# As Soon as Probable: Optimal Scheduling under Stochastic Uncertainty

Jean-Francois Kempf, Marius Bozga, and Oded Maler

CNRS-VERIMAG
University of Grenoble
France
**@imag.fr**

**Abstract.** In this paper we continue our investigation of stochastic (and hence dynamic) variants of classical scheduling problems. Such problems can be modeled as duration probabilistic automata (DPA), a well-structured class of acyclic timed automata where temporal uncertainty is interpreted as a bounded *uniform distribution* of task durations [18]. In [12] we have developed a framework for computing the expected performance of a *given* scheduling policy. In the present paper we move from *analysis* to *controller synthesis* and develop a dynamic-programming style procedure for automatically synthesizing (or approximating) *expected time optimal* schedulers, using an iterative computation of a *stochastic time-to-go* function over the state and clock space of the automaton.

## 1 Introduction

The allocation over time of reusable resources to competing tasks is a problem of almost universal importance, manifested in numerous application domains ranging from manufacturing (job shop) to program parallelization (task-graph). When the system admits uncertainty, for example, in task arrival time or duration, one cannot execute a fixed time-triggered schedule but needs to develop a scheduling *policy* (or *strategy*) that prescribes scheduling decisions at different *states* that the system may find itself in. The general problem falls into the scope of *controller synthesis* for timed systems [5] and its extensions toward time-optimality [4] and cost-optimality [14]. The reader is referred to [1] for a general framework for modeling and solving dynamic scheduling problems based on a restricted class of timed automata and to [17] for a more general exposition of control in the presence of adversaries.

The above mentioned work treats temporal uncertainty in the *set-theoretic* and *worst-case* tradition. A duration of a task can be *any* number in a given interval (no probability assigned) and the strategy is evaluated according to the worst performance induced by an instance of the external environment. Duration probabilistic automata (DPA) have been introduced in [18] to express stochastic scheduling problems and evaluate the expected performance of schedulers by methods similar to techniques used for generalized semi-Markov processes (GSMP) [10, 11, 7] such as [6, 2]. This approach refines the successor operator used in the verification of timed automata [9, 15] from a *zone transformer* into a *density transformer* and can compute, for example, the time distribution of reaching the final state and hence the expected termination time under a *given*

scheduler. In [12] we developed an alternative clock-free procedure for evaluating the performance of schedulers by computing *volumes* of *zones* in the duration space.

In the present paper we move to the *synthesis* problem where we seek an optimal scheduling policy which decides in what global situations to *start* an enabled step of a process and under what conditions to *wait* and let other processes use the resource first. To this end we define a *stochastic time-to-go* function which assigns to any state of the schedule (global state of the automaton *and* the values of active clocks) the density of the time to total termination starting from this state and following the optimal strategy. These functions are piecewise-continuous and we show how they can be computed backwards from the final state.

The rest of the paper is organized as follows. Section 2 provides preliminary definitions. Section 3 introduces single processes, shows how to model them by automata and solve the (degenerate) optimal scheduling problem by backward value iteration. In Section 4 we introduce several processes running in parallel and admitting resource conflicts. We model them as products of automata and define schedulers that, in each state, resolve the non-determinism associated with the decision whether to start a step. In Section 5 we define the basic iterative step for computing the value (optimal expected time-to-go) in a state of the automaton based on the value of its successors. The value/policy iteration algorithm using these operators defines the optimal strategy. In Section 6 we study the computational aspects of the algorithm. First we characterize the class of time densities obtained by applying the passive race-analysis part of the iteration. Then we prove a non-laziness property of optimal schedulers for this class of problems, which facilitates the approximate solution of the optimization part of the iteration. A discussion of future work closes the paper.

## 2 Preliminaries

**Definition 1 (Bounded Support Time Density).** *A time density is a function* $\psi : \mathbb{R}_+ \to \mathbb{R}_+$ *satisfying*

$$\int_0^\infty \psi[t]dt = 1.$$

*A time density is of bounded support when* $\psi(t) \neq 0$ *iff* $t \in I$ *for some interval* $I = [a, b]$. *A partial time density satisfies the weaker condition:* $\int \psi[t]dt < 1$. *A bounded-support time density is uniform if* $\psi[t] = 1/(b-a)$ *inside its support* $[a, b]$.

We use a non-standard notation for *distributions*:

$$\psi[\leq t] = \int_0^t \psi[t']dt' \quad \psi[> t] = 1 - \psi[\leq t]$$

with $\psi[\leq t]$ indicating the probability of a duration which is at most $t$. We use $\mathbf{c}$ to denote the "deterministic" (Dirac) density which gives the constant $c$ with probability 1. The *expected value* of a time density $\psi$ is $\mathbb{E}(\psi) = \int \psi[t] \cdot tdt$.

We will use time densities to specify durations of tasks (process steps) as well as the remaining time to termination given some state of the system. To this end we need the following *operators* on densities: 1) *Convolution*, to characterize the duration of two or

more tasks composed sequentially, and 2) *Shift*, to reflect the change in the remaining time *given* that the process has *already* spent some amount of time $x$.

**Definition 2 (Convolution and Shift).** *Let $\psi$ , $\psi_1$ and $\psi_2$ be uniform densities supported by $I = [a, b]$, $I_1 = [a_1, b_1]$ and $I_2 = [a_2, b_2]$, respectively.*

- *The convolution $\psi_1 * \psi_2$ is a density $\psi'$ supported by $I' = I_1 \oplus I_2 = [a_1+a_2, b_1+b_2]$ and defined as*
$$\psi'[t] = \int_0^t \psi_1[t']\psi_2[t-t']dt'$$
- *The residual density (shift) of $\psi$ relative to a real number $0 \leq x < b$ is $\psi' = \psi_{/x}$ such that $\psi'[t] = \psi[x+t] \cdot \gamma_{a,b}(x)$ where*
$$\gamma_{a,b}(x) = \begin{cases} 1 & \text{if } 0 < x \leq a \\ \frac{b-a}{b-x} & \text{if } a < x < b \end{cases}$$

When $x < a$, $\psi_{/x}$ is a simple shift of $\psi$. When $x > a$ we already know that the actual duration is at least $x$ (restricted to the sub-interval $[x, b]$) and hence we need to normalize. Note also that $(\psi * \mathbf{c})[t] = (\mathbf{c} * \psi)[t] = \psi[t-c]$ and that $\mathbf{0}$ is the identity element for convolution. We write $\psi' = \psi_{/x}$ more explicitly as

$$\psi'[t] = \begin{cases} 0 & \text{when } x + t \leq a \\ \frac{1}{b-a} & \text{when } x \leq a, a < x + t \leq b \\ \frac{1}{b-x} & \text{when } a < x, x + t \leq b \\ 0 & \text{when } b < x + t \end{cases}$$

One can verify that the shift satisfies: $\psi_{/x}[t-x] = \gamma_{a,b}(x) \cdot \psi[t]$ and $(\psi_{/x})_{/y} = \psi_{/(x+y)}$. Note that the expressive weakness (and computational advantage) of the exponential distribution is due to $\psi_{/x} = \psi$. A subset of a hyper-rectangle is called a *zone* if it is expressible as a *conjunction* of *orthogonal* and *difference* constraints, namely constraints of the form $x_i \leq c$, $x_i - x_{i'} \leq c$, etc.

## 3 Processes in Isolation

Processes in our model are inspired by the jobs in the job-shop problem. Each process consists of an ordered sequence of steps, indexed by $K = \{1, \ldots, k\}$, such that a step can start executing only after its predecessor has terminated.[1]

**Definition 3 (Process).** *A sequential stochastic process is a pair $P = (\mathcal{I}, \Psi)$ where $\mathcal{I} = \{I_j\}_{j \in K}$ is a sequence of duration intervals and $\Psi = \{\psi_j\}_{j \in K}$ is a matching sequence of densities with $\psi_j$ being the uniform density over $I_j = [a_j, b_j]$, indicating the duration of step $j$.*

Probabilistically speaking, step durations can be viewed as a finite sequence of *independent* uniform random variables $\{y_j\}_{j \in K}$ that we denote as points $y = (y_1, \ldots, y_k)$ ranging over a *duration space* $D = I_1 \times \cdots \times I_k \subseteq \mathbb{R}^k$ with density $\psi(y_1, \ldots, y_k) = \psi_1(y_1) \cdots \psi_k(y_k)$. A state-based representation of a process is given by simple DPA.

[1] See [1] for a straightforward generalization to partial-order precedence constraints.

**Definition 4 (SDPA).** *A simple duration probabilistic automaton (SDPA) of $k$ steps is a tuple $\mathcal{A} = (\Sigma, Q, \{x\}, Y, \Delta)$ where $\Sigma = \Sigma_s \uplus \Sigma_e$ is the alphabet of* start *and* end *actions with $\Sigma_s = \{s_1, \ldots, s_k\}$ and $\Sigma_e = \{e_1, \ldots, e_k\}$. The state space is an ordered set $Q = \{\overline{q}_1, q_1, \overline{q}_2, \ldots, q_k, \overline{q}_{k+1}\}$ with $\overline{q}_j$ states considered* idle *and $q_j$ states are* active*, $x$ is a clock variable and $Y = \{y^1, \ldots, y^k\}$ is a set of auxiliary random variables. The transition relation $\Delta$ consists of tuples of the form $(q, g, r, q')$ with $q$ and $q'$ being the source and target of the transition, $g$ is a guard, a precondition (external or internal) for the transition and $r$ is an action (internal or external) accompanying the transition. The transitions are of two types:*

1. Start *transitions: for every idle state $\overline{q}_j$, $j < k+1$ there is one transition of the form $(\overline{q}_j, s_j, \{x\}, q_j)$. The transition, triggered by a scheduler command $s_j$, activates clock $x$ and sets it to zero;*
2. End *transitions: for every active state $q_j$, there is a transition, conditioned by the clock value, of the form $(q_j, x = y_j, e_j, \overline{q}_{j+1})$. This transition renders clock $x$ inactive and outputs an $e_j$ event.*

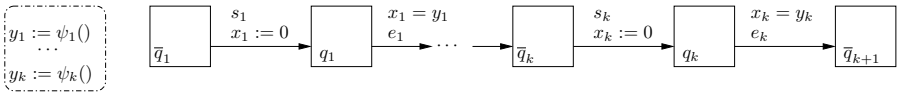



**Fig. 1.** A simple DPA.

For each step $j$ we draw a duration $y_j$ according to $\psi_j$. Upon a scheduler command $s_j$ the automaton moves from a waiting state $\overline{q}_j$ to active state $q_j$ in which clock $x$ advances with derivative 1. The *end* transition is taken when $x = y_j$, that is, $y_j$ time after the corresponding *start* transition. An extended state of the automaton is a pair $(q, x)$ consisting of a discrete state and a clock value which represents the time elapsed since the last *start* transition. The extended state-space of the SDPA is thus

$$S = \{(\overline{q}_j, \bot) : j \leq k+1\} \cup \{(q_j, x) : j \leq k \wedge x \leq b_j\}$$

where $\bot$ indicates the inactivity of the clock in waiting/idle states.

Note the difference between transition labels $s_j$ and $e_j$: the *start* transitions are *controllable* and are issued by the scheduler that we want to optimally synthesize while the *end* transitions are generated by the *uncontrolled* external (to the scheduler) environment represented by random variable $y_j$. Without a scheduler, the SDPA is *undetermined* and can issue a *start* transition any time. The derivation of an optimal scheduler is done via the computation of a *time-to-go* function that we first illustrate on the degenerate case of one process in isolation. In this case each state has only one successor and any waiting between steps unnecessarily increases the time to termination.

**Definition 5 (Local Stochastic Time to Go).** *The local stochastic time-to-go function associates with every state $(q, x)$ a time density $\mu(q, x)$ with $\mu(q, x)[t]$ indicating the*

*probability to terminate within $t$ time* given *that we start from $(q, x)$ and apply the optimal strategy.*

This function admits the following inductive definition:

$$\mu(\overline{q}_{k+1}, \bot) = \mathbf{0} \tag{1}$$

$$\mu(\overline{q}_j, \bot) = \mu(q_j, 0) \tag{2}$$

$$\mu(q_j, x)[t] = \int_0^t \psi_{j/x}[t'] \cdot \mu(\overline{q}_{j+1}, 0)[t - t']dt' \tag{3}$$

Line (1) indicates the final state while (2) comes from the fact that in the absence of conflicts the optimal scheduler need not wait and should start each step immediately when enabled. Equation (3) computes the probability for termination at $t$ based on the probabilities of terminating the current step in some $t'$ and of the remaining time-to-go being $t - t'$. It can be be summarized in a functional language as

$$\mu(q_j, x) = \psi_{j/x} * \mu(\overline{q}_{j+1}, \bot) = \psi_{j/x} * \mu(q_{j+1}, 0) \tag{4}$$

The successive application of (4) yields, not surprisingly, $\mu(q_1, 0) = \psi_1 * \cdots * \psi_k$.

**Definition 6 (Local Expected Time to Go).** *The expected time-to-go function is $V : Q \times X \to \mathbb{R}_+$ defined as*

$$V(q, x) = \int \mu(q, x)[t] \cdot t dt = \mathbb{E}(\mu(q, x)).$$

This measure satisfies $V(q_j, x) = \mathbb{E}(\psi_{j/x}) + V(q_{j+1}, 0)$ where the first term is the expected termination time of step $j$ starting from $x$. For the initial state this yields

$$V(q_1, 0) = \mathbb{E}(\psi_1 * \cdots * \psi_k) = \mathbb{E}(\psi_1) + \cdots + \mathbb{E}(\psi_k) = \sum_{j=1}^{k} (a_j + b_j)/2.$$

## 4 Conflicts and Schedulers

We extend the model to $n$ processes, indexed by $N = \{1..n\}$, that may run in parallel except for steps which are mutually conflicting due to the use of the same resource. For simplicity of notation we assume all processes to have the same number $k$ of steps.

**Definition 7 (Process System).** *A process system is a triple $(\mathcal{P}, M, h)$ where*

$$\mathcal{P} = P^{\mathbf{1}} || \cdots || P^{\mathbf{n}} = \{(\mathcal{I}^{\mathbf{i}}, \Psi^{\mathbf{i}})\}_{i \in N}$$

*is a set of processes, $M$ is a set of resources, and $h : N \times K \to M$ is a function which assigns to each step the resource that it uses.*

We use notations $P^{\mathbf{i}}_j$ to refer to step $j$ of process $i$ and $\psi^{\mathbf{i}}_j$ and $I^{\mathbf{i}}_j = [a^{\mathbf{i}}_j, b^{\mathbf{i}}_j]$ for the respective densities and their support intervals. Likewise we denote the corresponding controllable and uncontrollable actions by $s^{\mathbf{i}}_j$ and $e^{\mathbf{i}}_j$, respectively. Without loss of generality we assume there is one instance of each resource type, hence any two steps $P^{\mathbf{i}}_j$ and $P^{\mathbf{i}'}_{j'}$ such that $h(i,j) = h(i',j')$ are in *conflict* and cannot execute simultaneously. Each process is modeled as an SDPA $\mathcal{A}^{\mathbf{i}} = (\Sigma^{\mathbf{i}}, Q^{\mathbf{i}}, \{x^{\mathbf{i}}\}, Y^{\mathbf{i}}, \Delta^{\mathbf{i}})$ and the global system is obtained as a product of those restricted to conflict-free states. We write global states as $q = (q^{\mathbf{1}}, \ldots, q^{\mathbf{n}})$ and exclude states where for some $i$ and $i'$, $q^{\mathbf{i}} = q^{\mathbf{i}}_j$, $q^{\mathbf{i}'} = q^{\mathbf{i}'}_{j'}$ and steps $P^{\mathbf{i}}_j$ and $P^{\mathbf{i}'}_{j'}$ are conflicting. We say that action $s^{\mathbf{i}}_j$ (respectively, $e^{\mathbf{i}}_j$) is enabled in $q$ if $q^{\mathbf{i}} = \overline{q}^{\mathbf{i}}_j$ (resp. $q^{\mathbf{i}} = q^{\mathbf{i}}_j$). Since only one transition per process is possible in a global state, we will sometime drop the $j$-index and refer to those as $s^{\mathbf{i}}$ and $e^{\mathbf{i}}$.

**Definition 8 (Duration Probabilistic Automata).** *A duration probabilistic automaton (DPA) is a composition $\mathcal{A} = \mathcal{A}^{\mathbf{1}} \circ \cdots \circ \mathcal{A}^{\mathbf{n}} = (\Sigma, Q, X, Y, \Delta)$ of $n$ SDPA with the action alphabet being $\Sigma = \bigcup_i \Sigma^{\mathbf{i}}$. The discrete state space is $Q \subseteq Q^{\mathbf{1}} \times \cdots Q^{\mathbf{n}}$ (with forbidden states excluded). The set of clocks is $X = \{x^{\mathbf{1}}, \ldots, x^{\mathbf{n}}\}$, the extended state-space is $S \subseteq S^{\mathbf{1}} \times \cdots S^{\mathbf{n}}$ and the auxiliary variables are $Y = \bigcup_i Y^{\mathbf{i}}$ ranging over the joint duration space $D = D^{\mathbf{1}} \times \cdots \times D^{\mathbf{n}}$. The transition relation $\Delta$ is built using interleaving, that is, a transition $(q, g, r, q')$ from $q = (q^{\mathbf{1}}, \ldots, q^{\mathbf{i}}, \ldots, q^{\mathbf{n}})$ to $q' = (q^{\mathbf{1}}, \ldots, q'^{\mathbf{i}}, \ldots, q^{\mathbf{n}})$ exist in $\Delta$ if a transition from $(q^{\mathbf{i}}, g, r, , q'^{\mathbf{i}})$ exists in $\Delta^{\mathbf{i}}$, provided that $q'$ is not forbidden.*

The DPA thus defined (see Fig. 2-a) is not probabilistically correct as it admits non-determinism of a non probabilistic nature: in a given state the automaton may choose between several *start* transitions or decide to wait for an *end* transition (the termination of an active step). A *scheduler* selects *one* action in any state and then the only non-determinism that remains is due to the probabilistic task durations. A discussion on different types of schedulers can be found in [12].

**Definition 9 (Scheduler).** *A scheduler for a DPA $\mathcal{A}$ is a function $\Omega : S \to \Sigma_s \cup \{\mathbf{w}\}$ such that for every $s \in \Sigma_s$, $\Omega(q,x) = s$ only if $s$ is enabled in $q$ and $\Omega(q,x) = \mathbf{w}$ (wait) only if $q$ admits at least one active component.*

Composing the scheduler with the DPA (see Fig. 2-b) renders it input-deterministic in the sense that any point $y \in D$ induces a unique[2] run of the automaton.

**Definition 10 (Steps, Runs and Successors).** *The steps of a controlled DPA $\mathcal{A} \circ \Omega$, induced by a point $y \in D$ are of the following types:*

- *Start steps:* $(q,x) \xrightarrow{s^{\mathbf{i}}_j} (q',x')$ *iff* $q^{\mathbf{i}} = \overline{q}^{\mathbf{i}}_j$ *and* $\Omega(q,x) = s^{\mathbf{i}}_j$;
- *End steps:* $(q,x) \xrightarrow{e^{\mathbf{i}}_j} (q',x')$ *iff* $q^{\mathbf{i}} = q^{\mathbf{i}}_j$ *and* $x^{\mathbf{i}} = y^{\mathbf{i}}_j$;
- *Time steps:* $(q,x) \xrightarrow{t} (q, x+t)$ *iff* $\forall i\ (q^{\mathbf{i}} = q^{\mathbf{i}}_j \Rightarrow x^{\mathbf{i}} + t < y^{\mathbf{i}}_j)$.

[2] We define a priority order among the $e^{\mathbf{i}}$-actions so that in the (measure zero) situation where two actions are taken simultaneously we impose the order to guarantee a unique run and avoid artifacts introduced by the interleaving semantics.

*The* run *associated with $y$ is a sequence of steps starting at $(\overline{q}_1^{\mathbf{1}}, \ldots, \overline{q}_1^{\mathbf{n}})$ and ending in $(\overline{q}_{k+1}^{\mathbf{1}}, \ldots, \overline{q}_{k+1}^{\mathbf{n}})$.*
*The $t$-$i$-*successor *of a state $(q, x)$ denoted by $\sigma^{\mathbf{i}}(t, q, x)$, is the state $(q', x')$ reached from $(q, x)$ after a time step of duration $t$ followed by a a transition of $P^{\mathbf{i}}$ .*

The duration of a run is the sum of the time steps and it coincides with the termination time of the last process, known as *makespan* in the scheduling literature.

## 5 Expected Time Optimal Schedulers

In [12] we developed a method to compute the expected termination time under a *given* scheduler by computing volumes in the duration space. We now show how to optimally synthesize such schedulers from an uncontrolled DPA description. To this end we extend the formulation of stochastic time-to-go from a single process (3) to multiple processes (7).

We define a partial order relation on global states based on the order on the local states with $q \preceq q'$ if for every $i$, $q^{\mathbf{i}} \preceq q^{\mathbf{i}'}$. For extended states we let $(q, x) \preceq (q', x')$ if either $q \prec q'$ or $(q = q' \wedge x \leq x')$. The *forward cone* of a state $q$ is the set of all states $q'$ such that $q \prec q'$. The immediate successors of a state $q$ are the states reachable from it by one transition. A *partial scheduler* is a scheduler defined only on a subset of $Q$. To optimize the decision of the scheduler in a state we need to compare the effect of the possible actions on the time-to-go.

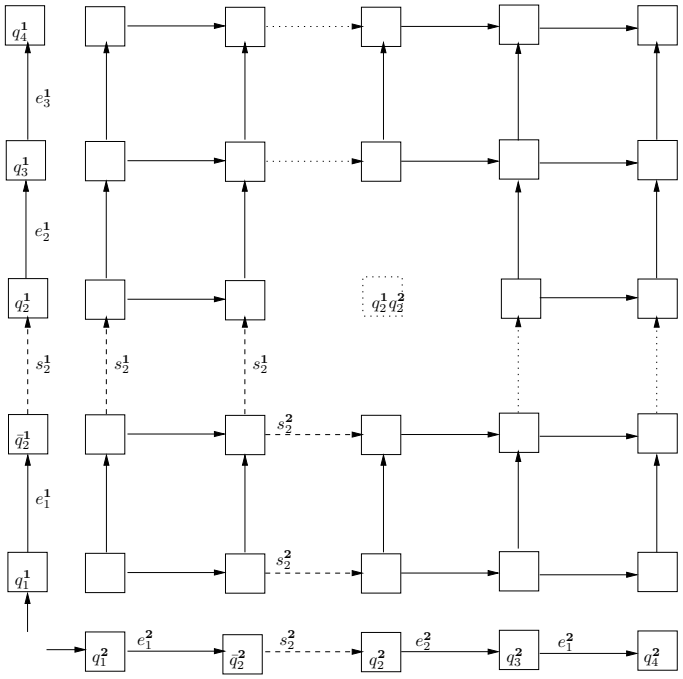
**Definition 11 (Local Stochastic Time-to-Go).** *Let $\mathcal{A}$ be a DPA with a partial strategy whose domain includes the forward cone of a state $q$. With every $i$, $x$ and every $s \in \Sigma_s \cup \{\mathbf{w}\}$ enabled in $q$, the time density $\mu^{\mathbf{i}}(q, x, s) : \mathbb{R}_+ \to [0, 1]$ characterizes the stochastic time-to-go for process $P^{\mathbf{i}}$ if the controller issues action $s$ at state $(q, x)$ and continues from there according to the partial strategy.*

Note that for any successor $q'$ of $q$ the optimal action has already been selected and we denote its associated time-to-go by $\mu^{\mathbf{i}}(q', x')$. Once $\mu^{\mathbf{i}}(q, x, s)$ has been computed for every $i$, the following measures, all associated with action $s$, can be derived from it.

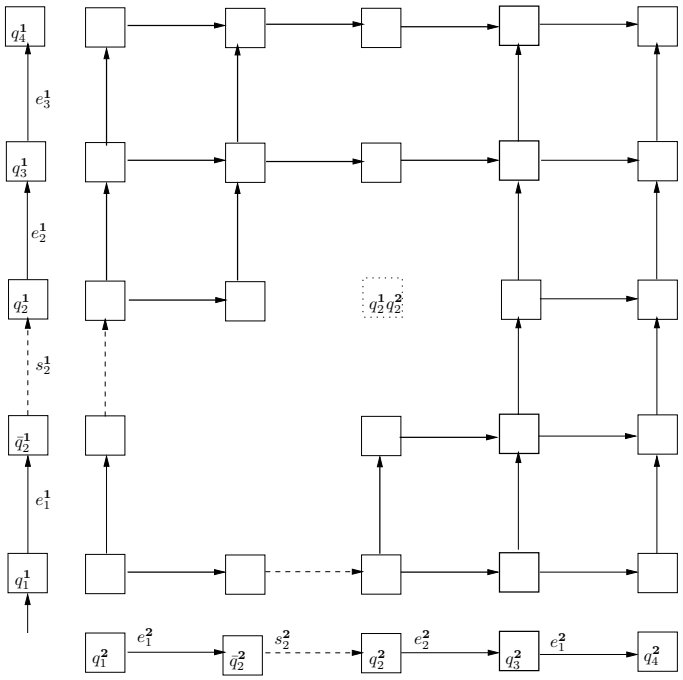
**Definition 12 (Global Stochastic Time-to-Go).** *With every state $(q, x)$ and action $s$ enabled in it, we define*

- *The stochastic time-to-go for total termination (makespan) which is the expected duration of the last task: $\mu(q, x, s) = \max\{\mu^{\mathbf{1}}(q, x, s), \ldots, \mu^{\mathbf{n}}(q, x, s)\}$;*
- *The expected total termination time: $V(q, x, s) = \int t \cdot \mu(q, x, s)[t]dt$*

The computation of $\mu$ for a state $q$, based on the stochastic time-to-go of its successors, is one of the major contributions of the paper. The hard part is the computation of the time-to-go associated with *waiting* in a state where several processes are active. In this *race* situation the automaton may leave $q$ via different transitions and $\mu$ should be computed based on the probabilities of these transitions (and their timing) and the time-to-go from the respective successor states.




(a)




(b)

**Fig. 2.** (a) The DPA for two parallel processes admitting a resource conflict and their respective second steps, with the conflict state $(q_2^1, q_2^2)$ removed. The dashed arrows indicate *start* transitions which should be under the control of a scheduler while the dotted arrows indicate post-conflict *start* transitions; (b) The automaton resulting from composition with a FIFO scheduler which starts a step as soon as it is enabled.

With each state $(q, x)$ we associate a family $\{\rho^{\mathbf{i}}(q,x)\}_{i\in N}$ of partial time densities with the intended meaning that $\rho^{\mathbf{i}}(q,x)[t]$ is (the density of) the probability that the *first* process to terminate its current step is $P^{\mathbf{i}}$ *and* that this occurs within $t$ time. This is relative to the fact that the time elapsed since the initiation of each active step is captured by the respective clock value in $x$.[3]

**Definition 13 (Race Winner).** *Let $q$ be a state where $n$ processes are active, each in a step admitting a time density $\psi^{\mathbf{i}}$. With every clock valuation $x = (x^{\mathbf{1}}, \ldots, x^{\mathbf{n}}) \leq (b^{\mathbf{1}}, \ldots, b^{\mathbf{n}})$ and every $i$ we associate the partial density:*

$$\rho^{\mathbf{i}}(q,x)[t] = \psi^{\mathbf{i}}{}_{/x^{\mathbf{i}}}[t] \cdot \prod_{i'\neq i} \psi^{\mathbf{i}'}{}_{/x^{\mathbf{i}'}}[> t]$$

This is the probability that $P^{\mathbf{i}}$ terminates in $t$ time and every other process $P^{\mathbf{i}'}$ terminates within some $t' > t$.

**Definition 14 (Computing Stochastic Time-to-go).** *For every $i$, the function $\mu^{\mathbf{i}}$ is defined inductively as*

$$\mu^{\mathbf{i}}((\ldots \overline{q}^{\mathbf{i}}_{k+1} \ldots), x) = \mathbf{0} \tag{5}$$

$$\mu^{\mathbf{i}}(q, x, s^{\mathbf{i}'}) = \mu^{\mathbf{i}}(\sigma^{\mathbf{i}'}(0, q, x))) \tag{6}$$

$$\mu^{\mathbf{i}}(q, x, \mathbf{w})[t] = \sum_{i'=1}^{n} \int_0^t \rho^{\mathbf{i}'}(q,x)[t'] \cdot \mu^{\mathbf{i}}(\sigma^{\mathbf{i}'}(t', q, x))[t - t']dt' \tag{7}$$

For any global state where $P^{\mathbf{i}}$ is in its final state, $\mu^{\mathbf{i}}$ is zero (5). Each enabled *start* action $s^{\mathbf{i}'}$ leads immediately to the successor state and the cost-to-go is inherited from there (6). For waiting we make a convolution between the probability of $P^{\mathbf{i}'}$ winning the race and the value of $\mu^{\mathbf{i}}$ in the post-transition state and sum up over all the active processes (7). The basic iterative step in computing the value function and strategy is summarized in Algorithm 1. A *dynamic programming* algorithm starting from the final state and applying the above procedure will produce the expected-time optimal strategy. Since we are dealing with *acyclic* systems, the question of convergence to a fixed point is not raised and the only challenge is to show that the defined operators are computable.

## 6 Computing Optimal Strategies

Algorithm 1 splits into three parts, the third being merely book-keeping. In the first we essentially *compute* the outcome of *waiting* (by race analysis) and of *starting*. As we

[3] Note that the dependence on the time already elapsed is in sharp contrast with the *memoryless exponential* distribution where this time does *not* matter for the future. For those distributions the time-to-go is associated only with the discrete state and is much easier to compute, see [1] for the derivation of optimal schedulers for timed automata with exponential durations.

**Algorithm 1: Value Iteration**

**Input**: A global state $q$ such that $\Omega(q', x)$ and $\mu^{\mathbf{i}}(q', x)$ have been computed for each of its successors $q'$ and every $i$
**Output**: $\Omega(q, x)$, and $\mu^{\mathbf{i}}(q, x)$

% COMPUTE:
**forall** $s \in \Sigma_s \cup \{\mathbf{w}\}$ enabled in $q$
 **for** $i = 1..n$
  compute $\mu^{\mathbf{i}}(q, x, s)$ according to (6-7)
 **end**
 compute $\mu(q, x, s)$ ($\max$ of random variables)
 compute $V(q, x, s)$ (expected makespan)
**end**
% OPTIMIZE:
**forall** $x \in Z_q$
 $V(q, x) = \min_s(V(q, x, s))$
 $s_* = \arg\min_s V(q, x, s)$
 $\Omega(q, x) = s_*$
**end**
% UPDATE:
**for** $i = 1..n$
 $\mu^{\mathbf{i}}(q, x) = \mu^{\mathbf{i}}(q, x, s_*)$
**end**

shall see, starting from a specific class of time densities, this computation can be done in a symbolic/analytic way, resulting in *closed-form expressions* over $x$ and $t$ for the values of $\mu^{\mathbf{i}}$, $\mu$ and $V$ associated with each action. The second part involves *optimization*: to classify extended states according to the action that optimizes $V$ in them. For this part we prove a monotonicity property which facilitates the task of approximating the boundaries between the optimality domains of the various actions.

Each of the functions we have defined is in fact an infinite family of functions parameterized by a state $q$ and clock valuation $x$ ranging over the rectangle $Z_q$. They can be characterized as follows.

**Definition 15 (Zone-Polynomial Time Densities).** *A function $\mu : Z \to (\mathbb{R}_+ \to [0, 1])$ over a rectangular clock space $Z$ is zone-polynomial if it can be written as*

$$\mu(x^{\mathbf{1}}, \ldots, x^{\mathbf{n}})[t] = \begin{cases} f_1(x^{\mathbf{1}}, \ldots, x^{\mathbf{n}})[t]) \text{ if } Z_1(x^{\mathbf{1}}, \ldots, x^{\mathbf{n}}) \text{ and } l_1 \leq t \leq u_1 \\ f_2(x^{\mathbf{1}}, \ldots, x^{\mathbf{n}})[t]) \text{ if } Z_2(x^{\mathbf{1}}, \ldots, x^{\mathbf{n}}) \text{ and } l_2 \leq t \leq u_2 \\ \ldots \\ f_L(x^{\mathbf{1}}, \ldots, x^{\mathbf{n}})[t]) \text{ if } Z_L(x^{\mathbf{1}}, \ldots, x^{\mathbf{n}}) \text{ and } l_L \leq t \leq u_L \end{cases}$$

*where*

- *For every $r$, $Z_r(x^{\mathbf{1}}, ..., x^{\mathbf{n}})$ is a zone included in the rectangle $Z$, which moreover satisfies either $Z_r \subseteq [x^{\mathbf{i}} \leq a^{\mathbf{i}}]$ or $Z_r \subseteq [a^{\mathbf{i}} \leq x^{\mathbf{i}}]$, for every $i = 1..n$.*

– *For every $r$, the bounds $l_r$, $u_r$ of the $t$ interval are either nonnegative integers $c$ or terms of the form $c - x^{\mathbf{i}}$, with $i = 1..n$, $c \in \mathbb{Z}^+$. Moreover, the interval $[l_r, u_r]$ must be consistent with the underlying zone, that is, $Z_r \subseteq [l_r, u_r]$.*
– *For every $r$, $f_r(x^{\mathbf{1}}, ..., x^{\mathbf{n}})[t] = \sum_k \frac{P_k(x^{\mathbf{1}},...,x^{\mathbf{n}})}{Q_r(x^{\mathbf{1}},...,x^{\mathbf{n}})} t^k$ where $P_k$ are arbitrary polynomials and $Q_r$ is a characteristic polynomial associated with zone $Z_r$ defined as $\prod_i (b^{\mathbf{i}} - \max\{x^{\mathbf{i}}, a^{\mathbf{i}}\})$. Note that for each zone, the $\max$ is attained uniformly as either $a^{\mathbf{i}}$ or $x^{\mathbf{i}}$.*

**Theorem 1 (Closure of Zone-Polynomial Densities).** *Zone-polynomial time densities are closed under (6) and (7).*

**Sketch of Proof**: Operation 6 is a simple substitution. Closure under summation is also evident - you just need to refine the partitions associated with the summed functions and make them compatible and then apply the operation in each partition block. The only intricate operation is the quasi-convolution part of (7). The function $\mu^{\mathbf{i}}(\sigma^{\mathbf{i}'}(t', q, x))[t - t']$ is *not* a zone polynomial time density. Due to the time progress by $t'$ enforced by the substitution $\sigma^{\mathbf{i}'}$ it might happen that polynomials of the form $(b_i - (x^{\mathbf{i}} - t'))$ appear in the denominators. But, in all feasible cases, they will be simplified through multiplication by $\rho^{\mathbf{i}'}(q, x)[t']$ which contains the same polynomials as factors (within $\psi^{\mathbf{i}}{}_{/x^{\mathbf{i}}}[\geq t']$, see Def. 13). Hence, integration of $t'$ is always trivially completed as $t'$ occurs only on numerator polynomials and/or powers of the form $t'^k$ and $(t - t')^k$. Moreover, after integration, the remaining constraints on $t$ and $x$ can also be rewritten to match the required form of zone-polynomial time densities. ⌟

Next we move to the *optimization* part of the iteration. Consider a state $q$ where process $P^{\mathbf{i}}$ has to decide whether or not to start a step while other processes are active in their respective steps. The optimal strategy $\Omega$ in this state partitions the clock space of the other processes into $\Omega^{-1}(s^{\mathbf{i}})$ and $\Omega^{-1}(\mathbf{w})$, extended states where waiting or starting are preferred. The boundary corresponds to the set of solutions of the piecewise-polynomial equation $V(q, x, s) = V(q, x, \mathbf{w})$. Our approach is to *approximate* the optimal strategy based on sampling: we cut the clock space $Z_q$ into an $\varepsilon$-grid and for each grid point $x$ we compare $V(q, x, s)$ and $V(q, x, \mathbf{w})$ and select the optimal value (Fig. 3-(a)). Once the optimal action has been computed for all grid points we complete the strategy for the rest of the clock-space by selecting in each $\varepsilon$-cube the action which is optimal for its leftmost corner (Fig. 3-(b)). To estimate the deviation from the optimal strategy we first bound the derivative of $V$ with respect to any of the clocks.

**Lemma 1 (Derivative of $V$).** *Let $V$ be the value function associated with a problem. Then for every $(q, x)$ and for every $i$* $\dfrac{\partial V}{\partial x^{\mathbf{i}}}(q, x) \geq -1$

**Sketch of Proof**: Consider first the local value function of a process in isolation. In any state $q$, the clock space splits into two parts. When $x < a$, the derivative is naturally $-1$. When $x > a$ the rate of progress is slower (because progress is combined with accumulation of optimism-contradicting information) and the magnitude of the deriva-

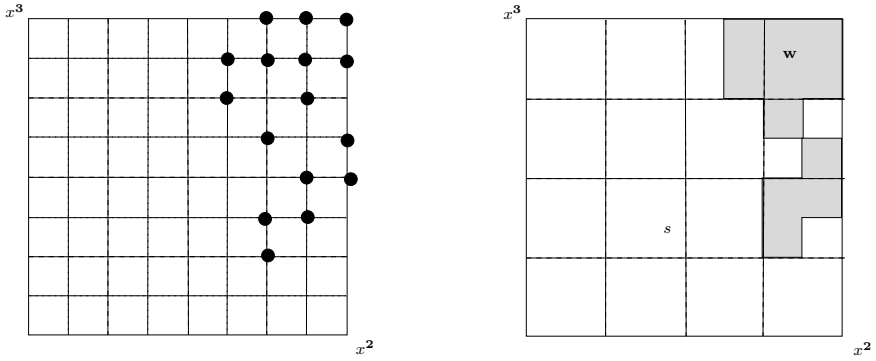



**Fig. 3.** Approximating the optimal action for $P^{\mathbf{1}}$ in the clock space of $P^{\mathbf{2}}||P^{\mathbf{3}}$: (a) Computing the optimal action for all grid points with the dark circle indicating $\Omega^{-1}(\mathbf{w})$; (b) Approximating the optimal strategy with the dark cubes indicating $\Omega'^{-1}(\mathbf{w})$.

tive is smaller.[4] When running together, the progress of each process is bounded by its progress in isolation or by the progress of another process that blocks it. The progress of the expected termination of the last process (makespan) is bounded by the progress of each of the processes. ⌟

**Lemma 2 (Approximation).** *Let $x'$ be a point in the clock space and let $x < x'$ be its nearest grid point on the left and hence $d(x, x') < \varepsilon$. Assume without loss of generality that the optimal strategy $\Omega$ satisfies $\Omega(q, x) = s$ and $\Omega(q, x') = \mathbf{w}$. Consider an approximate strategy $\Omega'$ such that $\Omega'(q, x') = s$, then the respective value functions of the strategies satisfy $V'(q, x') - V(q, x') \leq \varepsilon$.*

**Proof**: According to what we know about the optimal strategy we have

$$V(q, x', \mathbf{w}) < V(q, x', s) < V(q, x, s) < V(q, x, \mathbf{w})$$

and from Lemma 1, $V(q, x, s) - V(q, x', \mathbf{w}) \leq \varepsilon$ and so is $V(q, x', s) - V(q, x', \mathbf{w}) = V'(q, x') - V(q, x')$. ⌟

Before we state the consequent main result, we prove an important property of optimal strategies which can reduce the number of grid points for which the optimal strategy should be computed. This "non-laziness" property, formulated in the deterministic setting in [1], simply captures the intuition that preventing an enabled process from taking a resource is *not* useful *unless* some other process benefits from the resource during the waiting period.[5] The proof in a non-deterministic setting is more involved.

[4] The derivative of $V$ represents the progress toward the average duration, minus the growth of the average itself when $x > a$.

[5] Or, in scheduling under uncertainty, if some information gathered during the period has increased the expected time-to-go associated with waiting, which is impossible in our setting. Non-lazy schedules are also known as *active* schedules.

**Definition 16 (Laziness).** *A scheduling policy $\Omega$ is lazy at $(q, x)$ if $\Omega(q, x + t) = s^{\mathbf{i}}$ for some $i$ and $\Omega(q, x + t') = \mathbf{w}$ for every $t' \in [0, t)$. A schedule is non-lazy if no such $(q, x)$ exists.*[6]

**Theorem 2 (Non Lazy Optimal Schedulers).** *The optimal value $V$ can be obtained by a non-lazy scheduler.*

To prove the theorem we need a lemma whose proof can be found in the appendix.

**Lemma 3 (Value of Progress).** *Let $q$ be a state and let $x$ and $x'$ be two clock valuations which are identical except for ${x'}^{\mathbf{i}} = x^{\mathbf{i}} + \delta$. Then the value of $(q, x')$ is at least as good as the value of $(q, x)$, that is, $V(q, x') \leq V(q, x)$.*

The lemma can be wrongly interpreted as saying that always a more advanced state has a better value. This is true in general only for progress that does *not* induce a possibility of *blocking* other processes: advancing the clock of an already-active step or starting a step on a resource that has no other future users. Starting a step on a resource that has other users in its horizon is a type of progress which is outside the scope of the lemma.
**Proof of Theorem 2**: Imagine a strategy $\Omega$ which is lazy at $(q, x)$ and takes its earliest *start* at $(q, x+t)$, as illustrated in Fig. 4-(a). Following an alternative strategy that starts at $x$, we will find ourselves after $t$ time in a state where one clock is more advanced (Fig. 4-(b)) and hence satisfying the condition of Lemma 3. If we keep all instances of laziness partially ordered according to $\preceq$ and apply the above modification starting from the minimal elements of the state space, we gradually push the earliest laziness toward later states until it is completely eliminated. ⌟


**Fig. 4.** Proof of theorem: the state reached after starting at $x + t$ (a) is less advanced than after starting at $x$ (b).

To illustrate the limited scope of the lemma and theorem consider a task whose duration is characterized by a *discrete* probability with probability $p$ for $a$ and $1 - p$ for $b$. In this case, the value function associated with waiting is

$$V(x) = \begin{cases} p(a - x) + (1 - p)(b - x) & \text{when } x < a \\ 0(a - x) + 1(b - x) & \text{when } x > a \end{cases}$$

Here at $x = a$ there is a *jump* in $V$ from $(1 - p)(b - a)$ to $(b - a)$ which is, intuitively, due to the accumulation of information: after $a$ time, the non-occurrence of an *end* event

[6] This definition of laziness can be extended from a one-dimensional interval $[x, x + t]$ to any pair of clock valuations $x$ and $x'$ such that $x'$ is reachable from $x$ by a time step.

tells us that the duration is certainly $b$. Such a situation contradicts the lemma, because for $x < a < x'$ we may have $V(x') > V(x)$. This jump in the expected time-to-go for waiting can also justify laziness: when $x < a$ the expected value of waiting can be better than for starting, but after $x = a$ the relation between these values may change.

**Corollary 1 (Forward-Backward Closure).** *Let $(q, x)$ be an extended state such that $V(q, x, \mathbf{w}) < V(q, x, s)$ and hence the optimal scheduler satisfies $\Omega(q, x) = \mathbf{w}$. Then $\Omega(q, x') = \mathbf{w}$ for any $x' = x+t$. Likewise if $V(q, x, \mathbf{w}) > V(q, x, s)$ then $\Omega(q, x') = s$ for all $x' = x - t$.*

This property can be used to reduce the number of grid points for which the strategy should be computed: $\Omega(q, x) = \mathbf{w}$ implies $\Omega(q, x') = \mathbf{w}$ for any grid point of the form $x' = x + m\varepsilon$ and likewise if $\Omega(q, x) = s$, starting is also optimal for any $x' = x - m\varepsilon$. Using an adaptation of the multi-dimensional binary search algorithm of [16], originally devised to approximate Pareto fronts for multi-criteria optimization problems, one can complete the marking of grid points with less than $(1/\varepsilon)^n$ evaluations of the strategies.

**Theorem 3 (Main Result).** *Let $\Omega$ be the expected-time optimal scheduler whose value at the initial state is $V$. For any $\varepsilon$, one can compute a scheduler $\Omega'$ whose value $V'$ satisfies $V' - V \leq \varepsilon$.*

**Proof**: Apply Algorithm 1 while going backwards from the final state. In the optimization part use sampling with grid size $\varepsilon/nk$. The division by $nk$ is needed to tackle the (theoretical) possibility of approximation error accumulation along paths. ⌟

## 7 Discussion

To the best of our knowledge, this work presents the first automatic derivation of optimal controllers/schedulers for "non-Markovian" continuous-time processes. We have laid down the conceptual and technical foundation for treating *clock-dependent* time densities and value functions and investigated the properties of optimal schedulers. We list below some ongoing and future work directions:

1. The algorithm as described here works individually on each global state, which is a recipe for quick state explosion. A more sophisticated algorithm that works on the whole decision subspace associated with an action $s$ and taking advantage of forward/backward closure, will be much more efficient. Although the stronger property of upward-downward closure does not follow, unfortunately, from non-laziness, the multi-dimensional binary search algorithm of [16] could provide good approximations whose error analysis will require a more refined characterization of the partition induced by the optimal strategy.
2. Implementation and experimentation: we currently have a prototype implementation building upon the infra-structure developed in [12] for computing integrals over zones. Once completed, we intend to compare cost/quality tradeoffs of our algorithm with statistical methods such as [8, 13] that evaluate and synthesize schedulers based on sampling the duration space. Such experiments will determine whether symbolic techniques can be part of tools for design-space exploration [13].

3. Extending the result to cyclic systems will require a definition of the value associated with infinite runs and some new techniques for proving convergence to a fixed point in the spirit of [3]. Introducing stochasticity in task arrival will enrich queuing theory with the expressive advantage of automata.
4. One can think of alternative measures for evaluating the performance of the scheduler. For example, rather than considering the expected makespan ($\max$ over all processes) we can optimize the average termination time of all tasks, or even employ a multi-dimensional vector of termination times in the multi-criteria spirit.

**Acknowledgement**: The project benefitted from the support of the French ANR project EQINOCS and numerous useful comments given by E. Asarin.

## References

1. Y. Abdeddaïm, E. Asarin, and O. Maler. Scheduling with timed automata. *Theoretical Computer Science*, 354(2):272–300, 2006.
2. R. Alur and M. Bernadsky. Bounded model checking for GSMP models of stochastic real-time systems. In *HSCC*, pages 19–33, 2006.
3. E. Asarin and A. Degorre. Volume and entropy of regular timed languages: Analytic approach. In *FORMATS*, 2009.
4. E. Asarin and O. Maler. As soon as possible: Time optimal control for timed automata. In *HSCC*, pages 19–30, 1999.
5. E. Asarin, O. Maler, A. Pnueli, and J. Sifakis. Controller synthesis for timed automata. In *Proc. IFAC Symposium on System Structure and Control*, pages 469–474, 1998.
6. L. Carnevali, L. Grassi, and E. Vicario. State-density functions over DBM domains in the analysis of non-Markovian models. *IEEE Trans. Software Eng.*, 35(2):178–194, 2009.
7. C.G. Cassandras and S. Lafortune. *Introduction to Discrete Event Systems*. Springer, 2008.
8. A. David, K. Larsen, A. Legay, M. Mikučionis, and Z. Wang. Time for statistical model checking of real-time systems. In *CAV*, pages 349–355, 2011.
9. C. Daws, A. Olivero, S. Tripakis, and S. Yovine. The tool KRONOS. In *Hybrid Systems*, pages 208–219, 1995.
10. R. German. Non-markovian analysis. In E. Brinksma, H. Hermanns, and J.-P. Katoen, editors, *Euro Summer School on Trends in Computer Science*, pages 156–182, 2000.
11. P.W. Glynn. A GSMP formalism for discrete event systems. *Proceedings of the IEEE*, 77(1):14–23, 1989.
12. J.-F. Kempf, M. Bozga, and O. Maler. Performance evaluation of schedulers in a probabilistic setting. In *FORMATS*, pages 1–15, 2011.
13. Jean-Francois Kempf. *On Computer-Aided Design-Space Exploration for Multi-Cores*. PhD thesis, University of Grenoble, October 2012.
14. K.G. Larsen, G. Behrmann, E. Brinksma, A. Fehnker, T. Hune, P. Pettersson, and J. Romijn. As cheap as possible: Efficient cost-optimal reachability for priced timed automata. In *CAV*, 2001.
15. K.G Larsen, P. Pettersson, and W. Yi. UPPAAL in a nutshell. *International Journal on Software Tools for Technology Transfer (STTT)*, 1(1):134–152, 1997.
16. J. Legriel, C. Le Guernic, S. Cotton, and O. Maler. Approximating the Pareto front of multi-criteria optimization problems. In *TACAS*, 2010.
17. O. Maler. On optimal and reasonable control in the presence of adversaries. *Annual Reviews in Control*, 31(1):1–15, 2007.
18. O. Maler, K.G. Larsen, and B.H. Krogh. On zone-based analysis of duration probabilistic automata. In *INFINITY*, pages 33–46, 2010.

## Appendix: Proof of Lemma 1

**Lemma 1 (Value of Progress)** *Let $q$ be a state and let $x$ and $x'$ be two clock valuations which are identical except for $x'^{\mathbf{i}} = x^{\mathbf{i}} + \delta$. Then the value of $(q, x')$ is at least as good as the value of $(q, x)$, that is, $V(q, x') \leq V(q, x)$.*

**Sketch of Proof**: We prove it by induction on the number of transitions remaining between $q$ and the final state.
*Base case*: When there is only one *end* transition pending and one active clock working toward a duration characterized by a time density $\psi$, the value is defined as

$$V(q,x) = \int_{\max(x,a)}^{b} \psi_{/x}(y)(y-x)dy = \begin{cases} (a+b)/2 - x & \text{when } x \leq a \\ (b-x)/2 & \text{when } x \geq a \end{cases}$$

and the derivative $dV/dx$ is negative ($-1$ and then $-1/2$). Hence $V(q, x') < V(q, x)$.
*Inductive case*: suppose the lemma holds for all states beyond $q$. We will show that each action taken in $(q, x')$ can lead to a state at least as advanced as the state reached via the same action from $(q, x)$. For an immediate *start* transition this is evident. Suppose we take action $\mathbf{w}$ at both $(q, x')$ and $(q, x)$. Then there are three possibilities: if the race winner at $(q, x')$ is some $P^{\mathbf{i}'}$, $i' \neq i$, taking an $e^{\mathbf{i}'}$ transition to $q'$, then the same $P^{\mathbf{i}'}$ will win the race from $(q, x)$ within the same time, landing in $q'$ with value of $x^{\mathbf{i}}$ less advanced by $\delta$ and the inductive hypothesis holds, see Figure 5-(a) and the corresponding runs $\xi_x$ and $\xi_{x'}$, depicted below projected on the two relevant clocks:

$$\xi_{x'} : (x^{\mathbf{i}} + \delta, x^{\mathbf{i}'}) \xrightarrow{t} (x^{\mathbf{i}} + \delta + t, x^{\mathbf{i}'} + t) \xrightarrow{e^{\mathbf{i}'}} (x^{\mathbf{i}} + \delta + t, \perp)$$

$$\xi_{x} : (x^{\mathbf{i}}, x^{\mathbf{i}'}) \xrightarrow{t} (x^{\mathbf{i}} + +t, x^{\mathbf{i}'} + t) \xrightarrow{e^{\mathbf{i}'}} (x^{\mathbf{i}} + t, \perp).$$

Suppose, on the contrary, that $P^{\mathbf{i}}$, the process whose clock is more advanced in $x'$ than in $x$, wins the race from $(q, x')$ within $t$ time. If $P^{\mathbf{i}}$ is also the winner from $(q, x)$, the $e^{\mathbf{i}}$ transition will be taken by $\xi_x$ after $t + \delta$ time. By waiting at $q'$ for $\delta$ time, run $\xi_{x'}$ can reach the same state as $\xi_x$, see Figure 5-(b) and the corresponding runs:

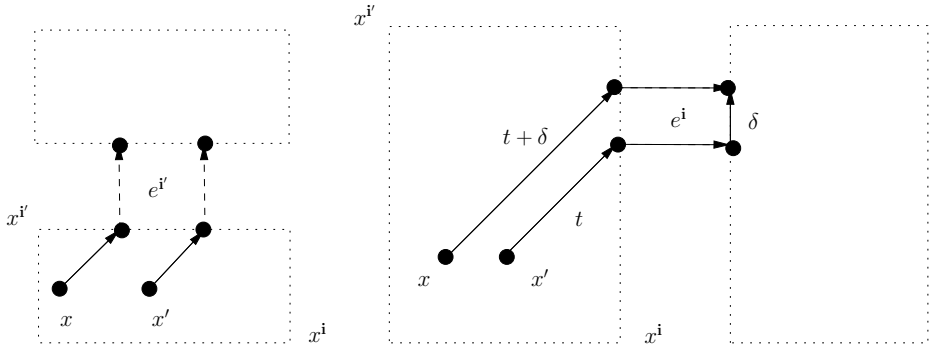
$$\xi_{x'} : (x^{\mathbf{i}} + \delta, x^{\mathbf{i}'}) \xrightarrow{t} (x^{\mathbf{i}} + \delta + t, x^{\mathbf{i}'} + t) \xrightarrow{e^{\mathbf{i}}} (\perp, x^{\mathbf{i}'} + t) \xrightarrow{\delta} (\perp, x^{\mathbf{i}'} + t + \delta)$$

$$\xi_{x} : (x^{\mathbf{i}}, x^{\mathbf{i}'}) \xrightarrow{t+\delta} (x^{\mathbf{i}} + \delta + t, x^{\mathbf{i}'} + \delta + t) \xrightarrow{e^{\mathbf{i}}} (\perp, x^{\mathbf{i}'} + t + \delta).$$

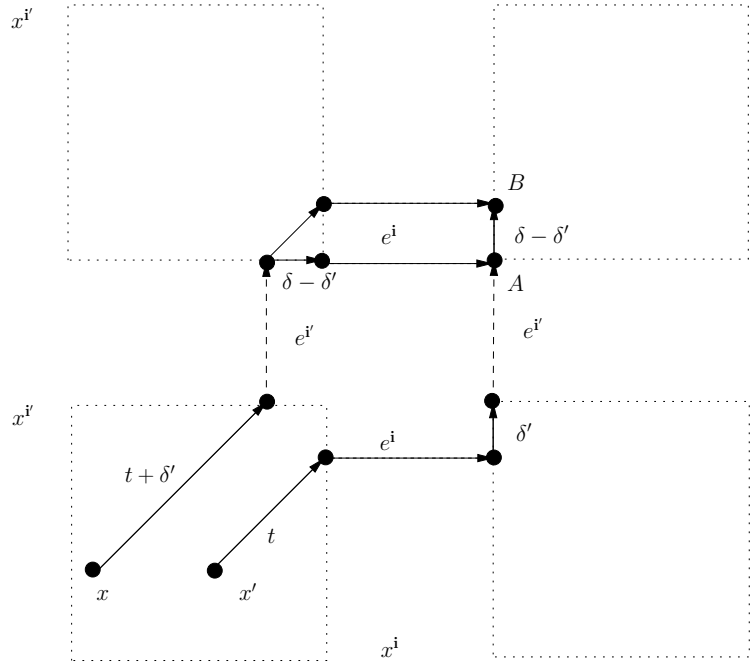
Finally, suppose the race winner from $(q, x)$ is $P^{\mathbf{i}'}$ within some $t + \delta'$ time, $\delta' < \delta$, leading to state $q''$. This splits into two sub-cases. If at $q''$ the run $\xi_x$ does not start a new step, then, by waiting at $q'$ for $\delta - \delta'$ time, $\xi_{x'}$ can reach within $t + \delta'$ the same extended state that $\xi_x$ has reached in more time $t + \delta$:

$$\xi_{x'} : (x^{\mathbf{i}} + \delta, x^{\mathbf{i}'}) \xrightarrow{t} \xrightarrow{e^{\mathbf{i}}} (\perp, x^{\mathbf{i}'} + t) \xrightarrow{\delta'} (\perp, x^{\mathbf{i}'} + t + \delta') \xrightarrow{e^{\mathbf{i}'}} (\perp, \perp)$$

$$\xi_{x} : (x^{\mathbf{i}}, x^{\mathbf{i}'}) \xrightarrow{t+\delta'} \xrightarrow{e^{\mathbf{i}'}} (x^{\mathbf{i}} + \delta' + t, \perp) \xrightarrow{\delta-\delta'} (x^{\mathbf{i}} + \delta + t, \perp) \xrightarrow{e^{\mathbf{i}}} (\perp, \perp).$$




(a) (b)




(c)

**Fig. 5.** Proof of lemma: (a) $P^{\mathbf{i}'}$ wins from both $x$ and $x'$; (b) $P^{\mathbf{i}}$ wins from both. (c) $P^{\mathbf{i}}$ wins from $x'$ and $P^{\mathbf{i}'}$ wins from $x$. Here either $\xi_{x'}$ reaches point $A$ in less time than does $\xi_x$, or both reach point $B$ at the same time.

If, on the other hand, during the interval of duration $\delta - \delta'$ in which $\xi_x$ catches up with the progress of $\xi_{x'}$ in $P^{\mathbf{i}}$, $\xi_x$ makes an $s^{\mathbf{i}'}$ transition to start a next step of $P^{\mathbf{i}'}$, so can $\xi_{x'}$ and they will reach the same state:

$$\xi_{x'} : (x^{\mathbf{i}} + \delta, x^{\mathbf{i}'}) \xrightarrow{t} \xrightarrow{e^{\mathbf{i}}} (\bot, x^{\mathbf{i}'} + t) \xrightarrow{\delta'} (\bot, x^{\mathbf{i}'} + t + \delta') \xrightarrow{e^{\mathbf{i}'}} \xrightarrow{s^{\mathbf{i}'}} (\bot, 0) \xrightarrow{\delta - \delta'} (\bot, \delta - \delta')$$

$$\xi_x : (x^{\mathbf{i}}, x^{\mathbf{i}'}) \xrightarrow{t+\delta'} \xrightarrow{e^{\mathbf{i}'}} \xrightarrow{s^{\mathbf{i}'}} (x^{\mathbf{i}} + \delta' + t, 0) \xrightarrow{\delta - \delta'} (x^{\mathbf{i}} + \delta + t, \delta - \delta') \xrightarrow{e^{\mathbf{i}}} (\bot, \delta - \delta').$$

These two cases are illustrated in Figure 5-(c).

The bottom line of this case analysis is that for every action taken in $(q, x)$ and $(q, x')$ and for every $y$, there exist durations $t_y$ and $t'_y$ such that $0 \leq t'_y \leq t_y$, a state $p_y$ such that $q \prec p_y$ and clock valuations $z_y$ and $z'_y$ such that $z_y \leq z'_y$ in the value of one clock (as in the premise of the lemma) and

$$(q, x) \xrightarrow{t_y} (p_y, z_y) \quad \text{and} \quad (q, x') \xrightarrow{t'_y} (p_y, z'_y).$$

The cost to go from $(q, x)$ can be written as

$$V(q, x) = \int \psi(y) \cdot (t_y + V(p_y, z_y)) dy$$

and since for every $y$, we have $t'_y \leq t_y$ and, moreover, following the inductive hypothesis $V(p_y, z'_y) \leq V(p_y, z_y)$, we can conclude that $V(q, x') \leq V(q, x)$. ∎