Electronic Communications of the EASST
Volume 72 (2015)

Proceedings of the
15th International Workshop on
Automated Verification of Critical Systems (AVoCS 2015)

Permissive strategies in timed automata and games[1]

Patricia Bouyer, Erwin Fang[2] and Nicolas Markey

15 pages

Guest Editors: Gudmund Grov, Andrew Ireland
ECEASST Home Page: http://www.easst.org/eceasst/ ISSN 1863-2122

[1] Partly supported by FET project Cassting (FP7-601148) and ERC project EQualIS (FP7-308087).
[2] Most of the work presented in the paper has been done while this author was a student at ENS Cachan and RWTH Aachen.

Permissive strategies in timed automata and games[‡]

Patricia Bouyer[1], **Erwin Fang**[2§] **and Nicolas Markey**[1]

[1]LSV, CNRS & ENS Cachan, France
[2]Institute of Information Security, ETH Zurich, Switzerland

Abstract: Timed automata are a convenient framework for modelling and reasoning about real-time systems. While these models are now well-understood, they do not offer a convenient way of taking timing imprecisions into account. Several solutions (e.g. parametric guard enlargement) have been proposed over the last ten years to take such imprecisions into account. In this paper, we propose a novel approach for handling robust reachability, based on *permissive strategies*. While classical strategies propose to play an action at an exact point in time, permissive strategies consider intervals of possible dates when to play the selected action. In other words, the controller specifies an interval of time delays for actions to be executed in a more flexible way. With such a permissive strategy, we associate a penalty, which is the inverse of the length of the proposed interval, and accumulates along the run. We show that in that setting, optimal strategies can be computed in polynomial time for one-clock timed automata.

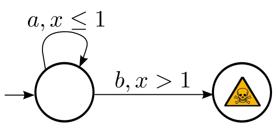
Keywords: timed automata, timed games, permissive strategies, multi-move, timed penalty games, timed robustness

1 Introduction

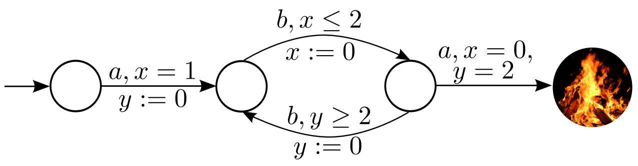
Validation of real-time embedded systems has been an active research area for many years now. Model checking real-time systems was proposed in [ACD90] as a possible approach to verify properties of such system models. Another approach to construct timed systems correctly is by synthesizing executions or winning strategies of a controller given a specification or winning condition.

There is an increasing interest in synthesis based on games within the computer science and control theory communities, since games are a suitable paradigm for modeling reactive systems that maintain a continuous interaction with the environments [FLM14]. The synthesis problem is somehow dual to verification: while in verification, one asks whether some property φ is satisfied in a model $\mathcal{M}$, i.e., $\mathcal{M} \models \varphi$, the synthesis problem considers a property and a plant or game area as input and asks whether a strategy can be computed that controls the system in order to satisfy the property. In a game-theoretic context this corresponds to the existence of a strategy for a player. In this work, we consider timed automata, as defined by Alur and Dill [AD94], and the reachability winning objective. The main objective is to synthesize winning strategies that are robust w.r.t. to timing perturbations.

[‡] Partly supported by FET project Cassting (FP7-601148) and ERC project EQualIS (FP7-308087).
[§] Most of the work presented in the paper has been done while this author was a student at ENS Cachan and RWTH Aachen.



(a) Infinitely many actions can be executed within one time unit.



(b) Under perfect conditions the fire-state is not reachable. However, if the clocks x and y do not evolve exactly at the same speed, the fire state is reachable after executing the inner loop finitely often.

Fig. 1: Two examples are shown that are valid in the timed automaton model. However, both of these abstraction do not reflect the reality.

A timed automaton is a finite automaton extended with a finite set of clocks. It is a convenient paradigm to model systems with real-time constraints and to reason about these algorithmically. Efficient model-checking tools such as HyTech [HHW97], Kronos [BDM+98] and Uppaal [LPY97] are available. Still, a drawback of timed-automata is that their semantics are idealistic: these models are assumed to have arbitrary precision for delays, and immediate transitions. This leads, among other unrealistic behaviors, to the paradox that infinitely many actions can be executed within a finite amount of time. Furthermore, timed automata also assume that time can be measured exactly. This means that a system can enforce a controller to choose punctual delays. However, these are not realistic assumptions since computers are digital and values can only be stored in variables of finite size. Figure 1 shows these undesired behaviors on two concrete instances of timed automata.

Therefore, investigating on robustness issues on timed automata is crucial, and it has been an active area of research over the last ten years. The quest is to include certain meaningful notions of robustness or tolerance with respect to timing perturbations into the timed-automata model. A prominent approach is the so-called guard enlargement, i.e., the transformation of each guard of the form $a \leq x \leq b$ into $a - \delta \leq x \leq b + \delta$, for some parameter $\delta > 0$. Safety of the resulting enlarged automaton entails *robust safety* of the original automaton, i.e., safety even in the presence of timing perturbations. Several decidability and complexity results have been obtained for this notion of robustness. Efficient algorithms are being implemented in the tool Shrinktech [San15]. Robust reachability has also been proved to be decidable [BMS12]: there, the aim is to synthesize a strategy that will be able to counteract the (parametric) timing perturbations and reach a target location. We discuss these and other related works in more detail in Section 3.

Our contribution. In this paper, we also focus on robust reachability, but using *permissive strategies*. As opposed to strategies classically used in most kind of games, permissive strategies propose several possible moves to be played from a given configuration. In the timed setting, this is implemented by having strategies proposing an interval of possible dates at which the player allows her action to be played or executed. Each interval is assigned a penalty inversely proportional to the size of the interval. These penalties are summed up along the path until the target is reached.

In this setting, our aim is to compute the most permissive strategy for reaching a target location. We prove that the problem can be solved in polynomial time for one-clock timed automata (and games), and that an almost-optimal memoryless permissive strategy exists.

2 Permissive strategies and penalty games

Timed automata. Let $\mathfrak{C}$ be a finite set of variables (named *clocks* in the sequel). A *clock valuation* over $\mathfrak{C}$ is a mapping $\kappa\colon \mathfrak{C} \to \mathbb{R}_{\geq 0}$, assigning to each clock a non-negative real value. For $t \in \mathbb{R}_{\geq 0}$, we write $\kappa + t$ for the clock valuation that results from κ by adding t time units, i.e., $(\kappa+t)(c) = \kappa(c) + t$ for all $c \in \mathfrak{C}$. For a subset $U \subseteq \mathfrak{C}$, let $\kappa[U := 0]$ be the clock valuation that results from κ by resetting all clocks in U, i.e., $\kappa[U := 0](c) = \kappa(c)$ for all $c \in \mathfrak{C} \setminus U$, and $\kappa[U := 0](c) = 0$ for all $c \in U$. The set $\mathsf{Constr}(\mathfrak{C})$ of all *convex clock constraints* over $\mathfrak{C}$ is defined as the set of conjunctions of atomic constraints of the form "$c \sim n$" for $c \in \mathfrak{C}$, $n \in \mathbb{N}$, and $\sim \in \{<, \leq, =, \geq, >\}$. We write $\mathscr{I}$ for the set of all intervals of $\mathbb{R}_{\geq 0}$.

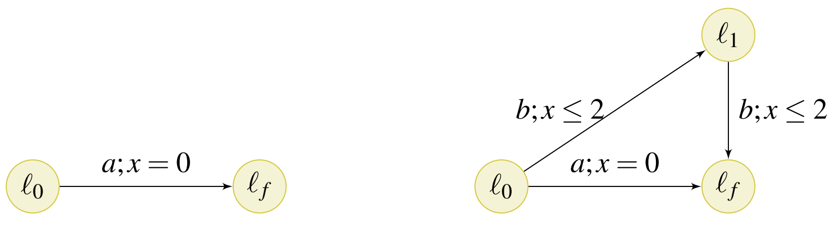
Definition 1 A *timed automaton* is a tuple $\mathscr{A} = \langle Q, \mathfrak{C}, \mathsf{Act}, E, \mathsf{Inv}\rangle$, where Q is a finite set of *locations*; $\mathfrak{C}$ is a finite set of *clocks*; Act is a finite set of *actions*; $E \subseteq Q \times \mathsf{Act} \times \mathsf{Constr}(\mathfrak{C}) \times 2^{\mathfrak{C}} \times Q$ is a transition relation; $\mathsf{Inv}\colon Q \to \mathsf{Constr}(\mathfrak{C})$ is a mapping that assigns an invariant to each location. The transition relation is required to be *deterministic*, which in our setting means that for any two transitions (q, a, g_1, r_1, q_1) and (q, a, g_2, r_2, q_2) in E with $q_1 \neq q_2$, the constraint $g_1 \wedge g_2$ is unsatisfiable,

A *configuration* of $\mathscr{A}$ is a pair $s = (q, \kappa) \in Q \times (\mathbb{R}_{\geq 0})^{\mathfrak{C}}$ such that $\kappa \models \mathsf{Inv}(q)$. A *move* is a pair $(d, a) \in \mathbb{R}_{\geq 0} \times \mathsf{Act}_i$. A move (d, a) is *enabled* in configuration (q, κ) if the following conditions hold: (1.) the invariant $\mathsf{Inv}(q)$ holds for all $\kappa + d'$ with $d' \in [0, d]$, and (2.) there is a (unique) transition $e = (q, a, g, r, q') \in E$ such that $\kappa + d \models g$ and $\kappa' = (\kappa + d)[r := 0] \models \mathsf{Inv}_i(q')$.

When those conditions are met, we write $(q, \kappa) \xrightarrow{d,a} (q', \kappa')$, which gives rise to an infinite-state transition system. Notice that we can assume that the second condition always holds, even if it means adding an extra sink location q_{sink}. We make this assumption in the sequel, as it simplifies the presentation.

A *run* from the initial configuration s_0 is an infinite sequence ρ of pairs $((d_i, a_i), s_i)_{i \geq 1}$ with $s_i \in Q \times (\mathbb{R}_{\geq 0})^{\mathfrak{C}}$ and $s_{i-1} \xrightarrow{d_i, a_i} s_{i+1}$ for all $i \geq 1$. For a finite prefix of a run (which we abusively call *finite run* in the sequel) $\pi = (\pi_j)_{1 \leq j \leq n}$, we write $\mathsf{last}(\pi)$ for the configuration s_n of the last element π_n of π. We let $|\pi| = n$. For a run π and an integer $1 \leq j \leq n$, we write $\pi_{\leq j}$ for the finite prefix of π up to the j-th transition.

Multi-moves and permissive strategies. In this paper, we consider a modified notion of moves, which we call *multi-moves*. In our timed setting, a *multi-move* is a pair (I, a) where I is a non-empty interval of $\mathbb{R}_{\geq 0}$ and a is an action. Intuitively, a multi-move (I, a) corresponds to the set of all moves (t, a) for all $t \in I$. Non-determinism is then solved by an opponent player, and the semantics of timed automata in this setting is defined as a game, as we now explain.



(a) No winning strategy for Player 1.

(b) Player 1 has a winning strategy by choosing $(b,[0,1])$ in ℓ_0 and ℓ_1.

Fig. 2: Two simple examples, where Player 1 has and has no winning strategy, for the sake of intuition

A multi-move (I,a) is enabled in configuration (q,κ) whenever for all $d \in I$, the move (d,a) is enabled in (q,κ). Any multi-move (I,a) enabled in (q,κ) gives rise to a transition $(q,\kappa) \xrightarrow{I,a} (q,\kappa,I,a)$; the latter configuration is an intermediary configuration, from which the opponent can select some $d \in I$ and activate the actual transition $(q,\kappa,I,a) \xrightarrow{d,a} (q',\kappa')$ where (q',κ') is the unique configuration such that $(q,\kappa) \xrightarrow{d,a} (q',\kappa')$. In this setting, a *play* from s_0 is an infinite sequence π of triples $((I_i,a_i),d_i,s_i)_{i\geq 1}$ such that $s_{i-1} \xrightarrow{I_i,a_i} (s_{i-1},I_i,a_i) \xrightarrow{d_i} s_i$ for all $i \geq 1$. A finite play is a finite prefix of a play, in the same way as finite runs. In particular, the last configuration $\mathsf{last}(\pi)$ is $s_{|\pi|}$.

A *permissive strategy* is a mapping σ that associates with each finite play π from s_0 a multi-move $\sigma(\pi) = (I,a)$ enabled in $\mathsf{last}(\pi)$. A finite play $\pi = (\pi_j)_{1\leq j\leq n}$, with $\pi_j = ((I_j,a_j),d_j,s_j)$ for all $1 \leq j \leq n$, is compatible with a permissive strategy σ if $\sigma(\pi_{\leq j}) = (I_j,a_j)$ for all $1 \leq j \leq n$. An (infinite) play π from s_0 is compatible with σ whenever all its finite prefixes are compatible with σ. Such a play is then called an *outcome* of σ from s_0. In this paper, we consider reachability objectives: given a target location g, a permissive strategy σ is said winning from s_0 whenever all its outcomes eventually visit location g.

Penalty of a permissive strategy. In the setting of timed robustness, our aim is to compute highly permissive strategies. A naive approach for comparing strategies is to compare the sizes of the intervals proposed by the strategies. This order would obviously not be total, and would not give rise to a notion of maximally permissive strategies. We prefer a semantic criterion, based on the quantitative measure of permissiveness.

We define the *penalty* of a multi-move (I,a) as follows:

$$\mathsf{penalty}(I,a) = \begin{cases} \frac{1}{|I|} & \text{if } I \text{ is not punctual, i.e., if } |I| > 0, \\ +\infty & \text{otherwise.} \end{cases}$$

With this definition, the larger the interval, the smaller the penalty. Of course, various other penalty functions could be considered. We elaborate on this point in Section 4.4.

In order to define the penalty of a permissive strategy, we extend the notion of penalty along

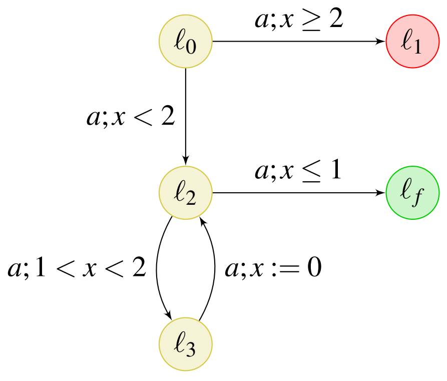


Fig. 3: Example of a timed automaton (transitions to the sink location are omitted for the sake of readability)

finite plays: given a permissive strategy σ and a finite play π, we define

$$\mathsf{penalty}_\pi(\sigma) = \sum_{j=0}^{|\pi|-1} \mathsf{penalty}\big(\sigma(\pi_{\leq j})\big).$$

(Notice that this definition does not need π to be an outcome of σ, even though it will be the case in the sequel). Again, other ways of accumulating penalties along a play could be considered.

Finally, we define the penalty of a permissive strategy. In order to have only finite paths (and finite penalty), we only consider *winning* permissive strategies, and consider the prefixes of the plays until their first visit to the target location. For a winning permissive strategy σ from initial configuration s_0, we define

$$\mathsf{penalty}_{s_0,g}(\sigma) = \sup_{\pi \in \mathsf{Out}_f(s_0,g,\sigma)} \mathsf{penalty}_\pi(\sigma)$$

where $\mathsf{Out}_f(s_0, g, \sigma)$ is the set of finite outcomes of σ from s_0 and ending at their first visit to g. The penalty of non-winning strategies is $+\infty$. The problem we tackle in this paper is the following:

Definition 2 (Computing the most permissive strategy - the decision problem) Given a timed automaton $\mathscr{A}$, a configuration s_0 and a target location g, and a threshold $p \in \mathbb{Q}$, the *most-permissive strategy problem* asks whether there exists a winning permissive strategy σ in $\mathscr{A}$ such that $\mathsf{penalty}_{s_0,g}(\sigma) \leq p$.

Example 3 *Figure 3 displays an example of a timed automaton with target location ℓ_f. Obviously, the target location ℓ_f is reachable, and can even be reached with a penalty of* 4 *(starting from $(\ell_0, x \mapsto 0)$); a corresponding strategy is to propose delay interval $[0, 1/2]$ in $(\ell_0, x \mapsto 0)$, and then $[0, (1-\kappa(x))/2]$ from (ℓ_2, κ). One easily sees that the penalty of this strategy is* 4 *(which is reached when Player* 2 *selects delay $1/2$ in ℓ_0). As we explain after Theorem 9, better strategies exist for this example.*

3 Related work

Robustness. Several previous works have proposed notions on defining robustness in timed automata. One of the first attempts was presented in [GHJ97], where a topological definition was introduced. The idea of this "tube semantics" is to accept a run if, and only if, all "neighbouring runs" are also accepted. The aim was to find a procedure for deciding language inclusion in this setting. However, this was shown to be undecidable later in [HR00].

Guard enlargement was then proposed by Puri [Pur98]. This semantics aims at over-approximating the behaviors of implementations of timed automata over (simplified) hardware [DDR04]. Notice that makes model-checking algorithms consider more runs, contrary to the tube semantics. Hence this is mainly aimed at reasoning about *robust safety* which is proven to be decidable in [Pur98, DDMR04]. Guard shrinking was then introduced in [SBM11]: the aim of shrinking is to counteract the enlargement that the model will be subject to when being implemented. Hence, the shrunk model is a good candidate to implement, provided that it preserves roughly the same behaviors as the original automaton. This was proven decidable in [SBM11]. Guard enlargement was also considered for reachability objectives [BMS12]. In this case, the aim is to reach a target location despite possible timing perturbations. A natural approach is to see this as a game, where one player tries to reach the target while the opponent introduces timing perturbations. This approach is also decidable. Based on this approach, a stochastic approach to the robustness of timed systems was proposed in [ORS14].

Our approach here shares similarities with that of [BMS12]: in both approaches, the aim is to end up with a strategy to reach a target without choosing the exact date at which transitions are taken. There are several important differences however: in particular, in our approach we add up the penalties along the runs, so that we favor shorter runs. We believe that having shorter strategies is a sensible choice in a setting where the imprecisions may accumulate when the run becomes longer. Also, guard enlargement considers the same enlargement for all the transitions, while we allow different lengths for the intervals.

Permissive strategies. While permissive strategies are a key notion in supervisory control [RW89, ELTV14], they have not been widely considered in reactive synthesis, with the exception of [BJW02, BKK11]. In those cases however, permissiveness is measured in terms of the set of behaviours allowed by the strategy. Hence maximally-permissive strategies need not exist, depending on the type of winning objectives. Our quantitative measure of permissiveness originates from [BDMR09, BMOU11], where the notion of penalty of multi-strategies is studied for discrete-time systems. This work was recently extended to Markov Decision Processes [DFK$^+$14].

4 Computing optimal permissive strategies

In this section, we study some properties of the most-permissive-strategy problem, and prove that it is decidable for one-clock timed autmata: we define a sequence of functions that we prove converges to the least penalty that can be achieve for reaching g. We then show that for one-clock timed automata, the computation is effective and that it terminates in a finite number of steps.

4.1 Least penalty for winning in i steps

Let $\mathscr{A}$ be a timed automaton, and g be the goal location. W.l.o.g., we assume that all the configurations of $\mathscr{A}$, except configurations involving q_{sink}, are winning for the objective of reaching location g. Given $a, b \in \mathbb{R}_{\geq 0}$, we write $\langle a, b \rangle$, with $\langle \in \{[, (\}$ and $\rangle \in \{],)\}$, for the interval between a and b which is either (half-)open or (half-)closed. For a clock valuation κ and a convex clock constraint φ, we define

$$\mathscr{D}(\kappa, \varphi) = \{I \in \mathscr{I} \setminus \{\emptyset\} \mid \forall t \in I.\ \kappa + t \models \varphi\}.$$

Then $\mathscr{D}(\kappa, \mathsf{Inv}(q))$ contains the set of intervals of delays that can be elapsed from (q, κ). We now define a sequence of functions $(\mathscr{P}_i)_{i \in \mathbb{N}}$ inductively as follows: for location g, we let $\mathscr{P}_i(g, \kappa) = 0$ for all $i \in \mathbb{N}$ and all valuation κ. For any location $q \neq g$, and for any valuation κ, we let

$$\mathscr{P}_0(q, \kappa) = +\infty$$
$$\mathscr{P}_{i+1}(q, \kappa) = \min_{a \in \mathsf{Act}} \inf_{I \in \mathscr{D}(\kappa, \mathsf{Inv}(q))} \left(\mathsf{penalty}(I, a) + \sup_{d \in I} \mathscr{P}_i(\mathsf{succ}(q, \kappa, d, a)) \right)$$

where $\mathsf{succ}(q, \kappa, d, a)$ is the configuration (q', κ') such that $(q, \kappa) \xrightarrow{d,a} (q', \kappa')$. We take the usual convention that the infimum over the empty set is $+\infty$.

Then, we let $\mathscr{P}(q, \kappa) = \lim_{i \to +\infty} \mathscr{P}_i(q, \kappa)$. Notice that this limit exists, as a consequence of the following lemma:

Lemma 4 *For any $n \in \mathbb{N}$, for any configuration (q, κ), the mapping $t \mapsto \mathscr{P}_n(q, \kappa + t)$ is non-decreasing and continuous, while the mapping $i \mapsto \mathscr{P}_i(q, \kappa)$ is non-increasing.*

Next we prove the correspondence between $\mathscr{P}_i$ and the optimal penalty of winning permissive strategies from a given configuration:

Lemma 5 *For any integer i and for any $\varepsilon > 0$, there exists a winning permissive strategy σ such that for any winning configuration s,*

$$\mathsf{penalty}_{s,g}(\sigma) \leq \mathscr{P}_i(s) + \varepsilon.$$

Proof. We prove the result by induction on i, the case where $i = 0$ being trivial. Assume that the result holds for some i. Pick $\varepsilon > 0$. Applying the induction hypothesis, we pick a winning permissive strategy σ such that

$$\mathsf{penalty}_{s,g}(\sigma) \leq \mathscr{P}_i(s) + \frac{\varepsilon}{2}$$

from any winning configuration s.

Pick a configuration $s = (q, \kappa)$. By definition of $\mathscr{P}_{i+1}$, there exists an action a_s and an interval I_s such that

$$\mathscr{P}_{i+1}(q, \kappa) \leq \mathsf{penalty}(I_s, a_s) + \sup_{d \in I_s} \mathscr{P}_i(\mathsf{succ}(q, \kappa, d, a_s)) \leq \mathscr{P}_{i+1}(q, \kappa) + \frac{\varepsilon}{2}.$$

We then define a new strategy σ' as follows:

$$\begin{aligned} \sigma'(s) &= (I_s, a_s) \\ \sigma'(s \cdot \rho) &= \sigma(\rho) \qquad \text{for any non-empty path } \rho \end{aligned}$$

By construction, this permissive strategy satisfies the expected inequality. □

Lemma 6 *For any winning configuration s, and for any permissive strategy σ that is winning from s, it holds*

$$\mathscr{P}(s) \leq \mathsf{penalty}_{s,g}(\sigma).$$

Proof. The proof is by induction on the number of steps needed by σ to reach g. More precisely, we prove that for any integer k, for any winning configuration s, and for any permissive strategy all of whose outcomes from s reach g within at most k steps, it holds

$$\mathscr{P}_k(s) \leq \mathsf{penalty}_{s,g}(\sigma).$$

The result follows from Lemma 4.

The case $k = 0$ holds trivially, since either $s = (g, \kappa)$ for some κ and $\mathscr{P}(s) = 0$, or there is no permissive strategy that is winning in zero steps. Assume that the result holds for some integer k, and consider a permissive strategy that is winning from $s = (q, \kappa)$ in $k+1$ steps. Let $(I, a) = \sigma(s)$. Then from any configuration $\mathsf{succ}(q, \kappa, d, a)$, the strategy σ' defined by $\sigma'(\rho) = \sigma(s \cdot \rho)$ is winning in at most k steps. It follows that $\mathscr{P}_k(\mathsf{succ}(q, \kappa, d, a)) \leq \mathsf{penalty}_{\mathsf{succ}(q,\kappa,d,a),g}(\sigma')$. Then

$$\begin{aligned} \mathsf{penalty}_{s,g}(\sigma) &= \sup_{\pi \in \mathsf{Out}_f(s,g,\sigma)} \sum_{j=0}^{|\pi|-1} \mathsf{penalty}\big(\sigma(\pi_{\leq j})\big) \\ &= \mathsf{penalty}(I, a) + \sup_{d \in I} \mathsf{penalty}_{\mathsf{succ}(q,\kappa,d,a),g}(\sigma') \end{aligned}$$

Hence $\mathsf{penalty}_{s,g}(\sigma) \geq \mathsf{penalty}(I, a) + \sup_{d \in I} \mathscr{P}_k(\mathsf{succ}(q, \kappa, d, a)) \geq \mathscr{P}_{k+1}(q, \kappa)$, as required. □

4.2 Memoryless permissive strategies for one-clock automata

Despite these good properties, the sequence $\mathscr{P}_k(q, \kappa)$ does not provide us with an algorithm for computing (or even approximating up to some positive ε) the optimal penalty from a given configuration. This is for two reasons: first, $\mathscr{P}_k(q, \kappa)$ only gives an over-approximation of $\mathscr{P}(q, \kappa)$, and we have no information about how close this approximation is from the exact value. But more importantly, computing $\mathscr{P}_{k+1}(q, \kappa)$ requires computing $\mathscr{P}_k(\mathsf{succ}(q, \kappa, d, a))$ for infinitely many moves (d, a). Hence the results of the previous section are by no means effective.

In this section, we prove that for one-clock timed automata, the sequence can be computed, and that the computation terminates in finitely many steps. The proof has several stages: we first prove that any winning multi-strategy can be made to use any resetting transition at most once, without increasing its penalty. Then, we prove that any location will be visited at most once between any two resetting transition. This bounds the number of steps after which the sequence $(\mathscr{P}_k)_k$ is constant.

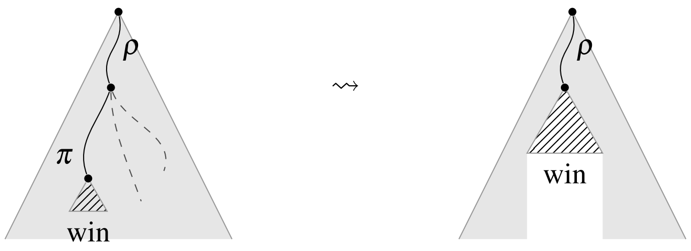


Fig. 4: Construction of $\sigma_{E'}$ for the proof of Lemma 7

4.2.1 Taking reset transitions at most once.

In this section, we prove that optimal permissive strategies can be made to visit any resetting transition at most once, along any outcome:

Lemma 7 *Let $\mathscr{E}$ be the set of resetting transitions of a game on a timed automaton $\mathscr{G}$ and let σ be a winning permissive strategy from some configuration s. We can build a winning permissive strategy σ' such that* $\mathsf{penalty}_{s,g}(\sigma') \leq \mathsf{penalty}_{s,g}(\sigma)$ *and any transition in $\mathscr{E}$ appears at most once along any finite outcome of* $\mathsf{Out}_{fin}(s,g,\sigma')$.

Proof. The proof is by induction: for a subset $E \subseteq \mathscr{E}$, we define our induction hypothesis as follows:

$$\begin{aligned}\exists \sigma_E \text{ s.t. } \forall \pi \in \mathsf{Out}_f(s,g,\sigma_E).\ &\text{any edge } e \in E \text{ is taken at most once along } \pi \\ &\text{and } \ \sigma_E \text{ is winning, and } \mathsf{penalty}_{s,g}(\sigma_E) \leq \mathsf{penalty}_{s,g}(\sigma). \qquad (\mathsf{IH}_E)\end{aligned}$$

Then σ satisfies $(\mathsf{IH}_\varnothing)$. Now assume that we have a strategy σ_E satisfying (IH_E) for some E. We pick an edge $e \in \mathscr{E} \setminus E$, and, writing $E' = E \cup \{e\}$, we build a strategy $\sigma_{E'}$ satisfying $(\mathsf{IH}_{E'})$.

For this, we first remark that because σ_E is winning (for a reachability objective), the edge e is visited finitely many times along any outcome in $\mathsf{Out}_f(s,g,\sigma_E)$. In other terms, for any finite outcome ρ ending after an occurrence of edge e, we can select a path π such that $\rho \cdot \pi$ is an outcome of σ_E, ending after an occurrence of e, and such that there is no occurrence of e in the subtree generated by σ_E after $\rho \cdot \pi$. We write $f_{\sigma_E}(\rho)$ for the path $\rho \cdot \pi$ constructed above.

Now, we build the strategy $\sigma_{E'}$. We arbitrarily pick a finite path ρ. If ρ does not visit edge e, then we let $\sigma_{E'}(\rho) = \sigma_E(\rho)$. if ρ visits edge e once, we write $\rho = \rho_1 \cdot \rho_2$, where edge e is the last transition in ρ_1, and define $\sigma_{E'}(\rho) = \sigma_E(f_{\sigma_E}(\rho_1) \cdot \rho_2)$. Finally, the value of $\sigma_{E'}$ over paths that visit e more than once is irrelevant.

It remains to prove that $\sigma_{E'}$ satisfies both conditions of (IH_E). First, pick a maximal outcome ρ in $\mathsf{Out}_f(s,g,\sigma_{E'})$. If ρ does not visit edge e, then it is also an outcome of σ_E, hence it visits any edge in E at most once, and the first property follows. If ρ visits e at least once, then one easily proves that writing $\rho = \rho_1 \cdot \rho_2$ where ρ_1 ends after the first visit to edge e, it holds that $f_{\sigma_E}(\rho_1) \cdot \rho_2$ is an outcome of σ_E. By construction, ρ_2 never visits edge e, and ρ_1 is a prefix of $f_{\sigma_E}(\rho_1)$. It follows that the edges in E' are visited at most once along ρ.

We use similar arguments for proving that the penalty of $\sigma_{E'}$ is less than or equal to that of σ_E. In order to prove this, we prove that for any outcome $\rho' \in \mathsf{Out}_f(s,g,\sigma_{E'})$, there is an outcome $\rho \in \mathsf{Out}_f(s,g,\sigma_E)$ such that the penalty of $\sigma_{E'}$ along ρ' is higher than that of σ_E along ρ. In case ρ never visits edge e, then letting $\rho' = \rho$, and noticing that $\sigma_{E'}$ plays as σ_E all along ρ, we get the result. If $\rho = \rho_1 \cdot \rho_2$, with ρ_1 ending after visiting edge e, then ρ_1 is an outcome of both strategies, and both strategies play the same moves along this outcome, so that they have the same penalties; similarly, ρ_2 is an outcome of $\sigma_{E'}$ after ρ_1 and an outcome of σ_E after $f_{\sigma_E}(\rho_1)$, and both strategies plays the same moves along those paths. Hence the penalty of σ_E is higher than that of $\sigma_{E'}$, which completes the proof. □

The inductive step of the constructive proof of Lemma 7 is visualized in Figure 4. The striped area shows the sub-game-tree after a resetting transition has been taken for the last time in the play.

4.2.2 No cycles between reset transitions.

We use similar arguments for proving that any location of a timed automaton $\mathcal{G}$ is visited at most once between any two consecutive resets of the clock:

Lemma 8 *Let σ be a winning permissive strategy from some configuration s. We can build a winning permissive strategy σ' such that $\mathsf{penalty}_{s,g}(\sigma') \leq \mathsf{penalty}_{s,g}(\sigma)$ and for any finite outcome π of $\mathsf{Out}_{fin}(s,g,\sigma')$,*

- *if π involves no resetting transitions, any location of $\mathcal{G}$ appears at most once along π;*
- *otherwise, along π, any location of $\mathcal{G}$ appears at most once between any two consecutive resets, before the first reset, and after the last reset.*

Lemma 8 intuitively tells us that at most one location visit for each location is sufficient if there is no resetting transition along a winning play. In general it might be necessary to visit one location several times. However, the number of visits can be bounded. Figure 5 shows such an example: location ℓ_1 has to be visited three times before ℓ_1 is reachable.

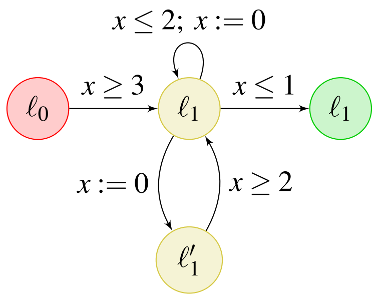


Fig. 5: An automaton where several visits to ℓ_1 are needed

4.2.3 Computation of $\mathscr{P}_i(q,\kappa)$.

The arguments above entail that the sequence $\mathscr{P}_i$ converges in finitely many steps. It remains to explain how we compute these functions. We write C for the set of constants appearing in the clock constraints of the automaton. Computing $\mathscr{P}_1(q,\kappa)$ is easy, as it suffices to find the action a

with the largest time interval I for which $\mathsf{succ}(q,\kappa,d,a) = (g,\kappa')$ for any $d \in I$. One easily notices that the lower bound of the largest I is either 0 or of the form $c - \kappa(x)$, for some constant $c \in C$. Similarly, the upper-bound is of the form $c' - \kappa(x)$ or $+\infty$. It follows that $\mathscr{P}_1(q,\kappa)$ is made of finitely many pieces, on which it is either a constant, or of the form $\frac{1}{d-\kappa(x)}$.

We prove by induction that $\mathscr{P}_i(q,\kappa)$ is always piecewise of the form $b_i^n + \frac{c_i^n}{d_i^n - \kappa(x)}$ (or $+\infty$), with finitely many pieces $\langle e_i^n, f_i^n \rangle$, with rational constants when $n \leq 1$, and algebraic constants for larger n.

As we just showed, this is the case of $\mathscr{P}_0$ and $\mathscr{P}_1$. Suppose this is the case at step n. Then

$$\mathscr{P}_{n+1}(q,\kappa) = \min_{a \in \mathsf{Act}} \inf_{\substack{I \in \mathscr{D}(\kappa, \mathsf{Inv}(q)) \\ I = \langle e,f \rangle}} \left(\mathsf{penalty}(\langle e,f \rangle, a) + \sup_{d \in I} \mathscr{P}_n(\mathsf{succ}(q,\kappa,d,a)) \right)$$

From (q,κ), several transitions may be possible, with guards of the form $x \in \langle \alpha_j, \beta_j \rangle$, leading to configurations (q_j, κ_j), with $\kappa_j(x) \in \{0, \kappa(x)\}$ depending on whether the clock is reset along the corresponding transition.

Following Lemma 4, we have that $\sup_{d \in \langle e,f \rangle} \mathscr{P}_i(\mathsf{succ}(q,\kappa,d,a))$ can only be achieved by taking the transition as late as possible, hence when d tends to f (the upper bound of I) or when $\kappa(x) + d$ tends to some constant in C (which corresponds to taking a transition as late as possible while it is available). The same argument entails that e (the lower bound of I) can be chosen as a constant in C: since the worst case is when the opponent plays as late as possible, e can be taken as low as possible as long as it does not enable a new transition.

In the end, there are only finitely many values to try for the action to play and the lower bound e of I (satisfying $e = 0$ or $\kappa(x) + e \in C$). For those choices, $f \mapsto \sup_{d \in \langle e,f \rangle} \mathscr{P}_n(\mathsf{succ}(q,\kappa,d,a))$ is easily computed as a function of f, following the remark above. It is piecewise of the form $b_i^n + \frac{c_i^n}{d_i^n - \kappa(x)}$ (but it need not be continuous at positions where $\kappa(x) + f \in C$). The function (of f) to optimize is then of the form

$$\frac{1}{f - (c - \kappa(x))} + b_i^n + \frac{c_i^n}{d_i^n - (\kappa(x) + f)}$$

This function is to be optimized over an interval with bounds of the form $\langle e_i^n, f_i^n \rangle$. The extremal values can be obtained at the bounds of the interval, or at the root of a polynomial of degree 2 obtained by computing the derivative of the above function.

Finally we obtain the following:

Theorem 9 *The optimal penalty (and a corresponding almost-optimal strategy) for reaching a target location in a timed automaton can be computed in polynomial time.*

Example 10 *We come back to our Example 3, and compute the optimal penalty for reaching the target. We initialize the computation by letting $\mathscr{P}_0(\ell,\kappa) = +\infty$ for all $\ell \neq \ell_f$; we also let $\mathscr{P}_i(\ell_f,\kappa) = 0$ for all i.*

Then only configurations (ℓ_2,κ) where $\kappa(x) \leq 1$ are winning in one step. For those configurations, $\mathscr{P}_1(\ell_2,\kappa) = \frac{1}{1-\kappa(x)}$ (hence it is $+\infty$ when $\kappa(x) = 1$). The other configurations have value $+\infty$.

At step 2, any configuration (ℓ_3,κ) is winning, since the clock is reset when going to ℓ_2. We have $\mathscr{P}_2(\ell_3,\kappa) = \mathscr{P}_1(\ell_2, x \leftarrow 0) = 1$. Configuration (ℓ_0,κ) with $\kappa(x) \leq 1$ are also winning *in two steps. The optimal penalty in two steps is computed as follows:*

$$\begin{aligned}\mathscr{P}_2(\ell_0,\kappa(x)) &= \inf_{0\le e\le f<2-\kappa(x)}\left(\frac{1}{f-e}+\sup_{e\le d\le f}\mathscr{P}_1(\ell_2,\kappa+d)\right)\\ &= \inf_{0\le f<2-\kappa(x)}\left(\frac{1}{f}+\mathscr{P}_1(\ell_2,\kappa+f)\right)\\ &= \inf_{0\le f\le 1-\kappa(x)}\left(\frac{1}{f}+\frac{1}{1-(\kappa(x)+f)}\right)\end{aligned}$$

One easily obtains that the infimum is reached when $f=\frac{1-\kappa(x)}{2}$, with $\mathscr{P}_2(\ell_0,\kappa)=\frac{4}{1-\kappa(x)}$.

The only new transition to consider at step 3 is the transition from ℓ_2 to ℓ_3. The penalty in ℓ_2 is computed as follows:

$$\mathscr{P}_3(\ell_2,\kappa)=\inf_{0\le e\le f<2-\kappa(x)}\left(\frac{1}{f-e}+\sup_{e\le d\le f}\mathscr{P}_2(\mathsf{succ}(\ell_2,\kappa,d,a))\right)$$

The successor of (ℓ_2,κ) may be either ℓ_f (with penalty 0) or ℓ_3 (with penalty 1); the latter will be chosen by the opponent as soon as $\kappa(x)+f>1$. Hence

$$\begin{aligned}\mathscr{P}_3(\ell_2,\kappa) &= \inf_{0\le e\le f<2-\kappa(x)}\left(\frac{1}{f-e}+\mathbb{1}_{(1-\kappa(x);+\infty)}(f)\right)\\ &= \inf_{0\le f<2-\kappa(x)}\left(\frac{1}{f}+\mathbb{1}_{(1-\kappa(x);+\infty)}(f)\right)\end{aligned}$$

(We denote $\mathbb{1}_I(f)$ for some interval I and value f as the function outputting 1 if $f\in I$ and 0 otherwise.) We optimize this function by considering two cases: for $f\le 1-\kappa(x)$, the penalty is $1/f$, which is minimized when $f=1-\kappa(x)$ with value $1/(1-\kappa(x))$; for $1-\kappa(x)<f<2-\kappa(x)$, the penalty is $1/f+1$, which is optimized when f tends to $2-\kappa(x)$ with value $1/(2-\kappa(x))+1$. In the end, when $\kappa(x)\le x_0=(3-\sqrt{5})/2\simeq 0.38$, the optimal multi-strategy is to play interval $[0,1-\kappa(x)]$, with penalty $1/(1-\kappa(x))$; when $\kappa(x)\ge x_0$, the optimal multi-strategy is to play interval $[0,2-\kappa(x))$, with penalty $1+1/(2-\kappa(x))$. Notice that the optimal penalty is a continuous function of $\kappa(x)$. Also notice that this gives the optimal penalty for winning from ℓ_2.

We now compute $\mathscr{P}_4$. Following Lemma 7, there is no hope of improving the penalty from location ℓ_3, so that only ℓ_0 has to be considered. We have:

$$\begin{aligned}\mathscr{P}_4(\ell_0,\kappa(x)) &= \inf_{0\le e\le f<2-\kappa(x)}\left(\frac{1}{f-e}+\sup_{e\le d\le f}\mathscr{P}_1(\ell_2,\kappa+d)\right)\\ &= \inf_{0\le f<2-\kappa(x)}\left(\frac{1}{f}+\mathscr{P}_3(\ell_2,\kappa+f)\right)\end{aligned}$$

When $\kappa(x)+f\in[0,x_0]$ (assuming $\kappa(x)\le x_0$), the function to optimize is $1/f+1/(1-\kappa(x)-f)$. This function has no local minimum for $\kappa(x)+f\in[0,x_0]$. Hence over $[0,x_0-\kappa(x)]$, the infimum is when $f=x_0-\kappa(x)$, and its value is $1/(x_0-\kappa(x))+1/(1-x_0)$. When $\kappa(x)+f\in[x_0,2)$, the function to optimize is $1/f+1+1/(2-\kappa(x)-f)$. The local minimum is reached when $f=(2-\kappa(x))/2$, which indeed satisfies $\kappa(x)+f\in[x_0,2)$ when $\kappa(x)<2$. In the end, we obtain $\mathscr{P}_4(\ell_0,\kappa)=\mathscr{P}(\ell_0,\kappa)=1+\frac{4}{2-\kappa(x)}$.

4.3 Extension to one-clock timed games

In the computations above, non-determinism is solved by an adversary with very limited capabilities. We explain below how our approach can be lifted to *timed games*, where the second player has more power. In (classical) timed games, at each step, both players propose a delay and an action (where the set of actions is partitioned between Player-1 actions and Player-2 actions); the player with the shortest delay (if any) then applies her move, and the game continues. Non-determinism (when both players propose the same delay) is solved by the second player [AFH⁺03]. This framework can be lifted to the setting of permissive strategies: the first player proposes a multi-move (I,a), while the second player proposes a move[6] (δ,α). In case $\delta < e$ for all $e \in I$, then the move of the second player is applied; in case $\delta > f$ for all $f \in I$, the second player selects a delay $d \in I$, and the move (d,a) is applied; finally, if $\delta \in I$, Player 2 may decide to either apply her move (δ,α), or to select some $d \in I$ with $d \leq \delta$, and to apply the move (δ,a).

Our results above for permissive strategies in timed automata can be extended to timed games, with the following changes: First, we adapt the computation of the sequence $\mathscr{P}_{i+1}$, in order to take the extended capabilities of the opponent. More precisely, instead of only maximizing $\mathscr{P}_i(\mathsf{succ}(q,\kappa,d,a))$ when d ranges over I, she now also has the opportunity to apply another move (δ,α) for any δ in or "before" I (*i.e.*, for which there exists $t \geq 0$ s.t. $\delta + t \in I$):

$$\mathscr{Q}_i(g,\kappa) = 0$$

$$\mathscr{Q}_0(q,\kappa) = +\infty \qquad \text{for all } q \neq g$$

$$\mathscr{Q}_{i+1}(q,\kappa) = \min_{a\in\mathsf{Act}_1} \inf_{I\in\mathscr{D}(\kappa,\mathsf{Inv}(q))} \Big(\mathsf{penalty}(I,a) + \max\big(\sup_{d\in I} \mathscr{Q}_i(\mathsf{succ}(q,\kappa,d,a)), \sup_{\delta\leq I} \max_{\alpha\in\mathsf{Act}_2} \mathscr{Q}_i(\mathsf{succ}(q,\kappa,\delta,\alpha))\big)\Big)$$

With this definition, the first statement of Lemma 4 fails: the global penalty of move (I,a) from $(q,\kappa+t)$ might be better than the penalty of move $(I+t,a)$ from (q,κ), because the latter offers more possibilities to Player 2. Still, in the one-clock setting, the result holds in case $\kappa(x)$ and $\kappa(x)+t$ are not separated by any constant of the automaton. In other terms, for one-clock games, $t \mapsto \mathscr{Q}_n(q,\kappa+t)$ is piecewise non-decreasing and piecewise continuous, with pieces defined by the constants of the automaton.

In the end, we can still look for the optimal choice of Player 2 within a finite set, and all the other arguments that we used in the one-player setting still apply. Finally, we obtain:

Theorem 11 *The optimal penalty (and a corresponding almost-optimal strategy) for reaching a target location in a timed automaton game can be computed in polynomial time.*

4.4 Discussions on other ways of computing penalties

Our choice of the way we compute penalties is only one among many relevant possibilities: for instance, we believe that assigning penalty $+\infty$ to punctual intervals is reasonable when dealing with robustness, but it might be argued that we should compare how much the player reduces the

[6] We only consider permissive strategies of the first player in this setting, as we only want to minimize the penalty for the protagonist.

interval compared to what she is allowed to do. In other terms, if playing punctual is the only possibility, then it should have bounded penalty. This requires extra definitions and arguments, which we leave for future work.

Accumulating penalties along a run is also something we could change. It fits well with finitary objectives such as reachability, and favors short paths. Another solution would be to take the maximum penalty along the outcomes, which could be handled by a trivial modification of our algorithm. Averaging would be yet another option, which looks more complex to compute.

5 Conclusion

This paper has introduced a new notion of robustness in timed automata based on permissive strategies. The contribution on the specification part is twofold: On the one hand, we extended the notion of permissive strategies to a timed formalism. On the other hand, this notion of permissiveness, which implies robustness, makes it possible to compare the robustness of two different system models based on the quantitative measure of the permissiveness. Furthermore, we have shown that the most permissive strategy with respect to the reachability acceptance condition on one clock can be efficiently computed. This presented approach can be seen as a complementary methodology to existing practical approaches such as presented in [BFLM11] or [CJL+09] for measuring the robust on models of timed systems.

Challenging problems in this approach remain when additional clocks will be included. At this point, reasoning about the system models might get computationally infeasible.

Bibliography

[ACD90] R. Alur, C. Courcoubetis, D. Dill. Model-checking for real-time systems. In *LICS'90*. Pp. 414–425. June 1990.

[AD94] R. Alur, D. L. Dill. A theory of timed automata. *Theoretical Computer Science* 126(2):183–235, 1994.

[AFH+03] L. de Alfaro, M. Faella, T. A. Henzinger, R. Majumdar, M. Stoelinga. The Element of Surprise in Timed Games. In Amadio and Lugiez (eds.), *CONCUR'03*. LNCS 2761, pp. 142–156. Springer, Aug.-Sept. 2003.

[BDM+98] M. Bozga, C. Daws, O. Maler, A. Olivero, S. Tripakis, S. Yovine. Kronos: A model-checking tool for real-time systems. In *Formal Techniques in Real-Time and Fault-Tolerant Systems*. Pp. 298–302. 1998.

[BDMR09] P. Bouyer, M. Duflot, N. Markey, G. Renault. Measuring permissivity in finite games. In *CONCUR'09*. Pp. 196–210. Springer, 2009.

[BFLM11] P. Bouyer, U. Fahrenberg, K. G. Larsen, N. Markey. Quantitative analysis of real-time systems using priced timed automata. *Communications of the ACM* 54(9):78–87, Sept. 2011.

[BJW02] J. Bernet, D. Janin, I. Walukiewicz. Permissive strategies: from parity games to safety games. *RAIRO-ITA* 36(3):261–275, 2002.

[BKK11] Ch. Baier, J. Klein, S. Klüppelholz. A Compositional Framework for Controller Synthesis. In Katoen and König (eds.), *CONCUR'11*. LNCS 6901, pp. 512–527. Springer, Sept. 2011.

[BMOU11] P. Bouyer, N. Markey, J. Olschewski, M. Ummels. Measuring Permissiveness in Parity Games: Mean-Payoff Parity Games Revisited. In Bultan and Hsiung (eds.), *ATVA'11*. LNCS 6996, pp. 135–149. Springer, Taipei, Taiwan, Oct. 2011.

[BMS12] P. Bouyer, N. Markey, O. Sankur. Robust reachability in timed automata: a game-based approach. In Czumaj et al. (eds.), *ICALP'12*. LNCS 7392, pp. 128–140. Springer, July 2012.

[CJL$^+$09] F. Cassez, J. Jessen, K. Larsen, J.-F. Raskin, P.-A. Reynier. Automatic Synthesis of Robust and Optimal Controllers An Industrial Case Study. In Majumdar and Tabuada (eds.), *Hybrid Systems: Computation and Control*. Lecture Notes in Computer Science 5469, pp. 90–104. Springer Berlin Heidelberg, 2009.

[DDMR04] M. De Wulf, L. Doyen, N. Markey, J.-F. Raskin. Robustness and Implementability of Timed Automata. In Lakhnech and Yovine (eds.), *FORMATS'04*. LNCS 3253, pp. 118–133. Springer, Sept. 2004.

[DDR04] M. De Wulf, L. Doyen, J.-F. Raskin. Almost ASAP Semantics: From Timed Models to Timed Implementations. In Alur and Pappas (eds.), *HSCC'04*. LNCS 2993, pp. 296–310. Springer, Mar.-Apr. 2004.

[DFK$^+$14] K. Dräger, V. Forejt, M. Kwiatkowska, D. Parker, M. Ujma. Permissive controller synthesis for probabilistic systems. In *TACAS'14*. Pp. 531–546. Springer, 2014.

[ELTV14] R. Ehlers, S. Lafortune, S. Tripakis, M. Y. Vardi. Bridging the Gap between Supervisory Control and Reactive Synthesis: Case of Full Observation and Centralized Control. In Lesage et al. (eds.), *WODES'14*. Pp. 222–227. IFAC, 2014.

[FLM14] M. Faella, S. La Torre, A. Murano. Automata-theoretic decision of timed games. *Theoretical Computer Science* 515:46–63, 2014.

[GHJ97] V. Gupta, T. A. Henzinger, R. Jagadeesan. Robust Timed Automata. In Maler (ed.), *HART'97*. LNCS 1201, pp. 331–345. Springer, Mar. 1997.

[HHW97] T. Henzinger, P.-H. Ho, H. Wong-Toi. HyTech: A model checker for hybrid systems. In Grumberg (ed.), *Computer Aided Verification*. Lecture Notes in Computer Science 1254, pp. 460–463. Springer Berlin Heidelberg, 1997.

[HR00] T. A. Henzinger, J.-F. Raskin. Robust undecidability of timed and hybrid systems. In *HSCC'00*. Pp. 145–159. Springer Berlin Heidelberg, 2000.

[LPY97] K. G. Larsen, P. Pettersson, W. Yi. UPPAAL in a nutshell. In *International Journal on Software Tools for Technology Transfer (STTT)*. Volume 1(1), pp. 134–152. Springer, 1997.

[ORS14] Y. Oualhadj, P.-A. Reynier, O. Sankur. Probabilistic Robust Timed Games. In *CONCUR'14*. LNCS 8704, pp. 203–217. Springer, 2014.

[Pur98] A. Puri. Dynamical Properties of Timed Automata. In Ravn and Rischel (eds.), *FTRTFT'98*. LNCS 1486, pp. 210–227. Springer, Sept. 1998.

[RW89] P. J. Ramadge, W. M. Wonham. The Control of Discrete Event Systems. *Proc. IEEE* 77(1):81–98, Jan. 1989.

[San15] O. Sankur. Symbolic Quantitative Robustness Analysis of Timed Automata. In *TACAS'15*. 2015.

[SBM11] O. Sankur, P. Bouyer, N. Markey. Shrinking Timed Automata. In Chakraborty and Kumar (eds.), *FSTTCS'11*. LIPIcs 13, pp. 375–386. LZI, Dec. 2011.