In **Bulletin of the EATCS** (74), p. 135-154, 2001.

Construction and Verification of Performance and Reliability Models

Holger Hermanns

University of Twente, Faculty of Computer Science, Formal Methods and Tools Group,
P.O. Box 217, 7500 AE Enschede, the Netherlands

hermanns@cs.utwente.nl

Abstract

Over the last two decades formal methods have been extended towards performance and reliability evaluation. This paper tries to provide a rather intuitive explanation of the basic concepts and features in this area. Instead of striving for mathematical rigour, the intention is to give an illustrative introduction to the basics of stochastic models, to stochastic modelling using process algebra, and to model checking as a technique to analyse stochastic models.

1 Introduction

Modern industrial systems, such as communication networks, transport systems, or manufacturing systems, are more and more operating in a stochastic context: communication lines can break, buffers can overflow, a lorry with material for a just-in-time production line might get stuck in a traffic jam. Each of these phenomena is stochastic by nature, its absence or presence can only be predicted up to some probability. Since these stochastic phenomena have impact on the system under consideration, it is nowadays commonly agreed that the systems themselves exhibit stochastic behaviour. As a consequence, performance and reliability studies of industrial systems have to take into account that rigid assessments ("It is impossible that the system fails") only hold under unrealistic assumptions.

The construction and analysis of models suited for performance and reliability studies of real-world phenomena is a difficult task. To a large extent this problem is attacked using human intelligence and experience. Due to increasing size and complexity of systems, this tendency seems even growing: performance as well as reliability modelling becomes a task dedicated to specialists, in particular for systems exhibiting a high degree of irregularity. Traditional performance models such as queueing networks lack hierarchical composition and abstraction means, significantly hampering the modelling of systems that are developed nowadays. Some notable results and concepts have been developed [8, 36, 37, 38], but they remain isolated from the system design cycle, due to the lack of a well-founded theory of hierarchy, composition and abstraction.

On the other hand, for describing the plain functional behaviour of systems various specification formalisms have been developed that are strongly focussed on modelling systems in a compositional, hierarchical manner. A prominent example of such specification formalisms is *process algebra* [19]. Developed on a strong mathematical basis, process algebra has emerged as an important framework to achieve compositionality. Process algebra

provides a formal apparatus for reasoning about structure and behaviour of systems in a compositional way.

During the last decade, *stochastic process algebra* has emerged as a promising way to carry out compositional performance and reliability modelling, mostly on the basis of continuous time Markov chains (CTMCs). Following the same philosophy as ordinary process algebra, the stochastic behaviour of a system is described as the composition of the stochastic behaviours of its components.

To analyse properties of formally specified models *model checking* is a very successful technique to establish the correctness of the model, relative to a given set of temporal logic properties the model is supposed to satisfy [12, 13]. Using efficient encoding techniques, model checking has been applied to industrial size designs involving more than 10^{30} states.

It appears valuable to apply efficient model checking techniques also to performance and reliability properties of industrial systems. Since performance and reliability models are stochastic in nature, the properties of interest are stochastic as well, and have to be described in an appropriate extension of a temporal logic. The model checking algorithm then involves the calculation (or approximation) of probabilities of certain properties to hold.

This paper tries to provide a rather intuitive explanation of the basic concepts and features of stochastic models, of stochastic modelling using process algebra, and of model checking as a technique to analyse stochastic models. For the sake of being illustrative the paper tends to treat various fine points much more simplistic than the advanced reader probably desires.

The paper is organised as follows. Section 2 introduces the basic concepts of stochastic models. Section 3 exemplifies the use of process algebra for modelling stochastic phenomena by means of a real-world example, and Section 4 describes the model checking approach to analyse stochastic models. Section 5 concludes the paper. This paper is a revised version of an invited contribution to the 5th International ERCIM Workshop on Formal Methods for Industrial Critical Systems (FMICS 2000) [25].

2 Stochastic models

A stochastic model is basically a means to describe the evolution of a real-world phenomenon as time[1] passes, with a particular emphasis on phenomena with stochastic timing characteristics. In other words, repeated observations of the same phenomenon can have varying timing characteristics, but their variation exhibits a specific kind of randomness.

As an example, consider a gambler that throws a die every minute. Observing the gambler, one might wish to study a phenomenon, such as the time it takes to throw a ⚅. Starting the observation at some arbitrary minute, one counts the minutes till the die shows a ⚅. Obviously, repeated observations will usually lead to different results, at least if gambling with a fair die. Nevertheless, the variation among these observations exhibits a specific kind of randomness: The time needed to throw a ⚅ is known to follow the so-called geometric probability distribution.

Probability distribution. A probability distribution is a function that assigns a probability (a real value between 0 and 1) to each element of some given set. For instance, the geometric probability distribution P assigns probabilities to natural numbers. For the

[1] It is a bit narrow minded to consider the time domain as the only possible domain of variability. Spatial Markov processes, for instance, are used to describe the evolution of a phenomenon as its position in some appropriate space changes, as opposed to the time.

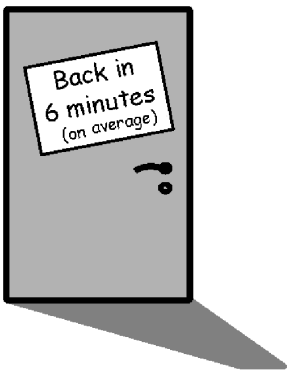

Figure 1: At the door of a gambler

gambler, these numbers enumerate the minutes he is already gambling (remind that he throws the die once per minute). For some t, $P(t)$ is the probability to see the first ⚅ after t minutes, and is given by:

$$P(t) \;=\; \Pr\{\text{see a } ⚅ \text{ within first } t \text{ minutes}\} \;=\; 1 - \left(\frac{5}{6}\right)^t, \qquad t \in \{0, 1, 2, \ldots\},$$

or complementary,

$$1 - P(t) \;=\; \Pr\{\text{still no } ⚅ \text{ after } t \text{ minutes}\} \;=\; \left(\frac{5}{6}\right)^t, \qquad t \in \{0, 1, 2, \ldots\}.$$

For instance, the probability of not having seen a ⚅ after $t = 2$ minutes, i.e., after throwing the die twice, equals 25/36.

To make the example a bit more interesting, assume that the gambler is throwing the die somewhere outside his office. Before leaving his office he has put a note on the door, as depicted in Figure 1. In fact, his intention is to return to his office as soon as the die shows a ⚅. Now let us assume that someone arrives at his door, finding the door closed. How long will he have to wait for the gambler? Probably just a minute, but probably (more likely) more than a minute, probably (unlikely) more than ten minutes. Since this experiment is governed by the above geometric distribution, the probability of having to wait more than a minute is 5/6, the probability of waiting more than ten minutes is $(5/6)^{10}$. Figure 2 depicts these probabilities for the first 15 minutes.

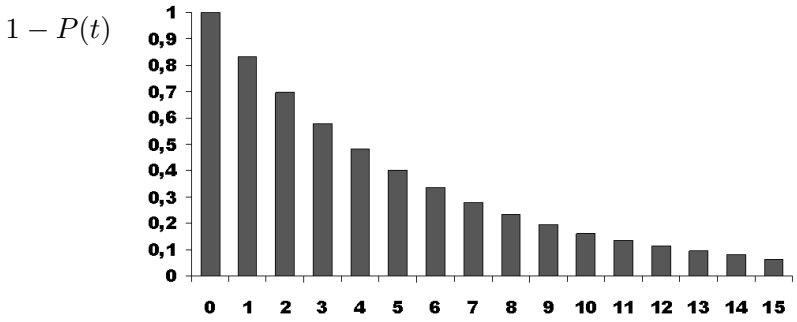

Figure 2: A geometric probability distribution[2]: Will the gambler still be absent at time t?

[2]The reader familiar with the standard pictorial representation might be surprised that the plot in

Markov chain. Having explained the gambler's behaviour, we are now in the position to specify a stochastic model of his behaviour. It is depicted in Figure 3. As many other (formal or semi-formal) models, the model is a graph, consisting of states and transitions. There are two states in this model. One state represents the absence of the gambler, one represents his presence in the office. The model contains three transitions representing possible events that might induce a change of state. One transition indicates that every minute the absent gambler has a 1-out-of-6 chance to return to his office. Another transition indicates that with probability 5/6 the absent gambler will miss the ⚅, and hence has to stay absent for at least another minute. In case he is back in his office, the third transition indicates that he stays there (ad infinitum). The small arrow on top of the left state indicates the initial state. i.e. the state occupied at time zero.

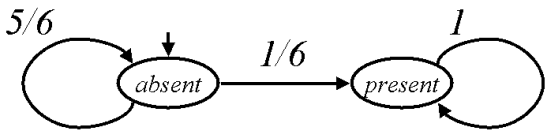

Figure 3: A discrete-time Markov chain describing the gambler's behaviour

The stochastic model of the gambler's behaviour is a very simple one. It is a Markov chain, named after A.A. Markov who studied models of this kind in the beginning of the last century [34]. More specific, it is a discrete-time Markov chain (DTMC), since state changes are only possible at discrete points in time: The gambler can return to his office precisely every minute only. DTMCs restrict the possible time points for state changes to a discrete subset of dense real time. As in our example, these time points are often (but not necessarily) equidistant.

Markov chain analysis. For a given stochastic model, such as a Markov chain, there is usually a variety of interesting properties that one might want to study. Two substantially different classes of properties can be distinguished. *Transient analysis* investigates the evolution of the model up to a given point in time. On the contrary, *steady-state analysis* focusses on the long-run average behaviour. It requires that on the long-run initial start-up effects (the transient phase) do not have a measurable impact.

A trivial steady-state property for the gambler is that with probability zero he will be absent on the long-run. As an example for a transient property, we have already indicated that the probability of still being absent after 10 minutes is $(5/6)^{10}$. A variant of steady-state analysis gives us that on average it takes the gambler six minutes to throw a ⚅. So, the sign on the office door is essentially right, the gambler will be back in six minutes, on average. (This can also be directly derived from the fact that the expected (or average) value of a geometric distribution is $1/p$ where $p = 1/6$ is the chance of a succesful event.)

Analysis techniques. In practise, three fundamentally different techniques are used to analyse stochastic models. They differ with respect to accuracy, applicability and computational requirements. Here, we only give a concise subjective summary on differences and similarities, and refer to Jain's textbook [33] for a more elaborate discussion.

Figure 2 (and Figure 5) tends to zero for $t \to \infty$, instead of tending towards 1. What is depicted is not the probability distribution $P(t)$ (nor its probability density), but its complement $1 - P(t)$. This is done for didactic reasons.

Simulation. The stochastic model is mimicked by a simulator throwing dice and producing statistics of simulated time spent in states. The fraction of simulated time spent in a particular state is used as an estimate for the state probability. This technique is generally applicable, in particular it is suitable also for non-Markov stochastic models. However, it should be noticed that good accuracy tends to require long simulation runs, and hence limits applicability in practise: To increase the accuracy of the simulation by a factor of n, one needs to increase the length of the simulation runs (and hence the run-time of the simulation) by a factor of n^2.

Numerical solution. The transient or steady-state behaviour of a stochastic model is obtained by an exact or approximate numerical algorithm where model parameters are instantiated with numerical values. This approach gives accurate results in general, up to numerical precision. Typically the solution time increases logarithmically with an increase in accuracy. On the other hand, its applicability is restricted to finite Markov chains (with a few exceptions, see e.g. [22, 32]). Furthermore the number of states of the model is a limiting factor, because of computational and especially storage requirements. A very readable textbook on numerical solution methods is [41].

Analytical solution. The transient or steady state property of interest is expressed as a closed formula over the parameters of the model. This is the most simple, accurate and elegant technique. However, analytical solutions are available only for highly restricted classes of stochastic models.

Absence of memory. Markov chains are widely used as stochastic models of real-world phenomena. This is mainly because they possess a distinguishing feature that simplifies both modelling and analysis. They obey the so called *memory-less property*: The future evolution of a Markov chain model is independent of the past, it only depends on the state currently occupied. This property is best explained in terms of the absent gambler. The probability that the gambler returns to his office after one minute from now is 1/6, independent of the fact that someone might be waiting for him in front of his door for ten minutes (or years) already. This is a direct consequence of the fact that a fair die has no memory; the die does not change if it has not shown a ⚅ for ages. This should not be mixed with the fact that the probability of actually having to wait for ten minutes is low, $(5/6)^{10}$. Under the assumption that this unlikely case becomes reality, it still needs another six minutes waiting time on average, as the sign on the door indicates.

Discrete vs. continuous time. Discrete-time Markov chains are convenient to describe the stochastic evolution of sequential systems. In each state, the outgoing transitions define how the probability mass will be spread at the next time instant. Since DTMCs evolve in a discrete time domain, the flow of probability is not continuous, instead it possesses jumps, and remains unchanged in the time interval between two relevant time points, such as between $t = 2$ and $t = 3$. This is relatively convenient for sequential systems. But it is not convenient in a concurrent probabilistic setting, for both theoretical as well as pragmatic reasons.[3]

As an example, imagine that the gambler's office door is checked by some customer. In case he finds the door closed he probabilistically decides to check again after either 24 or 48 seconds. Note that the basic time unit of this DTMC is 24 seconds. For instance, one might want to study the probability that the customer finds an open door after 72 seconds.

[3]DTMCs are appropriate for some types of concurrent systems possessing globally synchronising clocks, such as ATM-switches or certain manufacturing systems.

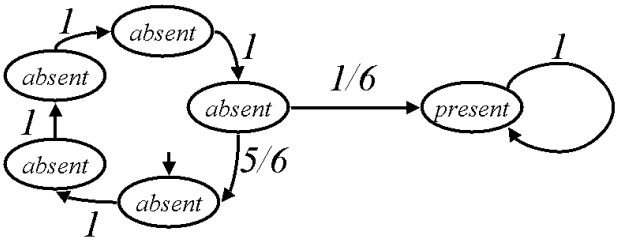

Figure 4: A discrete-time Markov chain describing the gambler's behaviour if observed every 12 seconds

Without specifying the model in all detail, we are already in the position to understand the problem: In order to develop a concurrent probabilistic model of both gambler and customer, we have to relate events that may happen at every 24 seconds to events that happen may every 60 seconds. One solution is to change the basic time unit of both models to 12 seconds, the greatest common divisor of their basic time units. In other words, the gambler's model is blown up to record in 4 additional states that while being absent, four times 12 seconds pass till he may throw the die in the last twelve seconds of the minute (cf. Figure 4).[4] After a similar change in the customer's sub-model, one can combine both models (by essentially taking the cross-product of states and the products of transition probabilities). To determine the concurrent stochastic behaviour at the next point in time (i.e. after 12 seconds) one synchronously updates the respective states in the two sub-models, because state changes now occur exactly at the same time. The probability for such a joined transition is given by the product of the transition probabilities in the sub-models.

This strategy has two practical limitations, at least. First, it tends to induce a tremendous blow-up of the size of the model, caused by the number of auxiliary states needed in general. Second, it fails if there is no greatest common divisor, for instance if the customer shows up every π seconds, or if time points are not equidistant. As a consequence, virtually all stochastic models of concurrent systems are developed in a continuous time domain, including models of modern computer systems (even though each component of such a system can be considered as working in discrete time, changing state according to fixed frequency clock ticks).

Continuous-time Markov chains. Continuous-time Markov chains (CTMCs) are Markov chains interpreted over continuous time, in contrast to DTMCs. They are widely used to model the stochastic behaviour of concurrent real-world phenomena, due to their mathematical simplicity, paired with modelling convenience.

How does the continuous-time variant of the gambler look like? In a continuous time setting, the absent gambler is able to return to his office at arbitrary time points. Still we may assume that he has a 1-out-of-6 chance to return within the first minute, and so on. Under these assumptions, we get the following probability distribution:

$$\Pr\{\text{still no } \boxdot \text{ after } t \text{ minutes}\} \;=\; (5/6)^t, \qquad t \in \mathbb{R}^+.$$

What is this? It perfectly resembles the geometric distribution appearing in the discrete time case, but it is different. The difference is that the domain of this function is the

[4]Note that this change encodes some kind of memory in an otherwise memory-less model: A sequence of states is used to keep track of the time already spent in the original state.

non-negative real line, instead of the natural numbers. In other words, the above function assigns a probability to all time points one may think of, instead of only to each minute. Hence, there is now a non-zero probability of returning within the first second already, namely $1 - (5/6)^{1/60}$. Instead of being a geometric distribution, this function belongs to the class of so-called (negative) exponential probability distributions, because $(5/6)^t$ can be rewritten to $e^{-\lambda t}$, with $\lambda = \ln 6 - \ln 5 \approx 0.18232$. The value λ is a parameter of the distribution, usually called 'rate'. For $t < 15$, the probabilities determined by this exponential probability distribution are depicted (by the dark plot) in Figure 5. The expected value of an exponential distribution (i.e. the average duration) is $1/\lambda$, the reciprocal value of the rate. So, the (continuously gambling) gambler returns after 5.48 minutes on average, not after six minutes.[5]

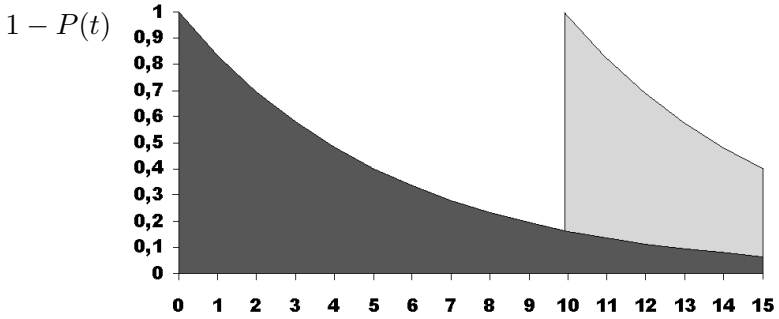

Figure 5: A negative exponential probability distribution with $\lambda = \ln 6 - \ln 5$: Will the gambler still be absent at time t?

A continuous-time Markov chain model of this absent gambler is depicted in Figure 6. It consists of two states, and one transition. The transition represents that the gambler can return to his office with rate λ. The gambler stays absent as long as needed to throw a ⚅. According to the value of λ the probability mass flows from state to state as time passes, that is, a fraction of $1 - e^{\lambda} = 1/6$ of the probability mass flows from the left state to the right state per minute.[6]

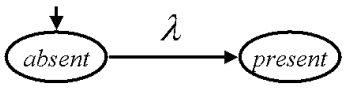

Figure 6: A continuous-time Markov chain describing the gambler's behaviour

Though the above example shows one of the simplest CTMCs one can think of, it exhibits all relevant ingredients: states and transitions, the latter labelled with rates of exponential distributions. It is worth to note that – in correspondence to geometric

[5]Remark that since the probability mass is flowing continuously, a sixth of the mass leaks prior to the first minute tick. Hence, to some extent the probability mass flows earlier than in the discrete-time case, where a sixth of the probability mass jumps a bit later, at each minute tick. As a consequence, the average time needed for the continuously gambling gambler is slightly smaller than 6 minutes. To obtain an average duration of 6 minutes, one has to adjust λ to 1/6.

[6]Since the gambler continuously tries to return to his office, there is no need to record by an explicit (looping) transition that he might fail for some (continuous) time. For CTMCs, this fact is implicit, while in the DTMC scenario it is not.

distributions – exponential distributions are memory-less: The future evolution of a CTMC model is independent of the past, it only depends on the state currently occupied. In terms of the gambler, the probability that the absent gambler returns to his office within the next minute is 1/6, independent of the fact he might have been absent for ages already.

Figure 5 allows us to illustrate the memory-less property in a pictorial way [1]. Consider the case that the gambler is still gambling after minute 10. We obtain the probability that he will still be gambling at time $10 + t$ by stretching the tail of the distribution (from time 10 to ∞) upwards in such a way that it reaches probability 1 for minute 10, i.e. $t = 0$. As a matter of fact, this stretching returns precisely the original distribution, as indicated by the light-grey plot in Figure 5, except that it is shifted by 10 minutes. The same graphical illustration holds for the geometric distribution, but for no other discrete or continuous distribution, because the exponential (resp. geometric) distribution is the only memory-less continuous (resp. discrete) distribution.

From a pragmatic point of view, the memory-less property is rather convenient. It simplifies analysis, but it also simplifies modelling. In particular, it fits well to concurrent stochastic phenomena: If two sub-models, both described in terms of CTMCs, are to be considered concurrently, one can simply interleave their evolution: If one sub-model changes from one state to another, the other sub-model is not affected. The fact that the latter has been staying in some state for some time (the time it took the former sub-model to change state) does not need to be recorded somehow, because it does not alter the future behaviour of the latter sub-model, due to the memory-less property.

Anyway, it should be clearly stated that absence of memory is an assumption that is by far not always justified when modelling real-world phenomena.[7] On the other hand, exponential distributions are the most appropriate guesses if only expected values are known. This is because among all distributions with a given expected value, the exponential distribution maximises entropy [23, 39]. Actually, in performance engineering practise one hardly has measurement data at hand that allows one to determine more than an expected value.

3 Formal specification of continuous-time Markov chains

In this section we illustrate the use of formal methods to model a specific aspect of a real-world example as a CTMC. Several formal notations exist that map on CTMCs, among them stochastic Petri nets and stochastic process algebra. Here we restrict ourselves to illustrate the use of process algebra; an introduction to the Petri net based approach can be found for instance in [1]. As opposed to standard Petri nets, process algebra allows one to compose models out of smaller sub-models, by means of general composition operators such as parallel composition and choice [19], and also more specific constructs, such as exception handling [21]. We will make use of these operators to model a simplified view on the performance and reliability of the Hubble space telescope.

The Hubble Space Telescope. The Hubble space telescope (HST) [40] is an orbiting astronomical observatory operating from the near-infrared into the ultraviolet (cf. Figure 7). Launched in 1990 and scheduled to operate through 2010, the HST carries a variety of instruments producing imaging, spectrographic, astrometric, and photometric data.

[7] It is possible to incorporate a notion of memory into the model, similar to what we have used to realise synchronisation of DTMCs (cf. footnote 4). In this way, general non-exponential probability distributions (so-called phase-type distributions) can be represented. The price to pay for this is usually a blow up of the model.

Figure 7: The Hubble space telescope [40].

The HST was first conceived in the 1940's. It was designed and built in the 1970's and 1980's, aiming at a life span of 15 years with on-orbit servicing taking place on 3 year intervals. The HST programme is a cooperation of the National Aeronautics and Space Administration(NASA) and the European Space Agency (ESA).[8]

Since the telescope has been launched in April 1990, three servicing missions were carried out: in December 1993, in February 1997, and in December 1999. During the last mission the stabilising unit of the HST was repaired. This was necessary, since severe problems with the reliability of the gyroscopes contained therein had forced the HST to turn into a sleep mode.

The gyroscopes are part of HST pointing system. They provide a frame of reference to determine where it is pointing and how that pointing changes as the telescope moves across the sky. They report any small movements of the spacecraft to the HST pointing and control system. The computers then command the spinning reaction wheels to keep the spacecraft stable or moving at the desired rate in order to avoid that the telescope pointing device staggers. This is of particular importance to avoid that pictures taken by the telescope are blurred. The gyroscopes work by comparing the HST motion relative to the axes of the spinning masses inside the gyroscopes.

The HST has a total of six gyroscopes, grouped into three fine guidance sensors. They are arranged in such a way that any three gyroscopes can keep the HST operating with full accuracy. Two fine guidance sensors had been replaced already during the first servicing mission in 1993. Till the end of the second servicing mission in 1997, all six gyroscopes were working normally, but then one after the other failed. Starting from January 1999 the HST had been operating with only 3 functional gyroscopes. As a consequence of a fourth gyroscope failure on November 13, 1999, HST turned itself into a sleep mode and the science program was suspended. Without operational gyroscope the telescope would have run the risk to crash. In December 1999, a space shuttle mission was sent to the HST to replace (among others) the complete stabilising unit. This mission was successful.

[8]Originally, the HST was designed to be returned to earth via the space shuttle every 5 years with on-orbit servicing every 2.5 years as well. This concept was later dropped as it was felt there was a too great risk of contamination and structural load to make the concept sound.

In order to judge whether the problems of the HST could have been expected beforehand, one might want to study the reliability of the stabilising unit by means of an abstract stochastic model. Here we construct a simple Markov chain model of the gyroscopes, and of their controller. The model is a toy example, developed to give a flavour of Markov chain modelling with process algebra. The model is developed in the algebra of interactive Markov chains (IMC) [24, 27], an orthogonal extension of both CTMCs and basic process algebra.

Basic processes. Each gyroscope might FAIL after an exponentially distributed amount of time (it is known that exponential distributions fit relatively well to failures of technical equipment). The failure rate λ is the same for all gyroscopes. A GYRO specification is as follows:

$$\text{GYRO} = (\lambda).\ \text{FAIL}.\ \text{STOP}$$

This specification corresponds to a graphical representation depicted in Figure 8. Apart from a transition labelled λ representing the delay prior to failure, there is a second kind of transition, indicated by a dotted arrow labelled FAIL. In abstract terms, this transition represents the potential of interaction, i.e. of synchronising with a partner transition (labelled with the same name) in a different sub-model. The potential of interaction between sub-models is one of the well-known features offered by a process algebraic approach [7, 35].

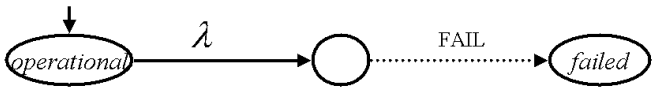

Figure 8: A simple interactive Markov chain describing the gyroscope's behaviour

Parallel composition. Six of these gyroscopes coexist independently in the stabilising unit, together with a controller that keeps track of the status of each gyroscope, by means of synchronisation on FAIL. This is realised using the operator |[FAIL]| for synchronisation, and ||| to denote independent parallelism (among the gyroscopes):

$$\begin{aligned}\text{STABILISER} \;=\; & \text{CONTROLLER} \\ & |[\text{FAIL}]| \\ & (\text{GYRO}|||\text{GYRO}|||\text{GYRO}|||\text{GYRO}|||\text{GYRO}|||\text{GYRO})\end{aligned}$$

The controller counts the number of failures, and mechanically turns the telescope into sleep mode in case four gyroscopes have failed. To turn into sleep mode requires some time. For the moment we just assume an exponential distribution with rate μ. We will explain shortly how to deal with other distributions. After turning on the sleep mode, the controller notifies the base station by means of a SLEEP signal. In the meantime, further gyroscope failures might occur. If the last gyroscope fails, a CRASH is assumed to be inevitable. The graphical representation of the controller is depicted in Figure 9.

$$\begin{aligned}\text{CONTROLLER} \;=\; & \text{FAIL}.\ \text{FAIL}.\ \text{FAIL}.\ \text{FAIL}. \\ & ((\mu).\ \text{SLEEP}.\ \text{STOP} \;|||\; \text{FAIL}.\ \text{FAIL}.\ \text{CRASH}.\ \text{STOP}))\end{aligned}$$

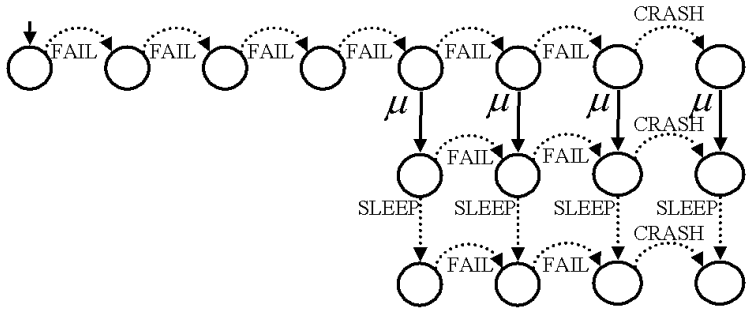

Figure 9: An interactive Markov chain describing the controller

To complete the picture, we consider the stabilising unit of the HST in the context of the base station. The base station listens to the SLEEP notification and reacts accordingly: launch a space shuttle mission to repair – and then restart – the telescope.

$$\text{BASE} \;=\; \text{SLEEP. PREPARE. LAUNCH. REPAIR. RESTART. BASE}$$

Exception handling. The complete specification consists of the STABILISER and the BASE station synchronising on SLEEP. Two events may alter the functioning of the system. If a CRASH occurs, the whole system is extinguished, but if the shuttle mission manages to repair the stabilising unit in time, the whole system will be restarted anew.[9]

$$\begin{array}{lll} \text{HST} = & \mathit{try} & \{\ \text{STABILISER} \;|[\text{SLEEP}]|\; \text{BASE}\ \} \\ & \mathit{catch}\ \ \text{RESTART} & \{\ \text{HST}\ \} \\ & \mathit{catch}\ \ \text{CRASH} & \{\ \text{STOP}\ \} \end{array}$$

Time constraints. Of course, preparing the shuttle mission takes time, and one might wish to incorporate the expected (random) delay in the model. To do so, we can use a constraint-oriented style, as advocated in [42]. This style allows one to add constraints on the timing of certain sequences of interactions, such as between PREPARE and LAUNCH by means of a dedicated operator, defined in [27]. For instance,

$$\begin{array}{ll} \mathit{on}\ \ \text{PREPARE}\ \ \mathit{delay}\ \ \text{LAUNCH}\ \ \mathit{by}\ \ (\nu).\ \text{STOP} \\ \mathit{in}\ \ \text{HST} \end{array}$$

adds an exponentially distributed delay with rate ν between PREPARE and LAUNCH. Semantically speaking, this will have the same effect as specifying BASE = SLEEP. PREPARE. (ν). LAUNCH. REPAIR. RESTART. BASE, but it is much more modular and flexible, in particular because it can be used to impose very general time constraints, instead of only exponentially distributed ones, see [27]. In short, one can insert an arbitrary (phase-type distributed) delay between PREPARE and LAUNCH, by replacing (ν). STOP in the above expression by some appropriate term (in fact, an encoding of the distribution as a CTMC).

For the sake of the presentation we do not add further time constraints, even though a realistic model would at least impose some nontrivial delay between LAUNCH and REPAIR, (as well as a non-exponential delay to set up the SLEEP mode.)

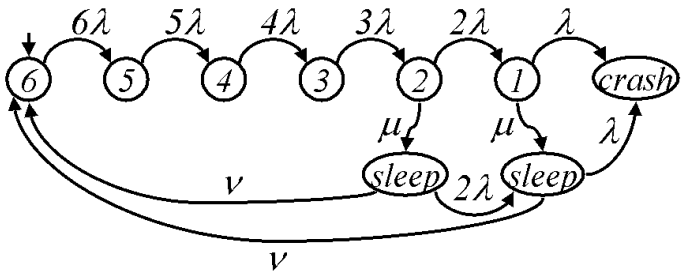

Figure 10: A continuous-time Markov chain corresponding to the stochastic behaviour of the telescope

Extracting the Markov chain. The complete HST specification gives rise to a stochastic model, a CTMC depicted in Figure 10. It is obtained from the specification by applying the formal semantics of the process algebra, and compressing the model by means of an appropriate weak bisimulation afterwards.[10] The states are labelled from left to right with the number of gyroscopes that are currently operational, except if the system is *sleep*ing, or *crash*ed.

Remark that in this CTMC the failure rate λ appears weighted with different multiplying factors. The intuitive reason is that if six gyroscopes are operational, the time to the first failure is six times smaller than if only one gyroscope is left. This increased failure rate for multiple identical components is correctly derived by the formal approach outlined above. The necessary compression algorithm is implemented in [9].

4 Performance and reliability via model checking

In this section we illustrate the use of model checking to analyse performance and reliability properties of CTMC models. We discuss the main ingredients of this approach, and apply model checking to the simple Hubble space telescope example of Section 3.

Temporal logic. The model checking approach relies on the use of temporal logic for specifying properties one is interested in. For this purpose temporal logic provides means to specify undesired (or – dually – desired) evolutions. Typical specifications of properties are `'something undesired never happens'` or `'eventually a desired state is reached'`. A temporal logic specification is usually considered in the context of a given model (provided by some process algebraic specification, for instance). The mechanic verification whether a model satisfies a temporal logic specification is called *model checking*. It is worth to mention that basic temporal logic does not allow one to reason about delays and time points (although the name might suggest the converse). It is 'temporal' in the sense that it allows one to refer to the ordering of events as the model evolves in time.

Temporal logics for Markov chains. In the context of Markov chain models, the temporal logic approach turns into a probabilistic temporal one. It is not sufficient to decide whether `'eventually a desired state is reached'`. Instead the probability of

[9] The semantics of this exception handling is defined in [17].

[10] As explained in [27], constructing the Markov chain requires to internalise all possible interactions beforehand. This is necessary but not always sufficient to extract a CTMC, since interactive Markov chains are strictly more expressive than CTMCs (because of the absence on non-determinism in CTMCs).

eventually reaching a desired state is much more interesting. For the gambler example in Figure 3 the standard interpretation of `'eventually the gambler will be present'` would return false, because it is in principle possible to stay absent ad infinitum. However, this evolution is extremely unlikely, it has probability zero. So, a quantitative interpretation of temporal logic is needed, quantifying the likelihood of satisfying a given property. This allows one to specify properties such as `'a desired state is eventually reached with at least probability 0.95'`.

Moreover, since the evolution of a Markov chain model in time is measurable (in the true sense of the word), it is possible to reason about time instances within the temporal logic. Timed properties such as `'with at most probability 0.2 the gambler will still be absent after 10 minutes'` are possible.

Continuous stochastic logic. The continuous stochastic logic (CSL), first proposed in [2] and further refined in [3, 5] provides means to reason about continuous-time Markov chain models. It is a branching time logic based on CTL [11] with dedicated means to specify time intervals, and to quantify probability. As explained in Section 2, there are two substantially different classes of properties of a CTMC: transient and steady-state properties. Therefore, CSL provides two complementary means to quantify the probability mass: a steady-state operator $\mathcal{S}$, to quantify the long-run likelihood, and a transient probability operator $\mathcal{P}$.

For instance, a steady-state property $\mathcal{S}_{\leq p}(\Phi)$ is true if the long-run likelihood of property Φ is at most p.[11] Φ can be a basic property (usually called atomic proposition) valid (or invalid) in some state. It can also be an arbitrary nested property of the logic. The transient probability operator is used to quantify the likelihood of evolving in a specified way, from a given state and a given time point on. For example $\mathcal{P}_{\leq p}(X\,\Phi)$ is true in a state if the probability of moving (in one step) to a state where Φ holds is at most p. Apart from $X\,\Phi$, there can be various other arguments for the operator $\mathcal{P}$, such as

- $\Diamond\,\Phi$ quantifies the probability for evolving in such a way that eventually a Φ-state is reached.
- $\Diamond^{[0,t]}\,\Phi$ characterizes the amount of probability for reaching a Φ-state within t time units.
- $\Phi_1\,\mathcal{U}\,\Phi_2$ characterizes the amount of probability for evolving only along Φ_1-states until a Φ_2-state is reached.
- $\Phi_1\,\mathcal{U}^{[t_1,t_2]}\,\Phi_2$ quantifies the probability for evolving only along Φ_1-states until a Φ_2-state is reached, under the additional constraint that Φ_1 holds at least up to time t_1, and Φ_2 holds at time t_2 the latest.

Model checking CSL. Model checking a CTMC with respect to a given CSL property involves various algorithms. Since the details are not of vital importance for a proper understanding of the approach – at least relative to the logical means to specify properties – we only give a concise overview of the ingredients.

As in other model checking strategies, a couple of graph algorithms are used. In addition, algorithms to quantify the probability mass of satisfying the above criteria are needed. In principle, these probabilities could be derived using simulation, numerical solution, or sometimes via analytical solutions. Since numerical solution of CTMCs is well-studied and generally applicable, it seems wise to use numerical solution methods

[11] Instead of '$\leq$' one may use arbitrary comparison operators, or specify intervals of probabilities instead.

to model check CSL properties [5]. In this way, model checking involves matrix-vector multiplications (for X), solutions of linear systems of equations (for $\Diamond$, $\mathcal{U}$ and for $\mathcal{S}$), and solutions of systems of Volterra integral equations (for $\mathcal{U}^{[\ldots]}$). Linear systems of equations can be solved iteratively by standard numerical methods [41]. Systems of integral equations can be solved either by piecewise integration after discretisation, or they can be reduced to standard transient analysis [3]. A prototypical model checker for CSL, $\mathsf{E} \vdash \mathsf{MC}^2$, is available [28]. We shall make use of $\mathsf{E} \vdash \mathsf{MC}^2$ to investigate properties of the Hubble space telescope.

Properties of the telescope model. CSL provides a rich framework to study performance and reliability properties of the HST. Here we consider a few illustrative cases. In order to allow the calculation of numerical values, we first need to fix the model parameters λ, μ, and ν of the CTMC in Figure 10. Assuming a basic time unit of one year, we set $\lambda = 0.1$, i.e., we assume that each gyroscope has an average lifetime of 10 years. (Remind that $1/\lambda$ gives the average duration of an exponential distribution with rate λ.) To turn on the sleep mode may require a hundredth of a year (a bit more than three days and a half) on average, hence we set $\mu = 100$. Further assuming that preparing the repair mission will take about two months, we set $\nu = 6$. Unless otherwise stated we consider the validity of CSL properties in the initial state, i.e. the state labelled *6* in Figure 10. The state labels appearing in this figure serve as atomic state propositions for the logic.

First, let us look into long-run averages. An interesting property, often called *availability*, is the probability that the system will be available – i.e. neither *crash*ed nor *sleep*ing – on the long-run average. In CSL we assure an availability higher than p by specifying

$$\mathcal{S}_{>p}(\neg\ (\ sleep\ \vee\ crash)\ ).$$

None of the states of the HST satisfies this property (whatever the value of p may be). This should not be surprising, because the telescope is not constructed for the long run. In fact, the availability of the telescope is zero, because on the long run, the modelled telescope will crash, all the probability mass will eventually be accumulated in the *crash*-state (cf. Figure 10).[12]

While checking standard availability does not make much sense for the HST, the *instantaneous availability* is of interest. Instantaneous availability is a typical transient property, it is the probability that the system is operational at a given time point t. This time point could for instance be given by the need to observe a rare astronomic event. Assuming that an interesting comet passes the telescope in five years, we specify

$$\mathcal{P}_{\geq 0.95}(\Diamond^{[5,5]}\neg\ (sleep\ \vee\ crash))$$

in order to assure that with at least probability 0.95 the telescope is neither *sleep*ing nor *crash*ed then. (Note that the time interval $[t, t]$ denotes just a single time point.) We compute a probability of more than 0.98 of satisfying this property, hence it is satisfied.

In the same vain, we may wonder about the probability to obtain blurred data at that time from the telescope, because less than three gyroscopes are operational, but sleep mode is not yet turned on. This is a very unlikely situation, and one might accept at most a probability of 10^{-6}. One way of characterising the relevant states is to isolate those

[12] Generally speaking, steady-state properties provide very useful insight in the model, in particular for the widespread class of models where the probability mass can flow forever without gradually leaking into some sink (so to speak), or where more than one sink exists. Each of these sinks may in general consist of a set of mutually reachable states.

(non-*sleep*) state that (with positive probability) can turn on the *sleep* mode in the next step. This gives us

$$\mathcal{P}_{\leq 10^{-6}}(\Diamond^{[5,5]}\ (\neg\ sleep\ \wedge\ \mathcal{P}_{>0}(X\ sleep)\ ),$$

a property that is not satisfied, because the probability of being in the specified states after 5 years is about 10^{-5}.

Another quantity of interest is the *time until first sleep*, i.e. the time span before the (fully operational) telescope has to be put into *sleep* mode for the first time. In reality, this happened within 2.7 years: All gyroscopes were operational at the end of the second servicing mission in early 1997, and the *sleep* mode was turned on in November 1999. We require a less than 10 % chance of such a first sleep within 2.7 years by specifying the property

$$\mathcal{P}_{<0.1}(\ \neg sleep\ \mathcal{U}^{[0,2.7]}\ sleep).$$

It turns out that this property is valid; $\mathsf{E} \vdash \mathsf{MC}^2$ computes that the probability of a first sleep within 2.7 years amounts to about 0.03. A related question is whether it was likely not to witness any gyroscope failure within the four years between the first (1993) and the second servicing mission (1997). We answer this question by checking whether the probability to leave state *6* within 4 years is between, say, 0.3 and 0.7. (Notice that leaving state *6* corresponds to a gyroscope failure).

$$\mathcal{P}_{[0.3,0,7]}(\Diamond^{[0,4]}\neg\ 6)$$

In fact, this property is invalid, because the probability of a gyroscope failure within 4 years is approximately 0.9, thus exceeding the upper bound 0.7.

As a last example property, be reminded that the HST is planned to stay on orbit through 2010. Hence, it seems worth to study whether a *crash* before reaching the year 2010 can hardly be expected. To do so, we check a property saying that there is at most a 1% chance that the system will *crash* within the next 10 years (given that the system was reset to state *6* in late 1999):

$$\mathcal{P}_{<0.01}(\Diamond^{[0,10]}\ crash).$$

This property is satisfied, the probability of crashing within 10 years is calculated by $\mathsf{E} \vdash \mathsf{MC}^2$ to be 0.00036. Remind that the model is a toy example, and that its timing parameters are not claimed to reflect reality.

5 Concluding remarks

In this paper, we have tried to give an illustrative introduction to the basics of stochastic models, to stochastic modelling using process algebra, and to model checking as a technique to analyse stochastic models.

A few questions have not been addressed to a satisfactory extent. In particular we have skipped the discussion how to label states of a CTMC generated from a process algebra in such a way that these labels can be used in temporal logic property specifications. One solution to this problem is to move from a state-based logic towards a transition-based formalism [29]. As a further direction of research, we have extended CSL to so-called Markov reward models, where states are parametrized with costs. This extension allows one to specify a broad set of performability properties [4].

Another important issue for industrial strength formal analysis is the availability of tool support. Currently, prototypical tool support is available for both the stochastic

modelling and the analysis phase: A couple of prototypes exist that allow a process algebraic modelling of CTMCs [6, 10, 26]. So far, performance models with up to 10^7 states have been modelled and analysed compositionally [27]. A prototypical model checker for Markov chains, $\mathsf{E} \vdash \mathsf{MC}^2$, is available [28], it has been used to check the above CSL properties of the Hubble space telescope. More effort is nevertheless needed to enhance modelling and analysis convenience. In addition, it seems favourable to link stochastic features to existing modelling and analysis tools that provide an open and extensible architecture. We are currently making efforts to incorporate stochastic modelling and analysis features into the CADP toolset [9, 20, 18].

Markov chain models have been the clear focus of this paper. Their memory-less property considerably simplifies both modelling and analysis, but the property also implies that some real-world phenomena can only roughly be approximated with Markov chains. Hence, there is a need to extend the framework sketched in this paper beyond Markov models. The work of D'Argenio et al. [15, 16, 17] develops a process algebra to specify non-Markov performance and reliability models in an elegant way. So, the benefits of a process algebraic formalism extend to performance and reliability modelling in general. Anyhow, the analysis of such models needs further investigations. First results in this direction, however, indicate that numerical solution methods are impractical in general [30]. We are currently developing an open analysis environment for such specifications, linking to UPPAAL [31] for real-time model checking and to MÖBIUS [14] for discrete event simulation and numerical analysis.

Acknowledgements. Pedro R. D'Argenio, Boudewijn Haverkort and Joost-Pieter Katoen have provided very valuable comments on an earlier version of this paper.

References

[1] M. Ajmone Marsan, G. Balbo, G. Conte, S. Donatelli, and G. Franceschinis. *Modelling with Generalized Stochastic Petri Nets*. Wiley, 1995.

[2] A. Aziz, K. Sanwal, V. Singhal and R. Brayton. Verifying continuous time Markov chains. In *Computer Aided Verification (CAV 96)*, LNCS 1102:269–276, Springer, 1996.

[3] C. Baier, B.R. Haverkort, H. Hermanns and J.-P. Katoen. Model checking continuous-time Markov chains by transient analysis. In *Computer Aided Verification (CAV 2000)*, LNCS 1855:358-372, Springer, 2000.

[4] C. Baier, B.R. Haverkort, H. Hermanns and J.-P. Katoen. On the logical characterisation of performability properties. In *International Colloquium on Automata, Languages, and Programming (ICALP 2000)*, LNCS 1853:780-792, 2000.

[5] C. Baier, J.-P. Katoen and H. Hermanns. Approximate symbolic model checking of continuous-time Markov chains. In *Concurrency Theory (CONCUR 99)*, LNCS 1664:146–162, Springer, 1999.

[6] M. Bernardo, W.R. Cleaveland, S.T. Sims, and W.J. Stewart. TwoTowers: A tool integrating function and performance analysis of concurrent systems. In Proc. of IFIP Joint Int. Conf. on Formal Description Techniques and Protocol Specification, Testing and Verification. North Holland, 1998.

[7] T. Bolognesi and E. Brinksma. Introduction to the ISO Specification Language LOTOS. *Computer Networks and ISDN Systems* 14:25-59, 1987.

[8] K.M. Chandy, U. Herzog, L. Woo. Parametric Analysis of Queuing Models. *IBM Journal of Research and Development* 19(1):36–42, 1975.

[9] M. Cherif, H. Garavel, and H. Hermanns. bcg_min – Minimzation of normal, probabilistic, or stochastic labeled transitions systems encoded in the BCG format. `http://www.inrialpes.fr/vasy/cadp/man/bcg_min.html`

[10] G. Clark, S. Gilmore, J. Hillston, and N. Thomas. Experiences with the PEPA performance modelling tools. *IEE Proceedings–Software* 146(1):11-19, February 1999.

[11] E.M. Clarke, E.A. Emerson and A.P. Sistla. Automatic verification of finite-state concurrent systems using temporal logic specifications. *ACM Tr. on Progr. Lang. and Sys.* 8(2):244-263, 1986.

[12] E.M. Clarke and R.P. Kurshan. Computer-aided verification. IEEE Spectrum 33(6):61–67, 1996.

[13] E.M. Clarke, O. Grumberg and D. Peled. *Model Checking*. MIT Press, 1999.

[14] D. Daly, D.D. Deavours, J.M. Doyle, P.G. Webster, and W.H. Sanders. Möbius: An extensible tool for performance and dependability modeling. In *Computer Performance Evaluation*, LNCS 1786:332–336, 2000.

[15] P.R D'Argenio. *Algebras and Automata for Timed and Stochastic Systems*. PhD-Thesis, University of Twente, November 1999.

[16] P.R. D'Argenio, J.-P. Katoen E. Brinksma. Specification and Analysis of Soft Real-Time Systems: Quantity and Quality. In *Proc. of the 20th IEEE Real-Time Systems Symposium*, pp. 104-114, Phoenix, Arizona, December 1999. IEEE Computer Society Press.

[17] P.R. D'Argenio, H. Hermanns, J.-P. Katoen, and R. Klaren. MoDeST – A modelling and description language for stochastic timed systems. To appear in *Proc. Joint International PAPM-PROBMIV 2001 workshop*, Springer LNCS, 2001.

[18] J.-C. Fernandez, H. Garavel, A. Kerbrat, R. Mateescu, L. Mounier and M. Sighireanu. CADP (Caesar/Aldébaran Development Package): A protocol validation and verification toolbox. In *Computer Aided Verification (CAV 96)*, LNCS 1102:437-440, Springer, 1996.

[19] W.J. Fokkink. *Introduction to Process Algebra*. Texts in Theoretical Computer Science, Springer, 2000.

[20] H. Garavel. OPEN/CÆSAR: An open software architecture for verification, simulation, and testing. In B. Steffen, ed, *Tools and Algorithms for the Construction and Analysis of Systems (TACAS 98)*, LNCS 1384:68–84, Springer, 1998.

[21] H. Garavel and M. Sighireanu. On the Introduction of Exceptions in E-LOTOS. In R. Gotzhein and J. Bredereke, editors, *Formal Description Techniques IX*, pp. 469-484, Chapman and Hall, 1996.

[22] B. Haverkort. SPN2MGM: Tool support for matrix-geometric stochastic Petri nets. In *Proc. of IEEE International Computer Performance and Dependability Symposium*, pp. 219–228, Urbana-Champaign, Illinois, September 1996. IEEE Computer Society Press.

[23] B. Haverkort. private communication. 2000.

[24] H. Hermanns. *Interactive Markov Chains*. PhD thesis, Universität Erlangen-Nürnberg, 1998.

[25] H. Hermanns. Performance and reliability model checking and model construction. In *Proc. 5th International ERCIM Workshop on Formal Methods for Industrial Critical Systems (FMICS 2000)*, GMD Report 91:11–28, 2000.

[26] H. Hermanns, U. Herzog, U. Klehmet, V. Mertsiotakis and M. Siegle. Compositional performance modelling with the TIPPtool. *Performance Evaluation* 39(1-4):5–35, 2000.

[27] H. Hermanns and J.-P. Katoen. Automated compositional Markov chain generation for a plain-old telephony system. *Science of Computer Programming* 36(1):97–127, 2000.

[28] H. Hermanns, J.-P. Katoen, J. Meyer-Kayser and M. Siegle. A Markov chain model checker. In *Tools and Algorithms for the Construction and Analysis of Systems (TACAS 2000)*, LNCS 1786:347-362, Springer, 2000.

[29] H. Hermanns, J.-P. Katoen, and J. Meyer-Kayser. Towards model checking stochastic process algebra, 2000. In *2nd Int. Conference on Integrated Formal Methods (IFM 2000)*, LNCS 1945:420-439, Springer, 2000.

[30] G.G. Infante-Lopez, H. Hermanns, and J.-P. Katoen. Beyond memoryless distributions: Model checking semi-Markov chains. To appear in *Proc. Joint International PAPM-PROBMIV 2001 workshop*, Springer LNCS, 2001.

[31] K.G. Larsen, P. Pettersson, and W. Yi. Uppaal in a nutshell. *Int. J. of Software Tools for Technology Transfer*, 1(1/2):134–152, 1997.

[32] C. Lindemann and R. German. Modeling discrete event systems with state-dependent deterministic service times. *Discrete Event Dynamic Systems: Theory and Applications* 3:249–270, July 1993.

[33] R. Jain. *The Art of Computer Systems Performance Analysis*. J. Wiley, New York, 1991.

[34] A.A. Markov. Extension of the limit theorems of probability theory to a sum of variables connected in a chain. 1907. reprinted in Appendix B of R. Howard. *Dynamic Probabilistic Systems*, volume 1: Markov Chains. John Wiley and Sons, 1971.

[35] E.-R. Olderog and C.A.R. Hoare. Specification Oriented Semantics for Communicating Processes. *Acta Informatica* 23:9–66, 1986.

[36] B. Plateau and K. Atif. Stochastic Automata Network for Modeling Parallel Systems. *IEEE Transactions on Software Engineering*, 17(10), 1991.

[37] W.H. Sanders, *Construction and solution of performability models based on stochastic activity networks*, Ph.D. thesis, University of Michigan, 1988.

[38] H.A. Simon and A. Ando. Aggregation of Variables in Dynamic Systems. *Econometrica* 29:111-138, 1961.

[39] A. N. Shiryaev. *Probability.* Graduate texts in Mathematics. Springer, 1989.

[40] Space Telescope Science Institute Home Page. `http://www.stsci.edu/`

[41] W.J. Stewart. *Introduction to the Numerical Solution of Markov Chains.* Princeton Univ. Press, 1994.

[42] C.A. Vissers, G. Scollo, M. van Sinderen, and E. Brinksma. Specification Styles in Distributed Systems Design and Verification. *Theoretical Computer Science*, 89(1):179–206, 1991.