Discrete Real-Time and Stochastic-Time Process Algebra for Performance Analysis of Distributed Systems

J. Markovski* and E.P. de Vink

Formal Methods Group, Technische Universiteit Eindhoven
Den Dolech 2, 5612 AZ Eindhoven, The Netherlands
tel: +31 40 247 5158, fax: +31 40 247 5361
j.markovski@tue.nl, evink@win.tue.nl

Abstract

We present a process algebra with conditionally distributed discrete-time delays and generally-distributed stochastic delays. The treatment allows for expansion laws for the parallel composition and the maximal progress operator. The approach is illustrated by a specification of the concurrent alternating bit protocol with generally-distributed unreliable channels in the language χ. We compare performance analysis using timed probabilistic reward graphs and discrete-event simulation.

1. Introduction

Over the past decade stochastic process algebras emerged as compositional modeling formalisms for systems that do not only require functional verification, but performance analysis as well. Many Markovian processes algebras were developed like EMPA, PEPA, IMC, etc. exploiting the memoryless property of the exponential distribution. Before long, the need for general distributions arose, as exponential delays were not sufficient to model, for example, fixed timeouts of Internet protocols or heavy-tail distributions present in the media streaming services. Prominent stochastic process algebras with general distributions include TIPP, GSMPA, SPADES, IGSMP, NMSPA, and MODEST [13, 7, 11, 6, 17, 4].

Despite the greater expressiveness, compositional modeling with general distributions proved to be challenging, as the memoryless property cannot be relied on [16, 8]. 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 [12]. 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 [13, 11, 17, 4], which enables the action transitions guarded by the clocks that expire first, and (2) pre-selection policy [7, 6], 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 [11, 4], i.e., the time left up to expiration, or (2) by keeping track of the spent lifetimes [13, 7, 6, 17], 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 [16, 8].

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 equivalent 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 [15]. 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 [18]. We refer to the samples as stochastic delays, a naming resembling standard timed delays.

*This research has been funded by the Dutch BSIK/BRICKS project AFM 3.2.

The relation between real-time and stochastic time has been studied in various settings: a structural translation from stochastic automata to timed automata with deadlines is given in [10]. This approach found its way into MODEST [4] as means to introduce real-time and stochastic time as separate constructs in the same formalism. Also, a translation from IGSMP into pure real-time models called interactive timed automata is reported in [6]. Our previous work studied the interplay between standard timed delays and discrete stochastic delays in [19, 18]. 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 [18].

The contribution of the present paper is twofold. Firstly, a sound and ground-complete process theory is provided that accommodates timed delays in a stochastic context, extending the work of [19, 18]. The theory provides an explicit maximal progress operator and an non-trivial expansion law for the parallel composition. Different 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 recover 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 theory 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 another one. This is required when modeling timed systems which correct behavior depends on the ordering of the durations of the timed delays, like for example, 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 is possible to specify which delays are the winners or losers of the imposed race.

We illustrate the theory by revisiting the $G/G/1/\infty$ queue from [19], treating it more elegantly, and by specifying a variant of the Concurrent Alternating Bit Protocol, CABP for short, that has fixed time-outs (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, restricted classes of models of general distributions are analytically solvable. Preliminary research on model checking of stochastic automata is reported in [9] and a proposal for model checking probabilistic timed systems is given in [21]. However, at the moment, the performance analysts turn to simulation when it comes to analyzing models with generally distributed delays. For the purpose of analyzing the specification of the CABP we depend on the toolset of the χ-language [1]. 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 process algebra with data [5]. In addition, in [22] a proposal was given to extend χ with a probabilistic choice to enable long-run performance analysis of probabilistic timed specifications using discrete-time probabilistic reward graphs (DTPRGs for short). We augment the prototype extension of the χ-toolset to cater for transient analysis too. The case study illustrates the new approach with channel distributions that are deterministic.

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 analyzes the CABP protocol and discusses its specification in the process algebra proposed and in the language χ. Section 5 wraps up with concluding remarks. The complete structural operational semantics and formal treatment of the theory can be found in [20].

Acknowledgments Many thanks to Jos Baeten and Nikola Trčka for fruitful discussions on the topic.

2. Timed and Stochastic Delays

In this section we informally introduce a number of notions in process theory that are used below. We refer the interested reader for further technical detail to [20].

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 $2^{\mathcal{V}}$.

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 $[{}^{W}_{L}]$ in case W is the set

of winners and L is the set of losers. We have occasion to write $[W]$ instead of $\left[{}^{W}_{\emptyset}\right]$ 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) $\left[{}^{X}_{Y}\right]$, (2) $\left[{}^{X,Y}_{\emptyset}\right]$, and (3) $\left[{}^{Y}_{X}\right]$. We then say that the race is resolved, as no more outcomes are possible. Thus, we can also write $\left[{}^{X}_{Y}\right] + \left[{}^{X,Y}_{\emptyset}\right] + \left[{}^{Y}_{X}\right]$ 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 $\left[{}^{X}_{Y}\right] + \left[{}^{X,Y}_{\emptyset}\right] + \left[{}^{Y}_{X}\right] + [Z]$ will yield the same behaviour. For example, the outcome of $\left[{}^{X}_{Y}\right] + [Z]$ is either (1) $\left[{}^{Z}_{X,Y}\right]$, (2) $\left[{}^{X,Z}_{Y}\right]$, or (3) $\left[{}^{X}_{Y,Z}\right]$. As outcomes of races may be involved in other races, we generalize the notion of a stochastic delay and refer to an arbitrary outcome $\left[{}^{W}_{L}\right]$ 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 in 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 has already be given in [19, 18] and subsequently extended in [20] to accommodate for explicitly stating the winners and the losers of a race and dynamic renaming of stochastic delays.

The race between two stochastic delays $\left[{}^{W_1}_{L_1}\right]$ and $\left[{}^{W_2}_{L_2}\right]$ is resolved, if the predicate $\mathrm{chk}^{\mathrm{R}}(\left[{}^{W_1}_{L_1}\right], \left[{}^{W_2}_{L_2}\right])$ is satisfied. By definition, $\mathrm{chk}^{\mathrm{R}}(\left[{}^{W_1}_{L_1}\right], \left[{}^{W_2}_{L_2}\right])$ holds iff

$(W_1 \cap W_2 \neq \emptyset$ and $(W_1 \cup W_2) \cap (L_1 \cup L_2) \neq \emptyset)$ or

$(W_1 \cap W_2 = \emptyset, W_1 \cap L_2 \neq \emptyset,$ and $W_2 \cap L_1 \neq \emptyset)$

as this exactly excludes $\left[{}^{W_1}_{L_1 \cup W_2 \cup L_2}\right]$, $\left[{}^{W_1 \cup W_2}_{L_1 \cup L_2}\right]$, and $\left[{}^{W_2}_{W_1 \cup L_1 \cup L_2}\right]$.

Naming of stochastic delays In our previous work [19, 18], we concluded that a process theory should have both dependent and independent stochastic delays to support an expansion law for the parallel composition and for compositional modeling. Dependent stochastic delays always exhibit the same duration in the same race when guided by the same random variable, whereas independent stochastic delays with the same name only have the same distribution, but not necessarily the same duration. To give an example, $\left[{}^{X,Y}_{Z}\right] + \left[{}^{X}_{U}\right]$ is the same race as $\left[{}^{X,Y}_{Z,U}\right]$ if we treat X as a dependent stochastic delay, whereas $\left[{}^{X}_{Z}\right] + [X] = \left[{}^{X}_{Z,Y}\right] + \left[{}^{X,Y}_{Z}\right] + \left[{}^{Y}_{X,Z}\right]$, provided that $\mathrm{F}_X = \mathrm{F}_Y$, when X is treated as an independent stochastic delay.

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, $|\left[{}^{X,Y}_{Z}\right]|_X$ denotes that X is a dependent stochastic delay, but Y and Z are independent. By default, every delay is considered as dependent. Hence, $\left[{}^{W}_{L}\right]$ actually means $|\left[{}^{W}_{L}\right]|_{W \cup L}$. Multiple scope operators intersect and, e.g., $||\left[{}^{X}_{Y}\right]|_X|_Y$ denotes the independent delay $\left[{}^{X}_{Y}\right]$ because $\{X\} \cap \{Y\} = \emptyset$.

Timed delays in a stochastic 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' expires 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. 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 $[X]$ can then be specified as

$$[X] = \sigma^X_\emptyset + \sigma^\emptyset_X.[X]\,,$$

where $\_.\_$ denotes the prefixing operation. Note that the race of $\sigma^X_\emptyset$ and $\sigma^\emptyset_X$ is resolved. In a generalized context, by the same reasoning we specify a stochastic delay $\left[{}^{W}_{L}\right]$ as

$$\left[{}^{W}_{L}\right] = \sigma^W_L + \sigma^\emptyset_{W \cup L}.\left[{}^{W}_{L}\right].$$

We will refer to σ^W_L as a timed delay in a stochastic 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 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.

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 [3]. 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 [3]. To support maximal progress, i.e., 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.

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 [19, 18, 20] for the formal semantics. Here, we give several examples to guide the reader's intuition. To illustrate the theory we 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 that does not do anything 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. The parallel composition is given by $\_ \parallel \_$. It allows passage of time only if both components do so. Finally, we introduce recursive variables as constants $R \in \mathcal{R}$ and we only deal with guarded recursion.

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 $R \in \mathcal{R}$. The set of closed terms is denoted by $\mathcal{C}$.

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 $\mathrm{I}(p)$ 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

$$\mathrm{D}(\epsilon) = \mathrm{D}(\delta) = \mathrm{D}(a.p) = \emptyset,\ \mathrm{D}(|p|_D) = \mathrm{D}(p) \cap D$$
$$\mathrm{D}(\sigma_L^W.p) = W \cup L,\ \mathrm{D}(\partial_H(p)) = \mathrm{D}(\theta_H(p)) = \mathrm{D}(p)$$
$$\mathrm{D}(p_1 + p_2) = \mathrm{D}(p_1 \parallel p_2) = \mathrm{D}(p_1) \cup \mathrm{D}(p_2)$$
$$\mathrm{I}(\epsilon) = \mathrm{I}(\delta) = \mathrm{I}(a.p) = \mathrm{I}(\sigma_L^W.p) = \emptyset$$
$$\mathrm{I}(p_1 + p_2) = \mathrm{I}(p_1 \parallel p_2) = \mathrm{I}(p_1) \cup \mathrm{I}(p_2)$$
$$\mathrm{I}(\partial_H(p)) = \mathrm{I}(\theta_H(p)) = \mathrm{I}(p),\ \mathrm{I}(|p|_D) = \bar{\mathrm{I}}(p, D)$$
$$\bar{\mathrm{I}}(\epsilon, D) = \bar{\mathrm{I}}(\delta, D) = \bar{\mathrm{I}}(a.p, D) = \emptyset$$
$$\bar{\mathrm{I}}(\partial_H(p), D) = \bar{\mathrm{I}}(\theta_H(p), D) = \bar{\mathrm{I}}(p, D)$$
$$\bar{\mathrm{I}}(p_1 + p_2, D) = \bar{\mathrm{I}}(p_1 \parallel p_2, D) = \bar{\mathrm{I}}(p_1, D) \cup \bar{\mathrm{I}}(p_2, D)$$
$$\bar{\mathrm{I}}(|p|_D, D') = \bar{\mathrm{I}}(p, D \cap D'),\ \bar{\mathrm{I}}(\sigma_L^W.p, D) = (W \cup L) \setminus D.$$

The dependent delays are computed as the delays connected by the outermost alternative or parallel composition that is not in scope, i.e. in the scope of the scope operator. The delays that are in the scope must be in the intersection of all dependence binding sets. For the independent delays we need an auxiliary function, as one has to keep track of this intersection [20]. We illustrate the situation by an example. Let $p = |[^X_{Y,Z}].\delta|_{X,Z}|_{X,Y}$. Then $\mathrm{D}(p) = \{X\}$ and $\mathrm{I}(p) = \{Y, Z\}$, since $\{X, Z\} \cap \{X, Y\} = \{X\}$.

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 $|\sigma_Y^X.\sigma^Y.a.\delta|_\emptyset$, we can use it as a building block of the system $|\sigma_Y^X.\sigma^Y.a.\delta|_\emptyset \parallel |\sigma_Y^X.\sigma^Y.a.\delta|_\emptyset$. However, for analysis we revert to the system $|(\sigma_Y^X.\sigma^Y.a.\delta) \parallel (\sigma_V^U.\sigma^V.a.\delta)|_\emptyset$, where $\mathrm{F}_X = \mathrm{F}_U$ and $\mathrm{F}_Y = \mathrm{F}_V$. Note that proper resolution of the race condition requires uniqueness of names of the racing delays (cf. [19, 18]). 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.

$|\delta|_D = \delta$ **A1**, $|\epsilon|_D = \epsilon$ **A2**, $|a.p|_D = a.p$ **A3**

$a.p = a.|p|_\emptyset$ **A4**, $\sigma_L^W.p = |\sigma_L^W.p|_{W \cup L}$ **A5**

$\sigma_L^W.p = \sigma_L^W.|p|_L$ **A6**, $||p|_{D_1}|_{D_2} = |p|_{D_1 \cap D_2}$ **A7**

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

$|\sigma^W_{L \cup \{X\}}.p|_D = |\sigma^W_{L \cup \{Y\}}.p[Y/X]|_D$
if $X, Y \notin L \cup D$ and $\mathrm{F}_X = \mathrm{F}_Y$ **A9**

$|p_1 + p_2|_D = |p_1|_D + |p_2|_D$
if $\mathrm{I}(|p_1|_D) \cap \mathrm{R}(|p_2|_D) = \mathrm{R}(|p_1|_D) \cap \mathrm{I}(|p_2|_D) = \emptyset$ **A10**

$(p + q) + r = p + (q + r)$ **A11**

$p + q = q + p$ **A12**, $p + \delta = p$ **A13**

$\epsilon + \epsilon = \epsilon$ **A14**, $a.p + a.p = a.p$ **A15**

$\sigma^{W_1}_{L_1}.p_1 + \sigma^{W_2}_{L_2}.p_2 = \sigma^{W_1 \cup W_2}_{L_1 \cup L_2}.(|p_1|_{L_1} + |p_2|_{L_2})$
if $W_1 \cap W_2 \neq \emptyset$ and $W_1 \cap L_2 = L_1 \cap W_2 = \emptyset$ **A16**

$\sigma^{W_1}_{L_1}.p_1 + \sigma^{W_2}_{L_2}.|p_2|_{L_2} = \sigma^{W_1}_{L_1 \cup W_2 \cup L_2}.(|p_1|_{L_1} + p_2)$
if $L_1 \cap W_2 \neq \emptyset$ and $W_1 \cap W_2 = W_1 \cap L_2 = \emptyset$ **A17**

$\sigma^{W_1}_{L_1}.p_1 + \sigma^{W_2}_{L_2}.p_2 = \sigma^{W_1}_{W_2 \cup L_2 \cup L_1}.(|p_1|_{L_1} + |p_2|_{L_2}) +$
$\sigma^{W_1 \cup W_2}_{L_1 \cup L_2}.(|p_1|_{L_1} + |p_2|_{L_2}) + \sigma^{W_2}_{L_2 \cup W_1 \cup L_1}.(|p_1|_{L_1} + |p_2|_{L_2})$
if $W_1 \cap W_2 = L_1 \cap W_2 = W_1 \cap L_2 = \emptyset$ **A18**

$\partial_H(\delta) = \delta$ **A19**, $\partial_H(a.p) = \delta$ if $a \in H$ **A20**

$\partial_H(\epsilon) = \epsilon$ **A21**, $\partial_H(a.p) = a.\partial_H(p)$ if $a \notin H$ **A22**

$\partial_H(\sigma_L^W.p) = \sigma_L^W.\partial_H(p)$ **A23**

$\partial_H(p_1 + p_2) = \partial_H(p_1) + \partial_H(p_2)$ **A24**

Table 1. Axioms for the sequential processes

We have

$$
\begin{array}{ll}
(\sigma_L^W.p)[Y/X] = \sigma_L^W.p & \text{if } X \notin W \cup L \\
(\sigma_L^W.p)[Y/X] = \sigma_L^{(W \setminus \{X\}) \cup \{Y\}}.p & \text{if } X \in W \\
(\sigma_L^W.p)[Y/X] = \sigma^W_{(L \setminus \{X\}) \cup \{Y\}}.p[Y/X] & \text{if } X \in L \\
|p|_D[Y/X] = |p[Y/X]|_D & \text{if } X \notin D \\
|p|_D[Y/X] = |p[Y/X]|_{(D \setminus \{X\}) \cup \{Y\}} & \text{if } X \in D,
\end{array}
$$

where the other cases are straightforward.

Axiomatization By now, we have gathered all the prerequisites to present the axioms for the operators, except for $\_ \parallel \_$ and $\theta_H(\_)$. (These operators can be dealt with using the expansion law discussed below for normal forms in which races have been 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 samen distribution. Axiom A9 is similar but now the renamed losing stochastic delay must be consistently renamed in the remainder too. Axiom A10 put 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–A24 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 $\mathrm{chk}^{\mathrm{R}}(W_k, L_k, W_\ell, L_\ell)$ 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 normal form; the summand ϵ is optional and $\sum_{i=1}^m p_i$ is shorthand for $p_1 + \ldots + p_m$ if $m > 0$, or δ otherwise. The availability of 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 [2]. Below, we use it to provide an expansion law for the parallel composition and the maximal progress operator.

Expansion laws The expansion of the parallel composition of $p_1 = |p|_D$ and $p_2 = |p'|_{D'}$, where $D \subseteq \mathrm{D}(p)$, $D' \subseteq \mathrm{D}(p')$, $\mathrm{I}(p_1) \cap \mathrm{R}(p_2) = \mathrm{R}(p_1) \cap \mathrm{I}(p_2) = \emptyset$, and $\mathrm{I}(p_1) \cap (D' \setminus D) = \mathrm{I}(p_2) \cap (D \setminus D') = \emptyset$, and p and p' have the normal forms: $p = \sum_{i=1}^m a_i.p_i + \sum_{j=1}^n \sigma^{W_j}_{L_j}.q_j(+\,\epsilon)$ and $p' = \sum_{k=1}^{m'} a'_k.p'_k + \sum_{\ell=1}^{n'} \sigma^{W'_\ell}_{L'_\ell}.q'_\ell(+\,\epsilon)$, with $p_i = |\bar{p}_i|_\emptyset$, $q_j = |\bar{q}_j|_{D_j}$, $p'_k = |\bar{p}'_k|_\emptyset$, and $q'_\ell = |\bar{q}'_\ell|_{D'_\ell}$ is given by

$p_1 \parallel p_2 = |p \parallel p'|_{D \cup D'}$, where $p \parallel p' =$

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

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

The expansion law of the maximal progress $\theta_I(p)$ [3] is given by $\theta_I(p) = |\theta_I(p')|_D$, where

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

Guarded recursion and delayable actions We introduce recursive specification by means of sets of recursive equations. We only consider guarded recursive specifications, meaning that 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 norm [2, 20].

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

$$[^{W}_{L}](p) = \sigma^{W}_{L}.p + \sigma_{W \cup L}.[^{W}_{L}](p)\,,$$

and define $[^{W}_{L}].p$ 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 $R_2 = [^{X}_{Y}](|p|_\emptyset + [Y](q)) + [X,Y](p+q) + [^{Y}_{X}]([X](p) + |q|_\emptyset)$. The solutions of R_1 and R_2 are

$$\begin{aligned} R_1 &= \sigma^{X}_{Y}.(|p|_\emptyset + [Y](q)) + \sigma^{X,Y}.(p+q) + \\ &\quad \sigma^{Y}_{X}.([X](p) + |q|_\emptyset) + \sigma_{X,Y}.R_1 \\ R_2 &= \sigma^{X}_{Y}.(|p|_\emptyset + [Y](q)) + \sigma^{X,Y}.(p+q) + \\ &\quad \sigma^{Y}_{X}.([X](p) + |q|_\emptyset) + \sigma_{X,Y}.R_2. \end{aligned}$$

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 [19, 18] and ground-completely axiomatized in [20]). For example,

$[^{W_1}_{L_1}](p_1) + [^{W_2}_{L_2}](p_2) = [^{W_1 \cup W_2}_{L_1 \cup L_2}](|p_1|_{L_1} + |p_2|_{L_2})$
if $W_1 \cap W_2 \neq \emptyset$ and $W_1 \cap L_2 = L_1 \cap W_2 = \emptyset$
$[^{W_1}_{L_1}](p_1) + [^{W_2}_{L_2}](p_2) = [_{L_1 \cup W_2 \cup L_2}{}^{W_1}](|p_1|_{L_1} + [^{W_2}_{L_2}](p_2))$
if $L_1 \cap W_2 \neq \emptyset$ and $W_1 \cap W_2 = W_1 \cap L_2 = \emptyset$
$[^{W_1}_{L_1}](p_1) + [^{W_2}_{L_2}](p_2) = [_{W_2 \cup L_2 \cup L_1}{}^{W_1}](|p_1|_{L_1} + [^{W_2}_{L_2}](p_2)) +$
$[^{W_1 \cup W_2}_{L_1 \cup L_2}](|p_1|_{L_1} + |p_2|_{L_2}) + [_{L_2 \cup W_1 \cup L_1}{}^{W_2}]([^{W_1}_{L_1}](p_1) + |p_2|_{L_2})$
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 [19]. The queue is specified as $Q = \theta_I(\partial_H(A \parallel Q_0 \parallel S))$, where

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

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 $C = \theta_I(\partial_H(A \| Q_0))$. We write the recursive specifications for $[X](s_1.A)$ and $\underline{r}_1(Q_1)$ and expand the parallel composition. We have $C = \sigma^{X}.c_1.C + \sigma_{X}.C$, 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

$$\begin{aligned} &Q = S_0 = [X](c_1.c_2.S_1) \\ &S_k = [^{X}_{Y}](c_1.S_{k+1}) + [^{X,Y}_{\emptyset}](c_1.s_3.c_2.S_k + s_3.c_1.c_2.S_k) + \\ &\qquad [^{Y}_{X}](s_3.c_2.S_{k-1}) \quad \text{when } k > 0 \end{aligned}$$

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 stochastic transition system is similar to the one from [19] and retains the same level of complexity.

4. Case-study: CABP

In this section we specify CABP first in the process theory $\mathrm{TCP}^{\mathrm{dst}}$ and then in the specification language χ. By restricting to deterministically timed delays, we show how to analytically obtain transient performance measures out of a χ-specification based on the proposal for long-run analysis in [22]. In the general case, we exploit discrete-event simulation in χ. For comparison, we perform Markovian analysis using an extension of the χ toolset[1] by turning all delays into exponential ones with the corresponding means.

Specification in $\mathrm{TCP}^{\mathrm{dst}}$ The Concurrent Alternating Bit Protocol, or CABP for short, is used for

[1] Provided by Nikola Trčka.

communicating data along an unreliable channel with a guarantee that no information is lost. The protocol relies on retransmission of data. An overview of CABP is depicted in Figure 1.

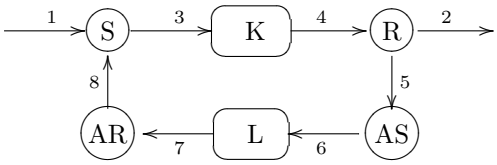

Figure 1. Scheme of the CABP

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 time-out, waiting for the correct 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 time-out. 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.

We can specify the CABP as below for a data set D. We note that the process theory does not contain an explicit probabilistic choice operator. To specify probabilistic behavior of the channel, we introduce time-outs 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 time-out expires with a delay distributed according to the conditional random variables $\langle X \mid X < t_k \rangle$ and $\langle X \mid X < t_k \rangle$, respectively, or they get lost with probability $1 - \mathrm{F}_X(t_k)$, and $1 - \mathrm{F}_Y(t_\ell)$, respectively. The CABP is specified as

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

where the recursive variables are parameterized by $b \in \{0,1\}$ and $d \in D$. 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 time-out of the sender, the time-out of the data channel, the processing time of the receiver, the time-out of the acknowledgement sender, and the time-out of the acknowledgement channel.

Specification in χ We illustrate some features of the language χ by presenting the χ specification of the sender process in Figure 2. For a detailed introduction to χ we refer the reader to [5].

```
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> )
]|
```

Figure 2. The sender process in χ

The process `sender` communicates with the other processes via three channels: `c1,c3,c8`. 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, which it packs in `tp` time units. Afterwards, a frame with the data and the alternating bit is send 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 time-out 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 time-out `ts` $= 10$ time units.

Analyzing timed systems in the long-run A proposal for analysis of timed χ specifications in the long-run was made in [22]. One considers a so-called discrete-time probabilistic reward graph that has deterministic timed transitions and instantaneous probabilistic transitions and only one type of transitions per state allowed. The DTPRG is yielded from the timed branching bisimulation reduced χ-specification and abstracting from internal transitions.

The prototype extension of χ that produces the DTPRGs uses special action transitions and self loops to simulate probabilistic choices and rewards. The pipeline for obtaining the DTPRG starts with the χ specification in which all transitions, except the one that are meant for the probabilistic choices and the rewards, are hidden. Afterwards, rewards are added as self loops during state space generation. The timed transition system that contains internal actions is then reduced using a timed branching bisimulation to a timed system without internal actions. The special actions are then replaced by probabilistic choices to obtain the DTPRG. We note that the method is not generally sound as there may be internal transitions and probabilistic choices originating from the same state in the specification. However, here it serves its purpose in illustrating the approach (cf. [22] for details).

Long-run performance analysis of a DTPRG is performed by translating it to a discrete-time Markov reward chain (DTMRC) with equivalent behavior. The translation from a DTPRG to a DTMRC is done in two steps. First, all timed transitions are unfolded to obtain the 'unfolded' DTMRC depicted in Figure 3b. Afterwards, the (formerly immediate) probabilistic transitions are aggregated, i.e., they are eliminated and their probabilities are propagated to the original timed transitions. We illustrate the translation in Figure 3 by an example from [22].

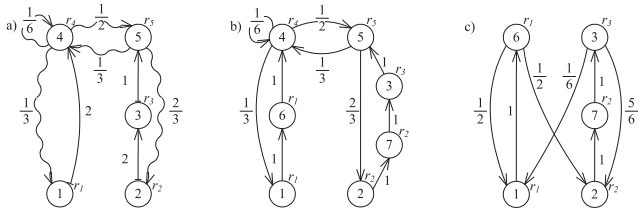

Figure 3. From DTPRG to DTMRC

The DTPRG depicted in Figure 3a has two types of states: timed (1, 2, and 3) and probabilistic (4 and 5). The translation from a DTPRG to a DTMRC is done in two steps. First, all timed transitions are unfolded to obtain the 'unfolded' DTMRC, as depicted in Figure 3b. Afterwards, the (formerly immediate) probabilistic transitions are aggregated, i.e., they are eliminated and their probabilities are propagated to the timed transitions as depicted in Figure 3. The unfolding in Figure 3b is obtained after every timed transition of the DTPRG in Figure 3a is represented as a sequence of probabilistic transitions with probability 1 in the DTMRC while preserving the state rewards. In the next step, an aggregation is done on the originally probabilistic transitions in the DTPRG, since in the DTPRG they do not take time, whereas in the DTMRC each transition takes one time unit. The aggregation basically splits the last transition of the unfolding of every timed transitions that precedes a probabilistic transition according to the accumulative probability of reaching another timed transition as depicted in Figure 3c. For more details and the matrix theoretic considerations underlying the problem the reader may consult [22].

The idea behind the long-run performance analysis of the DTPRG is to compute the long-run expected reward of the unfolded & aggregated DTMRC, and then to 'fold' it using a collector matrix that folds back the unfolded timed transitions. The folding collector matrix sums up the states which were obtained by the unfolding of the timed transitions. For example, the folding collector matrix for the DTMRC in Figure 3 is $V = \begin{pmatrix} 1&0&0\\ 0&1&0\\ 0&0&1\\ 1&0&0\\ 0&1&0 \end{pmatrix}$ (where states 6 and 7 have been renumbered to 4 and 5 in the matrix representation).

Transient analysis of DTPRGs We do transient analysis at some time step n of the DTPRG using the probability transition matrix of the unfolded & aggregated DTMRC at time step n. The idea is to first fold the transition matrix of the DTMRC and afterwards to compute the expected reward rate at time step n. To fold the transition matrix, we need a folding distributor matrix in addition to the folding collector matrix. The folding distributor matrix has 1 for the first states in the unfolding of the timed delays, and 0 otherwise. It supports the intuition that the process remains in a timed state until the timed delay expires by computing the joint transient probability at time step n for the unfolded states of each original timed state. In the example above, the folding distributor matrix is $U = \begin{pmatrix} 1&0&0&0&0\\ 0&1&0&0&0\\ 0&0&1&0&0 \end{pmatrix}$ as the starting states for unfolding of the timed delays are 1, 2, and 3.

To summarize, if the probability transition matrix at time step n of the unfolded & aggregated DTMRC is $P(n)$, then the probability transition matrix of the DTPRG $T(n)$ would be $T(n) = UP(n)V$, where U and V are the folding distributor and collector matrix. We note that the lumping condition does not hold for the folding distributor and collector matrix, i.e. $VUP(n)V \neq P(n)V$. Therefore, the folding must be computed separately for every time step, as we did not see an obvious way of obtaining $T(n+1)$ from $T(n)$.

Our approach can be mimicked, e.g., using the Markovian model checker PRISM [14] and the supporting formalisms. However, the translation to DTMRCs needs manual fine-tuning. In that setting, one model checks the underlying DTMRC, so there is no feedback on the performance measures of the original DTPRG.

Performance in a deterministic setting If we assume that the distributions of the channels are deterministic, then we can obtain its DTPRG representation and subsequently calculate its performance measures. First, we give in Figure 4, on the left, the long-run utilization of the data channel K. We assume that $t_p = t_r = 1$, $t_s = t_a = 10$, $t_k = 6$, $t_\ell = 2$, the distribution of the delay of the channel L is deterministic at 6, i.e. $\mathrm{P}(X{=}6) = 1$, and of the delay of the channel K 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 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 transient measure is not at all stable as depicted in Figure 4, on the right.

Discrete-event simulation When the channels are generally distributed we resort to discrete-event simulation in χ for performance analysis. Figure 5, on the left, 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 channel have the mean values of delay 6 and 2, as in the deterministic case. For comparison, we also performed Markovian analysis, again by using discrete event simulation, and the result is depicted in Figure 5, on the right. The exponential delays were chosen of the same mean values as the corresponding delays in the deterministic case.

Finally, to give a flavour of the results we show the dependence of the utilization of the channel K on the unreliability of the channel K at time step 200, where the unreliability of the acknowledgement channel L is fixed to 0.5. One sees that the long-run analysis using DTPRGs is close to the simulation results for the uniformly distributed channels. This is expected because they have the same mean value. As noted in [22], the Markovian analysis always underestimates the performance because the expected value of the maximum of two exponential delays is greater than the expected values of both delays, which increases the average cycle length of the system.

5. Conclusion

We presented 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 stochastic context. By construction, the theory conservatively extends standard timed process algebras of [3]. 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.

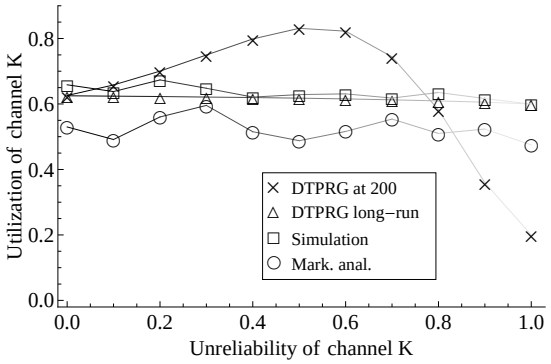

Figure 6. Utilization of the channel K at time 200 for unreliability 0.5 of the channel L

We modeled the concurrent alternating bit protocol in the process theory and, subsequently, in the specification language χ. We analyzed the protocol in the χ toolset by using discrete-event simulation for the generally distributed delays. By restricting to deterministic delays, we were able to analyze the protocol analytically with the discrete-time probabilistic reward graphs of [22]. We extended the χ-environment to also deal with transient performance analysis in addition to the existing methods for long-run analysis. Finally, we performed Markovian analysis by restricting to exponential delay and we compared all analysis results.

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 is 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.

References

[1] N. W. A. Arends. *A systems engineering specification formalism*. PhD thesis, Eindhoven University of Technology, 1996.

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

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

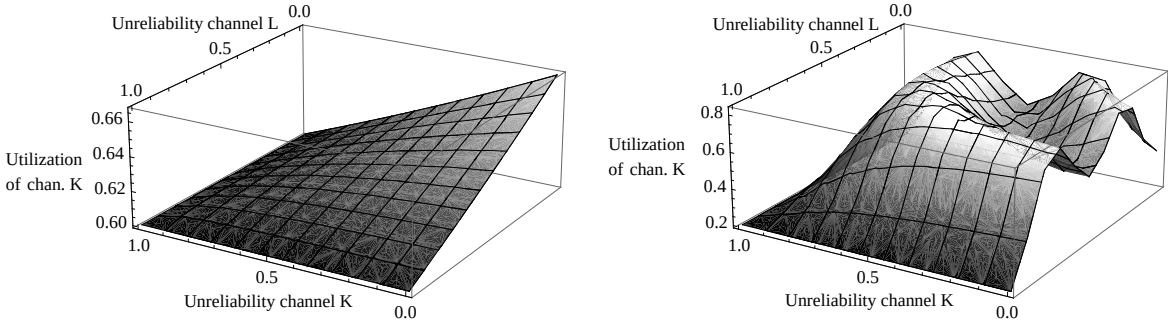

Figure 4. Long-run (left) and utilization at time step 200 (right) of the data channel K

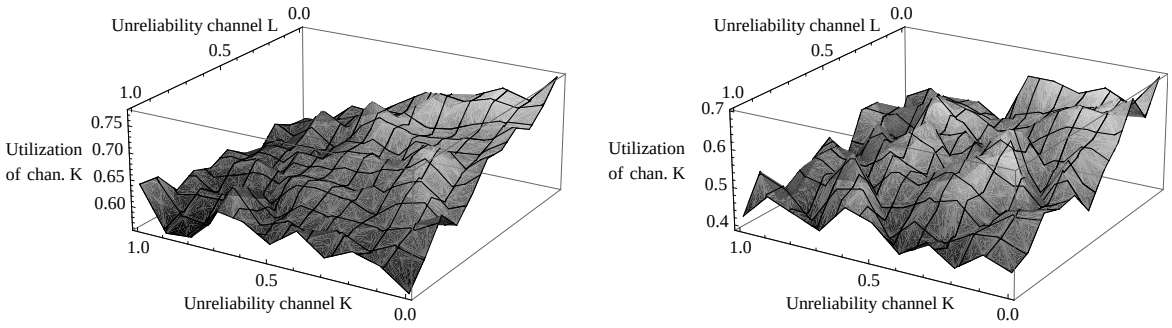

Figure 5. Utilization at 200 of channel K when the delays are uniform (left) and exponential (right)

[4] H. C. Bohnenkamp, P. R. D'Argenio, H. Hermanns, and J.-P. Katoen. MODEST: A compositional modeling formalism for hard and softly timed systems. *IEEE Transactions on Software Engineering*, 32:812–830, 2006.

[5] V. Bos and J. J. T. Kleijn. *Formal Specification and Analysis of Industrial Systems*. PhD thesis, Eindhoven University of Technology, 2002.

[6] M. Bravetti. *Specification and Analysis of Stochastic Real-time Systems*. PhD thesis, Universitá di Bologna, 2002.

[7] M. Bravetti, M. Bernardo, and R. Gorrieri. From EMPA to GSMPA: Allowing for general distributions. In *Proc. PAPM '97*, pages 17–33, Enschede, 1997.

[8] M. Bravetti and P. R. D'Argenio. Tutte le algebre insieme – concepts, discussions and relations of stochastic process algebras with general distributions. In *Validation of Stochastic Systems*, pages 44–88. LNCS 2925, 2004.

[9] J. Bryans, H. Bowman, and J. Derrick. Model checking stochastic automata. *ACM Transactions on Computational Logic*, 4(4):452–492, 2003.

[10] P. R. D'Argenio. From stochastic automata to timed automata: Abstracting probability in a compositional manner. In *Proc. WAIT 2003*, Buenos Aires, 2003.

[11] P. R. D'Argenio and J.-P. Katoen. A theory of stochastic systems, part II: Process algebra. *Information and Computation*, 203(1):39–74, 2005.

[12] P. W. Glynn. A GSMP formalism for discrete event systems. *Proceedings of the IEEE*, 77(1):14–23, 1989.

[13] H. Hermanns, V. Mertsiotakis, and M. Rettelbach. Performance analysis of distributed systems using TIPP. In *Proc. UKPEW'94*, pages 131–144. University of Edinburgh, 1994.

[14] A. Hinton, M. Kwiatkowska, G. Norman, and D. Parker. PRISM: A tool for automatic verification of probabilistic systems. In *Proc. TACAS'06*, pages 441–444. LNCS 3920, 2006.

[15] R. A. Howard. *Dynamic Probabilistic Systems*. Wiley, 1971.

[16] J.-P. Katoen and P. R. D'Argenio. General distributions in process algebra. In *Lectures on Formal Methods and Performance Analysis*, pages 375–429. LNCS 2090, 2001.

[17] N. López and M. Núñez. NMSPA: A non-Markovian model for stochastic processes. In *Proc. ICDS 2000*, pages 33–40. IEEE, 2000.

[18] J. Markovski and E. P. de Vink. Real-time in stochastic process algebra: Keeping track of winners and losers. Technical Report CS 07/13, Eindhoven University of Technology, 2007.

[19] J. Markovski and E. P. de Vink. Real-time process algebra with stochastic delays. In *ACSD 2007*, pages 177–186. IEEE, 2007.

[20] J. Markovski and E. P. de Vink. Discrete real-time and stochastic-time process algebra for performance analysis of distributed systems. Technical Report CS 08/10, Eindhoven University of Technology, 2008.

[21] J. Sproston. *Validation of Stochastic Systems*, chapter Model Checking for Probabilistic Timed Systems, pages 189–229. LNCS 2925, 2004.

[22] N. Trčka, S. Georgievska, J. Markovski, S. Andova, and E. P. de Vink. Performance analysis of Chi models using discrete-time probabilistic reward graphs. In *Proc. WODES'08*. IEEE, 2008.