2)) = I_1(l_1) \wedge I_2(l_2)$, and for all $l_1, l_1' \in L_1$, $l_2, l_2' \in L_2$, the set of edges is defined according to the following rules:*

1. *$\forall a \in \Sigma_1 \setminus \Sigma_2$, $((l_1, l_2), a, \varphi_1, \lambda_1, (l_1', l_2)) \in E$ iff $(l_1, a, \varphi_1, \lambda_1, l_1') \in E_1$.*
2. *$\forall a \in \Sigma_2 \setminus \Sigma_1$, $((l_1, l_2), a, \varphi_2, \lambda_2, (l_1, l_2')) \in E$ iff $(l_2, a, \varphi_2, \lambda_2, l_2') \in E_2$.*
3. *$\forall a \in \Sigma_1 \cap \Sigma_2$, $((l_1, l_2), a, \varphi_1 \wedge \varphi_2, \lambda_1 \cup \lambda_2, (l_1', l_2')) \in E$ iff $(l_1, a, \varphi_1, \lambda_1, l_1') \in E_1$ and $(l_2, a, \varphi_2, \lambda_2, l_2') \in E_2$.*

Theorem 8. *For all specifications $\mathcal{S}_1$, $\mathcal{S}_2$ and $\mathcal{T}$ such that $\mathcal{S}_1 \leq \mathcal{S}_2$ and $\mathcal{S}_1$ is composable with $\mathcal{T}$, we have that $\mathcal{S}_2$ is composable with $\mathcal{T}$ and $\mathcal{S}_1 \parallel \mathcal{T} \leq \mathcal{S}_2 \parallel \mathcal{T}$.*

Theorem 8 allows the independent implementability scenario: for any consistent specification $\mathcal{S}_1$ and $\mathcal{S}_2$, such that $\mathcal{S}_1$ is composable with $\mathcal{S}_2$, $\mathcal{S}_1 \parallel \mathcal{S}_2$ is consistent. Moreover, if $\mathcal{I}_1$ is an implementation that satisfies $\mathcal{S}_1$ and $\mathcal{I}_2$ is an implementation that satisfies $\mathcal{S}_2$, then $\mathcal{I}_1 \parallel \mathcal{I}_2$ $\mathsf{Sat}\mathcal{S}_1 \parallel \mathcal{S}_2$.

Finally, Theorem 9 show that this independent implementability can be extended to robust implementability:

Theorem 9. *For any Δ-robust consistent specification $\mathcal{S}_2$ and $\mathcal{S}_2$ such that $\mathcal{S}_1$ is composable with $\mathcal{S}_2$, let $\mathcal{I}_1$ be a Δ-robust implementation of $\mathcal{S}_1$ and $\mathcal{I}_2$ be a Δ-robust implementation of $\mathcal{S}_2$, then $\mathcal{I}_2 \parallel \mathcal{I}_2$ Sat_Δ $\mathcal{S}_1 \parallel \mathcal{S}_2$.*

3.5 Parametric Robustness Evaluation

Robustness problems, like robust consistency and robust satisfaction, can be solved with traditional timed games algorithms for a given value of the perturbation Δ. When considering Δ as a parameter we want to determine the maximum value of the perturbation such that these problems are satisfied.

Let $(\mathcal{A}^\Delta, W)$ be a parametric timed game, where $\mathcal{A}$ is a TIOA parametrized by Δ and W is a safety objective. We define $\Delta_{max} = \mathsf{Sup}\{\Delta \mid (\mathcal{A}^\Delta, W)$ *has a winning strategy*$\}$. Computing Δ_{max} would in general require to solve a parametric timed game, which is undecidable [1]. Therefore, considering that the problems are monotonic, we have propose in [25] a technique to estimate the maximum value of Δ with a given precision parameter. This procedure is described in Algorithm 1.

Algorithm 1: Evaluation of the maximum robustness

Input: $(\mathcal{A}^{\Delta}, W)$: parametric robust timed game,
Δ_{init}: initial maximum value,
ε: precision
Output: Δ_{good} such that $\Delta_{max} - \Delta_{good} \leq \varepsilon$

```
1 begin
2     Δ_good ← 0
3     Δ_bad ← Δ_init
4     while Δ_bad − Δ_good > ε do
5         (Δ_good, Δ_bad) ← RefineValues((A^Δ, W), Δ_good, Δ_bad)
6     end
7     return Δ_good
8 end
```

The algorithm assumes that the game $(\mathcal{A}^0, Bad)$ is won, and on contrary that $(\mathcal{A}^{\Delta_{init}}, Bad)$ is lost. At the heart of the algorithm the procedure RefineValues solves the game $(\mathcal{A}^{\Delta}, Bad)$ for a value $\Delta \in [\Delta_{good}, \Delta_{bad}]$ and updates the variables Δ_{good} and Δ_{bad} according to the result.

Different algorithms can be used to implement RefineValues. In [25] we have compared a basic binary search approach, with a counter strategy refinement approach. In this latter we analyze the winning strategies for the spoiler in order to determine the maximum value of Δ that invalidates these strategies. In practice, this technique implemented in the tool PyEcdar [27], allows Algorithm 1 to converge faster.

Δ_{good}	Δ_{bad}
0	6
0	3
0	1.5
0.75	1.5
0.75	1.125

We show in the table on the right how to run Algorithm 1 with binary search to check the robust consistency of the specification from Fig. 5a. First, we consider the robust game automaton on Fig. 5b. Δ_{init} is set to 6, which is the maximum constant in the model, and $\varepsilon = 0.5$. In the first iteration the algorithm considers $\Delta = 3$ and solves the game, which is lost. Therefore it updates the value of Δ_{bad} to 3. On the third iteration, for $\Delta = 0.75$ the game is won. In that case Δ_{good} is updated to 0.75. The algorithm stops when $1.125 - 0.75 \leq \varepsilon$.

Finally, in Table 3 we present the results of an experiment performed on an example of timed specifications that model the administration of a university (with the coffee machine specification M, presented in Fig. 3a, an administration specification A, a researcher specification R, and the structural compositions of these specifications). The results compare the performances of Algorithm 1 when checking robust consistency using either a binary search approach (BS) or a counter strategy refinement approach (CS).

Table 3. Comparing methods to check robust consistency of timed specifications

			$\Delta_{init} = 8$		$\Delta_{init} = 6$		$\Delta_{init} = 8$		$\Delta_{init} = 6$	
	Game size		$\varepsilon = 0.1$		$\varepsilon = 0.1$		$\varepsilon = 0.01$		$\varepsilon = 0.01$	
Model	loc.	edges	CR	BS	CR	BS	CR	BS	CR	BS
M	9	21	119ms	314ms	119ms	262ms	119ms	438ms	119ms	437ms
R	11	27	188ms	303ms	188ms	299ms	188ms	419ms	188ms	523ms
A	9	22	133ms	316ms	133ms	287ms	133ms	441ms	133ms	483ms
M ‖ **A**	41	158	10.1s	10.1s	10.1s	9.6s	10.4s	17.5s	10.4s	17.6s
R ‖ **A**	48	201	14.1s	12.1s	12.5s	11s	14.1s	19.6s	12.5s	19.4s
M ‖ **R**	44	152	10s	15.5s	9.81s	15.8s	10.3s	22.9s	9.78s	29.2s
M ‖ **R** ‖ **A**	180	803	54.4s	56.3s	54.6s	112s	55s	58.8s	55.7s	216s

Acknowledgment. This survey paper presents research which we have conducted with a number of coauthors; in alphabetical order, these are Alexandre David, Benoît Delahaye, Joost-Pieter Katoen, Kim G. Larsen, Ulrik Nyman, Mikkel L. Pedersen, Falak Sher, and Andrzej Wąsowski. We acknowledge their cooperation in this work; any errors in this presentation are, however, our own.

References

1. Alur, R., Henzinger, T.A., Vardi, M.Y.: Parametric real-time reasoning. In: STOC, pp. 592–601 (1993)
2. Bauer, S.S., Fahrenberg, U., Legay, A., Thrane, C.: General quantitative specification theories with modalities. In: Hirsch, E.A., Karhumäki, J., Lepistö, A., Prilutskii, M. (eds.) CSR 2012. LNCS, vol. 7353, pp. 18–30. Springer, Heidelberg (2012)
3. Caillaud, B., Delahaye, B., Larsen, K.G., Legay, A., Pedersen, M.L., Wąsowski, A.: Compositional design methodology with constraint Markov chains. In: QEST, pp. 123–132. IEEE Computer Society (2010)
4. Canetti, R., Cheung, L., Kaynar, D.K., Liskov, M., Lynch, N.A., Pereira, O., Segala, R.: Analyzing security protocols using time-bounded task-PIOAs. Discrete Event Dynamic Systems 18(1), 111–159 (2008)
5. Cassez, F., David, A., Fleury, E., Larsen, K.G., Lime, D.: Efficient on-the-fly algorithms for the analysis of timed games. In: Abadi, M., de Alfaro, L. (eds.) CONCUR 2005. LNCS, vol. 3653, pp. 66–80. Springer, Heidelberg (2005)
6. Cattani, S., Segala, R.: Decision algorithms for probabilistic bisimulation. In: Brim, L., Jančar, P., Křetínský, M., Kučera, A. (eds.) CONCUR 2002. LNCS, vol. 2421, pp. 371–385. Springer, Heidelberg (2002)
7. Cheung, L., Lynch, N.A., Segala, R., Vaandrager, F.W.: Switched PIOA: Parallel composition via distributed scheduling. Theor. Comput. Sci. 365(1-2), 83–108 (2006)
8. Cheung, L., Stoelinga, M., Vaandrager, F.W.: A testing scenario for probabilistic processes. J. ACM 54(6) (2007)
9. David, A., Larsen, K.G., Legay, A., Nyman, U., Traonouez, L.-M., Wąsowski, A.: Real-time specifications. Int. J. Softw. Tools Techn. Transfer (2013), `http://dx.doi.org/10.1007/s10009-013-0286-x`

10. David, A., Larsen, K.G., Legay, A., Nyman, U., Wąsowski, A.: Timed I/O automata: a complete specification theory for real-time systems. In: HSCC, pp. 91–100. ACM (2010)
11. de Alfaro, L., Henzinger, T.A., Stoelinga, M.: Timed interfaces. In: Sangiovanni-Vincentelli, A.L., Sifakis, J. (eds.) EMSOFT 2002. LNCS, vol. 2491, pp. 108–122. Springer, Heidelberg (2002)
12. de Moura, L., Bjørner, N.: Z3: An efficient SMT solver. In: Ramakrishnan, C.R., Rehof, J. (eds.) TACAS 2008. LNCS, vol. 4963, pp. 337–340. Springer, Heidelberg (2008)
13. Delahaye, B., Fahrenberg, U., Larsen, K.G., Legay, A.: Refinement and difference for probabilistic automata. In: Joshi, K., Siegle, M., Stoelinga, M., D'Argenio, P.R. (eds.) QEST 2013. LNCS, vol. 8054, pp. 22–38. Springer, Heidelberg (2013)
14. Delahaye, B., Katoen, J.-P., Larsen, K.G., Legay, A., Pedersen, M.L., Sher, F., Wąsowski, A.: Abstract probabilistic automata. In: Jhala, R., Schmidt, D. (eds.) VMCAI 2011. LNCS, vol. 6538, pp. 324–339. Springer, Heidelberg (2011)
15. Delahaye, B., Larsen, K.G., Legay, A., Pedersen, M.L., Wąsowski, A.: APAC: A tool for reasoning about abstract probabilistic automata. In: QEST, pp. 151–152. IEEE Computer Society (2011)
16. Fahrenberg, U., Larsen, K.G., Legay, A., Traonouez, L.-M.: Parametric and quantitative extensions of modal transition systems. In: Bensalem, S., Lakhnech, Y., Legay, A. (eds.) FPS 2014 (Sifakis Festschrift). LNCS, vol. 8415, pp. 84–97. Springer, Heidelberg (2014)
17. Fahrenberg, U., Legay, A., Thrane, C.: The quantitative linear-time–branching-time spectrum. In: FSTTCS. LIPIcs, vol. 13, pp. 103–114. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik (2011)
18. Fahrenberg, U., Thrane, C.R., Larsen, K.G.: Distances for weighted transition systems: Games and properties. In: QAPL. Electr. Proc. Theor. Comput. Sci., vol. 57, pp. 134–147 (2011)
19. Fecher, H., Leucker, M., Wolf, V.: *Don't know* in probabilistic systems. In: Valmari, A. (ed.) SPIN 2006. LNCS, vol. 3925, pp. 71–88. Springer, Heidelberg (2006)
20. Jansen, D.N., Hermanns, H., Katoen, J.-P.: A probabilistic extension of UML statecharts. In: Damm, W., Olderog, E.-R. (eds.) FTRTFT 2002. LNCS, vol. 2469, pp. 355–374. Springer, Heidelberg (2002)
21. Jonsson, B., Larsen, K.G.: Specification and refinement of probabilistic processes. In: LICS, pp. 266–277. IEEE (1991)
22. Katoen, J.-P., Klink, D., Leucker, M., Wolf, V.: Three-valued abstraction for continuous-time Markov chains. In: Damm, W., Hermanns, H. (eds.) CAV 2007. LNCS, vol. 4590, pp. 311–324. Springer, Heidelberg (2007)
23. Kaynar, D.K., Lynch, N., Segala, R., Vaandrager, F.: Timed I/O automata: A mathematical framework for modeling and analyzing real-time systems. In: RTSS, pp. 166–177. Society Press (2003)
24. Larsen, K.G.: Modal specifications. In: Sifakis, J. (ed.) CAV 1989. LNCS, vol. 407, pp. 232–246. Springer, Heidelberg (1990)
25. Larsen, K.G., Legay, A., Traonouez, L.-M., Wąsowski, A.: Robust synthesis for real-time systems. Theor. Comput. Sci. 515, 96–122 (2014)
26. Larsen, K.G., Thomsen, B.: A modal process logic. In: LICS, pp. 203–210. IEEE Computer Society (1988)

27. Legay, A., Traonouez, L.-M.: PyEcdar: Towards open source implementation for timed systems. In: Van Hung, D., Ogawa, M. (eds.) ATVA 2013. LNCS, vol. 8172, pp. 460–463. Springer, Heidelberg (2013)
28. Parma, A., Segala, R.: Axiomatization of trace semantics for stochastic nondeterministic processes. In: QEST, pp. 294–303. IEEE (2004)
29. Segala, R.: Probability and nondeterminism in operational models of concurrency. In: Baier, C., Hermanns, H. (eds.) CONCUR 2006. LNCS, vol. 4137, pp. 64–78. Springer, Heidelberg (2006)
30. Segala, R., Lynch, N.A.: Probabilistic simulations for probabilistic processes. NJC 2(2), 250–273 (1995)