Performance Evaluation of Schedulers in a Probabilistic Setting

Jean-Francois Kempf, Marius Bozga, and Oded Maler

CNRS-VERIMAG
University of Grenoble
France
@imag.fr

Abstract. We show how to evaluate the performance of solutions to finite-horizon scheduling problems where task durations are specified by bounded *uniform* distributions. Our computational technique, based on computing the volumes of zones, constitutes a contribution to the computational study of scheduling under uncertainty and stochastic systems in general.

1 Introduction

Scheduling, the allocation of limited reusable resources over time to competing tasks, is a universal activity. It is performed routinely in domains of very different scales in terms of time, space and energy. These include the allocation of airways and runways to flights, allocating machines to different product lines in a factory, and the efficient allocation of computation and communication resources to information-processing tasks. This latter activity is becoming of prime importance in many scales, ranging from world-wide cloud computing, via the realization of multiple distributed control loops, down to mapping and scheduling tasks on multi-core computers. In all such situations one wants to synthesize schedulers which are optimal or good in some sense, or at least to be able to compare the performance of proposed schedulers and choose the better ones. Performance and optimality of such schedulers are typically based on the quantity of work performed over time, which in the case of a finite amount of work can be expressed as *termination* time. Good schedules are typically associated with intensive, almost idle-free, utilization of critical bottleneck resources.

In a *deterministic* setting one assumes that everything is known *in advance* about the demand for work, including the tasks to be executed, their arrival times and the durations for which they occupy resources. In other words, once the scheduling policy itself is determined, the system admits a *unique* execution scenario (run, realization). Evaluating a scheduler based on this unique run is straightforward – just simulate it – while finding an optimal scheduler for any non-trivial scheduling problem (such as job-shop) is NP-hard or worse. However, determinism is rarely the case in real life and exact duration of tasks, their arrival times and many other features may vary to large extents. Each instance in this uncontrollable space yields a *different schedule* and the overall evaluation of a scheduler or a scheduling policy, which can be viewed as a strategy in a two-person timed game [7] with uncertainty viewed as an adversary, should be based on some *quantification* over all possible behaviors it induces [28].

This adversarial time-optimality problem has been tackled in [6, 1] using a *worst-case* approach on models of different types of uncertainty. In [6], using, the general model of *timed game automaton* [7] where the adversary is discrete, the following problem was proved to be decidable: synthesize a controller which is worst-case time-optimal in the sense that the maximal (over all possible runs induced by the adversary) time to reach a goal state is minimal. In [1] the case of job-shop scheduling with uncertain task *durations* each ranging over a bounded interval was treated. For this problem, worst-case optimality is defined trivially by the optimal solution to a *deterministic* scheduling problem associated with the worst case where all tasks take their respective maximal duration. One has to define a new notion of optimality (d-future optimal strategies) to make the optimal synthesis problem meaningful, resulting in a synthesis algorithm based on value iteration over sets of clock valuations (zones) which can be seen as an *offline* version of some kind of *model-predictive control*.

The use of worst-case reasoning is to some extent a residue of the safety-critical banner under which formal verification has been argued for, but in many (if not most) real-life situations, temporal uncertainty is modeled probabilistically as a distribution over the durations of each task and scheduler quality is measured accordingly, for example by the *expected* completion time or by its maximum over all but a small fraction of the runs. In this paper we develop and implement a computational framework in order to compute the performance of such schedulers, modeled by automata similar in structure to those used in [1] but whose durations are probabilistic. Such automata are sufficiently rich to express stochastic variants of well-known scheduling problems such as job-shop or task-graph. Formal definitions of these *duration probabilistic automata* and their semantics can be found in [29].

The study of continuous-time stochastic processes has been going on for many years in other branches of mathematics where simple computational questions like those we pose are not typically asked, as well as in closer domains such as probabilistic verification and performance evaluation [13, 11]. A well-studied class of such processes are *continuous-time Markov chains* (CTMC) where durations are distributed *exponentially*. Such distributions are memoryless in the sense that time spent waiting for a task to terminate does not influence the distribution on the remaining time. As a result they are easy to compute with and problems such as model-checking against qualitative [3] and quantitative [8] temporal properties or optimal controller synthesis for finite-horizon problems [1] are well understood. This forgetfulness assumption may be realistic and useful for modeling request arrivals in queuing models, but seems inappropriate for modeling the durations of several instances of the same computational task.[1]

In this paper we assume task durations to be *uniform* over a *bounded* interval, which is a natural "stochastization" of the set-theoretic temporal uncertainty of timed automata. Handling such systems we find ourselves in the realm of the so-called *generalized semi-Markov processes* (GSMP), a class of continuous-time stochastic processes [21, 22, 15, 25]. Similar computational studies of GSMPs include [2, 10], [14, 30] and [29]. The former are concerned with verifying temporal properties for some classes of GSMPs and develop techniques to determine whether the probability of a

[1] The academic paper industry is perhaps the prime example of an application domain where this hypothesis is useful.

property-violating behavior is zero. The work of [14, 30] is concerned with stochastic Petri nets for which a computational framework is developed for propagating densities in the marking graph. This work, as well as [29] on duration probabilistic automata, use densities on clocks which are auxiliary state variables. At each reachable state and zone in the clock space, the distribution over clock values is maintained and used to compute the distribution after the next transition. In contrast, the approach presented in this paper works directly on the space of the *duration* random variables and does not use clocks explicitly. Similar ideas were developed in [24] to compute the probability of test cases in timed systems.

The rest of the paper is organized as follows. Section 2 defines single and parallel processes, their behaviors (timed and qualitative) and presents a useful coordinate transformation between durations and time stamps. Section 3 shows how to derive the timing constraints associated with a qualitative behavior when processes execute independently without resource conflicts, and how to compute the volumes of the polytopes they define. Section 4 extends the framework to the more interesting case of resource conflicts that have to be resolved by dynamic scheduling strategies and presents very preliminary experimental results. A discussion of future directions concludes the paper.

2 Preliminaries

We consider a composition $S = P^{\mathbf{1}}||\cdots||P^{\mathbf{n}}$ of n *sequential stochastic processes*, each consisting of a sequence of steps. Each step has a probabilistic duration and cannot start before its predecessor terminates. We consider two execution frameworks:

1. *Independent execution*: all processes start simultaneously and each process starts a step as soon as its preceding step has terminated, regardless of the state of other processes;
2. *Coordinated execution*: the initiation of a step is controlled by a *scheduler* which may hold a step of one process in a waiting state until the termination of a step of another process that uses the same resource.

The second framework will allow us to compare schedulers but we start with the first because it is simpler, does not require knowledge of timed automata and hence is accessible to a wider audience. On this simpler model we will develop the basic computational machinery that will allow us to compute the probabilities of different *qualitative behaviors*, each corresponding to an equivalence class of timed behaviors associated with a particular *order* in which steps of different processes terminate.

Definition 1 (Uniform Distribution). *A* uniform *distribution inside an interval* $I = [a, b]$ *is characterized by a density* ψ *defined as*

$$\psi(y) = \begin{cases} 1/(b-a) & \textit{if } a \leq y < b \\ 0 & \textit{otherwise} \end{cases}$$

and in terms of distribution as

$$F(y) = \int_0^y \psi(\tau) d\tau = \begin{cases} 0 & \textit{if } y < a \\ (y-a)/(b-a) & \textit{if } a \leq y \leq b \\ 1 & \textit{if } b \leq y \end{cases}$$

Definition 2 (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 ψ_j being the uniform density over $I_j = [a_j, b_j]$, indicating the duration of step j.*

We consider *finite* processes with $K = \{1, \ldots, k\}$. 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 vectors $y = (y_1, \ldots, y_k)$ ranging over a *duration space*

$$D = I_1 \times \cdots \times I_k \subseteq \mathbb{R}^k$$

with density $\psi(y_1, \ldots, y_k) = \psi_1(y_1) \cdots \psi_k(y_k)$. Each point y in the duration space induces a *unique* behavior of the system written as a *time-event sequence* of the form

$$\xi_y = y_1\ e_1\ y_2\ e_2\ \cdots\ y_k\ e_k. \tag{1}$$

Time event sequences are alternations between time elapses represented by real numbers and discrete events that take no time. In the case of a single process $y_j \in I_j$ is the *duration* of step j and e_j is the *event* of terminating that step. The *timed language*[2] associated with the process consists of all the timed behaviors it may generate, namely $L = \{\xi_y : y \in D\}$. The *untimed language* associated with the process is $\underline{L}$, obtained by projecting away durations and retaining events and their order. In the case of a single process $\underline{L}$ is simply the singleton language $\{w\}$ where $w = e_1\ e_2 \cdots\ e_k$.

Mechanically speaking the process behaviors can be viewed as generated by the automaton of Fig. 1 in which being at state q_j corresponds to executing step j. Each run of the automaton is associated with a point y in the duration space. Upon entering q_j an auxiliary clock variable x is reset to zero and the termination transition labeled by e_j is taken exactly when $x = y_j$.

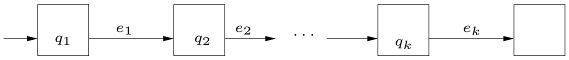

Fig. 1. An automaton view of a process.

Suppose we want to characterize the probability of a certain subset of L. For example those behaviors in which for every j the actual duration of step j is in some sub-interval $I'_j = [a'_j, b'_j] \subseteq I_j$. The total probability of these behaviors is simply the *volume* of the rectangle $I'_1 \times \cdots \times I'_k$ divided by the volume of the whole rectangle D. Probabilities of other subsets of the language can be more interesting but harder to compute. For example, the probability that the whole process terminates before some deadline r is simply the volume of the subset of D satisfying $y_1 + \cdots + y_k < r$ divided by the volume of D. Our technique is based on computing such volumes for a system of several parallel processes as described in the sequel.

[2] In the computer science tradition the term *language* is often used to denote a set of sequences or other objects that define dynamic behaviors.

It turns out to be easier to compute volumes after a coordinate transformation from the space of durations to the space of *time stamps* consisting of vectors $t = (t_1, \ldots, t_k)$ where t_j is the absolute occurrence time of event e_j, defined as $t_j = y_1 + y_2 + \cdots + y_j$. A behavior ξ_y can thus be written also as a sequence of time-stamped events[3]

$$\xi_t = (e_1, t_1), (e_2, t_2), \ldots, (e_k, t_k).$$

Assuming that all durations admit a positive lower bound $a_j > 0$, all time stamps satisfy *precedence constraints* of the form $t_j < t_{j+1}$.

Converting y to t and vice versa is done by the linear transformations $t = Ty$ and $y = T't$ where T and T' are matrices of the form

$$T = \begin{pmatrix} 1\,0\,0\,0 \\ 1\,1\,0\,0 \\ 1\,1\,1\,0 \\ 1\,1\,1\,1 \end{pmatrix} \quad T' = \begin{pmatrix} 1 & 0 & 0 & 0 \\ -1 & 1 & 0 & 0 \\ 0 & -1 & 1 & 0 \\ 0 & 0 & -1 & 1 \end{pmatrix}$$

These matrices are lower triangular (the value of t_j cannot depend on a duration $y_{j'}$ with $j' > j$) and their diagonal entries are equal to 1. The determinant of a triangular matrix is equal to the product of the diagonal entries which is 1 and hence the transformations are *volume preserving*. This means that the volume of the duration space D is equal to the volume of the *time-stamp space* C defined by the constraints

$$\varphi_C : \bigwedge_{j \in K} a_j \leq t_j - t_{j-1} \leq b_j$$

and computing the volume of any subset $C' \subseteq C$ amounts to computing the volume of its T' image $D' \subseteq D$. Let us remark that the density of t_j is the *convolution* of the densities $\psi_1, \ldots, \psi_j$ and its support is the Minkowski sum of $I_1, \ldots, I_j$.

The time-stamp space C and its subsets that we will encounter are defined as conjunctions of inequalities of the form $x \prec c$ or $x - x' \prec c$ where $\prec \in \{<, \leq, =, \geq, >\}$ and c is an integer constant. They define polytopes which are called *zones* (or timed polyhedra). Zones are used extensively in the analysis of timed automata [23, 17, 26]. They admit an efficient representation by *difference-bound matrices* (DBM) [19] and efficient algorithms based on shortest-path to remove redundant constraints [16].

Definition 3 (Process System). *A process system consists of n processes*

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

We use notations $P^{\mathbf{i}}_j$ to refer to step j of process i and $I^{\mathbf{i}}_j = [a^{\mathbf{i}}_j, b^{\mathbf{i}}_j]$ and $\psi^{\mathbf{i}}_j$ for the respective intervals and densities. To ease notation we assume all processes to have the same number k of steps. The event alphabet of the system is

$$\Sigma = \{e^{\mathbf{1}}_1, e^{\mathbf{1}}_2, \ldots, e^{\mathbf{n}}_{k-1}, e^{\mathbf{n}}_k\}$$

[3] These are the *timed traces* used originally in [4] to give semantics to timed automata. More about the relation between semantic models of timed behaviors can be found in [5].

consisting of all the termination events of the steps of the various processes.

A behavior of the system is induced by a point in the global duration space

$$y = (y_1^{\mathbf{1}}, y_2^{\mathbf{1}}, \ldots, y_{k-1}^{\mathbf{n}}, y_k^{\mathbf{n}}) \in \mathcal{D} = \prod_{i=1}^{n} \prod_{j=1}^{k} I_j^{\mathbf{i}} \subset \mathbb{R}^{nk},$$

which can be transformed into a point t in the time-stamp space

$$t = (t_1^{\mathbf{1}}, t_2^{\mathbf{1}}, \ldots, t_{k-1}^{\mathbf{n}}, t_k^{\mathbf{n}}) \in \mathcal{C} = T\mathcal{D}$$

where T is the appropriate block diagonal matrix.

When all processes start simultaneously, the time stamps are taken from the same global time reference and one can view a global run as merging local runs and sorting the events according to their time stamps, as illustrated in Fig. 2. The set of all such global behaviors is denoted by

$$L = L^{\mathbf{1}} || \cdots || L^{\mathbf{n}}.$$

All timed behaviors that admit the same *order* of events are said to exhibit the same *qualitative behavior*. This can be formalized as an operation among the untimed local languages. Let $\underline{L}^{\mathbf{i}} = \{e_1^{\mathbf{i}}\, e_2^{\mathbf{i}} \cdots e_k^{\mathbf{i}}\}$ be the untimed language associated with process $P^{\mathbf{i}}$: it consists of the unique qualitative behavior which satisfies the precedence constraints of $P^{\mathbf{i}}$. The potential qualitative behaviors of S constitute the language

$$\underline{L} = \underline{L}^1 || \cdots || \underline{L}^n$$

which is the *shuffle* of these languages, that is, the set of sequences consisting of one occurrence of each event in Σ and respecting the local precedence constraints for each process. Mathematically speaking, a qualitative behavior corresponds to a linear order[4] which is consistent with the partial order defined by the union of the precedence relations of all the tasks. Such an order is also known as *interleaving* in the theory of *concurrency* (motivated readers might want to consult [18] or [20]).

We use the term qualitative behavior also for any *prefix* of a sequence in $\underline{L}$. Such a prefix corresponds naturally to an *incomplete* run where not all processes have finished all their steps. From the standpoint of automata, qualitative behaviors correspond to *paths* in the transition graph of the *global automaton* associated with the system which is the (Cartesian) product $\mathcal{A} = \mathcal{A}^{\mathbf{1}} || \cdots || \mathcal{A}^{\mathbf{n}}$ of the automata associated with the individual processes as illustrated in Fig. 3. Unfortunately, these extremely important objects are not easy to draw for non-trivial dimensions. Incomplete behaviors correspond to paths not reaching the final state.

In a global *state* of the form $(q_{j_1}^{\mathbf{1}}, \ldots, q_{j_n}^{\mathbf{n}})$ each process i is busy executing its step j_i and there is a *race* between the termination transitions. The transition $e_{j_i}^{\mathbf{i}}$ that will win will be the first to satisfy the condition $x^{\mathbf{i}} = y_{j_i}^{\mathbf{i}}$. Since $x^{\mathbf{i}}$ has been reset to zero

[4] Since we are dealing with volumes, our neglect of the possibility of events occurring at *exactly* the same time and not paying too much attention to the distinction between strict and non strict inequalities is justified.

at $t^{\mathbf{i}}_{j_i-1}$ this condition will be fulfilled at time $t^{\mathbf{i}}_{j_i-1} + y^{\mathbf{i}}_{j_i} = t^{\mathbf{i}}_{j_i}$. The outcomes of all these races are completely determined by the value of y, and this determines the qualitative behavior which is exhibited. Had there been no timing constraints on task durations, that is, $I^{\mathbf{i}}_j = [0, \infty)$, the system would be completely *asynchronous* and all interleavings would, in principle, be possible. When durations are bounded, some qualitative behaviors may become strictly impossible due to the arithmetics of timing constraints while others will occur at low probability. In the sequel we develop methods for computing these probabilities.

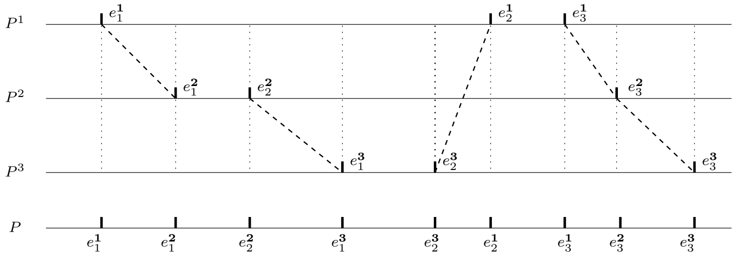

Fig. 2. A global behavior $w = e^{\mathbf{1}}_1\ e^{\mathbf{2}}_1\ e^{\mathbf{2}}_2\ e^{\mathbf{3}}_1\ e^{\mathbf{2}}_3\ e^{\mathbf{1}}_2\ e^{\mathbf{1}}_3\ e^{\mathbf{3}}_2\ e^{\mathbf{3}}_3$ obtained by merging local behaviors. The dashed line indicate the minimal set of additional inter-process constraints that characterize w.

3 Computing Volumes

The computation of the probability of a qualitative behavior w is performed in two steps. First we associate with it a zone $Z_w \subseteq \mathcal{C}$ consisting of all instances of t that yield this behavior. Then we integrate over this zone to find its volume.

Let $\varphi_{\mathcal{C}}$ be the constraint describing the whole time-stamp space:

$$\varphi_{\mathcal{C}} : \quad \bigwedge_{i \in N} \bigwedge_{j \in K} a^{\mathbf{i}}_j \leq t^{\mathbf{i}}_j - t^{\mathbf{i}}_{j-1} \leq b^{\mathbf{i}}_j$$

with $t^{\mathbf{i}}_0 = 0$ for every i. The zone Z_w for the qualitative behavior of Fig. 2 can be characterized by adding constraints that specify the particular order of events in w:

$$\varphi_w : \ \varphi_{\mathcal{C}} \wedge \ t^{\mathbf{1}}_1 < t^{\mathbf{2}}_1 < t^{\mathbf{2}}_2 < t^{\mathbf{3}}_1 < t^{\mathbf{2}}_3 < t^{\mathbf{1}}_2 < t^{\mathbf{1}}_3 < t^{\mathbf{3}}_2 < t^{\mathbf{3}}_3.$$

Some of these constraints appear already in $\varphi_{\mathcal{C}}$ and some are implied via transitivity by other constraints. After eliminating these redundant constraints one obtains the following description:

$$\varphi_w : \varphi_{\mathcal{C}} \wedge (t^{\mathbf{1}}_1 < t^{\mathbf{2}}_1) \wedge (t^{\mathbf{2}}_2 < t^{\mathbf{3}}_1) \wedge (t^{\mathbf{3}}_2 < t^{\mathbf{1}}_2) \wedge (t^{\mathbf{1}}_3 < t^{\mathbf{2}}_3) \wedge (t^{\mathbf{2}}_3 < t^{\mathbf{3}}_3).$$

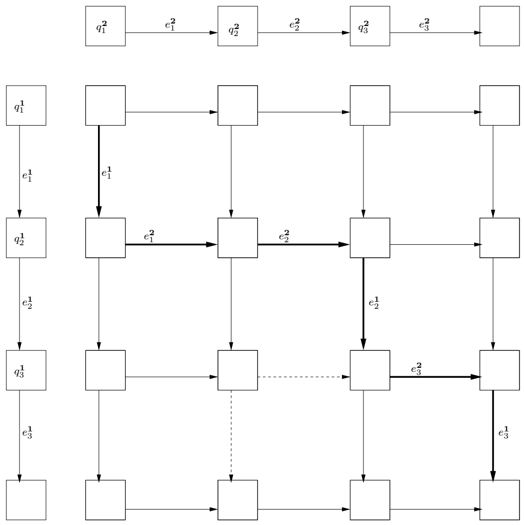

Fig. 3. The product automaton for a process system with $n = 2$, $k = 3$. The thick arrows indicate the path corresponding to the qualitative behavior $w = e_1^1\ e_1^2\ e_2^2\ e_1^3\ e_3^2\ e_2^1\ e_3^1\ e_2^3\ e_3^3$. The race between e_3^1 and e_2^2 in state (q_3^1, q_2^2) is indicated by the dashed arrows.

As illustrated in Fig. 2, the constraints that remain in φ_w are the inter-process constraints that are sufficient to characterize w. These constraints can be computed *incrementally* as we move along the prefix of a qualitative behavior. Let us follow the first two steps. Initially we have the empty word whose associated zone is $\mathcal{C}$ and hence its probability is 1. After the occurrence of the first event e_1^1 we know that P_1^1 terminated before P_1^2 and P_1^3. This leads to the constraints:

$$\varphi_{e_1^1} : \varphi_{\mathcal{C}} \wedge (t_1^1 < t_1^2)\ \wedge\ (t_1^1 < t_1^3) \tag{2}$$

After this first event we have a competition between e_1^2, e_1^3 and e_2^1. The winner of the race is the next event of w, e_1^2 and hence we add the constrains $t_1^2 < t_1^3$ and $t_1^2 < t_2^1$ and remove the constraint $t_1^1 < t_1^3$ which becomes redundant, yielding:

$$\varphi_{e_1^1 e_1^2} : \varphi_{\mathcal{C}} \wedge (t_1^1 < t_1^2)\ \wedge\ (t_1^2 < t_1^3)\ \wedge\ (t_1^2 < t_2^1).$$

In general whenever event e_j^i occurs, we add a constraint stating that t_j^i is smaller than the time stamps associated with all the pending events in the other processes. The incremental process is illustrated in Fig. 4.

This procedure is probabilistically correct in the following sense. For every $w \in \underline{L}$ the probability of all behaviors having w as a prefix is the relative volume of the corresponding zone Z_w, namely, $p(w) = |Z_w|/|\mathcal{C}|$. This holds trivially for the empty behavior when there are no constraints. For the inductive step observe that any qualitative behavior of the form $w\ e$ which extends w has to satisfy φ_w due to *causality* as well as additional constraints that guarantee that e is indeed the next event to win the race.

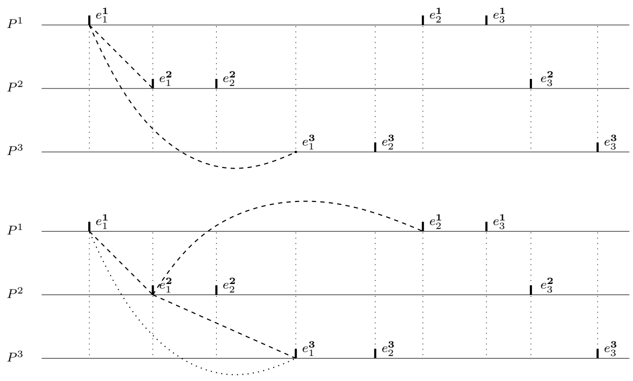

Fig. 4. Incremental constraint construction: constraints for $e_1^{\mathbf{1}}$ and then for $e_1^{\mathbf{1}}\ e_1^{\mathbf{2}}$. The constraint $t_1^{\mathbf{1}} < t_1^{\mathbf{3}}$ becomes redundant after the second event.

The constraints associated with all the extension of w form a *partition* of Z_w and all the probabilistic mass $p(w)$ is split among them, satisfying

$$\sum_e p(w\ e) = p(w).$$

In [29] a similar incremental approach that goes from a path/prefix to its successors has been developed using the clock auxiliary variables. The use of clocks required the concept of *density transformers* to account for the distribution of clock values before and after transitions (see also [2, 10, 14, 30]). These are not needed in the clock-free approach presented here. Those acquainted with the verification of timed automata using a forward computation of the simulation/reachability graph [17, 26] may notice that for every w the zone Z_w in the time-stamp space is empty exactly when its associated clock space zone in the reachability graph becomes empty. This suggests an alternative clock-free analysis algorithm for timed automata which is immediately applicable to acyclic systems but will require more work to be adapted to the cyclic case.

Having labeled qualitative behaviors by constraints we need to compute the volume of the zones. We illustrate this procedure on a concrete example with $n = 3$ and $k = 1$, hence $\mathcal{D} = \mathcal{C} = I_1^{\mathbf{1}} \times I_1^{\mathbf{2}} \times I_1^{\mathbf{3}}$, with concrete values

$$[a_1^{\mathbf{1}}, b_1^{\mathbf{1}}] = [2, 5],\ \ [a_1^{\mathbf{2}}, b_1^{\mathbf{2}}] = [3, 4],\ \text{ and }\ [a_1^{\mathbf{3}}, b_1^{\mathbf{3}}] = [4, 7].$$

The constraints associated with all qualitative behaviors where process $P^{\mathbf{1}}$ wins the first race are

$$\varphi_{e_1^{\mathbf{1}}}:\ \ (2 \le t_1^{\mathbf{1}} \le 5) \wedge (3 \le t_1^{\mathbf{2}} \le 4) \wedge (4 \le t_1^{\mathbf{3}} \le 7) \wedge (t_1^{\mathbf{1}} < t_1^{\mathbf{2}}) \wedge (t_1^{\mathbf{1}} < t_1^{\mathbf{3}}).$$

We pick an integration order $t_1^{\mathbf{3}} \prec t_1^{\mathbf{2}} \prec t_1^{\mathbf{1}}$, that is, the inside-out order of variable elimination, and rewrite $\varphi_{e_1^{\mathbf{1}}}$ as

$$\varphi_{e_1^{\mathbf{1}}}:\ \ (2 \le t_1^{\mathbf{1}} \le 5) \wedge (\max(3, t_1^{\mathbf{1}}) \le t_1^{\mathbf{2}} \le 4) \wedge (\max(4, t_1^{\mathbf{1}}) \le t_1^{\mathbf{3}} \le 7)$$

Then we split $I_1^{\mathbf{1}}$ into maximal segments where both $\max(3, t_1^{\mathbf{1}})$ and $\max(4, t_1^{\mathbf{1}})$ are uniform. In our example $[2, 5]$ splits into $[2, 3]$, $[3, 4]$ and $[4, 5]$ and the volume of the set can be written as

$$\left[\int_2^3\int_3^4\int_4^7 + \int_3^4\int_{t_1^{\mathbf{1}}}^4\int_4^7 + \int_4^5\int_{t_1^{\mathbf{1}}}^4\int_{t_1^{\mathbf{1}}}^7\right] dt_1^{\mathbf{3}} dt_1^{\mathbf{2}} dt_1^{\mathbf{1}} = 3 + \frac{3}{2} + 0 = \frac{9}{2}$$

which after dividing by $|\mathcal{C}| = 9$ gives a probability of $1/2$ for $e_1^{\mathbf{1}}$ winning the first race. Figure 5 illustrates two possible splits of a 2-dimensional zone into integration domains. The number of case splits and the forms of the integration domains may vary a lot depending on the chosen order.

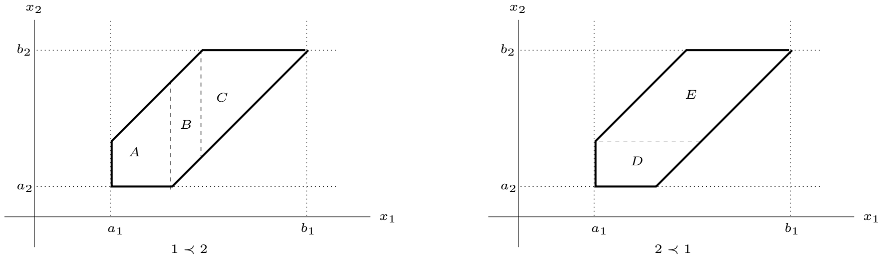

Fig. 5. Zone volume computation by splitting into integration domains in two different integration orders which yield 3 and 2 domains, respectively.

Theorem 1 (Probability of Qualitative Behaviors). *Given a system of stochastic sequential processes as in Def. 3 the probability of any of its qualitative behaviors is computable.*

The global termination time (makespan in the job-shop jargon) of a behavior is $\Theta = \max\{t_k^{\mathbf{1}}, \ldots, t_k^{\mathbf{n}}\}$. For all behaviors that are qualitatively equivalent the maximum is attained by the same variable, namely $t_k^{\mathbf{i}}$ for any behavior whose last event is $e_k^{\mathbf{i}}$. To compute the expected termination time we integrate $t_k^{\mathbf{i}}$ over Z_w and sum up over all w:

$$\mathbb{E}(\Theta) = \frac{1}{|\mathcal{C}|}\sum_{i=1}^{n}\sum_{w=w'e_k^{\mathbf{i}}}\int_{Z_w} t_k^{\mathbf{i}}.$$

Before moving to the coordinated execution framework let us mention some useful observations. So far we have treated qualitative behaviors in their finest granularity, taking note of the ordering between any pair of events. In many situations we are interested in *sets* of qualitative behaviors and their probability can often be computed more efficiently than summing up the probabilities of individual qualitative behaviors.

Suppose we want to characterize the set of all qualitative behaviors that pass through a global state $q = (q_{j_1}^{\mathbf{1}}, \ldots, q_{j_n}^{\mathbf{n}})$. Let $\underline{L}_j^{\mathbf{i}} = \{e_1^{\mathbf{i}} \cdots e_{j-1}^{\mathbf{i}}\}$ be the qualitative behavior of $P^{\mathbf{i}}$ that leads to $q^{\mathbf{i}}_j$. Then the set of qualitative behaviors that lead to q is

$$\underline{L}(q) = \underline{L}^{\mathbf{1}}_{j_1} || \cdots || \underline{L}^{\mathbf{n}}_{j_n}.$$

The constraints that characterize $\underline{L}(q)$ may forget the specific interleaving, that is, the specific order in which *past* events have occurred. The only constraints that are relevant are those that guarantee that the entrance of each process into its respective local state preceded the exit of all other processes from their respective states, that is,

$$\varphi_q : \varphi_{\mathcal{C}} \wedge \bigwedge_{i=1}^{n} \bigwedge_{i' \neq i} t^{\mathbf{i}}_{j_i-1} < t^{\mathbf{i}'}_{j_{i'}}.$$

Thus, to compute the expected termination time it suffices to partition the set of qualitative behaviors into n classes according to the identity of the *last* transition, letting Z^i be the zone defined by

$$\varphi^i : \ \varphi_{\mathcal{C}} \wedge \bigwedge_{i' \neq i} t^{\mathbf{i}'}_k < t^{\mathbf{i}}_k.$$

Then the expected termination time is

$$\mathbb{E}(\Theta) = \frac{1}{|\mathcal{C}|} \sum_{i=1}^{n} \int_{Z^i} t^{\mathbf{i}}_k. \tag{3}$$

A similar observation, made in the context of zone-based verification of timed automata, underlies the fact that the union of zones reached by interleavings of the same set of events is convex [31, 27, 9].

4 Conflicts and Schedulers

Now we adapt the framework to the case where steps of different processes may be at conflict due to requiring the same resource and hence cannot be executed simultaneously. Naturally, this situation is more intuitively expressed using automata, states and runs. In order not to discourage semantically challenged audiences we explain the automaton-based modeling very informally. Interested readers may consult [1] for a general framework for modeling and solving scheduling problems with timed automata as well as [29] for the formal definitions of *duration-probabilistic automata* (DPA) which is the model underlying this paper.

As a running example consider a system of two processes with three steps each, admitting a resource conflict between their respective second steps $P^{\mathbf{1}}_2$ and $P^{\mathbf{2}}_2$. Conflicts are modeled in automata using *forbidden states* in the global automaton, state $(q^{\mathbf{1}}_2, q^{\mathbf{2}}_2)$ in our example. To be able to prevent the automaton from entering this state[5] we refine the process model so that the initiation of step $P^{\mathbf{i}}_j$ does not occur *automatically* upon the termination of step $P^{\mathbf{i}}_{j-1}$. We thus modify the process automaton shown in Fig. 3 by inserting a *waiting state* $\bar{q}^{\mathbf{i}}_j$ between $q^{\mathbf{i}}_{j-1}$ and $q^{\mathbf{i}}_j$. The automaton can leave this state only when it receives a *start* command $s^{\mathbf{i}}_j$ from a scheduler as illustrated in Fig. 6-(a).

[5] We consider schedulers that by construction cannot make the system enter a forbidden state.

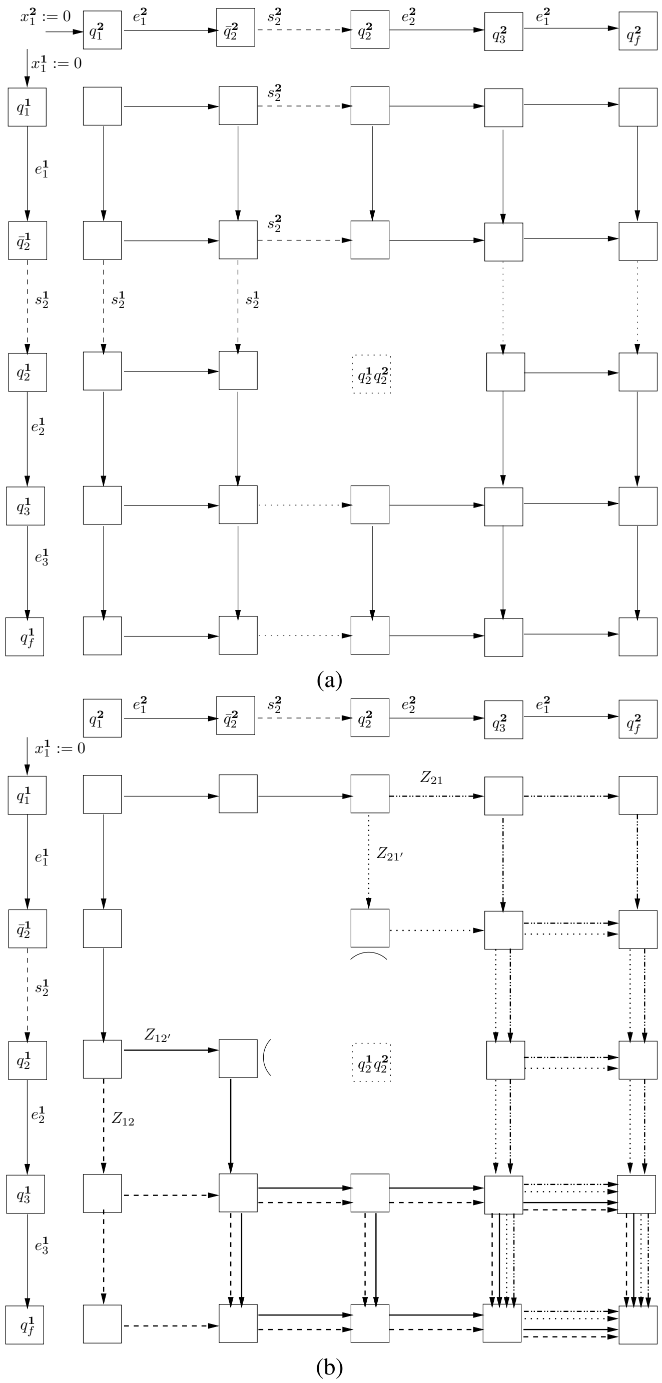

Fig. 6. (a) Two parallel processes admitting a resource conflict and their product automaton. The dashed arrows indicate *start* transitions which should be under the control of a scheduler while the dotted arrows indicate post-conflict *start* transitions; (b) The automaton resulting from composition with a FIFO scheduler and the 4 potential conflict resolution and resource utilization scenarios.

As long as the scheduler is not completely specified the system is *open* or using another terminology, admits both *probabilistic* and *set-theoretic* non-determinism. For example in state $(\bar{q}_2^{\mathbf{1}}, q_1^{\mathbf{2}})$ process $P^{\mathbf{1}}$ may either start its second step $(\bar{q}_2^{\mathbf{1}}, q_1^{\mathbf{2}}) \rightarrow (q_2^{\mathbf{1}}, q_1^{\mathbf{2}})$ or wait until step $P_1^{\mathbf{2}}$ terminates and let $P^{\mathbf{2}}$ take the resource first $(\bar{q}_2^{\mathbf{1}}, q_1^{\mathbf{2}}) \rightarrow (\bar{q}_2^{\mathbf{1}}, \bar{q}_2^{\mathbf{2}}) \rightarrow (\bar{q}_2^{\mathbf{1}}, q_2^{\mathbf{2}})$. A scheduler resolves this type of non-determinism by telling each process in a waiting state whether to take the resource and proceed to execution or wait until the resource is taken and released by another process.[6] Once such a scheduler is defined, the set-theoretic non-determinism is eliminated and the only non-determinism that remains is the one associated with task durations and thus it becomes possible to compute probabilities. To be more precise, probabilities can be computed also for non-deterministic schedulers that make a probabilistic choice, but we do not consider them here.

A scheduler is thus a mechanism which may observe the state of the system and decide whether to grant a resource to a process, possibly based on the level of progress of other processes. The most passive scheduler grants the resource to the *first* process whose corresponding step becomes enabled. Under such a FIFO scheduling policy it is the result of the race between $e_1^{\mathbf{1}}$ and $e_1^{\mathbf{2}}$ which determines the resource granting decision. The automaton obtained by composing the system with such a scheduler is shown in Figure 6-(b) where we have chosen to ignore the zero-measure situation when both processes terminate *exactly* at the same time (alternatively this situation can be handled by assigning an arbitrary priority when this is the case).

More active schedulers interfere with the execution order by imposing additional conditions upon the *start* transitions. Suppose that the duration of step $P_3^{\mathbf{1}}$ is much longer than that of $P_3^{\mathbf{2}}$ hence it would be reasonable to give $P_2^{\mathbf{1}}$ a priority over $P_2^{\mathbf{2}}$ even if the latter becomes enabled earlier. This priority can have different degrees of rigidity. A *strict* priority scheduler allows $s_2^{\mathbf{2}}$ only in global states where $P_2^{\mathbf{1}}$ has terminated, a condition that we write as $\mathcal{A}^{\mathbf{1}} > q_2^{\mathbf{1}}$. The automaton obtained by composing the system with such a scheduler is shown in Fig. 7. Note that strict priority schedulers make the automaton always “bypass” a conflict state from the same side.

Strict priority schedulers can be unnecessarily rigid for tasks with durations variability as they do not adapt to the actual evolution of the schedule. As an example for such adaptability consider the case where $P_1^{\mathbf{2}}$ terminates very early so that we can start $P_2^{\mathbf{2}}$ so that it will surely terminate before $P_2^{\mathbf{1}}$ becomes enabled and hence will not block it. Even if this is not guaranteed with certainty, a scheduler might want to start $P_2^{\mathbf{2}}$ if the expected delay incurred to $P^{\mathbf{1}}$ is small. Technically, the knowledge of the relative timing of $e_1^{\mathbf{2}}$ at decision time is encoded by the value of clock $x^{\mathbf{1}}$ reset upon starting $P_1^{\mathbf{1}}$. The larger is the value of $x^{\mathbf{1}}$, the more we are likely to block $P^{\mathbf{1}}$ and for a longer period. Hence the condition for issuing $s_2^{\mathbf{2}}$ by such a state-dependent scheduler will be of the form $(\mathcal{A}^{\mathbf{1}} > q_2^{1}) \vee (\mathcal{A}^{\mathbf{1}} < \bar{q}_2^{1} \wedge x^{\mathbf{1}} < d)$ for some constant d.

The labeling of states and qualitative behaviors with constraints in order to compute volumes, probabilities and expected termination times can be extended to handle all these types of schedulers. As an illustration consider the FIFO scheduler of Fig. 6-

[6] Note that we restrict ourselves to *non-lazy* schedulers: if they do not issue an $s_j^{\mathbf{i}}$ command at some point, they will not issue it later unless another process has utilized the resource. This class has been shown [1] to contain the optimal schedulers for the problems we are dealing with.

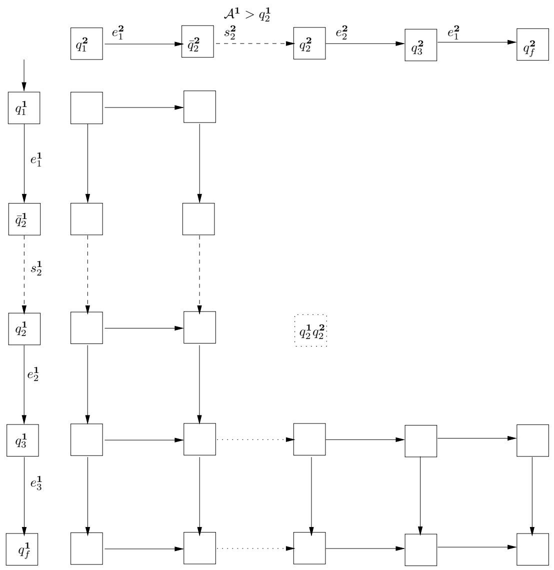

Fig. 7. (a) A scheduler that gives strict priority to $P^{\mathbf{1}}$. This is realized by the condition $\mathcal{A}^{\mathbf{1}} > q_2^{\mathbf{1}}$ which allows $P_2^{\mathbf{2}}$ to start only after $P_2^{\mathbf{1}}$ terminates.

(b) which admits 4 classes of qualitative behaviors (scenarios) that correspond to the outcomes of the conflict between $P^{\mathbf{1}}$ and $P^{\mathbf{2}}$ on the shared resource. These scenarios are characterized by the identity of the winner (for this scheduler it depends on the relation between $t_1^{\mathbf{1}}$ and $t_1^{\mathbf{2}}$) and by whether the loser termination time is delayed (depending on whether the winner releases the resource before the loser becomes enabled). These cases are summarized in Table 1 and depicted in Fig. 6-(b).

As the alert reader might have noticed, the transformation T from the duration space to the time-stamp space is different from the independent execution framework. It can nevertheless be shown to be volume preserving along the following lines. First, one can show that after adding inter-process precedence constraints causality is preserved and there is always a rearrangement of the indices such that the transformation matrix remains lower triangular. Secondly the notion of volume preservation can be easily generalized from linear to piecewise-linear transformations.

The above analysis can be generalized to m distinct resources and to multi-party conflicts on each of them. For each resource l one can compute the set U_l of all the utilization scenarios for this resource and their respective zones. A scenario corresponds to a particular order of resource utilization by conflicting steps and to the waiting delays incurred to these steps. Then the classes of potential qualitative behaviors of interest are the combinations of those, that is, $U = U_1 \times \cdots \times U_m$ with zones defined by intersection. While this sounds like a recipe for a severe combinatorial explosion, note that many scenarios will lead to empty zones, either for logical reasons (inter-process ordering of conflicting steps is incompatible with local precedence constraints) or due

winner	loser delayed		loser not delayed
P^1	$Z_{12'}$		Z_{12}
	$a_1^1 < t_1^1 < b_1^1$		$a_1^1 < t_1^1 < b_1^1$
	$a_1^2 < t_1^2 < b_1^2$		$a_1^2 < t_1^2 < b_1^2$
	$t_1^1 < t_1^2$		$t_1^1 < t_1^2$
	$t_1^1 + a_2^1 < t_2^1 < t_1^1 + b_2^1$		$t_1^1 + a_2^1 < t_2^1 < t_1^1 + b_2^1$
	$t_1^2 < t_2^1$		$t_2^1 < t_1^2$
	$t_2^1 + a_2^2 < t_2^2 < t_2^1 + b_2^2$		$t_1^2 + a_2^2 < t_2^2 < t_1^2 + b_2^2$
P^2	$Z_{21'}$		Z_{21}
	$a_1^1 < t_1^1 < b_1^1$		$a_1^1 < t_1^1 < b_1^1$
	$a_1^2 < t_1^2 < b_1^2$		$a_1^2 < t_1^2 < b_1^2$
	$t_1^2 < t_2^1$		$t_1^2 < t_2^1$
	$t_1^2 + a_2^2 < t_2^2 < t_1^2 + b_2^2$		$t_1^2 + a_2^2 < t_2^2 < t_1^2 + b_2^2$
	$t_2^2 < t_1^1$		$t_1^1 < t_2^2$
	$t_1^2 + a_2^2 < t_2^2 < t_1^2 + b_2^2$		$t_1^2 + a_2^2 < t_2^2 < t_1^2 + b_2^2$

Table 1. The zones corresponding to the four possible outcomes of the resource conflict of Fig. 7-(b). Constraints on t_3^1 and t_3^2 are omitted.

to the arithmetics of timing constraints (two conflicting tasks, one at the beginning and one at the end of their respective processes, are likely to be executed in one order). Naturally, for priority schedulers there will be fewer scenarios to analyze.

We have implemented a prototype tool which computes expected termination times as described in this paper. As input it takes a system description consisting of processes, steps, duration intervals and conflicts as well as a definition of a scheduling policy. Then for every utilization scenario it derives the corresponding zone, using the DBM library of IF [12] to normalize constraints and detect empty zones. Then it performs integration over the non-empty zones to compute probability and expected termination time. The integration uses the *GNU Multiple Precision Arithmetic Library* (GMP) to avoid rounding errors.

Let us describe our preliminary experiments. For each value of n from 1 to 5 and for each value of k from 1 to 40, we choose a number of conflicts (between 0 and 3) and a number of participants in each conflict (2 or 3). Each choice in this space defines a problem type for which we draw 10 concrete problems by randomly choosing the identity of the conflicting steps as well as step duration intervals of the form $[c-d, c+d]$ with c drawn uniformly in $[40, 50]$ and d in $[0, c/20]$. Then we try to compute expected termination times for a FIFO scheduler with a timeout of 3 minutes per problem on an old laptop.

The experiments with $n = 1$ compute the volume of one zone, the time-stamp space. Applying the reverse order integration we can compute up to dimension 63 in 0.4 seconds (currently this is a limitation of our DBM library). In general integration takes place in $\mathbb{R}^{nk}$ and its complexity depends on the following factors. First, the number of scenarios (orders of resource utilizations and their combinations) determines the number of zones whose volume we might need to compute, in case they are not detected beforehand to be empty. For each zone and each variable t we compute I_t, the projection of the zone on t. Then we define a partial order relation between these intervals such that $I_t < I_{t'}$ if the upper bound of I_t is smaller than the lower bound of $I_{t'}$. Then we construct a compatible linear order and integrate backwards. The chosen order determines the number of case splits but also the form of the integration domains and the polynomials obtained during integration. We experienced orders of integration that generate more splits but take less overall computation time. Since there is a lot of exploration and fine tuning ahead it is premature to report performance systematically. To give an idea, we mention some problem types for which we managed to compute for all the test cases.These include $(n, k) = (2, 12)$ with 2 conflicts, $(3, 6)$ and $(4, 6)$ with 3 binary conflicts or 2 ternary conflicts and $(5, 4)$ with 2 binary conflicts.

5 Concluding Remarks

We have presented a computational technique to evaluate schedulers in a non-Markovian setting based on splitting the space of valuations of the random variables and computing volumes. To the best of our knowledge no similar computational results have been reported. We mention some future work.

1. Integration over zones is the major computational activity in our procedure and its optimization is an interesting algorithmic problem.
2. To handle larger systems one needs to develop algorithms that do not explore all classes of qualitative behaviors but restrict the exploration to a high-volume small subset of those, whenever such exists.
3. While this work solves the *analysis* problem it would be more challenging to *synthesize* optimal schedulers automatically using value iteration. It is an open question whether such a backward iteration can be defined using the clock-free methods developed in this paper.
4. Another major challenge is to extend this framework to cyclic systems, define the appropriate performance measures and study their steady-state behavior.
5. Finally, it would be interesting to compare the analytic method developed here with statistical approaches based on random simulation. It is intriguing to see how many simulation runs are needed to approximate our results with a good confidence.

Acknowledgment This work grew out of numerous discussions with Kim Larsen and Bruce Krogh and benefitted from the constructive criticism of Eugene 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. R. Alur, C. Courcoubetis, and D.L. Dill. Model-checking for probabilistic real-time systems. In *ICALP*, pages 115–126, 1991.
4. R. Alur and D.L. Dill. A theory of timed automata. *Theoretical Computer Science*, 126(2):183–235, 1994.

5. E. Asarin, P. Caspi, and O. Maler. Timed regular expressions. *J. ACM*, 49(2):172–206, 2002.
6. E. Asarin and O. Maler. As soon as possible: Time optimal control for timed automata. In *HSCC*, pages 19–30, 1999.
7. 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.
8. C. Baier, B.R. Haverkort, H. Hermanns, and J.-P. Katoen. Model-checking algorithms for continuous-time Markov chains. *IEEE Trans. Software Eng.*, 29(6):524–541, 2003.
9. R. Ben-Salah, M. Bozga, and O. Maler. On interleaving in timed automata. In *CONCUR*, pages 465–476, 2006.
10. M. Bernadsky and R. Alur. Symbolic analysis for GSMP models with one stateful clock. In *HSCC*, pages 90–103, 2007.
11. P. Bouyer. *From Qualitative to Quantitative Analysis of Timed Systems*. Mémoire d'habilitation, Université Paris 7, 2009.
12. M. Bozga, S. Graf, and L. Mounier. IF-2.0: A validation environment for component-based real-time systems. In *CAV*, volume 2404 of *LNCS*, pages 343–348. Springer, July 2002.
13. E. Brinksma, H. Hermanns, and J.-P. Katoen, editors. *Lectures on Formal Methods and Performance Analysis*, volume 2090 of *LNCS*. Springer, 2001.
14. 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.
15. C.G. Cassandras and S. Lafortune. *Introduction to Discrete Event Systems*. Springer, 2nd edition, 2008.
16. T.T. Cormen, C.E. Leiserson, and R.L. Rivest. *Introduction to algorithms*. MIT Press, 1990.
17. C. Daws, A. Olivero, S. Tripakis, and S. Yovine. The tool KRONOS. In *Hybrid Systems*, pages 208–219, 1995.
18. V. Diekert and G. Rozenberg, editors. *The Book of Traces*. World Scientific, 1995.
19. D.L. Dill. Timing assumptions and verification of finite-state concurrent systems. In *Automatic Verification Methods for Finite State Systems*, pages 197–212, 1989.
20. J. Esparza and K. Heljanko. *Unfoldings – A Partial-Order Approach to Model Checking*. Springer, 2008.
21. R. German. Non-markovian analysis. In Brinksma et al. [13], pages 156–182.
22. P.W. Glynn. A GSMP formalism for discrete event systems. *Proceedings of the IEEE*, 77(1):14–23, 1989.
23. T.A. Henzinger, X. Nicollin, J. Sifakis, and S. Yovine. Symbolic model checking for real-time systems. *Information and Computation*, 111(2):193–244, 1994.
24. M. Jurdzinski, D. Peled, and H. Qu. Calculating probabilities of real-time test cases. In *FATES*, pages 134–151, 2005.
25. D. Kartson, G. Balbo, S. Donatelli, G. Franceschinis, and G. Conte. *Modelling with Generalized Stochastic Petri Nets*. John Wiley & Sons, 1994.
26. 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.
27. D. Lugiez, P. Niebert, and S. Zennou. A partial order semantics approach to the clock explosion problem of timed automata. *Theoretical Computer Science*, 345:27–59, 2005.
28. O. Maler. On optimal and reasonable control in the presence of adversaries. *Annual Reviews in Control*, 31(1):1–15, 2007.
29. O. Maler, K.G. Larsen, and B.H. Krogh. On zone-based analysis of duration probabilistic automata. In *INFINITY*, pages 33–46, 2010.
30. E. Vicario, L. Sassoli, and L. Carnevali. Using stochastic state classes in quantitative evaluation of dense-time reactive systems. *IEEE Trans. Software Eng.*, 35(5):703–719, 2009.
31. J. Zhao. Partial order path technique for checking parallel timed automata. In *FTRTFT*, pages 417–432, 2002.