Fundamenta Informaticae XX (2009) 1–29
IOS Press

Performance Evaluation of Distributed Systems Based on a Discrete Real- and Stochastic-Time Process Algebra

J. Markovski and E.P. de Vink

Formal Methods Group, Department of Mathematics and Computer Science
Eindhoven University of Technology, Den Dolech 2, 5612 AZ Eindhoven, The Netherlands
tel: +31 40 247 3360, fax: +31 40 247 5361
j.markovski@tue.nl, evink@win.tue.nl

Abstract. We present a process-algebraic framework for performance evaluation of discrete-time discrete-event systems. The modeling of the system builds on a process algebra with conditionally-distributed discrete-time delays and generally-distributed stochastic delays. In the general case, the performance analysis is done with the toolset of the modeling language χ by means of discrete-event simulation. The process-algebraic setting allows for expansion laws for the parallel composition and the maximal progress operator, so one can directly manipulate the process terms and transform the specification in a required form. This approach is illustrated by specifying and solving the recursive specification of the $G/G/1/\infty$ queue, as well as by specifying a variant of the concurrent alternating bit protocol with generally-distributed unreliable channels. In a specific situation when all delays are assumed deterministic, we turn to performance analysis of probabilistic timed systems. This work employs discrete-time probabilistic reward graphs, which comprise deterministic delays and immediate probabilistic choices. Here, we extend previous investigations on the topic, which only touched long-run analysis, to tackle transient analysis as well. The theoretical results obtained allow us to extend the χ-toolset. For illustrative purposes, we analyze the concurrent alternating bit protocol in the extended environment of the χ-toolset using discrete-event simulation for generally-distributed channels, the developed analytical method for deterministic channels, and Markovian analysis for exponentially-distributed delays.

1. Introduction

Over the past decade stochastic process algebras have emerged as compositional modeling formalisms for systems that not only require functional verification, but performance analysis as well. Many Markovian process algebras are developed like EMPA [9], PEPA [27], IMC [25], etc. exploiting the memoryless

Address for correspondence: J. Markovski, TU/e, P.O. Box 513, 5600 MB, Eindhoven, The Netherlands

property of the exponential distribution. Before long, the need for general distributions arose, as exponential delays are not sufficient to model, for example, fixed timeouts of Internet protocols or heavy-tail distributions present in media streaming services. Prominent stochastic process algebras and calculi with general distributions include TIPP [26], GSMPA [13], SPADES [20], IGSMP [12], NMSPA [31], and MODEST [10].

Despite the greater expressiveness, compositional modeling with general distributions proved to be challenging, as the memoryless property cannot be relied on [29, 14]. Typically, the underlying performance model is a generalized semi-Markov process that exploits clocks to memorize past behavior in order to retain the Markov property of history independence [23]. Similarly, the semantics of stochastic process algebras is given using clocks that represent the stochastic delays at the symbolic level. Such a symbolic representation allows for the manipulation of finite structures, e.g., stochastic automata or extensions of generalized semi-Markov processes. The concrete execution model is subsequently obtained by sampling the clocks, frequently yielding infinite probabilistic timed transition systems.

For the sampling of the clock two execution policies can be adopted: (1) race condition [26, 20, 31, 10], which enables the action transitions guarded by the clocks that expire first, and (2) pre-selection policy [13, 12], which preselects the clocks by a probabilistic choice. To keep track of past behavior, the clock samples have to be updated after each stochastic delay transition. One can do this in two equivalent ways: (1) by keeping track of residual lifetimes [20, 10], i.e., the time left up to expiration, or (2) by keeping track of the spent lifetimes [26, 13, 12, 31], i.e., the time passed since activation. The former manner is more suitable for discrete-event simulation, whereas the latter is acknowledged for its correspondence to real-time semantics [29, 14].

In this paper we consider the race condition with spent-lifetime semantics. However, we do not use clocks to implement the race condition and to determine the winning stochastic delay(s) of the race. Rather, we rely on an interpretation that uses conditional random variables and makes a probabilistic assumption on the winners followed by conditioning of the distributions of the losers on the time spent for the winning samples [28]. Thus, we no longer speak of clocks as we do not keep track of sample lifetimes, but we only cater for the ages of the conditional distributions [35]. We refer to the samples as stochastic delays, a naming resembling standard timed delays.

The relation between real and stochastic time has been studied in various settings: a structural translation from stochastic to timed automata with deadlines is given in [19]. This approach found its way into MODEST, where timed automata with deadlines are merged with stochastic automata in so-called stochastic timed automata as a means to introduce real and stochastic time as separate constructs. Also, a translation from IGSMP into pure real-time models called interactive timed automata is reported in [12]. The interplay between standard timed delays and discrete stochastic delays has been studied in [34, 35]. An axiomatization for a process algebra that embeds real-time delays with so-called context-sensitive interpolation into a restricted form of discrete stochastic time is given in [35].

The paper presents a performance evaluation framework based on process algebraic specifications and their analysis in an extended environment of the χ-toolset [8, 38]. The contribution of the paper is twofold. As a first contribution, a sound and ground-complete process algebra is provided that accommodates timed delays in a racing context, extending the work of [34, 35]. The theory provides an explicit maximal progress operator and a non-trivial expansion law for the parallel composition. Differently from other approaches, we derive stochastic delays as time-delayed processes with explicit information about the winners and the losers that induced the delay. We represent standard real-time as stochastic time inducing a trivial race condition in which the shortest sample is always exhibited by the same set

of delays and moreover has a fixed duration. The algebra also provides the possibility of specifying a partial race of stochastic delays, e.g., that one delay has always a shorter, equal, or longer sample than the other delay. This is required when modeling timed systems whose correct behavior depends on the relative ordering of the timed delays, e.g., in a time dependent controller. When the timed delays are simply replaced by stochastic delays, the total order of the samples is, in general, lost, unless it can be specified which delays are the winners or losers of the imposed race.

We illustrate the process theory by revisiting the $G/G/1/\infty$ queue from [34], treating it more elegantly now and providing a solution for the recursive specification by manipulating process terms using the proposed axiomatization. We also specify a variant of the concurrent alternating bit protocol that has fixed timeouts (represented by timed delays) and faulty generally-distributed channels (represented by stochastic delays), stressing the interplay of real-time and stochastic time.

Our second contribution concerns automated performance analysis. It is well known that only a small number of restricted classes of models of general distributions are analytically solvable. Preliminary research on model checking of stochastic automata is reported in [15] and a proposal for model checking probabilistic timed systems is given in [39]. However, at the moment, performance analysts turn to discrete-event simulation when it comes to analyzing models with generally-distributed delays. For analysis of the concurrent alternating bit protocol we depend on the toolset of the χ-language [8, 38, 11, 2]. At the start, χ was used to model discrete-event systems only, not supported by an explicit semantics. However, recently, it has been turned into a formal specification language set up as a hybrid process algebra with data [8, 38].

The connection between the timed discrete-event subset of χ and standard timed process algebras in vein of [4] is straightforward. In [42], a proposal was given to extend χ with a probabilistic choice to enable long-run performance analysis of probabilistic timed specifications. Here, we rely on this extension to provide a connection with the stochastic part of our process algebra as well. At this point, the co-existence of real and stochastic time in the same model plays a crucial role, which underlines the key position of the process algebra in the framework. The performance model is termed discrete-time probabilistic reward graph and it comprises deterministic delays and immediate probabilistic choices. It is suitable as an underlying performance model for stochastic delays with finite support set as used in the case study (even though the theory does not have such a limitation). In [42], discrete-time probabilistic reward graphs were employed for long-run analysis of industrial systems. Here, we extend the performance evaluation framework of [42] to cater for transient analysis as well. We accordingly augment the χ-toolset and apply it to the concurrent alternating bit protocol. The case study illustrates the new approach when the channel distributions are deterministic. Finally, we compare the analytical results with the ones obtained from discrete-event simulation and Markovian analysis using the same specification in χ. We visualize the proposed framework in Figure 1. We note that we rely on the CADP toolset [21] as a solver for the underlying/intermediate Markov reward processes.

The rest of this paper is organized as follows: Section 2 discusses background material and design choices. Section 3 introduces the process theory and revisits the $G/G/1/\infty$ queue example. Section 4 discusses transient analysis of discrete-time probabilistic reward graph in the performance evaluation framework. Section 5 analyzes the concurrent alternating bit protocol protocol and discusses its specification in the proposed process algebra and the language χ. Section 6 wraps up with concluding remarks. Due to substantial technical overhead, we do not give the operational semantics of the process-algebraic theory here. Instead, we focus on the axiomatization to illustrate its suitability for protocol specification. The complete structural operational semantics and formal treatment of the theory are available in [32].

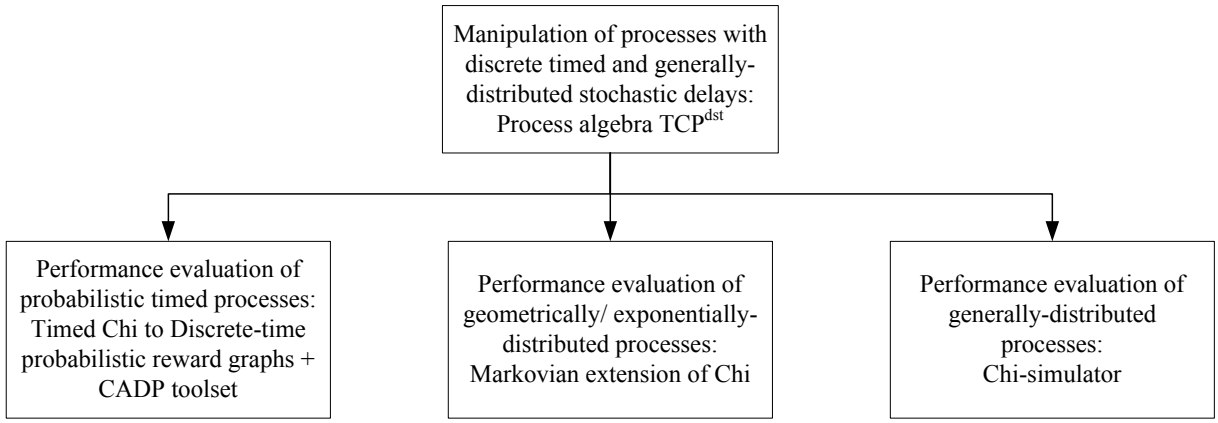

Figure 1. The proposed process-algebraic performance evaluation framework

2. Timed and Stochastic Delays

In this section we introduce a number of notions in process theory that are used below. We refer the interested reader for more technical detail to [32].

Preliminaries We use discrete random variables to represent durations of stochastic delays. The set of discrete distribution functions F such that $\mathrm{F}(n)=0$ for $n \leq 0$ is denoted by $\mathcal{F}$; the set of the corresponding random variables by $\mathcal{V}$. We use X, Y, and Z to range over $\mathcal{V}$ and F_X, F_Y and F_Z for their respective distribution functions. Also, W, L, V, and D range over $2^{\mathcal{V}}$. Given a set A, by A^n we denote vectors of size $n \in \mathbb{N}$ and by $A^{m \times n}$ matrices with m rows and n columns with elements in A. By $\mathbf{0}$ and $\mathbf{1}$ we denote vectors that consist of 0s and 1s.

Racing stochastic delays A stochastic delay is a timed delay of a duration guided by a random variable. We observe simultaneous passage of time for a number of stochastic delays until one or some of them expire. This phenomenon is referred to as the *race condition* and the setting as the *race*. For multiple racing stochastic delays, different stochastic delays may be observed simultaneously as being the shortest. The ones that have the shortest duration are called the *winners* and the others are referred to as the *losers*. The outcome of a race is completely determined by the winners and the losers and their distributions. So, we can explicitly represent the outcome of the race by a pair of sets W, L of stochastic delays. We write $\left[\begin{smallmatrix} W \\ L \end{smallmatrix}\right]$ in case W is the set of winners and L is the set of losers. We have occasion to write $[W]$ instead of $\left[\begin{smallmatrix} W \\ \emptyset \end{smallmatrix}\right]$ and omit the set brackets when clear from the context. Thus, $[X]$ represents a stochastic delay guided by the random variable X.

To express a race, we will use the operator $_ + _$. So, $[X] + [Y]$ represents the race between the stochastic delays X and Y. There are three possible outcomes of this race: (1) $\left[\begin{smallmatrix} X \\ Y \end{smallmatrix}\right]$, (2) $\left[\begin{smallmatrix} X, Y \\ \emptyset \end{smallmatrix}\right]$, and (3) $\left[\begin{smallmatrix} Y \\ X \end{smallmatrix}\right]$. Thus, we can also write $\left[\begin{smallmatrix} X \\ Y \end{smallmatrix}\right] + \left[\begin{smallmatrix} X, Y \\ \emptyset \end{smallmatrix}\right] + \left[\begin{smallmatrix} Y \\ X \end{smallmatrix}\right]$ instead of $[X] + [Y]$, as both expressions represent the same final outcomes of a race. If an additional racing delay Z is added, this also leads to equal outcomes, i.e., $[X] + [Y] + [Z]$ and $\left[\begin{smallmatrix} X \\ Y \end{smallmatrix}\right] + \left[\begin{smallmatrix} X, Y \\ \emptyset \end{smallmatrix}\right] + \left[\begin{smallmatrix} Y \\ X \end{smallmatrix}\right] + [Z]$ will yield the same behaviour. For example, the outcome of $\left[\begin{smallmatrix} X \\ Y \end{smallmatrix}\right] + [Z]$ is either (1) $\left[\begin{smallmatrix} Z \\ X, Y \end{smallmatrix}\right]$, (2) $\left[\begin{smallmatrix} X, Z \\ Y \end{smallmatrix}\right]$, or (3) $\left[\begin{smallmatrix} X \\ Y, Z \end{smallmatrix}\right]$. As outcomes of races may be involved in other races, we generalize the notion of a stochastic delay and refer to an arbitrary outcome $\left[\begin{smallmatrix} W \\ L \end{smallmatrix}\right]$ as a stochastic delay induced by the winners W and the losers L, or by W and L for short. Here, we decide not to dwell on

the formal semantics because of a substantial technical overhead to formalize the notion of dependencies of losers on the samples of the winners. The basis for the semantics is given in [34, 35] and subsequently extended in [32] to allow the explicit specification of the winners and the losers of a race.

To summarize, there are three possible combinations that give the relation between the winners and the losers: (1) $L_1 \cap W_2 \neq \emptyset$, which means that the race must be won by W_1 and lost by $L_1 \cup W_2 \cup L_2$, (2) $W_1 \cap W_2 \neq \emptyset$, which means that the race must be won by $W_1 \cup W_2$ together and lost by $L_1 \cup L_2$, and (3) $W_1 \cap L_2 \neq \emptyset$, which means that the race must be won W_2 and lost by $W_1 \cup L_1 \cup L_2$. Obviously, these 'restrictions' are disjoint and cannot be applied together. If more than one restriction holds, then they lead to ill-defined outcomes. For example, if both (1) and (2) hold at the same time, then L_1 and W_2 must exhibit the same sample and also W_1 and W_2 must exhibit the same sample. Then W_1 and L_1 must exhibit the same sample, which is a contradiction.

If at least two restrictions apply, then the outcomes cannot be combined as they represent disjoint events. In this case we say the race between the delays $\left[\begin{smallmatrix} W_1 \\ L_1 \end{smallmatrix}\right]$ and $\left[\begin{smallmatrix} W_2 \\ L_2 \end{smallmatrix}\right]$ with $W_1 \cup L_1 = W_2 \cup L_2$, is *resolved*. The extra condition ensures that the outcomes stem from the same race, i.e, they have the same racing delays. For example, $\left[\begin{smallmatrix} X \\ Y \end{smallmatrix}\right]$ and $\left[\begin{smallmatrix} Y, Z \\ X \end{smallmatrix}\right]$ cannot form a joint outcome. The delays do not stem from the same race, which renders their combination inconsistent. Resolved races play an important role as they enumerate every possible outcome of the race. We define a predicate $\mathrm{rr}(\left[\begin{smallmatrix} W_1 \\ L_1 \end{smallmatrix}\right], \left[\begin{smallmatrix} W_2 \\ L_2 \end{smallmatrix}\right])$ that checks whether two delays $\left[\begin{smallmatrix} W_1 \\ L_1 \end{smallmatrix}\right]$ and $\left[\begin{smallmatrix} W_2 \\ L_2 \end{smallmatrix}\right]$ are in a resolved race. It is satisfied if $W_1 \cup L_1 = W_2 \cup L_2$ and at least two of the following three restrictions from above hold: (1) $L_1 \cap W_2 \neq \emptyset$, (2) $W_1 \cap W_2 \neq \emptyset$, and (3) $W_1 \cap L_2 \neq \emptyset$.

Naming of stochastic delays Consider the process term $[X].p_1 \,\|\, [X].p_2$, where $[X]._$ denotes stochastic delay prefixing, $_ \,\|\, _$ denotes the parallel composition, and p_1 and p_2 are arbitrary process terms. We note that the alternative and the parallel composition impose the same race condition. In a standard way, the race is performed on two stochastic delays with the same distribution $\mathrm{F}_X \in \mathcal{F}$. However, both delays will not necessarily exhibit the same sample, unless F_X is Dirac. Intuitively, the process given by the above term is equivalent to process given by $[X].p_1 \parallel [Y].p_2$ with $\mathrm{F}_X = \mathrm{F}_Y$ leading to three possible outcomes.

However, in real-time semantics, timed delays (denoted by σ^n for a duration $n \in \mathbb{N}$) with the same duration are merged together. For example, $\sigma^m.p_1 \parallel \sigma^m.p_2$ is equivalent to $\sigma^m.(p_1 \parallel p_2)$. This parallel composition represents components that should delay together. Note that this is not obtained above in the stochastic setting. Previous investigation in this matter [34, 35, 32] points out that both dependent and independent stochastic delays are indispensable. The former enable an expansion law for the parallel composition; the latter support compositional modeling.

Dependent stochastic delays always exhibit the same duration in the same race when guided by the same random variable. In contrast, independent stochastic delays with the same name have the same distribution, but not necessarily the same duration. As an example, $\left[\begin{smallmatrix} X, Y \\ Z \end{smallmatrix}\right] + \left[\begin{smallmatrix} X \\ U \end{smallmatrix}\right]$ is the same race as $\left[\begin{smallmatrix} X, Y \\ Z, U \end{smallmatrix}\right]$ if we treat X as a dependent stochastic delay, whereas $\left[\begin{smallmatrix} X \\ Z \end{smallmatrix}\right] + [X] = \left[\begin{smallmatrix} X \\ Z, Y \end{smallmatrix}\right] + \left[\begin{smallmatrix} X, Y \\ Z \end{smallmatrix}\right] + \left[\begin{smallmatrix} Y \\ X, Z \end{smallmatrix}\right]$, provided that $\mathrm{F}_X = \mathrm{F}_Y$, when X is treated as an independent one.

We introduce an operator to specify dependent delays, denoted by $|_|_D$, in which scope the stochastic delays in D are treated as dependent. Thus, in the previous example, $\left|\left[\begin{smallmatrix} X, Y \\ Z \end{smallmatrix}\right]\right|_X$ denotes that X is a dependent stochastic delay, but Y and Z are independent. By default, every delay is considered as dependent. Hence, $\left[\begin{smallmatrix} W \\ L \end{smallmatrix}\right]$ actually means $\left|\left[\begin{smallmatrix} W \\ L \end{smallmatrix}\right]\right|_{W \cup L}$. Multiple scope operators intersect and, e.g., $\left|\left|\left[\begin{smallmatrix} X \\ Y \end{smallmatrix}\right]\right|_X\right|_Y$ denotes the independent delay $\left[\begin{smallmatrix} X \\ Y \end{smallmatrix}\right]$ because $\{X\} \cap \{Y\} = \emptyset$.

The dependence scope plays an important role in giving operational semantics to the terms. Recall

that the stochastic delay prefix $[{}^{W}_{L}].p$ denotes an outcome of a race between the stochastic delays in $W \cup L$, where the winners are given by W and the losers are given by L. Moreover, it denotes that there was passage of time for the losing delays in L that may continue to persist in p. This means that the losers do not have their original distribution in the resulting process p and that their distributions must be 'aged' by the duration of the sample exhibited by the winners W. Therefore, the names of the losing delays must be protected in p, i.e., they become dependent. This is achieved by writing $|p|_{L}$ as the remaining term after the expiration of the delay given by $[{}^{W}_{L}]$. Thus, $[{}^{W}_{L}].p$ is actually equivalent to $[{}^{W}_{L}].|p|_{L}$ as only the names in L must be preserved in p. Consequently, the stochastic delays not in L become independent. To support this interpretation of process terms, the stochastic delays that are not encompassed by any dependence scope are considered as dependent, i.e., $[{}^{W}_{L}].p$ is equivalent to $|[{}^{W}_{L}].p|_{W \cup L}$.

Timed delays in a racing context We first give an example of an execution of a stochastic delay.

Suppose that X is a random variable such that $\mathrm{P}(X{=}1) = \frac{1}{2}$ and $\mathrm{P}(X{=}2), \mathrm{P}(X{=}4), \mathrm{P}(X{=}5) = \frac{1}{6}$. We observe what happens after 1 unit of time. Then, either the stochastic delay expires with probability $\frac{1}{2}$ or it is aged by 1 time unit and it allows a passage of time as the random variable X', where $\mathrm{P}(X'{=}1),\ \mathrm{P}(X'{=}3),\ \mathrm{P}(X'{=}4) = \frac{1}{3}$. After one more time unit, the delay can either expire with probability that X did not expire in the first time unit multiplied by the probability that X' expired in the first time unit, i.e., $\mathrm{P}(X > 1) \cdot \mathrm{P}(X'{=}1) = \frac{1}{2} \cdot \frac{1}{3} = \frac{1}{6} = \mathrm{P}(X{=}2)$. We can proceed in the same fashion until we reach 5 time units with probability $\frac{1}{6}$.

Although being a simple exercise in probability, the example illustrates how to symbolically derive a stochastic delay using a timed delay of one unit of time. We denote by $\sigma^{X}_{\emptyset}$ the event where the delay expires in one time unit, i.e., the stochastic delay X wins a race in combination with a unit timed delay and there are no losers. By $\sigma^{\emptyset}_{X}$, we denote the event where the delay does not expire in one time unit, i.e., the stochastic delay X loses the race to a unit time delay and there are no additional winners. Then, at each point in time we have two possibilities: either the delay expires, or it does not expire and it is aged by one time unit. Intuitively, a stochastic delay prefix $[X].p$ can then be specified as $[X].p = \sigma^{X}_{\emptyset}.p + \sigma^{\emptyset}_{X}.[X].p$ for a given process term p. Note that the race of $\sigma^{X}_{\emptyset}$ and $\sigma^{\emptyset}_{X}$ is resolved. In a generalized context, following the same reasoning, we specify a stochastic delay prefix $[{}^{W}_{L}].p$ as

$$[{}^{W}_{L}].p = \sigma^{W}_{L}.p + \sigma^{\emptyset}_{W \cup L}.[{}^{W}_{L}].p\ .$$

Here, σ^{W}_{L} denotes the stochastic delays in W to be winning after one time unit delay with the stochastic delays in L losing. We will refer to σ^{W}_{L} as a timed delay in a racing context, or simply timed delay for short. Note that timed delays impose the same race condition as racing stochastic delays specified in their context. It turns out that in the process theory, it is sufficient to work only with timed delays and retrieve stochastic delays via guarded recursive specifications. We note that a timed delay of one time unit can be specified as $\sigma^{\emptyset}_{\emptyset}$. We omit the empty sets when clear from the context and we also write σ^{n} for $n \geq 1$ subsequent timed delays. We have to extend the resolved races condition to cover the situation when the set of winners is empty. So, we define that $\mathrm{rr}(\sigma^{W_1}_{L_1}, \sigma^{W_2}_{L_2})$ holds if $\mathrm{rr}([{}^{W_1}_{L_1}], [{}^{W_2}_{L_2}])$ holds, or $W_1 = \emptyset$ and $W_2 \cap L_1 \neq \emptyset$, or $W_2 = \emptyset$ and $W_1 \cap L_2 \neq \emptyset$.

Design choices The processes specified in our theory can perform timed delays, but can perform immediate actions as well, i.e., actions that do not allow any passage of time and can immediately (successfully) terminate. The choice between several actions is nondeterministic and depends on the environment as in standard process algebra. We favor time-determinism, i.e., the principle that passage of time alone cannot make a choice [4]. Also, we favor weak choice between immediate actions and

passage of time, i.e., we impose a nondeterministic choice on the immediate actions and the passage of time in the vein of the timed process algebras of [4]. To support maximal progress, i.e., to prioritize immediate actions over passage of time, we include a maximal progress operator in the theory together with encapsulation of actions, thereby disabling undesired actions. We derive delayable actions, similarly to stochastic delays, as recursive processes that can perform an immediate action at any point in time.

These design choices stem from timed process theory [4] as we aim to accomplish stochastic-time process theory as a conservative extension of real-time process theory. The conservative extension is an important prerequisite for co-existence of real- and stochastic-time delays as, otherwise, one must introduce them as separate constructs, e.g., similarly to the approach taken in MODEST with the introduction of stochastic timed automata [10].

3. Process Theory

In this section we introduce the process theory $\mathrm{TCP}^{\mathrm{dst}}$ of communicating processes with discrete real and stochastic time for race-complete process specifications that induce races with all possible outcomes. We refer the reader to [34, 35, 32] for the formal semantics. Here, we give several examples to guide the reader's intuition. To illustrate the theory we give the $G/G/1/\infty$ queue example.

Signature We continue by introducing the signature of the process theory $\mathrm{TCP}^{\mathrm{dst}}$. The deadlocked process is denoted by δ; successful termination by ϵ. Action prefixing is a unary operator scheme $a._-$, for every $a \in \mathcal{A}$, where $\mathcal{A}$ is the set of all possible actions. Similarly, timed delay prefixing is of the form $\sigma_{L}^{W}._-$ for $W, L \subseteq \mathcal{V}$ disjoint. The dependent delays scope operator scheme is given by $|_|_D$, for $D \subseteq \mathcal{V}$. The encapsulation operator scheme $\partial_H(_)$ for $H \subseteq \mathcal{A}$ suppresses the actions in H, whereas the maximal time progress operator scheme $\theta_H(_)$ gives priority to the actions in $H \subseteq \mathcal{A}$ over passage of time. The alternative composition is given by $_+_$, at the same time representing a nondeterministic choice between actions and termination, a weak choice between action and timed delays and a race condition for the timed delays. Parallel composition is given by $_ \parallel _$. It allows passage of time only if both components do so. Finally, we introduce guarded recursive variables as constants $R \in \mathcal{R}$.

The signature of $\mathrm{TCP}^{\mathrm{dst}}$ is given by

$$P ::= \delta \mid \epsilon \mid a.P \mid \sigma_{L}^{W}.P \mid |P|_{D} \mid \partial_H(P) \mid \theta_I(P) \mid P + P \mid P \parallel P \mid R,$$

where $a \in \mathcal{A}$, $W, L, D \subseteq \mathcal{V}$ with $W \cap L = \emptyset$, $H, I \subseteq \mathcal{A}$, and $R \in \mathcal{R}$. We write $\mathcal{C}$ for the closed terms.

Dependent and independent delays Before we present the process theory itself, we need some auxiliary operations to extract dependent and independent stochastic delays. By $\mathrm{D}(p)$ we denote the set of dependent delays of the term $p \in \mathcal{C}$, by $\mathrm{I}(p, \mathcal{V})$ ($\mathrm{I}(p)$ for short) its set of independent delays. The racing delays of a term are denoted by $\mathrm{R}(p) = \mathrm{D}(p) \cup \mathrm{I}(p)$. The functions $\mathrm{D}(p)$ and $\mathrm{I}(p)$ are given by

$$\begin{aligned}
&\mathrm{D}(\epsilon) = \mathrm{D}(\delta) = \mathrm{D}(a.p) = \emptyset, \quad \mathrm{D}(|p|_D) = \mathrm{D}(p) \cap D, \quad \mathrm{D}(\sigma_{L}^{W}.p) = W \cup L,\\
&\mathrm{D}(\partial_H(p)) = \mathrm{D}(\theta_H(p)) = \mathrm{D}(p), \quad \mathrm{D}(p_1 + p_2) = \mathrm{D}(p_1 \parallel p_2) = \mathrm{D}(p_1) \cup \mathrm{D}(p_2);\\
&\mathrm{I}(\epsilon, D) = \mathrm{I}(\delta, D) = \mathrm{I}(a.p, D) = \emptyset, \quad \mathrm{I}(\sigma_{L}^{W}.p, D) = (W \cup L) \setminus D, \quad \mathrm{I}(|p|_D, D') = \mathrm{I}(p, D \cap D'),\\
&\mathrm{I}(\partial_H(p), D) = \mathrm{I}(\theta_H(p), D) = \mathrm{I}(p, D), \quad \mathrm{I}(p_1 + p_2, D) = \mathrm{I}(p_1 \parallel p_2, D) = \mathrm{I}(p_1, D) \cup \mathrm{I}(p_2, D).
\end{aligned}$$

The dependent delays are computed as the delays connected by the outermost alternative or parallel composition that are not encapsulated by the scope operator. The delays that are in the scope operator

must be in the intersection of all dependence binding sets. For the independent delays we need an auxiliary set as a second parameter to keep track of this intersection [32]. We illustrate the situation by an example. Let $p = ||\sigma^X_{Y,Z}\delta|_{X,Z}|_{X,Y}$. Then $\mathrm{D}(p) = \{X\}$ and $\mathrm{I}(p) = \{Y, Z\}$ as $\{X, Z\} \cap \{X, Y\} = \{X\}$.

Renaming of independent delays The general idea of having both dependent and independent delays available is the following: For specification one can use multiple instances of a component using independent delays. As the delays are independent, there is no need to worry about the actual samples. For analysis however, it is advantageous to deal with dependent delays. For example, given the simple component $|\sigma^X_Y.\sigma^Y.a.\delta|_\emptyset$, we can use it as a building block of the system $|\sigma^X_Y.\sigma^Y.a.\delta|_\emptyset \parallel |\sigma^X_Y.\sigma^Y.a.\delta|_\emptyset$. However, for analysis we revert to the system $|\,(\sigma^X_Y.\sigma^Y.a.\delta) \parallel (\sigma^U_V.\sigma^V.a.\delta)\,|_\emptyset$, where $\mathrm{F}_X{=}\mathrm{F}_U$ and $\mathrm{F}_Y{=}\mathrm{F}_V$, in order to resolve the race condition. Note that proper resolution of the race condition requires uniqueness of names of the racing delays (cf. [34, 35]). It is clear that naming conflicts may arise when one puts the entire process under one scope operator, as in the example above. Therefore, it has to be checked whether there are independent delays with the same names. If such conflicts occur, then the independent delays introducing the clash must be renamed. Care has to be taken, that losing delays are renamed consistently as their names have been bound by the first race in which they participated. To this end, we define a renaming operation $p[Y/X]$ for $p \in \mathcal{C}$, that consistently renames the stochastic delay X into Y. We have

$$(\sigma^W_L.p)[Y/X] = \sigma^W_L.p \quad \text{if } X \notin W \cup L$$

$$(\sigma^W_L.p)[Y/X] = \sigma^{(W\setminus\{X\})\cup\{Y\}}_L.p \quad \text{if } X \in W \qquad |p|_D[Y/X] = |p[Y/X]|_D \quad \text{if } X \notin D$$

$$(\sigma^W_L.p)[Y/X] = \sigma^W_{(L\setminus\{X\})\cup\{Y\}}.p[Y/X] \quad \text{if } X \in L \qquad |p|_D[Y/X] = |p[Y/X]|_{(D\setminus\{X\})\cup\{Y\}} \quad \text{if } X \in D$$

where the other cases are straightforward.

Operational semantics We use a construct, called an *environment*, to keep track of the ages of the racing delays. Recall, σ^W_L denotes a unit delay after which a race was won by W and lost by L, for $W, L \subseteq \mathcal{V}$. However, because of time determinism, time passes equally for all racing delays in $W \cup L$ aging them by units of time. To denote that after a delay $\left[{W \atop L}\right]$, the same time that passed for the winners W has also passed for the losers L, we use an environment $\alpha\colon \mathcal{V} \to \mathbb{N}$. For each $X \in \mathcal{V}$, $\alpha(X)$ represents the amount of time that X has raced. We write $\mathcal{E}_\mathrm{s}$ for the set of all environments.

For example, the process term $\sigma^{X,Y}_Z.\sigma^U_Z.p$ has a racing timed transition in which X and Y are the winners and Z is the loser. In the resulting process $\sigma^U_Z.p$, the variable Z must be made dependent on the amount of time that has passed. This is denoted by $\alpha(Z) = 1$, provided that originally $\alpha(Z) = 0$. As Z again loses a race, this time to U, the transition induced by σ^U_Z updates $\alpha(Z)$ to 2.

The environment does not affect the outgoing transitions. It is used to calculate the correct distribution of the racing delays. The distribution of X, provided that $F_X(\alpha(X)) < 1$, at that point in time is given by $\mathrm{F}_X(n) = \frac{\mathrm{F}_X(n+\alpha(X)) - \mathrm{F}_X(\alpha(X))}{1-\mathrm{F}_X(\alpha(X))}$ for $n \in \mathbb{N}$. Thus, in order to compute the updated distribution of a racing delay X, one has to know its age.

The semantics of process terms is given by racing timed transition schemes. A state s of the transition scheme in an environment α is given by the pair $\langle s, \alpha\rangle \in S \times \mathcal{E}_\mathrm{s}$. The function $\mathrm{I}(s)$ gives the set of independent delays of the state s. Every state may have a termination option, denoted by the predicate $\downarrow$. There are two types of transitions: (1) $\xrightarrow{a}$, immediate action transitions labeled by $a \in \mathcal{A}$, that do not allow passage of time and model undelayable action prefixes; and (2) $\overset{W}{\underset{L}{\longmapsto}}$, (resolved) racing timed delay transitions, driven by the winners W and the losers L, that model racing timed delay prefixes. The timed delay transitions must be well-defined: for every $u \overset{W}{\underset{L}{\longmapsto}} u'$, the set of winners W and the set of losers L are

disjoint. Moreover, every two different transitions originating from the same state are in a resolved race. More precisely, if $u \overset{W_1}{\underset{L_1}{\longmapsto}} u_1 \neq u \overset{W_2}{\underset{L_2}{\longmapsto}} u_2$, then $\mathrm{rr}(\sigma^{W_1}_{L_1}, \sigma^{W_2}_{L_2})$ holds, implying that $W_1 \cup L_1 = W_2 \cup L_2$. Thus, for every state s there exists a set of racing delays $\mathrm{R}(s)$ satisfying $\mathrm{R}(s) = W \cup L$ for every $\langle s, \alpha\rangle \overset{W}{\underset{L}{\longmapsto}} \langle s', \alpha'\rangle$. Then, the set of dependent delays is given by $\mathrm{D}(s) = \mathrm{R}(s) \setminus \mathrm{I}(s)$.

We define a strong bisimulation relation on racing timed transition schemes. It requires racing timed delays to have the same age modulo names of the independent delays. This ensures that the induced races have the same probabilistic behavior. As usual, bisimilar terms are required to have the same termination options, action and timed transitions [37, 4].

A symmetric relation R on $S \times \mathcal{E}_\mathrm{s}$ is a bisimulation if, for every two states u_1, u_2 such that $R(u_1, u_2)$, it holds that: (1) if $u_1\!\downarrow$ then $u_2\!\downarrow$; (2) if $u_1 \xrightarrow{a} u_1'$ for some $u_1' \in S \times \mathcal{E}_\mathrm{s}$, then $u_2 \xrightarrow{a} u_2'$ for some $u_2' \in S \times \mathcal{E}_\mathrm{s}$; and (3) if $u_1 \overset{W_1}{\underset{L_1}{\longmapsto}} u_1'$ for some $u_1' \in S \times \mathcal{E}_\mathrm{s}$, then $u_2 \overset{W_2}{\underset{L_2}{\longmapsto}} u_2'$ for some $u_2' \in S \times \mathcal{E}_\mathrm{s}$. Moreover, u_1' and u_2' in (2)–(3) are again related by R. In (3) W_1 and L_1 differ from W_2 and L_2, respectively, only in the names of the independent racing delays, while comprising delays with the same distributions and ages. Also, an additional condition is imposed to ensure that the ages of the losers of u_1 that are racing as dependent delays in u_1' is preserved in u_1' as well. Two states u_1 and u_2 are bisimilar if there exists a bisimulation relation R that relates them. The complete technical details can be found in [32].

Axiomatization By now, we have gathered all the prerequisites to present the axioms for the operators, except for $_ \parallel _$ and $\theta_H(_)$. (These operators will be dealt with using the expansion laws discussed below for normal forms in which races are resolved.) Table 1 displays the axioms for the sequential processes. Axioms A1, A2, and A3 are standard. Axiom A4 states that there is no dependence of stochastic delays arising from an action. Axiom A5 states that all delays are treated as dependent by default. Axiom A6 states that the losers of a timed delay retain their names in the remaining process. Axiom A7 states that multiple scope operators intersect. Axiom A8 states that independent winning delays can be renamed into fresh names with the same distribution. Axiom A9 is similar but now the renamed losing stochastic delay must be consistently renamed in the remainder too. Axiom A10 puts stochastic delays in the same name space under the condition that there are no naming conflicts.

The standard axioms for associativity, commutativity, deadlock as the neutral element for the alternative composition, and the idempotence of the termination are given by the axioms A11–A14. Axiom A15 shows that a choice between the same alternatives is not a choice. Axioms A16–A18 show how races are resolved. In the case of A16 the winners have common variables, so they must win together provided that the joint stochastic delay is well-defined, i.e., there are no common stochastic delays between the winners and the losers. Note that in the remaining process p_i only the names of its losers L_i need to be preserved. Axiom A17 states that if the losers of the first timed delay have a common delay with the winners of the second, then all delays of the second delay are losers in the resulting delay. The last axiom states the result of a race in which there are no common variables between the winners and the losers of both timed delays. In that case, all outcomes of the race are possible. Finally, the axioms A19–A21 give the standard axioms for the encapsulation operator that suppresses the actions in H.

Head normal form Using the axioms, we can represent every term $p \in \mathcal{C}$ as $|p'|_B$, where $B \subseteq \mathrm{D}(p)$, and p' has the following head normal form

$$\textstyle\sum_{i=1}^{m} a_i.|p_i|_\emptyset + \sum_{j=1}^{n} \sigma^{W_j}_{L_j}.|q_j|_{D_j} \; (+\,\epsilon)\,,$$

with $\mathrm{rr}(\sigma^{W_k}_{L_k}, \sigma^{W_\ell}_{L_\ell})$ for $1 \leq k < \ell \leq n$ and $D_j \subseteq L_j \cap \mathrm{D}(q_j)$, and p_i and q_j for $1 \leq i \leq m$, $1 \leq j \leq n$ are again in head normal form; the summand ϵ is optional and $\sum_{i=1}^{m} p_i$ is shorthand for $p_1 + \ldots + p_m$

$$|\delta|_D = \delta \quad \mathbf{A1}, \qquad |\epsilon|_D = \epsilon \quad \mathbf{A2}, \qquad |a.p|_D = a.p \quad \mathbf{A3}, \qquad a.p = a.|p|_\emptyset \quad \mathbf{A4}$$

$$\sigma_L^W.p = |\sigma_L^W.p|_{W \cup L} \quad \mathbf{A5}, \qquad \sigma_L^W.p = \sigma_L^W.|p|_L \quad \mathbf{A6}, \qquad ||p|_{D_1}|_{D_2} = |p|_{D_1 \cap D_2} \quad \mathbf{A7}$$

$$|\sigma_L^{W \cup \{X\}}.p|_D = |\sigma_L^{W \cup \{Y\}}.p|_D \qquad \text{if } X, Y \notin W \cup D \text{ and } \mathrm{F}_X = \mathrm{F}_Y \quad \mathbf{A8}$$

$$|\sigma_{L \cup \{X\}}^W.p|_D = |\sigma_{L \cup \{Y\}}^W.p[Y/X]|_D \qquad \text{if } X, Y \notin L \cup D \text{ and } \mathrm{F}_X = \mathrm{F}_Y \quad \mathbf{A9}$$

$$|p_1 + p_2|_D = |p_1|_D + |p_2|_D \qquad \text{if } \mathrm{I}(|p_1|_D) \cap \mathrm{R}(|p_2|_D) = \mathrm{R}(|p_1|_D) \cap \mathrm{I}(|p_2|_D) = \emptyset \quad \mathbf{A10}$$

$$(p + q) + r = p + (q + r) \quad \mathbf{A11}, \qquad p + q = q + p \quad \mathbf{A12}$$

$$p + \delta = p \quad \mathbf{A13}, \qquad \epsilon + \epsilon = \epsilon \quad \mathbf{A14}, \qquad a.p + a.p = a.p \quad \mathbf{A15}$$

$$\sigma_{L_1}^{W_1}.p_1 + \sigma_{L_2}^{W_2}.p_2 = \sigma_{L_1 \cup L_2}^{W_1 \cup W_2}.(|p_1|_{L_1} + |p_2|_{L_2}) \qquad \text{if } W_1 \cap W_2 \neq \emptyset \text{ and } W_1 \cap L_2 = L_1 \cap W_2 = \emptyset \quad \mathbf{A16}$$

$$\sigma_{L_1}^{W_1}.p_1 + \sigma_{L_2}^{W_2}.p_2 = \sigma_{L_1 \cup W_2 \cup L_2}^{W_1}.(|p_1|_{L_1} + |p_2|_{L_2}) \qquad \text{if } L_1 \cap W_2 \neq \emptyset \text{ and } W_1 \cap W_2 = W_1 \cap L_2 = \emptyset \quad \mathbf{A17}$$

$$\begin{aligned}\sigma_{L_1}^{W_1}.p_1 + \sigma_{L_2}^{W_2}.p_2 = {} & \sigma_{W_2 \cup L_2 \cup L_1}^{W_1}.(|p_1|_{L_1} + |p_2|_{L_2}) + \sigma_{L_1 \cup L_2}^{W_1 \cup W_2}.(|p_1|_{L_1} + |p_2|_{L_2}) + \\ & \sigma_{L_2 \cup W_1 \cup L_1}^{W_2}.(|p_1|_{L_1} + |p_2|_{L_2}) \qquad \text{if } W_1 \cap W_2 = L_1 \cap W_2 = W_1 \cap L_2 = \emptyset \quad \mathbf{A18}\end{aligned}$$

$$\partial_H(\delta) = \delta \quad \mathbf{A19}, \qquad \partial_H(\epsilon) = \epsilon \quad \mathbf{A20}, \qquad \partial_H(p_1 + p_2) = \partial_H(p_1) + \partial_H(p_2) \quad \mathbf{A21}$$

$$\partial_H(\sigma_L^W.p) = \sigma_L^W.\partial_H(p) \quad \mathbf{A22}, \quad \partial_H(a.p) = \delta \text{ if } a \in H \quad \mathbf{A23}, \quad \partial_H(a.p) = a.\partial_H(p) \text{ if } a \notin H \quad \mathbf{A24}$$

Table 1. Axioms for sequential processes

if $m > 0$, or δ otherwise. The availability of a head normal form is technically important. On the one hand, it shows the possible outcomes of the race explicitly. On the other hand, it is instrumental for the uniqueness of guarded recursive specifications in the term model [5]. Below, we use it to provide an expansion law for the parallel composition and the maximal progress operator.

Expansion laws Let $\bar{p}_1 = |p|_D$ and $\bar{p}_2 = |p'|_{D'}$, where $D \subseteq \mathrm{D}(p)$, $D' \subseteq \mathrm{D}(p')$, and $\mathrm{I}(\bar{p}_1) \cap \mathrm{R}(\bar{p}_2) = \mathrm{R}(\bar{p}_1) \cap \mathrm{I}(\bar{p}_2) = \emptyset$, and assume that for p and p' we have the head normal forms $p = \sum_{i=1}^m a_i.p_i + \sum_{j=1}^n \sigma_{L_j}^{W_j}.q_j(+\epsilon)$ and $p' = \sum_{k=1}^{m'} a'_k.p'_k + \sum_{\ell=1}^{n'} \sigma_{L'_\ell}^{W'_\ell}.q'_\ell(+\epsilon)$, with $p_i = |\hat{p}_i|_\emptyset$, $q_j = |\hat{q}_j|_{D_j}$, $p'_k = |\hat{p}'_k|_\emptyset$, and $q'_\ell = |\hat{q}'_\ell|_{D'_\ell}$. The expansion of the parallel composition $\bar{p}_1 \parallel \bar{p}_2$ of $\bar{p}_1$ and $\bar{p}_2$ is then given by $\bar{p}_1 \parallel \bar{p}_2 = |p \parallel p'|_{D \cup D'}$, where

$$\begin{aligned}p \parallel p' = {} & \textstyle\sum_{i=1}^m a_i.(p_i \parallel p') + \sum_{k=1}^n a'_k.(p \parallel p'_k) + \sum_{\gamma(a_i, a'_k) \text{ def.}} \gamma(a_i, a'_k).(p_i \parallel p'_k)(+\epsilon) + \\ & \textstyle\sum_{W_j \cap W'_\ell \neq \emptyset, W_j \cap L'_\ell = L_j \cap W'_\ell = \emptyset} \sigma_{L_j \cup L'_\ell}^{W_j \cup W'_\ell}.(|q_j|_{L_j} \parallel |q'_\ell|_{L'_\ell}) + \\ & \textstyle\sum_{L_j \cap W'_\ell \neq \emptyset, W_j \cap W'_\ell = W_j \cap L'_\ell = \emptyset} \sigma_{L_j \cup W'_\ell \cup L'_\ell}^{W_j}.(|q_j|_{L_j} \parallel |q'_\ell|_{L'_\ell}) + \\ & \textstyle\sum_{W_j \cap L'_\ell \neq \emptyset, W'_\ell \cap W_j = W'_\ell \cap L_j = \emptyset} \sigma_{W_j \cup L_j \cup L'_\ell}^{W'_\ell}.(|q_j|_{L_j} \parallel |q'_\ell|_{L'_\ell}) +\end{aligned}$$

$$\sum_{W_j \cap W'_\ell = W_j \cap L'_\ell = L_j \cap W'_\ell = \emptyset} \Big(\sigma^{W_j}_{L_j \cup W'_\ell \cup L'_\ell}.(|q_j|_{L_j} \parallel |q'_\ell|_{L'_\ell}) + \sigma^{W_j \cup W'_\ell}_{L_j \cup L'_\ell}.(|q_j|_{L_j} \parallel |q'_\ell|_{L'_\ell}) + \sigma^{W'_\ell}_{W_j \cup L_j \cup L'_\ell}.(|q_j|_{L_j} \parallel |q'_\ell|_{L'_\ell}) \Big)$$

and the optional ϵ summand exists only if it exists in both p and p'.

The expansion law of the maximal progress $\theta_I(p)$ [4] is given by $\theta_I(p) = |\theta_I(p')|_D$, where

$$\theta_I(p') = \begin{cases} \sum_{i=1}^m a_i.\theta_I(p_i)(+\epsilon), & \text{if } a_i \in H \text{ for some } i \\ \sum_{i=1}^m a_i.\theta_I(p_i) + \sum_{j=1}^n \sigma^{W_j}_{L_j}.\theta_I(q_j)(+\epsilon), & \text{otherwise,} \end{cases}$$

and the optional ϵ summand exists if it exists in p.

Guarded recursion and delayable actions We introduce recursive specifications by means of sets of recursive equations. We only consider guarded recursive specifications. So, every recursive variable must be prefixed by either an action or by a timed delay in the specification. Such specifications have unique solutions in the so-called term model, relying on the existence of the head normal form [5, 32].

We define a set of delayable actions $\{\, \underline{a} \mid a \in \mathcal{A} \,\}$ by taking $\underline{a}.p$ to be the solution of the guarded recursive equation: $R = a.p + \sigma.R$. Thus, $\underline{a}(p) = a.p + \sigma.\underline{a}(p)$.

Stochastic delays We specify stochastic delays similarly to delayable actions above. We put

$$\tbinom{W}{L}(p) = \sigma^W_L.p + \sigma_{W \cup L}.\tbinom{W}{L}(p)\,,$$

and define $\tbinom{W}{L}.p$ as the solution of the above equation.

An example illustrates how to specify the desired stochastic behavior in this fashion. We consider the processes $R_1 = [X](p) + [Y](q)$ and $R_2 = \tbinom{X}{Y}(|p|_\emptyset + [Y](q)) + [X,Y](p+q) + \tbinom{Y}{X}([X](p) + |q|_\emptyset)$. The solutions of R_1 and R_2 are

$$R_1 = \sigma^X_Y.(|p|_\emptyset + [Y](q)) + \sigma^{X,Y}.(p+q) + \sigma^Y_X.([X](p) + |q|_\emptyset) + \sigma_{X,Y}.R_1$$
$$R_2 = \sigma^X_Y.(|p|_\emptyset + [Y](q)) + \sigma^{X,Y}.(p+q) + \sigma^Y_X.([X](p) + |q|_\emptyset) + \sigma_{X,Y}.R_2.$$

In absence of timed delays, we can manipulate the stochastic delays directly without having to resort to the recursive specifications at all (as it was originally proposed in [34, 35] and ground-completely axiomatized in [32]). For example,

$$\tbinom{W_1}{L_1}(p_1) + \tbinom{W_2}{L_2}(p_2) = \tbinom{W_1 \cup W_2}{L_1 \cup L_2}(|p_1|_{L_1} + |p_2|_{L_2}) \quad \text{if } W_1 \cap W_2 \neq \emptyset \text{ and } W_1 \cap L_2 = L_1 \cap W_2 = \emptyset$$
$$\tbinom{W_1}{L_1}(p_1) + \tbinom{W_2}{L_2}(p_2) = [_{L_1 \cup W_2 \cup L_2}^{W_1}](|p_1|_{L_1} + \tbinom{W_2}{L_2}(p_2)) \quad \text{if } L_1 \cap W_2 \neq \emptyset \text{ and } W_1 \cap W_2 = W_1 \cap L_2 = \emptyset$$
$$\tbinom{W_1}{L_1}(p_1) + \tbinom{W_2}{L_2}(p_2) = [_{W_2 \cup L_2 \cup L_1}^{W_1}](|p_1|_{L_1} + \tbinom{W_2}{L_2}(p_2)) + \tbinom{W_1 \cup W_2}{L_1 \cup L_2}(|p_1|_{L_1} + |p_2|_{L_2}) + [_{L_2 \cup W_1 \cup L_1}^{W_2}](\tbinom{W_1}{L_1}(p_1) + |p_2|_{L_2}) \quad \text{if } W_1 \cap W_2 = L_1 \cap W_2 = W_1 \cap L_2 = \emptyset$$

reflects how to deal with stochastic delay prefixes in the vein of the axioms A16–A18.

$G/G/1/\infty$ queue We proceed by specifying and solving the $G/G/1/\infty$ queue, also discussed in [34]. The queue is specified as $Q = \theta_I(\partial_H(A \parallel Q_0 \parallel S))$, where

$$A = [X](s_1.A), \quad S = \underline{r}_2([Y](s_3.S)), \quad Q_0 = \underline{r}_1(Q_1), \quad Q_{k+1} = \underline{r}_1(Q_{k+2}) + \underline{s}_2(Q_k) \quad \text{if } k \geq 0$$

and $H = \{s_1, r_1, s_2, r_2\}$ and $I = \{c_1, c_2, s_3\}$.

Let us first see how a stochastic delay synchronizes with a delayable action by solving the equation $C = \theta_I(\partial_H(A \parallel Q_0))$. We substitute the recursive specifications for $[X](s_1.A)$ and $\underline{r}_1(Q_1)$ and expand the parallel composition. We have $C = \sigma^{x}.c_1.C + \sigma_{x}.c_1.C$, i.e., $\theta_I(\partial_H(A \parallel Q_0)) = [X](c_1.\theta_I(\partial_H(A \parallel Q_1)))$. By using this result and the equations from above for handling stochastic delays, we obtain

$$Q = S_0 = [X](c_1.c_2.S_1),\ S_k = \tbinom{X}{Y}(c_1.S_{k+1}) + \tbinom{X,Y}{\emptyset}(c_1.s_3.c_2.S_k + s_3.c_1.c_2.S_k) + \tbinom{Y}{X}(s_3.c_2.S_{k-1}),$$

for $k > 0$ as the solution for the $G/G/1/\infty$ queue where $S_k = \theta_I(\partial_H(A \parallel Q_k \parallel [Y](s_3.S)))$. We note, however, that although the process terms specifying the queue are more elegant, the underlying racing timed transition system is similar to the transition system in [34] and retains the same level of complexity.

4. Performance Evaluation

For the purpose of performance analysis, we choose the framework of the language χ. It provides a means for Markovian analysis and discrete-event simulation from the same specification.

The language χ The language χ is a modeling language for control and analysis of industrial systems [8, 38]. It has been successfully applied to a large number of industrial cases, such as a car assembly line, a multi-product multi-process wafer fab [16], a fruit juice blending and packaging plant [22], and process industry factories [7]. Initially, χ came equipped with features for the modeling of discrete-event systems only, and was not supported by a formal semantics. Later, it was redesigned and converted to a formal timed specification language [11]. At present, χ can be characterized as a process algebra with data. In addition, it was extended to handle both discrete-event and continuous aspects, allowing for the modeling of hybrid systems [8].

Performance analysis of a χ model can be carried out either by simulation, or by analysis of the underlying continuous-time Markov (reward) chain. Simulation is a powerful method for performance analysis, but its disadvantages in comparison to analytical methods are well-known [6]. The approach based on Markov chains turns χ into a powerful Markovian process algebra in the vein of [25, 27]. It is analytical, and builds on a vast and well-established theory. However, the generation of a Markov chain from a χ model requires that all delays in the system are exponentially distributed. This is a serious drawback since in industrial systems, particularly in controllers, delays are often closer to being deterministic. Although it is possible to approximate deterministic delays by sequences of exponential delays, i.e., to model them by so-called phase-type distributions [36], this approach suffers from the state explosion problem. Many states are needed to approximate these delays sufficiently closely, and the generated Markov chain becomes large due to the full interleaving of stochastic transitions in parallel contexts.

Discrete-time probabilistic reward graphs In this paper, we build on an extension of the environment of timed χ proposed in [42] that employs discrete-time probabilistic reward graphs for long-run analysis of industrial systems. Here, we employ two methods introduced in [42] for long-run analysis of discrete-time probabilistic reward graphs by translation to discrete-time Markov reward chains [30]. The first one uses the notion of an unfolding that transforms each timed transition with duration n of the discrete-time probabilistic reward graph as a sequence of n time steps with probability 1 in the discrete-time Markov reward chain. The other one optimizes the former approach by replacing the timed delays with geometric

delays with the same mean. The former approach clearly increases the state space by introducing extra transitions, albeit in a specific manner, which can be exploited in the relevant computations. The latter translation does not increase the number of states, but as we discuss, is not suitable for transient analysis. In order to overcome this, we show how to obtain transient performance measures for 'unfoldings' of discrete-time probabilistic reward graphs by relating the transient measures of the obtained discrete-time Markov reward chain back to the original process.

Discrete-time probabilistic reward graphs have been proposed in [42] as a model for performance evaluation of industrial systems in which time delays are discrete and deterministic, while random behavior is expressed in terms of immediate probabilistic choices. Discrete-time probabilistic reward graphs are transition systems with two types of states: (1) probabilistic, which have finitely many probabilistic outgoing transitions and (2) timed, which have only one outgoing transition. In a discrete-time probabilistic reward graph, time itself does not decide a choice and, as such, there is no interleaving of timed transitions as in typical timed process algebras [3]. This is in contrast with the approach of Markovian process algebras, where all exponential delays are interleaved. As a consequence, compared to the Markovian approach which produces continuous-time Markov reward chains, the discrete-time probabilistic reward graph generated from a χ-model is considerably smaller (more than threefold for our case study). For our needs, we work with the following definition.

Definition 4.1. A discrete-time probabilistic reward graph is a tuple $\mathsf{G} = (\sigma, S, \dashrightarrow, \longmapsto, \rho)$, where (1) $\sigma \in \mathbb{R}^{1\times|S|}$ is an *initial state probability row vector* with $\sigma \geq 0$ and $\sigma\mathbf{1} = 1$; (2) $S = S_{\mathrm{p}} \cup S_{\mathrm{t}}$, where S_{p} and S_{t} are the disjoint sets of probabilistic and timed states, respectively; (3) $\dashrightarrow \subseteq S_{\mathrm{p}} \times (0,1] \times S$ is an (immediate) *probabilistic transition relation* with $\sum_{(s,p,s')\in\dashrightarrow} p = 1$ for every $s \in S_{\mathrm{p}}$; (4) $\longmapsto \subseteq S_{\mathrm{t}} \times \mathbb{N}^{+} \times S$ is a *timed transition relation* such that $s \overset{n}{\longmapsto} s'$ and $s \overset{m}{\longmapsto} s''$ (in infix notation) implies that $n = m$ and $s' = s''$; and (5) $\rho \in \mathbb{R}^{|S|\times 1}$ is a *state reward rate vector*.

The interpretation of a discrete-time probabilistic reward graph is as follows. In probabilistic states the process spends no time, and it jumps to another state according to the probabilistic transition relation. In a timed state the process spends as many time units as specified by the timed transition relation, and jumps to the unique subsequent state. The uniqueness requirement is to support the time-determinism property [37, 4, 3]. A reward is gained per time unit, as determined by the reward rate assigning function. Although we allow reward rates to be assigned also to probabilistic states, the process actually gains no reward as it spends no time in them. The aggregation method is capable of dealing with multiple subsequent and loops of probabilistic states, see Figure 2a. This provides for a better expressivity and modeling convenience [33]. These statements will also be supported by the aggregation method used below (cf. also [18, 41, 42]).

We visualize a discrete-time probabilistic reward graph as in Figure 2a. Here, states 1, 2, and 3 are timed, whereas states 4 and 5 are probabilistic. The reward rates are put in sans-serif at the top right corner of each state; the reward rate of the state i is r_{i}, for $1 \leq i \leq 5$.

Translation to discrete-time Markov reward chains To obtain the performance measures of a discrete-time probabilistic reward graph we exploit their relation with discrete-time Markov reward chains, which are well-established performance models. The discrete-time probabilistic reward graph is represented as an equivalent discrete-time Markov reward chain, which is then analyzed, and the results are interpreted back in the discrete-time probabilistic reward graph setting. The translation is performed in two steps: (1) the discrete-time probabilistic reward graph is transformed to a transition system to

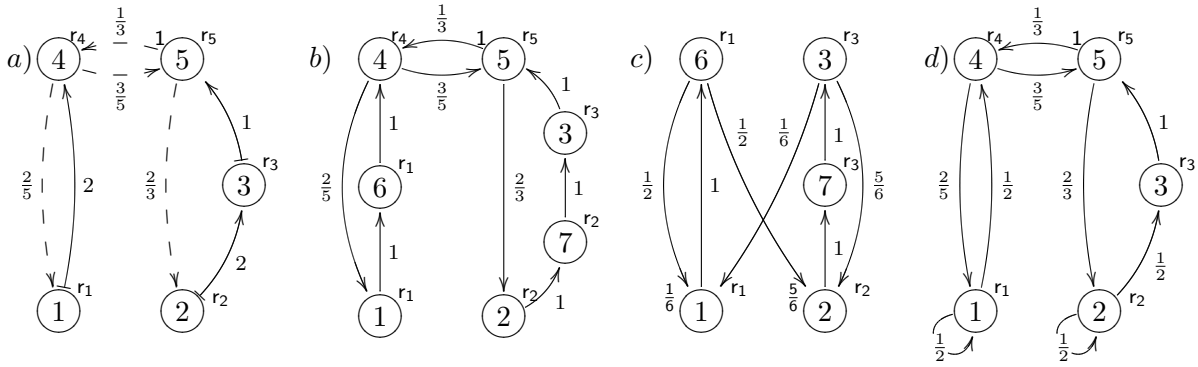

Figure 2. a) A discrete-time probabilistic reward graph, b) its unfolding, c) aggregated unfolding, and d) geometrization of a)

be interpreted as a discrete-time Markov reward chain, and (2) the discrete-time Markov reward chain is aggregated to truthfully represent the semantics of the discrete-time probabilistic reward graph as the immediate probabilistic transitions have to be eliminated. We need to interchangeably treat discrete-time Markov reward chains both as transition systems and in matrix terms. Here, we formally set up this framework and begin by defining a discrete-time Markov reward chain in terms of transition systems.

Definition 4.2. A discrete-time Markov reward chain $\mathsf{M} = (\sigma, S, \longrightarrow, \rho)$ is a tuple where (1) $\sigma \in \mathbb{R}^{1\times|S|}$ is the initial state probability row vector; (2) S is a finite set of states; (3) $\longrightarrow \subseteq S \times (0,1] \times S$ is the *probabilistic transition relation*; and (4) $\rho \in \mathbb{R}^{|S|\times 1}$ is the state reward vector.

Operationally, a discrete-time Markov reward chain waits one time unit in a state, gains the reward for this state determined by the reward vector ρ, and immediately jumps to another state with a probability specified by the relation $\longrightarrow$.

When required by the context, we will represent a discrete-time Markov reward chain as a triple (σ, P, ρ), where P is the probability transition matrix, i.e., the matrix representation of the probability transition relation, and ρ is the state reward vector. It is known that $\mathrm{P}(n)$, the transition probabilities after $n > 0$ time steps are given by $\mathrm{P}(n) = P^n$. Also, the long-run probability vector $\pi \in \mathbb{R}^{|S|}$, i.e., the average probability that the process resides in a given state after the system stabilizes, satisfies $\pi P = \pi$ [30, 17].

The main idea behind the translation from a discrete-time probabilistic reward graph G to a discrete-time Markov reward chain M is to represent a timed transition of duration n of G as a sequence of n states in M, connected by probabilistic transitions with probability 1, all having the same reward. The immediate probabilistic transitions of G remain unchanged by this transformation. Thus, the immediate probabilistic transitions of G are 'wrongly' transformed to probabilistic transitions of M that last one time unit. We come back to this problem later. First, we recall the naive transformation to a discrete-time Markov reward chain, which is referred to as the *unfolding* of a discrete-time probabilistic reward graph.

Definition 4.3. Let $\mathsf{G} = (\sigma_{\mathsf{G}}, S_{\mathsf{G}}, \dashrightarrow, \longmapsto, \rho_{\mathsf{G}})$ be a discrete-time probabilistic reward graph with $S_{\mathsf{G}} = \{s_1, \ldots, s_n\}$. Associate with every state $s_i \in S_{\mathsf{G}}$ a number $m_i \in \mathbb{N}^+$ as follows: if s_i is a probabilistic

state, then $m_i = 1$; if s_i is a timed state, then $m_i = m$ for the unique m such that $s_i \stackrel{m}{\longmapsto} s_k$, for some $s_k \in S_{\mathsf{G}}$. Then, the *unfolding* of G is the discrete-time Markov reward chain $\mathsf{U} = (\sigma_{\mathsf{U}}, S_{\mathsf{U}}, \longrightarrow, \rho_{\mathsf{U}})$ where $S_{\mathsf{U}} = \{ s_{ij} \mid 1 \leq i \leq n, 1 \leq j \leq m_i \}$ and (1) $\sigma_{\mathsf{U}}(s_{i1}) = \sigma_{\mathsf{G}}(s_i)$ and $\sigma_{\mathsf{U}}(s_{ij}) = 0$ for $1 < j \leq m_i$; (2) $s_{ij} \stackrel{1}{\longrightarrow} s_{ij+1}$ for $1 \leq j \leq m_i - 1$, and $s_{im_i} \stackrel{1}{\longrightarrow} s_{k1}$ if $s_i \stackrel{m}{\longmapsto} s_k$ or $s_{i1} \stackrel{p}{\longrightarrow} s_{k1}$ if $s_i \stackrel{p}{\dashrightarrow} s_k$; and (3) $\rho_{\mathsf{U}}(s_{ij}) = \rho_{\mathsf{G}}(s_i)$ for $1 \leq j \leq m_i$.

The set of probabilistic states of U is given by $S_{\mathsf{U},p} = \{s_{i1} \mid s_i \in S_{\mathsf{G},p}\}$ and the set of timed states is given by $S_{\mathsf{U},t} = S_{\mathsf{U}} \setminus S_{\mathsf{U},p}$. The *unfolding set* of s_i is given by $\mathrm{US}(s_i) = \{ s_{ij} \mid 1 \leq j \leq m_i \}$. The starting state of the unfolding of s_i is given by the function $\mathrm{us}(\mathrm{US}(s_i))$, which returns s_{i1}.

Remark 4.1. The states of the unfolding can be partitioned to probabilistic and timed states as in Definition 4.3. In the matrix representation $\mathsf{U} = (\sigma_{\mathsf{U}}, P, \rho_{\mathsf{U}})$, the transition matrix P induces two transition matrices P_{t} and P_{p}. The matrix P_{t} represents the unfolded timed transitions originating from timed states of $S_{\mathsf{G},\mathrm{t}}$, whereas P_{p} holds the translated immediate probabilistic transitions of the probabilistic states of $S_{\mathsf{G},\mathrm{p}}$. To obtain these matrices, the transition matrix P is first split to $P = P'_{\mathrm{t}} + P'_{\mathrm{p}}$ according to the timed and probabilistic transitions, respectively. The matrices P'_{t} and P'_{p} are adapted to transition matrices by adding 1s on the diagonal of the zero rows, where the other type of transitions is missing.

We illustrate the situation by an example.

Example 4.1. The unfolding of the discrete-time probabilistic reward graph from Figure 2a is given by the discrete-time Markov reward chain depicted in Figure 2b. The unfolded timed delays originating from states 1 and 2 introduce the new states 6 and 7, respectively. Here the set of timed states is $\{1, 2, 3, 6, 7\}$ and the set of probabilistic ones is $\{4, 5\}$. The timed and probabilistic transition matrices are given by

$$P_{\mathrm{t}} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \end{pmatrix} \qquad P_{\mathrm{p}} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ \frac{2}{5} & 0 & 0 & 0 & \frac{3}{5} & 0 & 0 \\ 0 & \frac{2}{3} & 0 & \frac{1}{3} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}.$$

As hinted above, the discrete-time Markov reward chain obtained by the unfolding, in general, does not truthfully represent the semantics of the original discrete-time probabilistic reward graph, in the sense that probabilistic states are immediate in the discrete-time probabilistic reward graph, whereas they last one unit of time in the discrete-time Markov reward chain. For example, in the discrete-time probabilistic reward graph in Figure 2a, state 5 can be reached from state 1 with probability $\frac{1}{2}$ after a delay of 2 time units (via $1 \stackrel{2}{\longmapsto} 4 \stackrel{1/2}{\dashrightarrow} 5$). However, in the unfolding this cannot be done in less than 3 time units (required for a sojourn in states 1, 6, and 4).

The solution to this problem is to eliminate the immediate probabilistic states appropriately. The elimination is achieved by the reduction-based aggregation method of [18, 41, 42], suitably adapted for the discrete-time setting [42]. Intuitively, in the new setting the method computes the accumulative probability of reaching one timed state from another and adjusts the delays. More specifically, the process

of aggregation is as follows: In an unfolding $\mathsf{U} = (\sigma, P, \rho)$ the transition probability matrix P is split to the transition matrices of the timed and probabilistic transitions P_{t} and P_{p}, respectively. Next, the Cesaro sum of the transition matrix induced by P_{p}, given by $\Pi = \lim_{n\to\infty} \frac{P_{\mathrm{p}}+P_{\mathrm{p}}^2+\ldots+P_{\mathrm{p}}^n}{n}$, is computed and its canonical product decomposition (L, R) is found (cf. [18, 41]). The canonical product decomposition is formally defined as follows.

Definition 4.4. Given a Markov chain $\mathsf{M} = (\sigma, P, \rho)$, such that $P = P_{\mathrm{t}} + P_{\mathrm{p}}$ for $P \in \mathbb{R}^{n\times n}$ as defined above, we define $\Pi = \lim_{n\to\infty} \frac{P_{\mathrm{p}}+P_{\mathrm{p}}^2+\ldots+P_{\mathrm{p}}^n}{n}$. Suppose $\mathrm{rank}(\Pi) = M$. Then, a canonical decomposition of Π is a pair of matrices (L, R) with $L \in \mathbb{R}^{M\times n}$ and $R \in \mathbb{R}^{n\times M}$ such that $L \geq 0$, $R \geq 0$, $\mathrm{rank}(L) = \mathrm{rank}(R) = M$, $L \times \mathbf{1} = \mathbf{1}$, and $\Pi = RL$.

Finally, the *aggregated* process is given by $\mathsf{M} = (\sigma R, LP_{\mathrm{t}}R, L\rho)$ as in [18, 41].

Remark 4.2. The Cesaro sum Π plays the role of the ergodic projection for the discrete-time case [30]. It represents the ergodic projection at one of the transition matrix P_{p} and it satisfies $\Pi P = P\Pi = \Pi$. This property is exploited for efficient computation. In [33] we also discuss the relationship between this approach and other approaches that eliminate immediate probabilistic state, e.g., vanishing states in Petri net theory [1]. There, we show that both methods converge in the limiting case when all immediate probabilistic states are eliminated, with the method employed in the setting of this paper being more general as there are no structural restrictions on the probabilistic transitions.

The next definition is adapted from [42].

Definition 4.5. Let G be a discrete-time probabilistic reward graph and $\mathsf{U} = (\sigma, P, \rho)$ be its unfolding where P induces P_{t} and P_{p}. Let $\Pi = \lim_{n\to\infty} \frac{P_{\mathrm{p}}+P_{\mathrm{p}}^2+\ldots+P_{\mathrm{p}}^n}{n}$. The translation by unfolding of G is the discrete-time Markov reward chain $\mathsf{M} = (\overline{\sigma}, \overline{P}, \overline{\rho})$, given by $\overline{\sigma} = \sigma R$, $\overline{P} = LPR$, and $\overline{\rho} = L\rho$, where (L, R) is a canonical product decomposition of Π.

The translation preserves the unfolding sets of the timed transitions of G and their starting states. Only the probabilistic states are eliminated and the transitions of the final states in the unfolding of the timed transitions in U are adjusted. Note that the unfolding has more states than the original process in the order of the sum of the duration of all timed transitions. We illustrate the translation by an example.

Example 4.2. The discrete-time Markov reward chain in Figure 2c is the aggregated chain of the one in Figure 2b. The aggregation eliminates the probabilistic states 4 and 5 and splits the incoming timed transitions from the states 6 and 3. The splitting is according to the accumulative (trapping) probabilities of 4 and 5 to the timed states 1 and 2 (which represent ergodic classes in the terminology of [18, 41]). Thus, in the aggregated chain there are two outgoing transitions from the states 6 and 3 to 1 and 2 (instead of a single one in the unfolding). The aggregation methods conform to the Markovian semantics that after a delay of one time unit there is an immediate probabilistic choice, which in the unfolding is explicitly stated by the immediate probabilistic transitions. It is straightforwardly checked that the discrete-time Markov reward chain in Figure 2c models the same system as the discrete-time probabilistic reward graph in Figure 2a when the discrete-time probabilistic reward graph is observed in the states 1, 2, and 3.

Remark 4.3. An alternative and more obvious, but possibly analytically and computationally intractable approach would be to translate and analyze discrete-time probabilistic reward graphs as deterministic semi-Markov reward chains [28]. However, to obtain the form of a semi-Markov reward chain, the aggregation by reduction still has to be applied to eliminate subsequent probabilistic transitions and probabilistic transitions must be introduced between subsequent timed transitions. Recently, a recurrence-relation-based tailored analysis approach for discrete-time semi-Markov processes has been proposed in [40].

The following lemma, adapted from [43], gives an important property of the long-run probability vector of the unfolding in terms of a relation between the states that belong to the same unfolding set. The result supports the assignment of the same reward to all states in an unfolding of a timed transition as in Definition 4.3.

Lemma 4.1. Let π be the long-run probability vector of the translation of a discrete-time probabilistic reward graph G. Then for every state $k \in S_{\mathsf{G},t}$ and $i, j \in \mathrm{US}(k)$ it holds that $\pi[i] = \pi[j]$.

Next, we recall how to relate the long-run performance measures of the translation back to the original process. Additionally, we show how to do the same in the transient case.

Performance metrics With the transformation to a discrete-time Markov reward chain in place, one can use the standard theory to compute performance measures. We focus on the expected reward rate at time step n or in the long-run.

If the resulting discrete-time Markov reward chain is ergodic, the expected reward at time step n is standardly computed as $\mathrm{R}(n) = \sigma \mathrm{P}(n) \rho$ and the long-run reward as $R^{\infty} = \pi\rho$, where (σ, P, ρ) is the translated discrete-time Markov reward chain, $\mathrm{P}(n)$ is its transition probability matrix, and π is its long-run probability vector [30]. In case the resulting process is not ergodic, one can always partition the original discrete-time probabilistic reward graph into subgraphs that produce ergodic and transient (or absorbing) processes, which themselves lead to ergodic processes, and analyze them separately. So, we do not consider the ergodicity condition as restrictive to our analysis and from now on we assume that we work only with ergodic processes when doing stationary analysis. After determining the performance metric, the obtained result has to be interpreted back in the discrete-time probabilistic reward graph setting.

This approach enables us to reason about the original discrete-time probabilistic reward graph G as we provide a backward relation between the discrete-time probabilistic reward graph G and its translation M. This is implemented by means of specially adapted distributor and collector matrices defined below (originally introduced as means to specify lumpings [30]). In our setting, they are employed as means to define the partition that is induced by the unfolded time transitions. The idea is to fold back the unfolded timed transitions and restore the effect of the probabilistic transitions in G by multiplying the transition matrix of M with these matrices. In that way, one can obtain the transition matrix of G and, consequently, its expected reward. As follows is the definition of this matrix and the required prerequisites. The approach is illustrated below in Example 4.3.

First we define the notions of a distributor and the collector (matrix). Given a partitioning of the state space of a discrete-time Markov chain, $\{C_1, \ldots, C_N\}$ say, we distinguish the following matrices. The collector matrix V defined as $V[i, j] = 1$ if $i \in C_j$, $V[i, j] = 0$ otherwise. The j-th column of V has an entry 1 for elements corresponding to states in C_j. A matrix U such that $U \geq 0$ and $UV = I$, with I denoting the identity matrix, is a distributor matrix for V. It can be readily seen that U is actually any

matrix of which the elements of the i-th row that correspond to elements in C_i sum up to 1, while the other elements of the row are 0.

The folding collector matrix of the unfolding U of G is defined as the collector of the partition induced by the unfolding sets. Due to the reduction-based aggregation, all probabilistic states have been eliminated to obtain the translation M. Consequently, the folding distributor and collector of U have too many states, as they also account for the already eliminated probabilistic transitions, and they have to be shrunk. Therefore, the rows and columns corresponding to the eliminated probabilistic transitions are omitted to obtain the folding distributor and collector of M.

The multiplication of the transition matrix of M with its folding collector produces the accumulative probability of residing in each unfolded timed state of M per unfolding set. So, the probabilities of residing in a timed state in the discrete-time probabilistic reward graph G can be extracted as the folded probability of the starting state of the unfolded timed transition. To carry this out, one has to multiply the folded transition matrix with the folding distributor to extract only the probabilities of the starting states. The folding distributor and collector matrices of the unfolding U and the translation M are defined as follows.

Definition 4.6. Let G be a discrete-time probabilistic reward graph, U its unfolding, and M its translation. The folding collector matrix V_{U} of U is given by $V_{\mathsf{U}}[i,j] = 1$ iff $j \in \mathrm{US}(i)$ and $V_{\mathsf{U}}[i,j] = 0$ otherwise, for $i,j \in S_{\mathsf{U}}$. The folding distributor U_{U} is given by $U_{\mathsf{U}}[i,j] = 1$ iff $j = \mathrm{us}(\mathrm{US}(i))$ and $U_{\mathsf{U}}[i,j] = 0$ otherwise. The folding distributor and collector matrix U_{M} and V_{M} of M are obtained by omitting the rows and columns of U_{U} and V_{U}, respectively, that correspond to the probabilistic states in $S_{\mathsf{U},p}$.

The folding collector V_{M} has the following property, which is a corollary of Lemma 4.1.

Corollary 4.1. Let G be a discrete-time probabilistic reward graph and M its translation. Let π be the long-run probability vector of M, V_{M} the folding collector of M, and U some distributor corresponding to V_{M}. Then, $\pi = \pi V_{\mathsf{M}} U$.

Intuitively, the corollary states that folding the long-run probabilities of the unfolded timed states in the translation can be done using the folding collector and an arbitrary distributor. So, we can reconstruct the behavior of the timed states in the original process G. However, the folding distributor and collector matrices cannot restore the behavior of the probabilistic states. Recall that we used the canonical decomposition (L,R) of the Cesaro sum Π to obtain the translation M from the unfolding U. To properly eliminate the effect of the probabilistic transitions the folding distributor U_{U} has to be multiplied by R to the right, obtaining $R_{\mathsf{M}} = U_{\mathsf{U}} R$, whereas the folding collector V_{U} is multiplied by L to the left obtaining $L_{\mathsf{M}} = L V_{\mathsf{U}}$.

Now, we have all prerequisites to propose a definition of $\mathrm{P}_{\mathsf{G}}(n)$, the transition matrix after n time steps of the discrete-time probabilistic reward graph G.

Definition 4.7. Let G be a discrete-time probabilistic reward graph, the discrete-time Markov reward chain U its unfolding and the discrete-time Markov reward chain M its translation by unfolding. Let (L,R) be the canonical decomposition of the transition matrix of probabilistic transitions of U and U_{U} and V_{U} the folding distributor and collector matrix. Then,

$$\mathrm{P}_{\mathsf{G}}(n) = R_{\mathsf{M}} \mathrm{P}_{\mathsf{M}}(n) L_{\mathsf{M}},$$

where $R_{\mathsf{M}} = U_{\mathsf{U}} R$ and $L_{\mathsf{M}} = L V_{\mathsf{U}}$ and $n \in \mathbb{N}$.

Notice that the matrices L_{M} and R_{M} no longer have the form of a distributor and a collector, unless every timed transition of G has a unit duration.

The following theorem gives the relation between the transient and long-run reward rate of a discrete-time probabilistic reward graph as induced by Definition 4.7 and the reward rates of its translation by unfolding. It supports Definition 4.7 and validates the calculation of $\mathrm{P}_{\mathsf{G}}(n)$.

Theorem 4.1. Let G be a discrete-time Markov reward chain and M its translation by unfolding. Then

$$\mathrm{R}_{\mathsf{G}}(n) = \mathrm{R}_{\mathsf{M}}(n) \qquad \text{and} \qquad R^{\infty}_{\mathsf{G}} = R^{\infty}_{\mathsf{M}}.$$

Proof:
We have $\sigma_{\mathsf{M}} = \sigma_{\mathsf{G}} R_{\mathsf{M}}$ and $\rho_{\mathsf{M}} = L_{\mathsf{M}} \rho_{\mathsf{G}}$, as can be seen from the definitions. By Corollary 4.1 we have for the long-run probability vector π_{G} that $\pi_{\mathsf{G}} = \pi_{\mathsf{M}} L_{\mathsf{M}}$. We obtain

$$R^{\infty}_{\mathsf{M}} = \pi_{\mathsf{M}} \rho_{\mathsf{M}} = \pi_{\mathsf{M}} L_{\mathsf{M}} \rho_{\mathsf{G}} = \pi_{\mathsf{G}} \rho_{\mathsf{G}} = R^{\infty}_{\mathsf{G}}.$$

Similarly, for the reward at time step $n \in \mathbb{N}$ we have

$$\mathrm{R}_{\mathsf{M}}(n) = \sigma_{\mathsf{M}} \mathrm{P}_{\mathsf{M}}(n) \rho_{\mathsf{M}} = \sigma_{\mathsf{G}} U_{\mathsf{M}} \mathrm{P}_{\mathsf{M}}(n) V_{\mathsf{M}} \rho_{\mathsf{G}} = \sigma_{\mathsf{G}} \mathrm{P}_{\mathsf{G}}(n) \rho_{\mathsf{G}} = \mathrm{R}_{\mathsf{G}}(n)\,.$$

This completes the proof. □

We illustrate the above by an example.

Example 4.3. The initial probability and reward vector of the discrete-time probabilistic reward graph depicted in Figure 2 are:

$$\sigma_{\mathsf{G}} = \begin{pmatrix} 0 & 0 & 0 & 0 & 1 \end{pmatrix} \qquad \rho_{\mathsf{G}} = \begin{pmatrix} r_1 & r_2 & r_3 & r_4 & r_5 \end{pmatrix}^{\mathsf{T}}$$

The folding distributor and collector matrix of the unfolding U in Figure 2b of the discrete-time probabilistic reward graph G in Figure 2a are given by U_{U} and V_{U} below with the canonical decomposition (L, R) of the Cesaro sum of the transition matrix of the immediate probabilistic transitions.

$$U_{\mathsf{U}} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \end{pmatrix} \quad V_{\mathsf{U}} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \end{pmatrix} \quad L = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \quad R = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ \frac{1}{2} & \frac{1}{2} & 0 & 0 & 0 \\ \frac{1}{6} & \frac{5}{6} & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}.$$

The folding distributor and collector matrices of the translation M depicted in Figure 2c are given by U_{M} and V_{M} and their adapted versions by R_{M} and L_{M} as follows:

$$U_{\mathsf{M}} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix} \quad V_{\mathsf{M}} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix} \quad R_{\mathsf{M}} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ \frac{1}{2} & \frac{1}{2} & 0 & 0 & 0 \\ \frac{1}{6} & \frac{5}{6} & 0 & 0 & 0 \end{pmatrix} \quad L_{\mathsf{M}} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \end{pmatrix}.$$

The initial probability vector σ_{M}, the transition matrix $\mathrm{P}_{\mathsf{M}}(3)$ at time step 3, and the reward vector ρ_{M} are given by

$$\sigma_{\mathsf{M}} = \sigma_{\mathsf{G}} R_{\mathsf{M}} = \begin{pmatrix} \frac{1}{6} & \frac{5}{6} & 0 & 0 & 0 \end{pmatrix} \qquad \mathrm{P}_{\mathsf{M}}(3) = \begin{pmatrix} 0 & 0 & 0 & \frac{1}{2} & \frac{1}{2} \\ \frac{1}{6} & \frac{5}{6} & 0 & 0 & 0 \\ \frac{1}{12} & \frac{1}{12} & \frac{5}{6} & 0 & 0 \\ \frac{1}{4} & \frac{1}{4} & \frac{1}{2} & 0 & 0 \\ 0 & \frac{1}{6} & \frac{5}{6} & 0 & \end{pmatrix} \qquad \rho_{\mathsf{M}} = L_{\mathsf{M}} \rho_{\mathsf{G}} = \begin{pmatrix} r_1 \\ r_2 \\ r_3 \\ r_1 \\ r_2 \end{pmatrix}.$$

For example, the probability transition matrix of G after 1, 2, and 3 time units is given by

$$\mathrm{P}_{\mathsf{G}}(1) = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ \frac{1}{2} & \frac{1}{2} & 1 & 0 & 0 \\ \frac{1}{2} & \frac{1}{2} & 0 & 0 & 0 \\ \frac{1}{6} & \frac{5}{6} & 0 & 0 & 0 \end{pmatrix} \qquad \mathrm{P}_{\mathsf{G}}(2) = \begin{pmatrix} \frac{1}{6} & \frac{5}{6} & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ \frac{1}{2} & \frac{1}{2} & 0 & 0 & 0 \\ \frac{1}{12} & \frac{5}{12} & \frac{1}{2} & 0 & 0 \\ \frac{1}{36} & \frac{5}{36} & \frac{5}{6} & 0 & 0 \end{pmatrix} \qquad \mathrm{P}_{\mathsf{G}}(3) = \begin{pmatrix} \frac{1}{6} & \frac{5}{6} & 0 & 0 & 0 \\ \frac{1}{2} & \frac{1}{2} & 0 & 0 & 0 \\ \frac{1}{12} & \frac{5}{12} & \frac{1}{2} & 0 & 0 \\ \frac{1}{3} & \frac{2}{3} & 0 & 0 & 0 \\ \frac{4}{9} & \frac{5}{9} & 0 & 0 & 0 \end{pmatrix}.$$

We can directly check the correspondence with the execution of the discrete-time probabilistic reward graph depicted in Figure 2. Note that the process never resides in the probabilistic states 4 and 5.

The long-run expected reward rate of the discrete-time probabilistic reward graph depicted in Figure 2a is obtained from the long-run probability vector π_{M} of its translation of Figure 2c. This vector is

$$\pi_{\mathsf{G}} = \pi_{\mathsf{M}} L_{\mathsf{M}} = \begin{pmatrix} \frac{1}{11} & \frac{3}{11} & \frac{3}{11} & \frac{1}{11} & \frac{3}{11} \end{pmatrix} L_{\mathsf{M}} = \begin{pmatrix} \frac{2}{11} & \frac{6}{11} & \frac{3}{11} & 0 & 0 \end{pmatrix}.$$

Note that the long-run probability vector of G has 0s for the places of the probabilistic states. The long-run expected reward rate of G is

$$R^{\infty}_{\mathsf{G}} = \pi_{\mathsf{G}} \rho_{\mathsf{G}} = \begin{pmatrix} \frac{2}{11} & \frac{6}{11} & \frac{3}{11} & 0 & 0 \end{pmatrix} \begin{pmatrix} r_1 & r_2 & r_3 & r_4 & r_5 \end{pmatrix}^{\mathsf{T}} = \frac{2}{11} r_1 + \frac{6}{11} r_2 + \frac{3}{11} r_3.$$

It is the same as the long-run probability vector of M, i.e.,

$$R^{\infty}_{\mathsf{M}} = \pi_{\mathsf{M}} \rho_{\mathsf{M}} = \begin{pmatrix} \frac{1}{11} & \frac{3}{11} & \frac{3}{11} & \frac{1}{11} & \frac{3}{11} \end{pmatrix} \begin{pmatrix} r_1 & r_2 & r_3 & r_1 & r_2 \end{pmatrix}^{\mathsf{T}} = \frac{2}{11} r_1 + \frac{6}{11} r_2 + \frac{3}{11} r_3.$$

The expected reward at time step 3 is

$$\sigma_{\mathsf{M}} \mathrm{P}_{\mathsf{M}}(3) \rho_{\mathsf{M}} = \frac{1}{6}(\frac{1}{2} r_1 + \frac{1}{2} r_2) + \frac{5}{6}(\frac{1}{6} r_1 + \frac{5}{6} r_2) = \frac{4}{9} r_1 + \frac{5}{9} r_2 = \sigma_{\mathsf{G}} \mathrm{P}_{\mathsf{G}}(3) \rho_{\mathsf{G}}.$$

We can visualize the full process of obtaining the performance measures of a discrete-time probabilistic reward graph by means of translation by unfolding in the left branch in Figure 3. In the figure we also depict the relation between the unfolded Markov reward chain and the original discrete-time probabilistic reward graph.

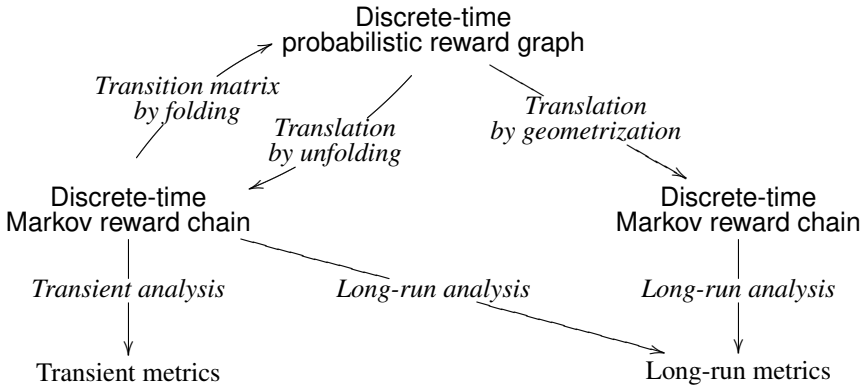

Figure 3. Performance measuring for discrete-time probabilistic reward graphs

The analysis of a discrete-time probabilistic reward graph by its translation to a discrete-time Markov reward chain using the approach described above introduces extra states that are required for the unfolding of the timed transitions. In the following section we give a brief overview of an optimized translation tailored for long-run analysis only.

Optimization by geometrization As discussed above the unfolding may have, in general, substantially more states than the original discrete-time probabilistic reward graph, as every delay of duration n introduces $n-1$ new states. To optimize the computation of long-run measures, a 'geometrization' of time delays is proposed in [42] to obtain a discrete-time Markov reward chain of, at most, the size of the original graph. The main idea is to replace discrete delays by geometrically distributed ones with the same mean instead of unfolding them.

The geometrization of a timed transition in G replaces the timed transition $s \stackrel{n}{\longmapsto} s'$ in G by two transitions $s \stackrel{1/n}{\longrightarrow} s'$ and $s \stackrel{(n-1)/n}{\longrightarrow} s$. This transformation induces a geometric sojourn time in the state with mean equal to the duration of the timed transition. As before, to obtain the final discrete-time Markov reward chain it is required to eliminate the probabilistic transitions. However, this translation is not adequate for transient analysis as it does not truthfully depict the semantics of G. Still, it was shown that the long-run expected reward of the discrete-time Markov reward chains obtained by translating the same discrete-time probabilistic reward graph by unfolding and geometrization is the same.

As an example, consider again the discrete-time probabilistic reward graph from Figure 2a. The discrete-time Markov reward chain in Figure 2d depicts its geometrization. The translation by geometrization is depicted by the right branch in Figure 3. The following theorem from [42] states that the two translations indeed commute, i.e., they give rise to discrete-time Markov reward chains with the same long-run performance measure.

Theorem 4.2. Let G be a discrete-time probabilistic reward graph, M_1 its translation by unfolding, and M_2 its translation by geometrization. Then $R^{\infty}_{\mathsf{M}_1} = R^{\infty}_{\mathsf{M}_2}$.

5. The Concurrent Alternating Bit Protocol

In this section, we specify the concurrent alternating bit protocol both in the process theory $\mathrm{TCP}^{\mathrm{dst}}$ and in the specification language χ. Our case study of the concurrent alternating bit protocol combines the process-algebraic setup of Section 2 and Section 3, on the one hand, and the performance evaluation framework of Section 4, on the other. By restricting to deterministic timed delays, we show how to analytically obtain transient performance measures. For the rest, we exploit discrete-event simulation in χ. For comparison purposes, we perform Markovian analysis using an extension of the χ toolset by turning all delays into exponential ones with mean values equal to the duration of the timed delays.

Protocol description The concurrent alternating bit protocol is used for communicating data along an unreliable channel with a guarantee that no information is lost relying on retransmission of data. An overview of the concurrent alternating bit protocol is depicted in Figure 4.

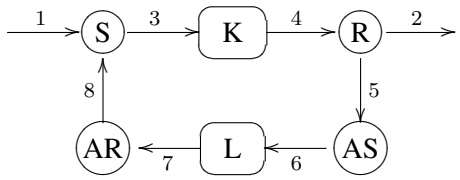

Figure 4. Scheme of the concurrent alternating bit protocol

```
sender ( c1, c3, c8: chan ) =
|[ altbit: bool = false, data: nat, ack: bool,
   tp: nat = 1, ts: nat = 10
 | c1?data; delay tp; c3!<data,altbit>;
   ( delay ts; c3!<data,altbit> |
     c8?ack; altbit := not altbit:
       c1?data; delay tp; c3!<data,altbit>
   )*; deadlock
]|
```

Figure 5. The sender process in χ

The arrival process sends the data at port 1 to the sender process S. The sender adds an alternating bit to the data and sends the package to receiver R via the channel K using port 3. It keeps re-sending the same package with a fixed timeout, waiting for the acknowledgement that the data has been correctly received. The channel K has some probability of failure and it transfers the data with a generally-distributed delay to the port 4. If the data is successfully received by R, then it is unpacked and the data is sent to the exit process via port 2. The alternating bit is sent as an acknowledgement back to the sender using the acknowledgement sender AS. The receiver R communicates with AS using port 5. The acknowledgement is sent via the unreliable channel L using port 6. Similarly to S, the acknowledgement process re-sends data after a fixed timeout. The acknowledgement is communicated to the acknowledgement receiver process AR. If the received acknowledgement is the one expected, then AR informs the sender S that it can start with the transmission of the next data package.

Process-algebraic specification We can specify, in the setup of Section 2 and 3, the concurrent alternating bit protocol as below for a data set D. Recall that the process theory does not contain an explicit probabilistic choice operator. To specify probabilistic behavior of the channel, we introduce timeouts to the channels K and L with duration t_k and t_ℓ, respectively. Thus, the messages are sent via the channels K and L before the timeout expires with a delay distributed according to the conditional random variables $\langle\, X \mid X < t_k \,\rangle$ and $\langle\, Y \mid Y < t_\ell \,\rangle$, respectively, or they get lost with probability $1 - \mathrm{F}_X(t_k)$, and $1 - \mathrm{F}_Y(t_\ell)$, respectively. Notably, to eliminate a possible nondeterministic choice in the timeout of

the channels (between two transitions labeled by i, see specification of K and L), it must be the case that $P(X = t_k) = 0$ and $P(Y = t_\ell) = 0$. The concurrent alternating bit protocol is specified as

$$\mathit{CABP} = \theta_I(\partial_H(S \parallel K \parallel R \parallel AS \parallel L \parallel AR))$$

with

$$\begin{aligned}
&S = S_0, \qquad S_b = \textstyle\sum_{d\in D}\ \underline{r}_1(d).\sigma^{t_p}.\underline{s}_3(d,b).T_{d,b}, \qquad T_{d,b} = \sigma^{t_s}.\underline{s}_3(d,b).T_{d,b} + \underline{r}_8(ack).S_{1\text{-}b}\\
&K = \textstyle\sum_{e\in D\times\{0,1\}}\ \underline{r}_3(e).\theta_i([X].i.\underline{s}_4(e).K + \sigma^{t_k}.i.K)\\
&R = R_0, \qquad R_b = \textstyle\sum_{d\in D}\ \underline{r}_4(d,b).\sigma^{t_r}.\underline{s}_5(ack).\underline{s}_2(d).R_{1\text{-}b} + \sum_{d\in D}\ \underline{r}_4(d,1\text{-}b).R_b\\
&AS = AS_1, \qquad AS_b = \underline{r}_5(ack).\underline{s}_6(1\text{-}b).AS_{1\text{-}b} + \sigma^{t_a}.\underline{s}_6(b).AS_b\\
&L = \textstyle\sum_{b\in\{0,1\}}\ \underline{r}_5(b).\theta_i([Y].i.\underline{s}_6(b).L + \sigma^{t_\ell}.i.L)\\
&AR = AR_0, \qquad AR_b = \underline{r}_7(b).\underline{s}_8(ack).AR_{1\text{-}b} + \underline{r}_7(1\text{-}b).AR_b,
\end{aligned}$$

where the recursion variables are parameterized by $d \in D$ and $b \in \{0, 1\}$,

$$\begin{aligned}
I \;=\;& \{\, r_1(d), s_2(d) \mid d \in D \,\} \cup \{\, c_3(d,b), c_4(d,b) \mid b \in \{0,1\}, d \in D \,\} \cup\\
& \{\, c_6(b), c_7(b) \mid b \in \{0,1\} \,\} \cup \{\, c_5(ack), c_8(ack) \,\}, \text{ and}\\
H \;=\;& \{\, s_3(d,b), s_4(d,b), r_3(d,b), r_4(d,b) \mid b \in \{0,1\}, d \in D \,\} \cup\\
& \{\, r_6(b), r_7(b), s_6(b), s_7(b) \mid b \in \{0,1\} \,\} \cup \{\, r_5(ack), r_8(ack), s_5(ack), s_8(ack) \,\}.
\end{aligned}$$

The deterministic timed delays with duration t_p, t_s, t_k, t_r, t_a, and t_ℓ represent the processing time of the sender, the timeout of the sender, the timeout of the data channel, the processing time of the receiver, the timeout of the acknowledgement sender, and the timeout of the acknowledgement channel. The internal action i enables the probabilistic choices induced by the timeouts as discussed above.

Specification and analysis in χ We illustrate some features of the language χ by discussing the χ specification of the sender process given in Figure 5. It is based on the version of timed χ of [11].

The process `sender` communicates with the other processes via three channels: `c1,c3,c8` (see Figure 4). The alternating bit is defined as a boolean variable and the data set is assumed to be the set of natural numbers. The sender waits for an arrival of a new data element, which it packs in `tp` time units. Afterwards, a frame with the data and the alternating bit is sent via channel `c3`. Here, the process enters the iterative construct represented by `(...)*` and it either resubmits the data every `ts` time units or it waits for an acknowledgement at channel `c8` from the acknowledgement receiver process. If the acknowledgement is received before the timeout expires, the process flips the alternating bit, packs the new data in `tp` time units, and sends it again via channel `c3`. Note that in the example, the processing time `tp` = 1 and the timeout `ts` = 10 time units.

The standard semantics of (discrete-event) χ is in terms of timed transition systems [8, 4]. The main idea underlying the construction of a discrete-time probabilistic reward graph from a timed transition system, as proposed here, is to hide all actions, i.e., to rename them to the special internal action τ, and then use the concept of timed branching bisimulation [3, 41] to reduce the system while abstracting from its internal transitions. If there is no real nondeterminism in the model, a timed transition system without any action labeled transition is obtained, i.e., a discrete-time probabilistic reward graph without probabilistic transitions. If there is one or more nondeterministic transition left, then the system is

underspecified. In that case, the resolution of the remaining nondeterministic choices depends on the environment, so its performance cannot be measured in the standard way. At this point, one can either revise the model to resolve the issue of underspecification or turn to performance analysis of processes comprising nondeterministic choices like the theory of Markov decision processes [28]. However, there the goal is to find an optimal scheduler for the nondeterministic transitions in order to achieve a given goal, a topic which is beyond the scope of this paper.

Since χ has no features to model probabilistic choice, the random behavior of the data and acknowledgement channel is modeled in χ by a nondeterministic choice. When the corresponding discrete-time probabilistic reward graph is generated from the χ model these nondeterministic choices must be appropriately replaced by probabilistic ones. For this we slightly adjust the method described in the previous paragraph. Instead of hiding all actions, the special actions used to indicate probabilistic branching remain visible. After the minimization, the probabilities that were intentionally left out are put as labels on the nondeterministic transitions, see Figure 6 below. Again, if there is still nondeterminism remaining in the model, we cannot proceed the performance analysis. Note that although the method is not always sound (in case of multiple probabilistic transitions leaving from the same state) as it requests manipulation on the resulting graph, it serves its purpose for this and similar examples. Of course, another approach is to extend χ with an explicit probabilistic choice operator (e.g., the one in [24]). However, this requires drastic changes of the language and tools, and as such goes beyond the scope of this paper. Notably, the framework makes use of probabilistic choices, but only for simulation purposes.

The standard χ language does not directly support reward specification either. We take a similar approach as for the absence of a probabilistic choice, and add rewards by manipulating the χ specification (again side-stepping changes in χ), see Figure 6 below. We add, for each reward criterion, an ever repeating parallel component to the specification. The result is that in the timed transition system yielded, every state has a self-loop labeled by a special action denoting the reward rate of the state. These actions will are not hidden by branching bisimulation reduction. As in the case for the probabilistic choice, a systematic technique rendering the above can in principle be incorporated into the χ environment.

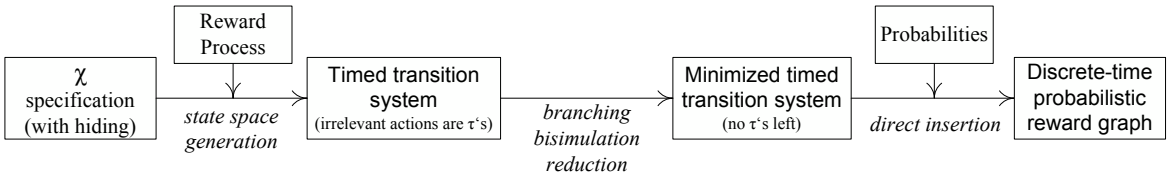

Figure 6. Generation of a discrete-time probabilistic reward graph from a χ specification

The complete pipeline of generating discrete-time probabilistic reward graphs from χ specifications is illustrated in Figure 6. Currently, we employ scripts tweaked into the χ environment that insert probabilities and rewards, in order to automatically produce the desired discrete-time probabilistic reward graph from a given χ specification.

Measuring utilization of the data channel K If we assume that the distributions of the channels in the concurrent alternating bit protocol are deterministic, then we can obtain its underlying discrete-time probabilistic reward graph as a performance model, and subsequently calculate its performance measures. First, we give in Figure 7, the long-run utilization of the data channel K. We assume that $t_p =$

$t_r = 1$, $t_s = t_a = 10$, $t_k = 6$, $t_\ell = 2$, that the distribution of the delay of the channel K is deterministic at 6, i.e., $\mathrm{P}(X{=}6) = 1$, and that the distribution of the delay of the channel L is deterministic at 2, i.e., $\mathrm{P}(Y{=}2) = 1$. To obtain the utilization of the data channel, we place reward 1 for every state in the unfolding of the timed delays with duration 6, which is the delay of the data channel K. We note that, although the surface is smooth in the long-run analysis, if we observe the utilization at time step 200, we see that the transient measure is not at all stable as depicted in Figure 8.

Remark 5.1. We can easily compute the utilization in the extremes for the stationary analysis, which further validates the model. If the unreliability of any channel is 1, meaning that no message is actually sent correctly, then every 10 time units the sender re-sends the message via channel K, which lasts 6 time units, resulting in utilization of 0.6. In case both channels are completely reliable, one needs 1 time unit to prepare the message, another 6 time units to send it via channel K, and 2 time units to send the acknowledgement. This amounts to sending a message every 9 time units, i.e., utilization of $\frac{6}{9} \approx 0.67$.

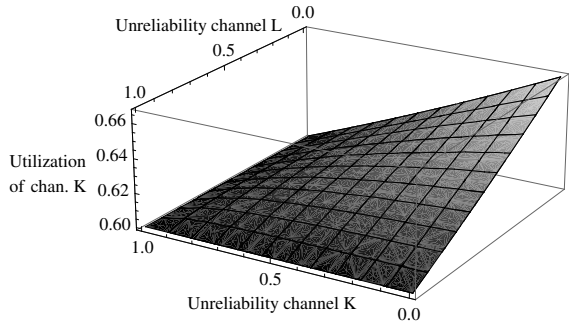

Figure 7. Long-run utilization of the data channel K

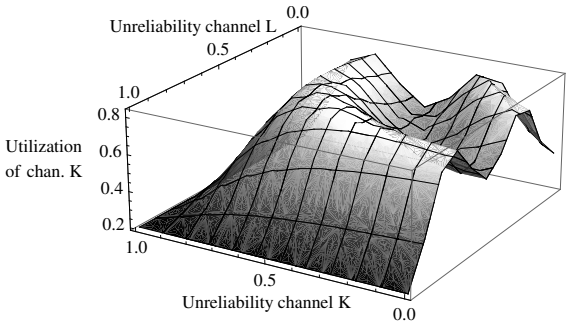

Figure 8. Utilization of the channel K at time 200

When the channels are generally-distributed we resort to discrete-event simulation in χ for performance analysis. Figure 9 gives the utilization of the data channel K, when the distribution of the delay of the data channel is uniform between 2 and 10 and the distribution of the delay of the acknowledgement channel is uniform between 1 and 4. Thus, the uniform distributions of the data and the acknowledgement channels have the mean values of delay 6 and 2, respectively, as in the deterministic case.

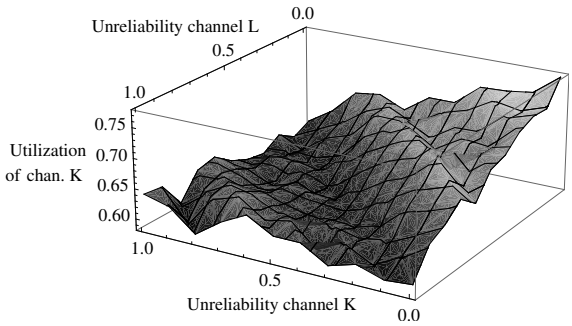

Figure 9. Utilization of the data channel K at time step 200 with uniformly distributed delays

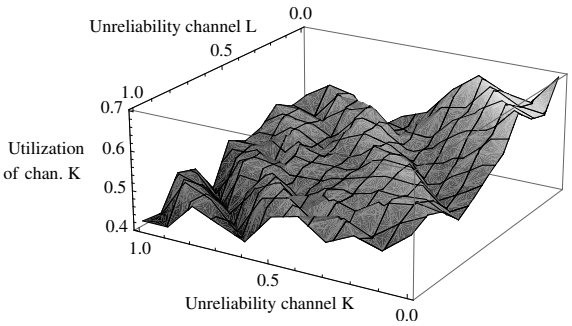

Figure 10. Utilization of the data channel K at time step 200 with exponentially distributed delays

For comparison, we also performed Markovian analysis, again by using discrete event simulation, and the result is depicted in Figure 10. The exponential delays were chosen of the same mean values as the corresponding delays in the deterministic case.

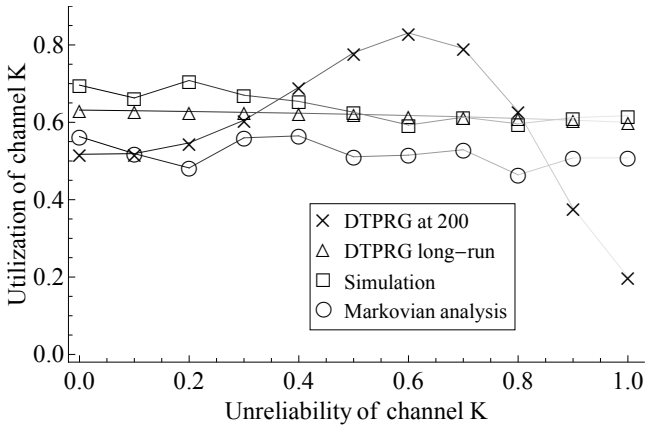

Figure 11. Utilization of the channel K at time 200 for unreliability 0.5 of the channel L

To give a flavor of the results, we discuss the dependence of the utilization of the channel K on the unreliability of the channel K at time step 200 in Figure 11 for each approach. Note, the unreliability of the acknowledgement channel L is fixed to 0.5. One sees that the long-run analysis using discrete-time probabilistic reward graphs is close to the simulation results for the uniformly distributed channels. This is to be expected because they have the same mean value. The Markovian analysis always underestimates the performance because the expected value of the maximum of two exponential delays is greater than maximum of the expected values of both delays. This slightly increases the average cycle length of the system in the following way. When considering the maximum of two deterministic delays, then this is the greater of the two delays. However, when doing the same for exponential distributions, the maximum always overestimates the greater exponential delay. This happens when considering the sender process timeouts, which in effect results in greater timeout in sending the message and, therefore, a lower utilization of the data channel.

6. Conclusion

We proposed a performance evaluation framework that is based on a process theory that enables specification of distributed systems with discrete timed and stochastic delays. The process theory axiomatizes sequential processes comprising termination, immediate actions, and timed delays in a racing context. By construction, the theory conservatively extends standard timed process algebras of [4]. We provided expansion laws for the parallel composition and the maximal progress operator. We derived delayable action and stochastic delay using timed delay prefixes and guarded recursive specifications. Using the formalism, the $G/G/1/\infty$ queue was handled quite conveniently.

For performance evaluation of the process terms we relied on the environment of the language χ, employing discrete-event simulation in the case of generally-distributed delays. We augmented the χ-environment to cater for transient performance analysis of systems exhibiting probabilistic timed behav-

ior, in addition to existing long-run analysis. The extension was supported by a model termed discrete-time probabilistic reward graph, comprising immediate probabilistic choices and deterministic delays. We gave transient analysis of these models by translating them to discrete-time Markov reward chains. We also provided a backward translation, relating the original process to the obtained Markov process, by calculating the transition matrix of the discrete-time probabilistic reward graph.

As a case study, we modeled the a variant of the concurrent alternating bit protocol with generally-distributed unreliable channels both in the process theory as well as in the specification language χ. We analyzed the protocol in the χ toolset by using discrete-event simulation when the channels were generally distributed. By restricting to deterministic delays, we were able to analyze the protocol analytically in the proposed framework of discrete-time probabilistic reward graphs. Finally, we performed Markovian analysis by restricting to exponential delays and we compared the results of the respective analysis.

As future work, we plan to introduce the hiding operator that produces internal transitions and to develop a notion of branching or weak bisimulation in that setting. This should pave the way for bigger case studies on Internet protocol verification and analysis as detailed performance specification becomes viable by using both generally-distributed stochastic delays and standard timeouts. We can also exploit existing real-time specification as the theory is sufficiently flexible to allow extension of real-time with stochastic time while retaining any imposed ordering of the original delays.

Acknowledgments Many thanks to Jos Baeten for fruitful discussions on the topic. We are indebted to the reviewers for their constructive comments and suggestions.

References

[1] Ammar, H., Huang, Y., Liu, R.: Hierarchical Models for Systems Reliability, Maintainability, and Availability, *IEEE Transactions on Circuits and Systems*, **34**(6), 1987, 629–638.

[2] Arends, N.: *A Systems Engineering Specification Formalism*, Ph.D. Thesis, Eindhoven University of Technology, 1996.

[3] Baeten, J., Bergstra, J., Reniers, M.: Discrete Time Process Algebra with Silent Step, in: *Proof, Language, and Interaction: Essays in Honour of Robin Milner*, MIT Press, 2000, 535–569.

[4] Baeten, J., Middelburg, C. A.: *Process Algebra with Timing*, Monographs in Theoretical Computer Science, Springer, 2002.

[5] Baeten, J. C. M., Bergstra, J. A., Klop, J. W.: On the consistency of Koomen's fair abstraction rule, *Theoretical Computer Science*, **51**(1), 1987, 129–176.

[6] Banks, J., Carson II, J., Nelson, B., Nicol, D.: *Discrete-Event System Simulation*, Prentice Hall, 2000.

[7] van Beek, D., van der Ham, A., Rooda, J.: Modelling and Control of Process Industry Batch Production Systems, *15th Triennial World Congress of the International Federation of Automatic Control*, Barcelona, 2002.

[8] van Beek, D., Man, K. L., Reniers, M., Rooda, J., Schiffelers, R. R. H.: Syntax and Consistent Equation Semantics of Hybrid Chi, *Journal of Logic and Algebraic Programming*, **68**, 2006, 129–210.

[9] Bernardo, M., Gorrieri, R.: A tutorial on EMPA: A theory of concurrent processes with nondeterminism, priorities, probabilities and time, *Theoretical Computer Science*, **202**(1–2), 1998, 1–54.

[10] Bohnenkamp, H., D'Argenio, P., Hermanns, H., Katoen, J.-P.: MODEST: A Compositional Modeling Formalism for Hard and Softly Timed Systems, *IEEE Transactions on Software Engineering*, **32**, 2006, 812–830.

[11] Bos, V., Kleijn, J. J. T.: *Formal Specification and Analysis of Industrial Systems*, Ph.D. Thesis, Eindhoven University of Technology, 2002.

[12] Bravetti, M.: *Specification and Analysis of Stochastic Real-time Systems*, Ph.D. Thesis, Università di Bologna, 2002.

[13] Bravetti, M., Bernardo, M., Gorrieri, R.: From EMPA to GSMPA: Allowing for General Distributions, *Proceedings of PAPM'97*, Enschede, 1997.

[14] Bravetti, M., D'Argenio, P.: Tutte le algebre insieme: Concepts, Discussions and Relations of Stochastic Process Algebras with General Distributions, in: *Validation of Stochastic Systems - A Guide to Current Research* (C. Baier, B. Haverkort, H. Hermanns, J.-P. Katoen, M. Siegle, Eds.), vol. 2925 of *Lecture Notes of Computer Science*, Springer, 2004, 44–88.

[15] Bryans, J., Bowman, H., Derrick, J.: Model Checking Stochastic Automata, *ACM Transactions on Computational Logic*, **4**, 2003, 452–492.

[16] van Campen, E.: *Design of a Multi-Process Multi-Product Wafer Fab*, Ph.D. Thesis, Eindhoven University of Technology, 2000.

[17] Chung, K.: *Markov Chains with Stationary Probabilities*, Springer, 1967.

[18] Coderch, M., Willsky, A. S., Sastry, S. S., Castanon, D.: Hierarchical Aggregation of Singularly Perturbed Finite State Markov Processes, *Stochastics*, **8**, 1983, 259–289.

[19] D'Argenio, P.: From Stochastic Automata to Timed Automata: Abstracting probability in a Compositional manner, *Proceedings of WAIT 2003*, Buenos Aires, 2003.

[20] D'Argenio, P., Katoen, J.-P.: A Theory of Stochastic Systems, Part II: Process Algebra, *Information and Computation*, **203**(1), 2005, 39–74.

[21] Fernandez, J., Garavel, H., Kerbrat, A., Mounier, L., Mateescu, R., Sighireanu, M.: CADP - a Protocol Validation and Verification Toolbox, *Proceedings 8th of CAV'96* (R. Alur, T. A. Henzinger, Eds.), 1102, 1996.

[22] Fey, J. J. H.: *Design of a Fruit Juice Blending and Packaging Plant*, Ph.D. Thesis, Eindhoven University of Technology, 2000.

[23] Glynn, P.: A GSMP Formalism for Discrete Event Systems, *Proceedings of the IEEE*, **77**, 1989, 14–23.

[24] Hansson, H.: *Time and Probability in Formal Design of Distributed Systems*, Elsevier, 1994.

[25] Hermanns, H.: *Interactive Markov Chains: The Quest for Quantified Quality*, vol. 2428 of *Lecture Notes in Computer Science*, Springer, 2002.

[26] Hermanns, H., Mertsiotakis, V., Rettelbach, M.: Performance Analysis of Distributed Systems Using TIPP, *Proceedings of UKPEW'94*, University of Edinburgh, 1994.

[27] Hillston, J.: *A Compositional Approach to Performance Modelling*, Cambridge University Press, 1996.

[28] Howard, R.: *Dynamic Probabilistic Systems*, Wiley, 1971.

[29] Katoen, J.-P., D'Argenio, P.: General Distributions In Process Algebra, in: *Lectures on Formal Methods and Performance Analysis* (E. Brinksma, H. Hermanns, J.-P. Katoen, Eds.), vol. 2090 of *Lecture Notes in Computer Science*, 2001, 375–429.

[30] Kemeny, J., Snell, J.: *Finite Markov Chains*, Springer, 1976.

[31] López, N., Núñez, M.: NMSPA: A Non-Markovian Model for Stochastic Processes, *Proceedings of ICDS 2000*, IEEE Computer Society, 2000.

[32] Markovski, J.: *Real and Stochastic Time in Process Algebras for Performance Evaluation*, Ph.D. Thesis, Eindhoven University of Technology, 2008.

[33] Markovski, J., Trčka, N.: Aggregation methods for Markov reward chains with fast and silent transitions, *Proceedings of MMB2008: Measurement, Modeling and Evaluation of Computer and Communication Systems*, VDE Verlag, 2008.

[34] Markovski, J., de Vink, E.: Real-Time Process Algebra with Stochastic Delays, *Proceedings of ACSD 2007*, IEEE, 2007.

[35] Markovski, J., de Vink, E.: Extending Timed Process Algebra with Discrete Stochastic Time, in: *Proceedings of AMAST 2008* (J. Meseguer, G. Rosu, Eds.), vol. 5140 of *Lecture Notes of Computer Science*, 2008, 268–283.

[36] Neuts, M.: *Matrix-Geometric Solutions in Stochastic Models, an Algorithmic Approach*, John Hopkins University Press, 1981.

[37] Nicollin, X., Sifakis, J.: An Overview and Synthesis of Timed Process Algebras, in: *Real-Time: Theory in Practice* (J. W. de Bakker, C. Huizing, W. R. de Roever, G. Rozenberg, Eds.), vol. 600 of *Lecture Notes of Computer Science*, 1992, 526–548.

[38] Schiffelers, R., Man, K.: *Formal Specification and Analysis of Hybrid Systems*, Ph.D. Thesis, Eindhoven University of Technology, 2006.

[39] Sproston, J.: Model Checking for Probabilistic Timed Systems, in: *Validation of Stochastic Systems* (C. Baier, B. Haverkort, H. Hermanns, J.-P. Katoen, M. Siegle, Eds.), vol. 2925 of *Lecture Notes of Computer Science*, 2004, 189–229.

[40] Tai, A., Tso, K., Sanders, W.: A Recurrence-Relation-Based Reward Model for Performability Evaluation of Embedded Systems, *Proceedings of DSN'08*, IEEE Computer Society, 2008.

[41] Trčka, N.: *Silent Steps in Transition Systems and Markov Chains*, Ph.D. Thesis, Eindhoven University of Technology, 2007.

[42] Trčka, N., Georgievska, S., Markovski, J., Andova, S., de Vink, E.: Performance Analysis of Chi models using Discrete Time Probabilistic Reward Graphs, *Proceedings of WODES'08* (B. Lennartson, M. Fabian, K. Åkesson, A. Giua, R. Kumar, Eds.), IEEE Computer Society, 2008.

[43] Trčka, N., Georgievska, S., Markovski, J., Andova, S., de Vink, E.: *Performance Analysis of χ Models using Discrete-Time Probabilistic Reward Graphs*, Technical Report CS 08/02, Eindhoven University of Technology, 2008.