Towards Reliable Modelling with Stochastic Process Algebras

Jeremy Thomas Bradley

Department of Computer Science

University of Bristol

October, 1999

A dissertation submitted to the University of Bristol in accordance with the requirements for the degree of Doctor of Philosophy in the Faculty of Engineering.

Abstract

In this thesis, we investigate reliable modelling within a stochastic process algebra framework. Primarily, we consider issues of variance in stochastic process algebras as a measure of model reliability. This is in contrast to previous research in the field which has tended to centre around mean behaviour and steady-state solutions.

We present a method of stochastic aggregation for analysing generally-distributed processes. This allows us more descriptive power in representing stochastic systems and thus gives us the ability to create more accurate models.

We improve upon two well-developed Markovian process algebras and show how their simpler paradigm can be brought to bear on more realistic synchronisation models. Now, reliable performance figures can be obtained for systems, where previously only approximations of unknown accuracy were possible.

Finally, we describe reliability definitions and variance metrics in stochastic models and demonstrate how systems can be made more reliable through careful combination under stochastic process algebra operators.

Acknowledgements

My three years in the department in Bristol have been a lot of fun and the person I have most to thank for this is my friend and mentor, Neil Davies.

I should also acknowledge the funding from NATS for my project and especially the help of Suresh Tewari (NATS) and Gordon Hughes (SSRC).

On the research side, I have to thank Judy Holyer, Peter Thompson, Ian Holyer, Dave Tweed, Adel Jomah and Pauline Francis-Cobley, here in Bristol, as well as Graham Clark, Stephen Gilmore, Jane Hillston, Nigel Thomas, Rob Pooley and Helen Wilson whose comments, suggestions and encouragement have all contributed enormously.

Personally, I would like to express my gratitude to Paul Dias, Chris Vowden, Simon Sleight, Anders and Anne-Mette Spilling and my brother Tim, all of whom saw both sides of the PhD and helped me survive.

To my parents, I owe a huge debt of thanks for supporting and encouraging me tirelessly throughout my education, not to mention putting up with me for the last three weeks of thesis-writing.

Finally, for Helen, who was always willing to listen, kept me going when it all seemed fruitless and displayed incredible patience and strength from 7000 miles away—thank you for everything!

To Helen

Declaration

I declare that the work in this dissertation was carried out in accordance with the Regulations of the University of Bristol. The work is original except where indicated by special reference in the text and no part of the dissertation has been submitted for any other degree.

Any views expressed in the dissertation are those of the author and in no way represent those of the University of Bristol.

The dissertation has not been presented to any other University for examination either in the United Kingdom or overseas.

Signed: Date:

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Modelling Communicating Systems

Communicating systems are traditionally modelled using a black-box philosophy that uses the mantra that *observation is everything*. This means that a system is effectively defined by its outputs and its inputs and nothing else. Any system which produces the same output in response to identical inputs is considered an identical system. It may well be that the system achieves its output using completely different algorithms or hardware, but as long as the external observable behaviour is preserved, that is all that matters. This is encapsulated in the observational bisimilarity concept presented by Milner in CCS [76, 77, 78].

1.2 Black-Box Modelling

This observationalist philosophy is not restricted to computer science or computer systems—as might be expected, it has roots in experimental sciences too. If a collection of atoms is excited and the resulting emitted light is analysed spectrally, the various peaks in the wavelength spectrum betray the constituent elements of the original atoms. In this case the input is the energy that is expended on the atoms and the output constitutes the photons that result. The implicit assumption is that the same entity is generating each peak each time the experiment is performed. This confidence is gained from repeated experimentation on elements which also have behaved the same under the same inputs. This is the key; all experimentation can be considered to be an input to a system and the resulting behaviour, whether physical or chemical, the output.

On a more grand scale, physicists' attempts to construct theories of the universe fall into the same category. The models which they come up with can only ever be at best observationally equivalent (bisimilar in CCS terms) to reality. It can never be actually determined whether the correct internal mechanisms have also been replicated. In the event that an experiment could be devised which would betray a mechanism, then it would no longer be an *internal* mechanism and would now constitute observable behaviour that any correct theory or model would have to replicate or predict.

In artificial intelligence, observational bisimilarity is the basis for one of the more famous thought experiments. Alan Turing proposed that if a system could behave in a way indistinguishable from a human, then it would have achieved intelligence. What makes this hard to verify in practice is that there is no obvious formal definition of "indistinguishable" and similarly no formal definition of what intelligent behaviour is or how much it can vary and still be called intelligent.

This is where the formal definitions of bisimilarities of CCS and other calculi come in to play. They allow relations to be defined which relate similarly behaving systems, so in effect giving a rigorous meaning to the phrase "similarly behaving".

1.3 Fault-Tolerance and Timely Behaviour

So how does this relate to the reliability requirements that an organisation like the CAA[1] requires of its systems and software?

There are two classes of reliability. There is the traditional functional correctness of a formally verified system, where a logic tool such as Z or indeed CCS is used to guarantee that correct operation will occur. Then there is the timely behaviour of a system or temporal reliability, which reasons about how long a system is likely to take to perform a task. It is the study of this temporal reliability which is the subject of this project.

For the CAA both types of reliability are an issue. Both temporal and functional reliability should make up their *flavour* of observational bisimilarity. Using such a flavour, two systems might be identical if they both produce the same outputs given the same inputs within a specific amount of time and with a given probability.

A system may, with complete surety, find a minimum length solution to the Travelling Salesman Problem, but if it takes many years to do so for a required network, then this may only be of limited value. The point is that the timing, or relative timing, of a result or output from a system (physical or computational) is as much part of the observable behaviour as the actual result itself.

This can be seen by going back to an example from the physical world. If a system contains atoms of Carbon-14, then it is known that it is an element of Carbon-14 because the half-life of the radioactive decay can be measured, that is, the relative timings of the decays identify the element, not the type of decay (in this case beta decay)—which is the same for many other isotopes of other elements.

This extra timing expressiveness is absent from functional process algebras such as CCS and adding it in has been the subject of a great deal of research

[1] The Civil Aviation Authority—NATS, a division of the CAA, sponsors this project.

in recent years. This thesis is particularly concerned with having the correct type of timing model to allow us to talk about temporal reliability in as general a way as possible.

To achieve this, we investigate timed stochastic extensions to CCS and the like. These are known as stochastic process algebras and, in their most general form, represent a completely general modelling environment for representing real-world phenomena with either precision or uncertainty, as required.

1.4 Our Research Goal

Our research goal is first to investigate a sufficiently expressive modelling paradigm, stochastic process algebras, that will allow us to express measures of reliability. Reliability and performance inevitably go hand in hand, often involving a trade-off of one for the other. Therefore, in this research, we investigate both the accuracy of current performance modelling stochastic process algebras, and methods for expressing and extracting reliability measures from systems.

1.5 How this Document is Structured

Chapter 1 gives a high level perspective on some of the broader scientific and philosophical issues which motivate this work.

Chapter 2 formally introduces some of the modelling techniques mentioned in the introduction. It sets out general techniques for modelling communicating systems and then gives an overview of other methodologies which make use of timing and stochastic information. This chapter provides the necessary background in which we set our work on reliability modelling in the rest of the thesis.

Chapter 3 describes an analysis technique known as *stochastic aggregation* that we use to compare some of the stochastic process algebra modelling techniques from chapter 2. It provides interesting insights into some of the added problems encountered when working with stochastically defined systems. We go on to use this method continually in chapters 4 and 5.

Chapter 4 compares two Markovian process algebras, PEPA and MTIPP, introduced in chapter 2, to measure their reliability in performance analysis. It applies the stochastic aggregation techniques of chapter 3 to analyse the accuracy of the two systems. This is one aspect of reliability—the ability of a paradigm to reproduce accurately the correct results, or if making an approximation to have an idea about how close the answer is to reality.

Chapter 5 specifically considers model reliability from a stochastic process algebra perspective using the techniques of chapter 3. Definitions and ways of measuring reliability are suggested and some example systems are analysed. In particular it is seen how the variance of a system plays a crucial role in the reliability and predictability of a system.

Finally, chapter 6 summarises what we have covered and discusses how well we have been able to achieve our reliability modelling goal. Future lines of investigation are discussed.

1.6 Notation

Throughout this thesis, we use standard mathematical notation to represent probabilistic concepts, in a style similar to that of Trivedi [97] and many others. In particular:

Random variables are defined by $X \sim \exp(\lambda)$, for instance, which means, the random variable X samples from the exponential distribution with parameter λ. Other types of distribution used include: Gamma, $\Gamma(n, \lambda)$; Uniform, $\mathrm{U}[a, b]$; Deterministic, $\delta(\nu)$; Hyperexponential, $\mathrm{Hyper}(\alpha_i; \lambda_i)$.

Distribution Functions are represented by f, F and L. The probability distribution function of the random variable X would be $f_X(t)$; the cumulative distribution function of X would be $F_X(t)$; and the Laplacian of X would be $L_X(\omega)$. Occasionally we use $\overline{F}_X(t)$ to mean the complement of the cumulative distribution function, so $\overline{F}_X(t) = 1 - F_X(t)$.

Chapter 2

Stochastic Process Algebras

2.1 Introduction

In this chapter, we introduce traditional process algebras as a technique for modelling communicating systems. We then discuss how process algebras were augmented to include timing information. This serves as an introduction to the main issue of the chapter: stochastic process algebras.

We will show why stochastic process algebras are more expressive than either standard or timed process algebras. We will examine the current state of the art in stochastic process algebras, concentrating on the analysis techniques used. This is so that, in the final section, we can discuss the physical interpretation of these analytical results and show why they are not sufficient on their own to model reliability.

Having looked at system modelling from a process algebra perspective, we will give a summary of two other stochastic techniques, stochastic Petri nets and queueing networks. This completes our picture of stochastic modelling since we then have a basis on which to compare stochastic process algebras with other techniques.

2.2 Process Algebras

Why use process algebras? There are many formalisms for representing computer systems or communicating systems, so we need to justify selecting process algebras. Milner [76] identifies two key areas which he regarded as central ideas behind his development of CCS: *observation* and *synchronised communication*.

As discussed in chapter 1, modelling observable interaction is a very natural way of defining the operation of an object, much more so because it lends itself so easily to abstraction. Higher level modelling becomes simply a matter of drawing larger black boxes around your components and hiding more internal working.

The concept of synchronised communication that Milner talks about embodies the whole mechanism of component-based modelling and concurrent composition. Models can be succinctly defined in terms of simple components which interact with each other to define the operation of the whole.

This issue of *parsimony* (in this context, succinctness of expression) is also important, both parsimony of the model and of the algebra. The first allows for an easy conceptual construction and subsequent understanding of a model while the second encourages further use of the paradigm. These are clearly both essential qualities if a formalism is to be successful.

Then there are the reasons that any formalism is used at all—the very fact that it provides a formal definition of a system means that ambiguities in design have to be sorted out and errors removed. The understanding of a system is always considerably improved by the use of a formal process and this is bound to make for more reliable and consistent projects. Also as a result of the formal expression process, a system can be formally and in many cases automatically reasoned about. Properties such as livelock and deadlock can be checked for; if a system invariant is specified it can sometimes be possible to verify formally the complete operation of the system against that invariant (although this is usually restricted by practical computation time and space limits).

By no means least there is the æsthetic elegance of the language itself: the fact that such a few language expressions can, when combined, define a complex communicating system—at least functionally. Examples of these functional process algebras include CCS (Calculus of Communicating Systems) [76], CSP (Communicating Sequential Processes) [60, 61] and ACP (Algebra for Communicating Processes) [7]; we use CCS below as an introduction to the notation.

2.2.1 CCS—A Calculus of Communicating Systems

Milner defines CCS [76, 78] as follows:

$$P \quad ::= \quad \mathbf{0} \;\Big|\; a.P \;\Big|\; P+P \;\Big|\; P|P \;\Big|\; P\backslash L \;\Big|\; P[f] \;\Big|\; A \tag{2.1}$$

Prefix $a.P$ represents the evolution of either an input or output action where a is an input event and $\overline{a}$ is an output. $a.P \xrightarrow{a}$P means $a.P$ emits an action a and proceeds to P.

Choice $P + Q$ is a choice operator and means that if $P \xrightarrow{a} P'$ then $P+Q \xrightarrow{a} P'$ or if $Q \xrightarrow{b} Q'$ then $P+Q \xrightarrow{b} Q'$.

Synchronisation $P|Q$ denotes concurrent composition. If $P \xrightarrow{a} P'$ and $Q \xrightarrow{\overline{a}} Q'$ then $P|Q \xrightarrow{\tau} P'|Q'$ where τ is a silent internal action. If the actions do not form an input/output pair then the components evolve unaffected by one another.

Restriction $P\backslash L$. Specifically useful for abstraction and operation hiding: actions in the set L are not observable outside the system P, i.e. they do not evolve.

Relabelling $P[f]$ is a process P with its actions relabelled by a function f. Useful for modularisation and component reuse.

Constant $A \stackrel{\text{def}}{=} P$ is used to specify labels for parts (or agents) of a component or system.

In some extensions there is also an explicit recursion operator, $\mathbf{fix}(X = P)$, which behaves as P with occurrences of the variable X replaced by $\mathbf{fix}(X = P)$.

2.3 Timed Process Algebras

2.3.1 Temporal CCS

The development of Temporal CCS [95, 96] is an interesting one since it encapsulates a subtle change in semantics to reflect a different model of timing.

In the first version of Temporal CCS [95], a system is presented as follows:

$$P \ ::= \ \mathbf{0} \,\Big|\, X \,\Big|\, \lambda[t].P \,\Big|\, P + P \,\Big|\, P|P \,\Big|\, P\backslash L \,\Big|\, P[S] \,\Big|\, \mathbf{fix}_X P \tag{2.2}$$

where the prefix $\lambda[t].P$ represents a process which takes at least time t to emit an action λ and enter state P. The other combinators are similar to traditional CCS.

A year later, the second version [96] had the following differences in definition:

$$P \ ::= \ \mathbf{0} \,\Big|\, X \,\Big|\, \lambda.P \,\Big|\, (t).P \,\Big|\, \delta.P \,\Big|\, P + P \,\Big|\, P|P \,\Big|\, \ldots \tag{2.3}$$

Here now, the timing aspect of the system is removed from the prefix operation. There is now an independent form $(t).P$ which represents a delay of exactly time t. The $\delta.P$ describes a passive component which can wait for an arbitrary amount of time before proceeding into state P. This is a necessary addition for synchronising components, where one component gets to a synchronising action first and has to wait for the other process to catch up.

The major philosophical change was this devolution of action and time. This was motivated by a result from quantum mechanics, as described in [96]:

> Computation involves energy change, and there is a result of quantum mechanics which states that energy changes and time cannot be measured simultaneously. Thus it seems reasonable when producing an observation-based model of time and computation not to permit simultaneous observation of these two activities.

This obviously has implications for the type of timed process algebra we will select for use in reliability modelling so we will discuss this issue later with respect to stochastic process algebras (section 2.4.2).

A final form of Temporal CCS [79] used a similar system definition to that of [96]:

$$P \;::=\; \mathbf{0} \;\Big|\; X \;\Big|\; \lambda.P \;\Big|\; (t).P \;\Big|\; \delta.P \;\Big|\; P+P \;\Big|\; P|P \;\Big|\; \dots \tag{2.4}$$

The prefix operation is represented by $\lambda.P$, the $(t).P$ and $\delta.P$ are as before from [96]. The summation operator $P + P$ is, for the first time, defined in temporal context and called *strong choice*. $A + B$ will behave as A if A evolves first and B if B evolves before A.

2.3.2 Timed CCS

Whereas Temporal CCS expresses actions in terms of precise evolutions of time, Timed CCS [21] models actions with intervals of time. This is an important difference, since we can now deal with uncertainty in the modelling process.

$$P \;::=\; \mathbf{0} \;\Big|\; X \;\Big|\; a(t)|_q^p.P \;\Big|\; P+P \;\Big|\; P|P \;\Big|\; P\backslash L \;\Big|\; P[S] \;\Big|\; \mathbf{fix}_X P \tag{2.5}$$

Here, the prefix operation, written $a(t)|_q^p.P$, represents an action a which can occur within an interval $p < t < q$ after a has been initialised.

With the uncertainty in action execution, the model starts to describe almost stochastic properties. This leads us nicely on to stochastic process algebras which precisely represent this uncertainty, as we will see.

2.4 Stochastic Process Algebras

2.4.1 Introduction

Stochastic process algebras differ from standard process algebras or timed process algebras by being able to represent *spatial uncertainty* explicitly—which event happens next—and *temporal uncertainty*—when an event happens [50]. They do this by assigning a random variable amount of time to the events of a system and having a method of probabilistic selection when a choice of action is available.

Most stochastic process algebras sample the random variables for the action times from exponential distributions with different rates. These are called Markovian process algebras and examples include PEPA [52], MTIPP [47], EMPA [9] and MPA [20]. More recently there have been developments in process algebras which allow for more generally-distributed action times; for instance, originally, TIPP [37] and ET-LOTOS [2] then later, a Stochastic Causality-Based Process Algebra [19], Stochastic π-Calculus [83] and GSMPA [18].

2.4.2 Time and Action

In most stochastic process algebras, time and action have been recombined in the prefix operation. This might be seen as being contrary to the quantum mechanical argument invoked by Tofts [96]—outlined in section 2.3.1.

The issue centres around whether time and action should be simultaneously observable given that energy change and time are not, in a quantum mechanical system. The first thing to note is that this argument only applies at the quantum level of interaction since Planck's constant is so small. Thus, for all modelling tasks on a non-quantum scale, combining time and action is a perfectly good approximation of reality. Indeed, the model itself is more likely to be a far greater source of error. Also the combination of action and time is a natural abstraction to make from the modeller's point of view—a calculation takes a particular amount of time.

If models do need to be constructed at a quantum scale, then how better to express the uncertainty principle [28] than by using a stochastic process algebra. If there is inherent uncertainty over the timing of an action, then this can be represented using an appropriate probability density function.

For all these reasons most SPAs (stochastic process algebras) maintain a direct link between action and timing[1].

2.4.3 Markovian Process Algebras

A very good introduction to performance modelling with Markovian process algebras is presented in Hermanns et al 1996 [44]. Markovian process algebras can be generally represented using the syntax:

$$P \quad ::= \quad (a, \lambda).P \,\Big|\, P + P \,\Big|\, P \,||_S\, P \,\Big|\, P/L \,\Big|\, A \tag{2.6}$$

Prefix $(a, \lambda).P$ is the prefix operation and describes an action a taking a random amount of time to occur. The random amount of time is sampled from an exponential distribution of rate λ and can therefore be in the range $0 < t < \infty$. This Markovian property also has the advantage that, if after a time t' the event has not occurred, then the remaining conditional distribution is identical to the original exponential distribution (see page 15 for details of the *memoryless property*).

Choice $P + P$ represents a competitive choice between two processes. The first process to evolve interrupts the other and the slower process is discarded from the system. This is known as a *race condition*.

Synchronisation $P \,||_S\, P$ is a synchronisation between two processes involving only the actions in the set S. The exact nature of the synchronisation differs from algebra to algebra, but it is usually intended that

[1] Hermanns' Interactive Markov Chains [43] is one of the few exceptions to this.

the rate of the overall synchronised event will in some way reflect the rate of the slower component.

A discussion of synchronisation strategies in the different Markovian process algebras can be found in Hillston 1994 [53]. One of the slightly contentious issues surrounding Markovian process algebras is the fact that the distribution describing the longer of two exponential distributions is itself not exponential. This means that technically MPAs do not form a closed model[2], however this is worked around in Markovian process algebras by approximating the synchronisation event with an exponential distribution. This issue and the consequences of the approximation are the subject of chapter 4.

Hiding P/L hides the actions of P that occur in the set L. This is an important modelling tool which allows internal events to be abstracted away from other components. A hidden action becomes a τ in the same way as it does in CCS, however successive τ's can only be merged into a single τ (as in observational bisimulation in CCS) under conditions of insensitivity (see section 3.1.1, which describes insensitivity).

Constant $A := P$ is the constant agent. It allows labels to be assigned to agents. By defining self-referential agents, recursion can be modelled.

The Performance Model

A performance model is obtained by establishing a translation from the algebra to a continuous time Markov chain (CTMC). First the set of reachable states X_i is constructed by eliminating component synchronisation using an expansion law (discussed in the Markovian case below). The elements of the infinitesimal generator matrix (for further details of Markov chain solution,

[2]Stochastic process algebras which minimally incorporate immediate transitions as well as exponential transitions such as EMPA [9, 8] and IMC [43, 45] can represent the maximum of two exponential distributions precisely.

see for example [68, 63]) are assigned from the rates of the state transition system:

$$G_{ij} = r(X_i, X_j) \quad : i \neq j \tag{2.7}$$

where $r(X_i, X_j)$ represents the total rate of transition from state X_i to X_j:

$$r(X_i, X_j) = \sum_k \sum_{X_i \xrightarrow{(a_k, \lambda_l)} X_j} \lambda_l \tag{2.8}$$

Finally the diagonal elements are just:

$$G_{ii} = -\sum_j r(X_i, X_j) \tag{2.9}$$

The steady-state probabilities $\boldsymbol{\pi}$ can be recovered from the equation:

$$\boldsymbol{\pi} G = \mathbf{0} \tag{2.10}$$

subject to $\sum_i \pi_i = 1$. These combined equations can be solved using Gaussian elimination to obtain $\boldsymbol{\pi}$. Now applying an appropriate reward structure to the steady-state probabilities will theoretically extract the required quantitative performance figures. A reward structure is just a weighting of the states to reflect how important a particular state is for a particular property. To construct a performance figure p, we would use a reward structure $\boldsymbol{r}_p$:

$$p = \boldsymbol{r}_p \cdot \boldsymbol{\pi} = \sum_i r_{p_i} \pi_i \tag{2.11}$$

The Memoryless Property

In stochastic terms, if $X \sim \exp(\lambda)$ and t' has elapsed then we are now interested in the random variable $Y = X - t'$, given that $X > t'$, thus the distribution $Y|X > t'$:

$$\begin{aligned}
F_{Y|X>t'}(t) &= \mathbb{P}(Y \leq t | X > t') \\
&= \frac{\mathbb{P}(t' < X \leq t + t')}{\mathbb{P}(X > t')} \\
&= \frac{1}{\overline{F}_X(t')} \int_{t'}^{t+t'} f_X(x)\,\mathrm{d}x \\
&= e^{\lambda t'}(F_X(t+t') - F_X(t')) \\
&= e^{\lambda t'}(-e^{-\lambda(t+t')} + e^{-\lambda t'}) \\
&= 1 - e^{-\lambda t}
\end{aligned} \tag{2.12}$$

So $Y|X > t' \sim \exp(\lambda)$ as required. This is the *memoryless property* of the exponential distribution and it is the only distribution to have this property. It is important to the process algebra paradigm because it allows for a relatively easy integration of stochastic processes into a CCS-like framework.

The expansion law of CCS rewrites a concurrent composition of components in terms of a single component with a choice of evolution according to which component evolves first.

$$C_1|\cdots|C_n = a_1.(C_1'|\cdots|C_n) + \cdots + a_n.(C_1|\cdots|C_n') \tag{2.13}$$

for $C_1 \xrightarrow{a_1} C_1', \ldots, C_n \xrightarrow{a_n} C_n'$.

This rewriting is called an *interleaving semantics* and is only made possible in Markovian process algebras because of the memoryless property of the action durations.

If the memoryless property did not exist then a concurrent composition of stochastic processes could not be rewritten in such a way. If one component evolved before another enabled component then the description of the residual distribution of the beaten component would have to take into account the distribution which beat it. This means that a memory of all the evolved processes would have to be incorporated in any component which was concurrently enabled.

As it is, with exponential processes, a pre-empted process still behaves like an exponential process and thus the same expansion law can be applied, without any need to incorporate traces of previous evolution.

PEPA

PEPA (Performance Enhanced Process Algebra) was presented primarily in [52, 55] and also [51, 50, 54]. PEPA has the following syntax:

$$P \ ::= \ (a,\lambda).P \,\Big|\, P+P \,\Big|\, P \underset{S}{\bowtie} P \,\Big|\, P/L \,\Big|\, A \tag{2.14}$$

The operators are identical to the description of a general Markovian process algebra of section 2.4.3 except that the synchronisation operator is now precisely defined. Cooperation, $P \underset{S}{\bowtie} Q$, is PEPA's method of specifying component synchronisation. Operationally, if $P \xrightarrow{(a,\lambda)} P'$, $Q \xrightarrow{(a,\mu)} Q'$ and $a \in S$ then $P \underset{S}{\bowtie} Q \xrightarrow{(a,\min(\lambda,\mu))} P' \underset{S}{\bowtie} Q'$, so the synchronised agent takes the rate of the slower component.

The synchronisation is slightly complicated by the existence of many possible a evolutions from the P and Q states. For this case, the *apparent rate*, $r_a(P)$, of a component is defined:

$$r_a(P) = \sum_{P \xrightarrow{(a,\lambda_i)}} \lambda_i \tag{2.15}$$

The reason for the name, apparent rate, is that when many identically labelled actions, a, are enabled simultaneously in a competitive choice, the evolution is indistinguishable from a single action a being enabled with rate $r_a(P)$. However, if this situation occurs in a cooperation of components then a particular a evolution has to be selected to participate in the cooperation. So for any given evolution $P \xrightarrow{(a,\lambda_i)}$ the probability that it is the first one to occur and thus participate in a cooperation is:

$$\mathbb{P}(X_i < \min(X_j : j \neq i)) = \frac{\lambda_i}{\sum_r \lambda_r} = \frac{\lambda_i}{r_a(P)} \tag{2.16}$$

where $X_r \sim \exp(\lambda_r)$ and $\min(X_j : j \neq i) \sim \exp(\sum_{r \neq i} \lambda_r)$.

So in the general case, if $P \xrightarrow{(a,\lambda)}$, $Q \xrightarrow{(a,\mu)}$ and $a \in S$ then $P \underset{S}{\bowtie} Q \xrightarrow{(a,\nu)}$, where:

$$\nu = \frac{\lambda}{r_a(P)} \frac{\mu}{r_a(Q)} \min(r_a(P), r_a(Q)) \tag{2.17}$$

The term:

$$\frac{\lambda}{r_a(P)} \frac{\mu}{r_a(Q)} \tag{2.18}$$

is a normalisation factor. The observable rate of the a-cooperation is in fact just $\min(r_a(P), r_a(Q))$, however (for m a-evolutions from P and n a-evolutions from Q) this is in turn made up of a choice between mn evolutions where $P \longrightarrow P_i$ and $Q \longrightarrow Q_j$ for specific i, j such that:

$$\sum_{i,j} \frac{\lambda_i}{r_a(P)} \frac{\mu_j}{r_a(Q)} \min(r_a(P), r_a(Q)) = \min(r_a(P), r_a(Q)) \tag{2.19}$$

which is the overall apparent rate of the cooperation mentioned earlier.

MTIPP

MTIPP (Markovian Timed Processes for Performance Evaluation) [47] has the syntax:

$$P \ ::= \ \mathbf{0} \,\Big|\, (a,\lambda).P \,\Big|\, P + P \,\Big|\, P \,||_S\, P \,\Big|\, \mathbf{rec}X : P \,\Big|\, X \tag{2.20}$$

The main difference from PEPA lies in the synchronisation operator $P \,||_S\, Q$ in which the synchronised agent inherits the product of the component rates. We investigate the merits of this construction in comparison to PEPA in chapter 4. The **rec** operator is the recursion operator and performs an identical task to that of the **fix** operator of CCS.

Markovian Process Algebras as a Modelling Paradigm

The trouble with Markovian process algebras is that it is difficult either to specify high level models accurately or to handle larger low level models tractably.

In common with other formal modelling paradigms, initially it is useful to be able to model at a high level of abstraction and rarely is it necessary to model the very lowest level events. So, at a higher level, the atomic actions will be a representation of a combination of more primitive actions (in some aggregated form). These primitive actions may be exponential in character but most combinations of them will not.

In summary, since the paradigm can only describe exponential and often therefore low-level actions, it generates huge state-spaces and model descriptions. These are, by definition, difficult to handle both computationally and conceptually. Further, it is not in general possible to abstract away from the low-level description to make the model simpler or smaller, because the modelling paradigm does not support the expressiveness to be able to do so.

So Markovian process algebras have their problems—we will see how other stochastic process algebras overcome some of these problems in further sections. However, they are by far the easiest to solve for steady-state distributions via direct translation to a CTMC and tools exist to do just that [33, 46]. Also, there is considerable research being carried out in the use of approximation and simplification techniques to reduce the model complexity problem (product-form solutions [91, 58] through techniques such as stochastic reversibility [67, 59], quasi-reversibility [40] and quasi-separability [94, 93]). Ultimately there may be considerable future in these product-form solutions as bounding models which are considerably easier to manipulate than the more complicated generally-distributed stochastic process algebras.

2.4.4 Generalised SPAs

Generalised stochastic process algebras incorporate not only exponential evolutions but also instantaneous transitions. This is usually done in order to include a probabilistic branch construct into a language, as in Rettelbach's PM-TIPP [85] or Bernardo et al's EMPA [9].

This then allows for greater distribution expressiveness in stochastic process algebra modelling and so generalised SPAs are part of the development history of stochastic process algebras. Once a probabilistic branch is formally capable of being modelled then distributions such as Coxian stage-types [25, 24, 97] can be expressed. These in turn can be used to approximate non-memoryless distributions and are therefore of interest to us when it comes to reliable modelling paradigms.

The probabilistic branch (represented using $\oplus$ throughout the thesis) in equation (2.21) represents a system which becomes B with probability p and C with probability $1 - p$. This choice represents an instantaneous change of state, and thus a transition, in process algebra terms.

$$A \stackrel{\text{def}}{=} [p]B \oplus [1-p]C \tag{2.21}$$

In PM-TIPP, an equivalence is set up to eliminate instantaneous transitions in the context of branching and translate them into identical competitive choice structures. Equation (2.22) shows an example of this elimination process:

$$(\lambda, a).([p]A \oplus [1-p]B) \quad \sim \quad (\lambda p, a).A + (\lambda(1-p), a).B \tag{2.22}$$

However, Hermanns et al [48] and subsequently [43] enriched an MTIPP-like language with immediate actions which need not necessarily occur within the confines of a branch:

$$P \quad ::= \quad \mathbf{0} \;\Big|\; (a, \lambda).P \;\Big|\; \underline{a}.P \;\Big|\; P + P \;\Big|\; P \,||_S\, P \;\Big|\; P/L \tag{2.23}$$

where $\underline{a}.P$ represents the immediate occurrence of an a action. This then has the advantage of allowing a precise representation of a slower-action or Last-to-Finish (in the language of chapter 4) synchronisation.

It might be thought that immediate transitions might be easily approximated by using exponential distributions with very high rates. This leads to *stiffness* problems when translated into the underlying Markov process. Numerical solutions of such a system can become overrun by the much larger rate figures from the instantaneous transitions and the detail of the solution from the slower, smaller rates can be lost.

In using such an algebra to model with Coxian stage-type distributions, however, we run the risk of generating huge underlying Markov chains for even very simple models. A better solution still is to use a process algebra which can work with general distributions directly.

2.4.5 Generally-Distributed SPAs

Generally-distributed stochastic process algebras are clearly of greatest interest to us since they are far more expressive than Markovian process algebras and are therefore more likely to be able to model a system accurately. An accurate underlying stochastic model is going to be essential for meaningful reliability analysis.

Although there are a few generally-distributed stochastic process algebras that have defined the semantics necessary to deal with non-memoryless distributions (TIPP [37], GSMPA [17, 18], Stochastic π-Calculus [83] and stochastic bundle event structures [19]), few also provide a framework for generating stochastic figures, such as steady-state probabilities, and none consider issues of reliability modelling.

In chapter 3, we will investigate a framework for describing and analysing the generally-distributed stochastic processes in stochastic process algebras. We will then apply this to judge the accuracy of Markovian process algebra performance figures in chapter 4 and then go on to develop reliability metrics in chapter 5.

2.5 Other Stochastic Process Descriptions

2.5.1 Queueing Processes

Queues are a set of stochastic processes, known as Birth-Death processes, which have been analysed extensively [23, 70, 71, 5, 38] since Kendall's 1951 article outlining problems in the field [69].

Although they are considered too restrictive a model for general system modelling, the wealth of stochastic results in the area will be useful to us when trying to verify generally-distributed process algebra results in chapter 3.

A queue consists of an arrival process, a buffer and one or more service processes. A customer arrives according to some specified distribution, is queued in a (possibly infinite) buffer and is eventually serviced and removed from the system.

Kendall laid down the foundations for describing a general queueing process [69]. This has become known as *Kendall notation*. Using this form, a queue is represented by the string: A/S/N[/K[/n]], where the last one or two identifiers may be omitted.

A describes the distribution of the arrival process. Some common values are M for Markovian, GI or G for general independent, D for deterministic.

S describes the distribution of the service process.

N denotes the number of servers servicing the buffer.

K represents the size of the buffer. If omitted it is ∞ by default.

n represents the number of customers. If omitted it is ∞ by default.

If the buffer is empty at any stage then the service process has to idle until a customer arrives. Even then a full service time has to expire before that customer is serviced. If the buffer is full, then arriving customers may be discarded or blocked.

2.5.2 Stochastic Extensions to Petri Nets

Petri nets were first conceived by Carl Petri [81]. They predate traditional process algebras at being able to model concurrent systems. However they differ from process algebras since they are specifically tailored to model process causality rather than state transitions. They concentrate on resource and process dependency and are largely orthogonal to process algebras, where the emphasis is on composable inter-communicating components.

Petri nets are directed bipartite graphs. The nodes are split into two categories, *places* and *transitions*, with directed edges between the two categories. Tokens flow through the graph, waiting in the places until all the places incident on a transition contain a token. At this stage the transition fires and at least one of the tokens from each of the incident places moves to the places immediately downstream of the firing transition.

Each arrangement of tokens is called a *marking* and each marking is equivalent to a state of a process. The collection of all reachable markings is identical to the reachable states of a process. An example of a Petri net is shown in figure 2.1; the circles are the places and the rectangular bars represent the transitions.

Stochastic firings were first introduced into Petri nets by Molloy [80]. These SPNs allowed transitions to fire with exponential delay and, similar to Markovian process algebras, an underlying Markov chain model could be extracted and solved for a steady-state distribution.

GSPNs (Generalised Stochastic Petri Nets) [4, 1] had both immediate and exponential transitions present in the same net and still allowed the steady-state probabilities to be derived. These were later extended to DSPNs (Deterministic Stochastic Petri Nets) [3] which could incorporate Markovian and deterministic transitions [74].

Although the disciplines of stochastic process algebras and stochastic Petri nets are very different in nature, they should still be capable of similar analytical tasks. In a stochastic process algebra context SPNs are very useful

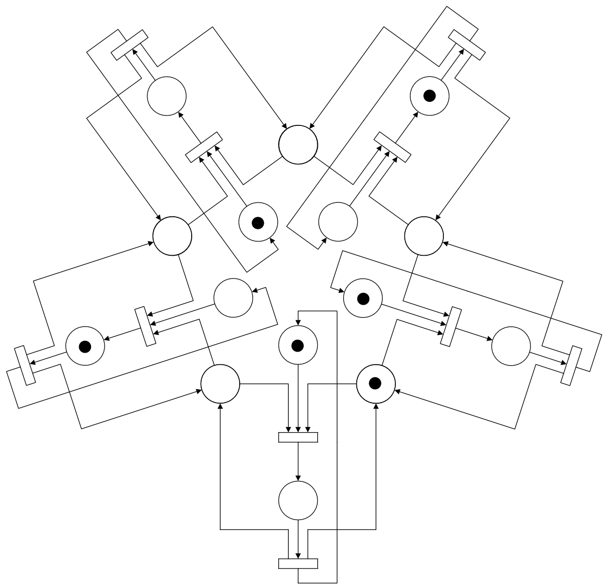

Fig. 2.1. A Petri net representation of the Dining Philosophers' problem.

therefore in being able to verify that identically specified systems generate the same performance results.

2.6 Conclusion

In summarising the formalisms and methodologies in this chapter, there are a few points to note about the current research emphasis and its impact on reliable modelling.

While stochastic process algebras are our preferred choice of stochastic modelling tool, for all the reasons given in section 2.2—parsimony, expressiveness, formal reasoning, compositional style—the solution techniques currently used are not necessarily the be-all and end-all of stochastic analysis.

Steady-state distributions are essentially a measure of long term average behaviour and can indeed be used to obtain expected values for throughput and usage. However, such statistics will also only apply in the long term and no statement is made about how long a given system might take to reach this steady-state.

Ironically, as the Ehrenfest Paradox [67] indicates, a system in steady-state equilibrium is also at its point of maximum entropy. This means that we actually know least about it in this state, compared to when we started the system off or any other time in the systems history.

While long term statistics are definitely part of the picture when understanding stochastic systems, there is a need to augment this information with some idea of how much the system can vary while running. Measures of average behaviour, by definition, remove any variant, extreme or tail-end behaviour. This variation from the mean is exactly the type of behaviour that reliability analysis is especially concerned with.

In chapter 5, we look at specifically this issue: how we can augment steady-state information with concepts of extreme behaviour.

In the meantime, chapter 3 investigates steady-state solution techniques more thoroughly and devises a method of stochastic analysis, called stochastic aggregation, which we can apply to generally-distributed systems.

We are also interested in ensuring that accurate performance information can be obtained from current tools. To this end, chapter 4 uses the stochastic aggregation of chapter 3 to investigate the effects of the approximate synchronisation models of Markovian process algebras, MTIPP and PEPA.

Chapter 3

Analysing Stochastic Systems

3.1 Introduction

In this chapter, we present a method of *stochastic aggregation* which effectively reduces the structural complexity of a generally-distributed stochastic system by embedding it into a probability distribution function within the system. By reducing the structure to a simple normal form, we can tractably undertake both performance and reliability analysis.

As discussed in chapter 2, if we are to have any success in modelling reliability in systems, we need to be able to describe those systems accurately. Thus we require the ability to model non-Markovian durations. Several process algebras exist which semantically describe systems with generally-distributed transitions [37, 2, 19, 83, 18]. However, the problem of analysing these systems in as general a way as possible is a mathematically hard one and many of these systems cannot provide any solution techniques to produce, for instance, performance statistics.

We introduce stochastic transitions systems as an underlying model for describing an arbitrary stochastic process. We then consider how we might produce steady-state probabilities for this paradigm using a method of stochastic aggregation (section 3.1.1, originally presented in [12]). By doing this, we immediately give ourselves the ability to compare our results with steady-state solutions from other paradigms, such as Markovian process algebras and generally-distributed queues (sections 3.3.3 and 3.3.2). This gives us a level of confidence in our stochastic transitions system technique.

In the latter part of the chapter, we provide an alternative progressive solution method for Markovian process algebras which does not involve continuous-time Markov chains. Also, with the aid of a queueing system, we demonstrate why not having an interleaving semantics for generally-distributed concurrent processes becomes such a computational problem.

We show that we can obtain the complete steady-state distribution at the component level of a generally-distributed stochastic process algebra (section 3.2.7). There still remains the problem of how to analyse the synchronising components that make up a fully-functioning system. To this end, we round off the chapter by demonstrating a *component aggregation* technique (section 3.3.4): how it might be possible to analyse a generally-distributed process algebra at a component level of the model. This would be similar to a product form solution since it does not require the complete interleaved state space generation that hampers other process algebras. Instead, components can be analysed one at a time and the collective results from each component would together form a steady-state distribution.

3.1.1 Stochastic Aggregation

Stochastic aggregation is a process of combining stochastic processes and structures into a single process. It is a well-known result from Trivedi [97] that program structures, such as *if-then-else* statements, *while-loops* and *for-loops*, can be represented stochastically and then aggregated into a single equivalent process. This allows us to aggregate sequences and combinations of such structures into single processes.

What we are able to do is show that for cyclic systems expressed in our stochastic transition system, for each state, we can aggregate the system into a normal form. These normal forms are special precisely because they can be analytically solved to give stationary distributions. They also represent a way of maintaining the complete stochastic nature of a system, but in a reduced structural form.

The key to this aggregation technique is not to lose sight of the original system totally. By carefully aggregating around a selected state, the normal form is generated still containing a replica of that state, relative to the rest of the system. Solving this normal form for the stationary distribution now yields the correct stationary probability for the original state. This is then repeated for the remaining states, to obtain the entire stationary distribution for the whole system.

The method presented differs from a technique such as *insensitivity* analysis [75, 41, 87] in its general goal, since it specifically seeks not to change the model under consideration. Insensitivity when applied to GSMPs (generalised semi-Markov processes) [75] looks for structures of generally-distributed processes which, when replaced with exponential processes with the same mean, will still have the same steady-state distribution. Insensitivity has been applied with significant result to models of both generalised stochastic Petri nets by Henderson [42] and stochastic process algebras by Clark [22]. However, given that ultimately we seek to obtain more than just steady-state information (specifically in chapter 5), we require a framework which specifically does not alter the model being analysed. By design, then, stochastic aggregation preserves the stochastic nature of a system.

3.2 Stochastic Transition Systems

3.2.1 Introduction

In this section, we present a solution technique for a generally-distributed stochastic transition system. The method was inspired by an example originally presented in Cox and Miller [26]; it has been extended to use general-distributions and, through aggregation, provides a solution technique for more than the original two state system.

A transition system is defined to describe individual evolutions from state to state. Time is represented along each transition by a continuous random variable and it is these random variables which can take any distribution (figure 3.1 shows an example of a stochastic transition system).

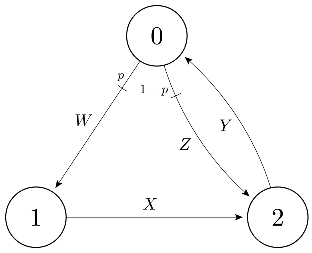

Fig. 3.1. A stochastic transition system with random variable transitions and probabilistic branching.

In this instance, as with other stochastic models, our primary goal is to obtain a stationary distribution for the system. To this end, we describe a simple example model (the Cox-Miller Normal Form) which can be solved directly to give the stationary distribution. The remainder of the section is devoted to showing how general models can be simplified to similar normal forms, which can also be solved to give stationary distributions.

We call this process of structural simplification *stochastic aggregation*. It is carried out in such a way that, from the stationary probabilities of the normal form, stationary values for the original system can be deduced.

3.2.2 Motivation

In designing a stochastic transition system, we are keen to avoid many of the complexities that make other generally-distributed systems (queueing systems, for instance) very difficult to solve. In particular, by steering clear of any notion of concurrent evolution, it is seen that the memoryless issue can be avoided at this stage. Although it is clear that we are reducing the expressiveness of our language by doing so, a solution technique for some class of generally-distributed system is still of interest to us. We will see later that we are still able to model even generally-distributed concurrent processes with careful mapping.

While studying some results and methods from single-server finite queues, the question kept on arising: 'What *is* the actual distribution between this state and the next one?'. It became apparent that saying that a queue had a generally-distributed service time was attributing a high level description to the queue, much as a process algebra might do. Precisely as a result of complexities like memory, the actual inter-state distributions were distinct, complex and certainly not obvious from the stated general distribution (see discussion on queueing systems in section 3.3.2, especially).

From section 2.4.3, we know that Markovian queues or systems do not have to cope with this complication of distribution memory since they are *memoryless*. In saying that a queue is Markovian, the transitions of its stochastic transition system are immediately attributed exponential distributions. It turns out there are many such simplifications and finesses that can be performed by using Markovian processes, which, of course, is why they are so useful and so heavily studied. However they do have the drawback of hiding a lot of the underlying stochastic complexity; complexity which has a direct bearing on reliable modelling, when we analyse generally-distributed systems.

From observing that some kind of underlying state transition system was behind the operation of such generally-distributed systems, it is a natural step to want to describe and solve such a model. In this way the use of techniques such as embedded Markov chains might be avoided. This is desirable because, though a clever application, the embedded Markov chain in queues is very reliant on the simple Birth-Death structure of the queue. Such nice structural properties cannot be assumed for general process graphs, which are typically described by process algebras.

With this in mind, it seemed necessary to abstract concepts like action labels, competitive choice and parallel composition up to a process algebra level. It will be seen in later sections how such higher level functionality can be reintroduced.

3.2.3 Definition

We define a stochastic transition system to be a directed labelled graph. For the purposes of stationary distribution analysis let the digraph be bi-connected, by which we mean that any vertex (state) is reachable from any other vertex, via a directed path. This latter condition is the equivalent of the structural ergodic condition for PEPA [52] and ensures that a stationary distribution exists.

In operation, a process will start in a designated state and proceed to follow one and only one path round the graph. At no stage, therefore, does memory become an issue since the process is only executing one transition at a time, in sequence.

In algebraic terms we define a state, S to be:

$$S ::= \{X\}.S \;\Big|\; [p]S \oplus [1-p]S \;\Big|\; A \tag{3.1}$$

where X represents a continuous positive random variable and p, $0 < p < 1$ is a probability and A is a constant label.

The first operator is analogous to the prefix operator in CCS [78] and represents a sequential process, which pauses for a duration X before proceeding to the next transition.

The second operator, $\oplus$, represents branching. As can be seen from the definition, we have adopted probabilistic branching, mainly for reasons of simplicity given above. The selection of a path is instantaneous and samples from a Bernoulli trial (parameter, p) to decide which branch to take.

For the algebraic model to match the graph description, it remains to be demonstrated how n-way branching in the digraph is treated. For this, we simply need a structural associativity law to enable us to treat the n-way branch in a pairwise fashion.

Let:

$$S \stackrel{\text{def}}{=} [p]T \oplus [1-p]\{Z\}.S_3 \tag{3.2}$$

$$T \stackrel{\text{def}}{=} [q]\{X\}.S_1 \oplus [1-q]\{Y\}.S_2 \tag{3.3}$$

We require that:

$$S \stackrel{\text{def}}{=} [m]\{X\}.S_1 \oplus [1-m]U \tag{3.4}$$

$$U \stackrel{\text{def}}{=} [n_1]\{Y\}.S_2 \oplus [n_2]\{Z\}.S_3 \tag{3.5}$$

such that $n_1 + n_2 = 1$.

Now:

$$\mathbb{P}(S_1 \text{ is reached from } S) = pq \tag{3.6}$$

$$\mathbb{P}(S_2 \text{ is reached from } S) = p(1-q) \tag{3.7}$$

$$\mathbb{P}(S_3 \text{ is reached from } S) = 1-p \tag{3.8}$$

given that the branch probabilities are independent Bernoulli trials. Now we can see that, for the second model to be consistent with the first:

1. For state S_1:

$$m = pq \tag{3.9}$$

2. For states S_2, S_3:

$$n_1(1 - pq) = p(1 - q) \tag{3.10}$$

$$n_2(1 - pq) = 1 - p \tag{3.11}$$

Now:

$$n_1 + n_2 = \frac{p - pq}{1 - pq} + \frac{1 - p}{1 - pq} = 1 \tag{3.12}$$

as required.

From this result, a general n-way probabilistic branch, $\bigoplus_{i=1}^{n}[p_i]S_i$ where $\sum_{i=1}^{n} p_i = 1$, can be defined recursively, for $n \geq 2$:

$$\bigoplus_{i=1}^{n}[p_i]S_i \stackrel{\text{def}}{=} [p_n - 1]\left(\bigoplus_{i=1}^{n-1}\left[\frac{p_i}{p_n - 1}\right] S_i\right) \oplus [p_n]S_n \tag{3.13}$$

$$\bigoplus_{i=1}^{2}[p_i]S_i \stackrel{\text{def}}{=} [p_1]S_1 \oplus [p_2]S_2 \tag{3.14}$$

3.2.4 Stochastic Aggregation

Here, we describe a technique called stochastic aggregation, which forms the basis of our solution technique for stochastic transition systems. The rules presented come from Trivedi 1982 [97] where they described program execution; they have been rewritten below to describe a stochastic transition system.

The associativity of probabilistic branching (section 3.2.3) means that it is sufficient to consider two-way branching in equations (3.17) and (3.19).

Stochastic aggregation was born out of a need to try and reduce system complexity by combining (or aggregating) the edges and states of the transition graph. In doing so, it replaces a structure of the graph with a smaller equivalent structure, which takes the same amount of time to execute as the original.

Throughout, stochastic aggregation is defined in terms of Laplace transforms.

Convolution

Two sequential processes are aggregated into one:

$$A \stackrel{\text{def}}{=} \{X_1\}.\{X_2\}.B \quad \longrightarrow \quad A \stackrel{\text{def}}{=} \{Y\}.B \tag{3.15}$$

From the standard convolution result of Laplace transforms [98], we get:

$$L_Y(\omega) = L_{X_1+X_2}(\omega) = L_{X_1}(\omega)L_{X_2}(\omega) \tag{3.16}$$

Branching

A branch point where both branches eventually recombine, can be aggregated into a single process:

$$A \stackrel{\text{def}}{=} [p]\{X_1\}.B \oplus [1-p]\{X_2\}.B \quad \longrightarrow \quad A \stackrel{\text{def}}{=} \{Y\}.B \tag{3.17}$$

where:

$$L_Y(\omega) = pL_{X_1}(\omega) + (1-p)L_{X_2}(\omega) \tag{3.18}$$

Terminating Loop

A process that loops with probability $(1-p)$, or takes an alternative path with probability p, can be aggregated into a single process:

$$A \stackrel{\text{def}}{=} [p]\{X_1\}.B \oplus [1-p]\{X_2\}.A \quad \longrightarrow \quad A \stackrel{\text{def}}{=} \{Y\}.B \tag{3.19}$$

It can be derived from [97] that:

$$L_Y(\omega) = \frac{pL_{X_1}(\omega)}{1-(1-p)L_{X_2}(\omega)} \tag{3.20}$$

3.2.5 Equilibrium Distributions

Before we launch into our first instance of solving a system for its equilibrium distribution, we must first discuss some properties of equilibrium distributions themselves. These are taken from Cox and Miller 1965 [26].

Given a state transition system:

1. For all initial conditions the probability $p_i(t)$ that state i is occupied at time t tends to a distribution $\{\pi_j : 1 \leq j \leq n\}$ as $t \longrightarrow \infty$.
2. If the initial state has the probability distribution $\{\pi_j : 1 \leq j \leq n\}$ then $p_i(t) = \pi_i$ for all t.
3. After a long time t_0, the proportion of time spent in any state i converges to π_i as $t_0 \longrightarrow \infty$.
4. The equilibrium probabilities, π_i, if they exist, satisfy a system of ordinary simultaneous linear equations.

These properties lead to the following empirical interpretations of stationary distributions:

1. Given a large number of independent realisations (traces) of a system: at a large time t_0, the proportion of systems in state i tends to π_i.
2. Given a single long realisation, the proportion of time spent in state i tends to π_i.

It is the second of these properties which we use to define our stationary distribution in the Cox-Miller Normal Form example presented in the next section.

3.2.6 Cox-Miller Normal Form

Cox-Miller Normal Form is the simplest generally-distributed cyclic process that can be solved directly to give an equilibrium distribution. It was described specifically in Cox and Miller [26, pp.172–173] for the Markovian case and is extended to general distributions here.

The system consists of two states, 0 and 1, and has two distinct generally-distributed transitions S and T, as shown in figure 3.2. Two sequences of independent identically distributed random variables are constructed X_1, $X_2, \dots$ and $Y_1, Y_2, \dots$, where $F_{X_i}(t) = F_S(t)$ and $F_{Y_i}(t) = F_T(t)$ for all i. X_i represents the ith transition from state 0 to state 1 and Y_i represents the returning ith transition back from state 1 to state 0. Thus the sequence of transitions in any given realisation (or trace) of the system would be: $X_1, Y_1, X_2, Y_2, \dots$ and so on.

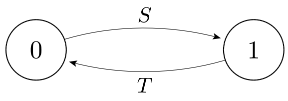

Fig. 3.2. The Cox-Miller Normal Form.

Suppose for simplicity that the system starts in state 0 and undergoes $2n$ transitions. The total amount of time it will have spent in state 0 is $X_1 + \cdots + X_n$ and similarly the total amount of time it will have spent in state 1 is $Y_1 + \cdots + Y_n$. We now construct two random variables $\Pi_i^n, i = 0, 1$, representing the proportion of time spent in states 0 and 1, after $2n$ transitions.

$$
\begin{aligned}
\Pi_0^n &= \frac{X_1 + \cdots + X_n}{X_1 + Y_1 + X_2 + Y_2 + \cdots + X_n + Y_n} \\
&= \frac{\frac{1}{n}\sum_{i=1}^n X_i}{\frac{1}{n}\sum_{i=1}^n X_i + \frac{1}{n}\sum_{i=1}^n Y_i} \\
&= \frac{\overline{X}}{\overline{X} + \overline{Y}}
\end{aligned} \tag{3.21}
$$

Similarly:

$$
\Pi_1^n = \frac{\overline{Y}}{\overline{X} + \overline{Y}} \tag{3.22}
$$

where $\overline{X}$ and $\overline{Y}$ are sample means of the n independent random variables.

We now utilise the Strong Law of Large Numbers [30, p.243] to relate the sample means of X_i and Y_i to the distribution means of S and T.

Thm. 1 (The Strong Law of Large Numbers). *Let X_k be a sequence of mutually independent random variables with a common distribution. If the expectation $\mu = \mathbb{E}(X_k)$ exists, then for every $\varepsilon > 0$ as $n \longrightarrow \infty$:*

$$
\mathbb{P}\left(\left|\frac{X_1 + \cdots + X_n}{n} - \mu\right| < \varepsilon\right) \longrightarrow 1 \tag{3.23}
$$

Now applying this to our sample means $\overline{X}$ and $\overline{Y}$, for all $\varepsilon > 0$:

$$
\begin{aligned}
&\lim_{n \longrightarrow \infty} \mathbb{P}(|\overline{X} - \mathbb{E}(S)| < \varepsilon) = 1 \\
&\Rightarrow \mathbb{P}\left(\lim_{n \longrightarrow \infty} \overline{X} = \mathbb{E}(S)\right) = 1
\end{aligned} \tag{3.24}
$$

Similarly:

$$
\mathbb{P}\left(\lim_{n \longrightarrow \infty} \overline{Y} = \mathbb{E}(T)\right) = 1 \tag{3.25}
$$

So when the limit is applied to Π_i^n, we get:

$$
\mathbb{P}\left(\lim_{n \longrightarrow \infty} \Pi_0^n = \frac{\mathbb{E}(S)}{\mathbb{E}(S) + \mathbb{E}(T)}\right) = 1 \tag{3.26}
$$

$$\mathbb{P}\left(\lim_{n\longrightarrow\infty} \Pi_1^n = \frac{\mathbb{E}(T)}{\mathbb{E}(S)+\mathbb{E}(T)}\right) = 1 \tag{3.27}$$

From our definition of an equilibrium distribution:

$$\lim_{n\longrightarrow\infty} \Pi_i^n = \pi_i \tag{3.28}$$

So we can now obtain π_0 and π_1:

$$\pi_0 = \frac{\mathbb{E}(S)}{\mathbb{E}(S)+\mathbb{E}(T)} \tag{3.29}$$

$$\pi_1 = \frac{\mathbb{E}(T)}{\mathbb{E}(S)+\mathbb{E}(T)} \tag{3.30}$$

With hindsight, we can see that since the steady-state distribution of Cox-Miller Normal Form is a function solely of the means of the constituent distributions, then it is insensitive to the type of the general distributions, S and T. This means that we could also have obtained the solution by substituting the general distributions with exponential distributions of the same mean and solving the resulting Markovian process to get the same answer as presented in equations (3.29) and (3.30).

While the steady-state distribution is certainly a useful comparative metric, this insensitivity is a good demonstration of quite how much distribution information is lost in the final result. The steady-state is very much a measure of average behaviour and this bears out our summary of it, given in section 2.6.

3.2.7 Stochastic Normal Forms

Motivation

In this section, we show that general systems can be reduced to a normal form, using stochastic aggregation. The simplest of these normal forms is the Cox-Miller Normal Form, which we solved in the previous section. In the next section we will show how the other normal form cases can themselves be solved using the Cox-Miller result.

To understand the process of aggregation to the normal form, we should explain the stochastic motivation. This in turn will explain the structure of the normal forms.

Given any chosen state in a system of which we wish to find the steady-state probability, we have to maintain the stochastic characteristics of that state within the system. In a stochastic transition system, the system remains in a given state for a random amount of time, i.e. the *sojourn time* from the state. The exact distribution of this amount of time is defined by the distributions on the leaving transitions. Therefore, a state in such a system is defined by its out-edges and preserving the stochastic characteristic of the state means preserving its out-edges.

Under this restriction we would like to be able to simplify the remainder of the system as much as possible. The rest of this section is devoted to showing that for each out-edge from a state, the rest of the system can be stochastically aggregated to a single state with a single return edge. So we create a class of systems with the same number of two-state cycles as there are out-edges from the state that we were originally carefully "aggregating around", shown in figure 3.3.

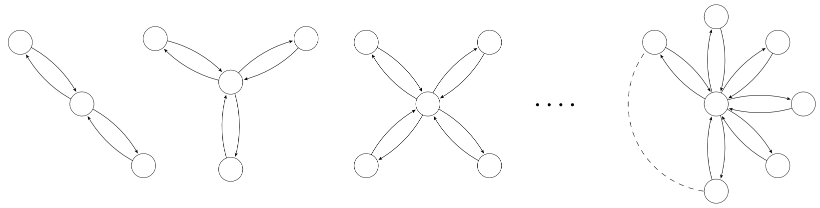

Fig. 3.3. The set of normal forms for a state with finite branching degree.

In doing this, we have fulfilled our wish to preserve the stochastic nature of the state, since each out-edge still exists explicitly in the reduced system. We have also preserved the context of the state within the original system, since the "remainder of the system" behaviour is modelled in aggregated form by each of the return edges in the cycles. These edges effectively represent the aggregated system behaviour given that a particular out-edge was traversed.

Reduction Rules

Now we have a good idea of exactly what we want to reduce our system to, and why, we can abstract the problem to a completely graph-based domain for proof.

Five reduction rules are constructed and shown in figure 3.4.

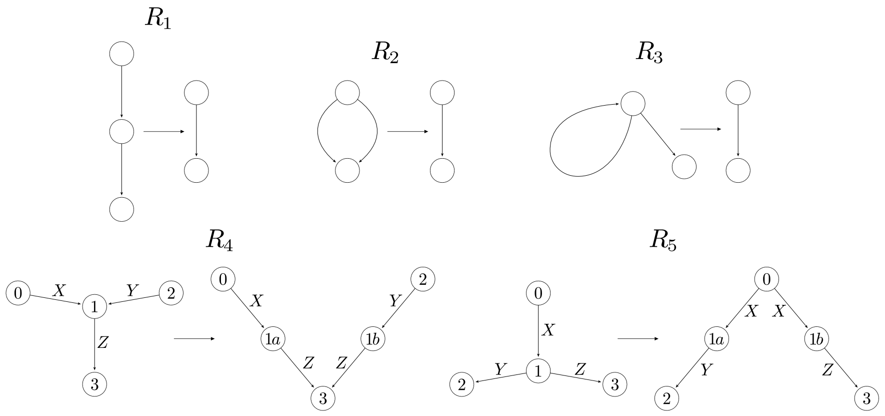

Fig. 3.4. The digraph reduction rules.

R_1 represents the convolution rule of equation (3.15). R_2 represents the branching rule of equation (3.17). R_3 represents the terminating loop reduction of equation (3.19). R_4 and R_5 represent some simple laws of graph sharing which are true for a stochastic transition system.

We demonstrate R_4 and R_5 in stochastic terms in tables 3.1 and 3.2; they are basically a reduction of graph sharing. This result extends easily to having many in-edges and out-edges into state 1, by appealing to the associativity of probabilistic branching.

Tab. 3.1. The stochastic equivalence of the two paths under reduction rule R_4.

R_4	**Before Reduction**		**After Reduction**	
$0 \to 3$	$0 \to 1 \to 3$	$L_X(z)L_Z(z)$	$0 \to 1a \to 3$	$L_X(z)L_Z(z)$
$2 \to 3$	$2 \to 1 \to 3$	$L_Y(z)L_Z(z)$	$2 \to 1b \to 3$	$L_X(z)L_Z(z)$

Tab. 3.2. The stochastic equivalence of the two paths under reduction rule R_5.

R_5	**Before Reduction**		**After Reduction**	
$0 \to 2$	$0 \to 1 \to 2$	$L_X(z)pL_Y(z)$	$0 \to 1a \to 2$	$pL_X(z)L_Z(z)$
$0 \to 3$	$0 \to 1 \to 3$	$L_Y(z)qL_Z(z)$	$0 \to 1b \to 3$	$qL_X(z)L_Z(z)$

Proof of Digraph Reduction to Normal Forms

This proof is taken from [99], where issues of convergence of the reduction method are also discussed. It should be stressed that this is only a proof mechanism, we do not envisage this as an algorithm for solution.

In the argument below, a *circuit* is defined to be a cyclic path of a digraph in which no node is ever traversed twice through the same pair of in- and out-edges. There are a finite number of such circuits in a strongly connected digraph. We also define the *parents* of a vertex, v, to be the set of adjacent vertices which are connected to v via their out-edges. The *children* of v are the set of vertices which have v as a parent.

Let G be a strongly connected or biconnected digraph:

- It is possible to get to any vertex from any other vertex.
- For a chosen vertex v in G, v is an element of each member of the set C of circuits which contain v and not an element of any member of the set C' of circuits which do not contain v.

- the circuits of $G = C \cup C'$, i.e. the graph is partitioned.
- We require to remove C' by aggregation and reduce C to one of the normal forms of figure 3.3, where the out-degree (number of out-edges) of v is preserved.

It can be seen that R_2 and R_3 reduce the number of circuits in a digraph, whereas R_1, R_4 and R_5 preserve the number of circuits. By applying the reduction rules to a general digraph, we can only reduce or maintain the number of circuits.

We describe the following digraph reduction method, to remove C':

1. Select a chosen vertex v.
2. Remove all immediately reducible cycles with R_2, R_3.
3. Select the circuit $c \in C'$ of shortest length (so c does not contain v).
4. Select a vertex $x \in c$ and label as the root of the circuit. Let $u := x$:
   (a) Go to u', the child of u, in circuit c; if u' has more than one parent then split using R_4 into a vertex with u as its only parent and a vertex with the remaining parents of u'.
   (b) Let $u :=$ child of vertex u, in c, whose parents we may just have increased. Repeat (4a) until all vertices in c, except x, have been processed (as in figure 3.5).
5. Essentially we now perform (4) in reverse using R_5 (figure 3.6). As before, let $u := x$:
   (a) Go to u', the parent of u, in circuit c; if u' has more than one child then split using R_5 into a vertex with u as its only child and a vertex with the remaining children of u'.
   (b) Let $u :=$ parent of vertex u, in c, whose children we may just have increased. Repeat (5a) until all vertices in c, except x, have been processed.

6. Apply R_1 to all the vertices of circuit c except x, until x is the only vertex left in c, with a single edge to itself.

7. Apply R_3 to remove c from C'.

8. Repeat steps (3)–(7), until C' is empty.

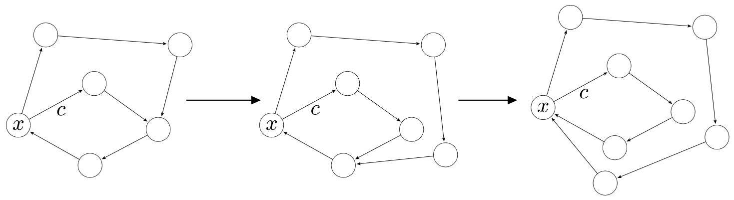

Fig. 3.5. Rewriting the circuit c to have no vertices with multiple parents except x.

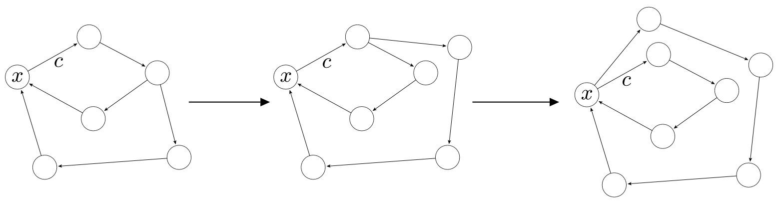

Fig. 3.6. Rewriting the circuit c to have no vertices with multiple children except x.

Now we need to restructure G (which now consists entirely of C, i.e. circuits containing v), to look like one of the normal forms of figure 3.3. Our goal is to maintain the out-edges from v unaggregated.

1. Select the shortest circuit of length > 2 through v and reapply step (4)–(6) with $x := v$.

2. Apply R_1 to all vertices of this circuit except v and child of v.

3. Apply R_2 to remaining in-edges to v.

4. Repeat for each circuit through v until G is in normal form.

Solution of Normal Forms

In section 3.2.6, it was shown how to solve the simplest normal form, the Cox-Miller Normal Form. This normal form only occurs when aggregating around a state with a single out-transition.

We use this fact to solve the remaining normal forms. The normal forms of figure 3.3 each consist of a central vertex of out-degree $n \geq 2$. All the adjacent vertices to the central state, i.e. the next ones out along the out-edges, have out-degree one.

This is the key. Using the stochastic aggregation technique of section 3.2.4, specifically the results from equations (3.19) and (3.15), we can take each of these out-degree one vertices in turn and aggregate to obtain a Cox-Miller Normal Form. It is a simple matter now to find the stationary probabilities for each of the degree one vertices and thus obtain the central vertex stationary probability by summing and subtracting from 1.

More specifically, label the central vertex with out-degree $n \geq 2$ as v. Label the n adjacent vertices to v, v_i for $1 \leq i \leq n$. For each v_i, aggregate to obtain a Cox-Miller Normal Form and solve for the stationary probability π_{v_i}. Now $\pi_v = 1 - \sum_{i=1}^{n} \pi_{v_i}$.

3.2.8 Worked Example

We now demonstrate the stochastic aggregation technique on the example state transition system shown in figure 3.7.

$$A \sim \Gamma(\alpha, \lambda) \tag{3.31}$$

$$B \sim \text{Hyper}\left(\frac{2}{3}, \frac{1}{3}; \lambda, \mu\right) \tag{3.32}$$

$$C \sim \text{U}\left[\frac{1}{\lambda}, \frac{1}{\mu}\right] \tag{3.33}$$

$$D \sim \delta\left(\frac{1}{\mu}\right) \tag{3.34}$$

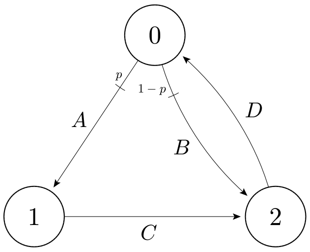

Fig. 3.7. A generally-distributed stochastic transition system.

where $0 < \mu < \lambda$ and $\alpha > 0$. A has a Gamma distribution; B has an hyperexponential distribution; C has a uniform distribution; and D is a deterministic process which occurs at $t = 1/\mu$ with probability 1. The random variables have the following transforms:

$$L_A(z) = \left(\frac{\lambda}{\lambda + z}\right)^{\alpha} \tag{3.35}$$

$$L_B(z) = \frac{2}{3}\frac{\lambda}{\lambda + z} + \frac{1}{3}\frac{\mu}{\mu + z} \tag{3.36}$$

$$L_C(z) = \frac{\lambda\mu}{(\lambda - \mu)z}\left[e^{-z/\lambda} - e^{-z/\mu}\right] \tag{3.37}$$

$$L_D(z) = e^{-z/\mu} \tag{3.38}$$

First we select state 1, then aggregate around the state, to create the combined process ψ_1, as displayed in figure 3.8. Using the stochastic aggregation rules (3.15,3.19):

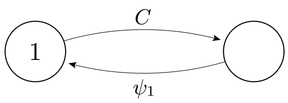

Fig. 3.8. Aggregating states 0 and 2 to give a Cox-Miller Normal Form for state 1.

$$L_{\psi_1}(z) = \frac{pL_A(z)L_D(z)}{1 - (1-p)L_B(z)L_D(z)} \tag{3.39}$$

The aggregated system for state 1 is now in Cox-Miller Normal Form. We now have to extract the expectations from the transforms in order to obtain the stationary probability for state 1:

$$\mathbb{E}(C) = -\frac{\mathrm{d}}{\mathrm{d}z} L_C(z)\bigg|_{z=0} = \frac{1}{2}\left(\frac{1}{\mu} + \frac{1}{\lambda}\right) \tag{3.40}$$

$$\mathbb{E}(\psi_1) = -\frac{\mathrm{d}}{\mathrm{d}z} L_{\psi_1}(z)\bigg|_{z=0} = \frac{((3\alpha - 2)\mu - \lambda)p + 2(2\lambda + \mu)}{3\lambda\mu p} \tag{3.41}$$

Now using equation (3.29), the stationary distribution of state 1 can be calculated:

$$\pi_1 = \frac{\mathbb{E}(C)}{\mathbb{E}(C) + \mathbb{E}(\psi_1)} = \frac{3(\lambda + \mu)p}{(\lambda + 6\alpha\mu - \mu)p + 8\lambda + 4\mu} \tag{3.42}$$

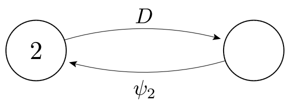

Fig. 3.9. Aggregating states 0 and 1 to give a Cox-Miller Normal Form for state 2.

Undertaking a similar aggregation for state 2, shown in figure 3.9, we obtain:

$$L_{\psi_2}(z) = pL_A(z)L_C(z) + (1 - p)L_B(z) \tag{3.43}$$

Thus:

$$\pi_2 = \frac{\mathbb{E}(D)}{\mathbb{E}(D) + \mathbb{E}(\psi_2)} = \frac{6\lambda}{(\lambda + 6\alpha\mu - \mu)p + 8\lambda + 4\mu} \tag{3.44}$$

At this stage, we can deduce that:

$$\pi_0 = 1 - (\pi_1 + \pi_2) = 2\frac{(3\alpha\mu - 2\mu - \lambda)p + \lambda + 2\mu}{(\lambda + 6\alpha\mu - \mu)p + 8\lambda + 4\mu} \tag{3.45}$$

However the full calculation of π_0 will be shown in order to demonstrate how states which do not aggregate directly to Cox-Miller Normal Forms can be solved. In this case, it is only possible to aggregate to a two-cycle normal form (figure 3.10), since state 0 has two out-edges that we wish to preserve. ψ_3 is given by:

$$L_{\psi_3}(z) = L_C(z)L_D(z) \tag{3.46}$$

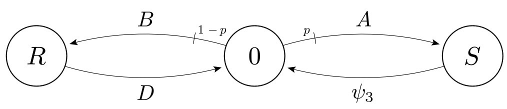

Fig. 3.10. Aggregating around state 0 to get a more complicated normal form.

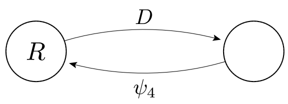

Fig. 3.11. Aggregating around state R, achieving a CMNF structure.

Now we treat the normal form of figure 3.10 just as any other system and aggregate to obtain Cox-Miller Normal Forms for the out-degree one states R and S. The R aggregate system in shown in figure 3.11, where:

$$L_{\psi_4}(z) = \frac{(1-p)L_B(z)}{1 - pL_A(z)L_{\psi_3}(z)} \tag{3.47}$$

Now:

$$\pi_R = \frac{\mathbb{E}(D)}{\mathbb{E}(D) + \mathbb{E}(\psi_4)} = \frac{6\lambda(1-p)}{(\lambda + 6\alpha\mu - \mu)p + 8\lambda + 4\mu} \tag{3.48}$$

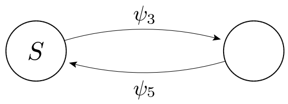

Fig. 3.12. Aggregating around state S, achieving a CMNF structure.

Similarly for state S in figure 3.12 where:

$$L_{\psi_5}(z) = \frac{pL_A(z)}{1 - (1-p)L_B(z)L_D(z)} \tag{3.49}$$

Thus:

$$\pi_S = \frac{\mathbb{E}(\psi_3)}{\mathbb{E}(\psi_3) + \mathbb{E}(\psi_5)} = \frac{3(3\lambda + \mu)p}{(\lambda + 6\alpha\mu - \mu)p + 8\lambda + 4\mu} \tag{3.50}$$

and:

$$\pi_0 = 1 - (\pi_R + \pi_S) = 2\frac{(3\alpha\mu - 2\mu - \lambda)p + \lambda + 2\mu}{(\lambda + 6\alpha\mu - \mu)p + 8\lambda + 4\mu} \tag{3.51}$$

which agrees with our earlier answer of equation (3.45).

3.3 Applications

3.3.1 Generally-Distributed Stochastic Process Algebras

By finding a mapping between a generally-distributed stochastic process algebra and a stochastic transition system, we would immediately achieve a solution technique for a generally-distributed process algebra, such as:

$$S ::= \ (x, F_X(t)).S \ \Big| \ S + S \ \Big| \ S||S \ \Big| \ A \tag{3.52}$$

The problem of solving a generally-distributed process algebra with the full complement of major operators is a difficult one and should not be in any way underestimated.

Given that the problem of solving a simple stochastic transition system with general distributions was reasonably straightforward, the difficulty in the process algebras problem must lie in the task of mapping such a process algebra on to its equivalent stochastic state space.

This does indeed turn out to be the case, and the mapping problem is demonstrated in the next section with a generally-distributed queue example.

As we might have expected, this problem is a result of our side-stepping the memory problem that is encountered when modelling concurrent general distributions. This means that we will be able to model a subset of a process algebra which does not contain the concurrent composition operator:

$$S ::= \ (x, F_X(t)).S \ \Big| \ S + S \ \Big| \ A \tag{3.53}$$

As we will see, this does not mean that concurrent processes cannot be modelled at all. In fact it is the concurrency strategy which determines whether we can express a system in a finite stochastic transition system. That is, whether a pre-empted component restarts its sojourn or continues unaffected. These two strategies are called pre-emptive restart with resampling and pre-emptive resume, and we will see the different effects of the two cases in action in chapter 4.

In the event that the generally distributed algebra is using pre-emptive restart, we have no difficulty providing a mapping to stochastic transition systems (section 3.A.3). It is the resume case which causes difficulty. We have no problem expressing the synchronisation action itself (see chapter 4 for a discussion of different synchronisation models), the difficulty lies in expressing the interleaving of unsynchronised actions. This is shown quite clearly in section 3.3.2, where we examine a generally-distributed queueing system (queues use a pre-emptive resume mechanism). It should be noted that for purely Markovian systems there is no stochastic difference between a resume and a restart strategy [55, p.24].

Constructing a Stochastic Transition System

The rules for mapping a restricted generally-distributed process algebra (equation (3.53)) to its stochastic transition system are as follows:

$$(x, F_X(t)).P \longrightarrow \{X\}.P \tag{3.54}$$

$$\begin{aligned}(x, F_X(t)).P + (y, F_Y(t)).Q \longrightarrow\ & [\mathbb{P}(X < Y)]\{X | X < Y\}.P \\ & \oplus [\mathbb{P}(Y < X)]\{Y | Y < X\}.Q\end{aligned} \tag{3.55}$$

Equation (3.54) is fairly self-explanatory. In the process algebra version, it describes a transition labelled x which proceeds with random variable X

(cumulative distribution function, $F_X(t)$) to a new state, P. In the stochastic transition system, the label is discarded and the process is represented by the evolution of the random variable X.

Equation (3.55) describes the mapping of competitive choice on to probabilistic branching and does deserve some discussion.

Competitive Choice

Process algebras have long used a mechanism called competitive choice for describing branching or choice points in a transition system. When a choice is to be made, two processes enter what is known as a *race condition*; whichever of the two finishes first is the successful branch. The losing or slower process is interrupted and discarded. Given that this is a popular representation of process choice it is as well to consider it, if only for reasons of compatibility. In the event that a process algebra were to use probabilistic choice then this would obviously have a much more straightforward conversion to the probabilistic branching model of the stochastic transition system.

Although this stochastic mapping has been used before in TIPP [37], it is worth proving a couple of properties to justify its use. The first is an observational property: that is, that the total time that a process spends in a competitive choice should be distributed as the minimum of the two competing processes. The second is that of associativity and its implications for the general n-way choice.

Calculating the minimum distribution of two independent random variables, we use the fact that: $\min(X, Y) > t \Leftrightarrow X > t$ and $Y > t$:

$$\begin{aligned}
F_{\min(X,Y)}(t) &= \mathbb{P}(\min(X, Y) \leq t) \\
&= 1 - \mathbb{P}(\min(X, Y) > t) \\
&= 1 - \mathbb{P}(X > t, Y > t) \\
F_{\min(X,Y)}(t) &= 1 - \overline{F}_X(t)\overline{F}_Y(t) \qquad (3.56)
\end{aligned}$$

where:

$$\overline{F}_X(t) = \int_t^\infty f_X(x)\,\mathrm{d}x = 1 - F_X(t) \tag{3.57}$$

Let us say that competitive choice between two random variables is represented by the notation:

$$C = X \odot Y \tag{3.58}$$

such that:

$$\begin{aligned} F_C(t) &= \mathbb{P}(X < Y)F_{X|X<Y}(t) + \mathbb{P}(Y < X)F_{Y|Y<X}(t) \\ &= \mathbb{P}(X \le t, X < Y) + \mathbb{P}(Y \le t, Y < X) \\ &= \int_0^t \overline{F}_Y(u)f_X(u) + \overline{F}_X(u)f_Y(u)\,\mathrm{d}u \\ &= \int_0^t -\frac{\mathrm{d}}{\mathrm{d}u}\left[\overline{F}_X(u)\overline{F}_Y(u)\right]\,\mathrm{d}u \end{aligned} \tag{3.59}$$

Since $\overline{F}_X(0)\overline{F}_Y(0) = 1$:

$$F_C(t) = 1 - \overline{F}_X(t)\overline{F}_Y(t) \tag{3.60}$$

From this we get the associativity result:

$$F_{X\odot(Y\odot Z)}(t) = 1 - \overline{F}_X(t)\overline{F}_Y(t)\overline{F}_Z(t) = F_{(X\odot Y)\odot Z}(t) \tag{3.61}$$

From this and [37], we state (it is easily shown) the general form of competitive choice. For $C = X_1 \odot \cdots \odot X_n$:

$$\begin{aligned} F_C(t) &= \mathbb{P}(X_1 < X_2, \ldots, X_1 < X_n)F_{X|X_1<X_2,\ldots,X_1<X_n}(t) + \cdots \\ &\quad + \mathbb{P}(X_n < X_1, \ldots, X_n < X_{n-1})F_{X|X_n<X_1,\ldots,X_n<X_{n-1}}(t) \\ &= 1 - \Pi_{i=1}^n \overline{F}_{X_i}(t) \end{aligned} \tag{3.62}$$

Thus, competitive choice between n processes can be mapped pairwise on to the equivalent probabilistic branching in the stochastic transition system.

3.3.2 Queueing Systems

M/G/1/K and G/G/1/K Queues

A first attempt at using the stochastic aggregation technique was in applying it to queueing systems. Queueing theory is one of the few theories of stochastic processes where generally-distributed transitions occur and can be solved using analytical methods. As such they were a useful and ultimately successful testing ground for the technique.

In this section we demonstrate a mapping on to a stochastic transition system for a G/G/1/2 queue. We provide a general analytic solution and then specialise it to a specific M/G/1/2 case. It is shown that the solution from the embedded Markov chain technique is identical to that produced by stochastic aggregation.

We further show that this technique cannot be applied to larger G/G/1/K queues, since the equivalent stochastic state space is infinite in size.

An Analytic Solution of a G/G/1/2 Queue

In this section, we describe a three-state single server queue with independent arrival process X and service process Y (figure 3.13). Converting this high level description to the stochastic transition system of figure 3.14 requires a little explanation.

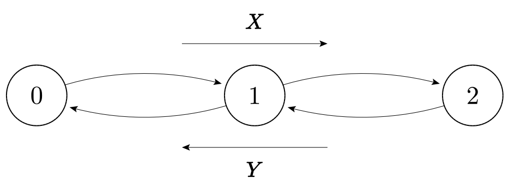

Fig. 3.13. The state diagram for a buffer of a G/G/1/2 queue.

Initially, it is possible to start in any state in the system, however these start states (indicated by concentric circles in figure 3.14) are often different in

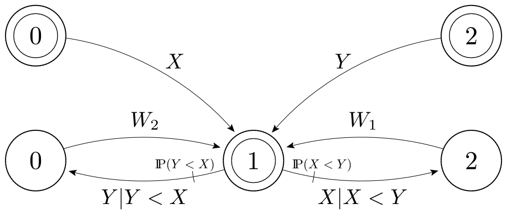

Fig. 3.14. The stochastic transition system for a G/G/1/2 queue.

stochastic terms to the same states during system operation. For instance, if the queue starts empty then the next transition is going to be an arrival sampled from X, however if the queue becomes empty during operation, the next arrival will be a different sample. On this occasion, the sample represents the remaining time left given that the X arrival process started while the system was in state 1 but was pre-empted by a service of the queue. This distribution W_2 and its counterpart W_1 will be defined precisely later.

The start states of 0 and 2 play no further part in any realisation (trace) since the system immediately moves to the recurrent states[1] on which the stationary distribution is defined.

With this particular queueing model we are able to do something we cannot do in general; that is, model concurrent processes. From state 1, X and Y always start afresh and spawn two processes each, according to whether $X < Y$ or $Y < X$. This is explained by a partitioning argument to construct the conditional probabilities and maps easily on to the probabilistic branching paradigm:

$$A_X = (A_X \cap B) \cup (A_X \cap \overline{B}) \tag{3.63}$$

where $A_X = \{\omega : X(\omega) \leq t\}$, $B = \{\omega : X(\omega) < Y(\omega)\}$ where ω is a sample point of the sample space. Applying the probability measure to the set of events, A_X, we get:

$$\begin{aligned} F_X(t) &= \mathbb{P}(X \leq t, X < Y) + \mathbb{P}(X \leq t, Y < X) \\ &= \mathbb{P}(X < Y)\mathbb{P}(X \leq t | X < Y) + \mathbb{P}(Y < X)\mathbb{P}(X \leq t | Y < X) \\ &= \mathbb{P}(X < Y)F_{X|X<Y}(t) + \mathbb{P}(Y < X)F_{X|Y<X}(t) \end{aligned} \tag{3.64}$$

[1]In the language of Markov chains; in our system, this is the biconnected subgraph.

Similarly:

$$F_Y(t) = \mathbb{P}(X < Y)F_{Y|X<Y}(t) + \mathbb{P}(Y < X)F_{Y|Y<X}(t) \tag{3.65}$$

So in strict order we either get samples from $X|X < Y$ and then $Y|X < Y$ or $Y|Y < X$ and then $X|Y < X$. This is not quite the end of the story since we also have the memory issue to contend with now. In figure 3.14, in going from state 1 through 0 to 1 again, the whole event from leaving state 1 to returning to state 1 represents the larger sample from $X|Y < X$, and the transition from 1 to 0 represents the sample from $Y|Y < X$ which happened first. So the second transition in that path, labelled W_2, represents a type of "difference" between the two distributions; the nature of this difference operation is described below.

We define the operator $\ominus$ on a pair of independent positive random variables as follows:

$$C = A \ominus B \tag{3.66}$$

such that, in terms of Laplace transforms:

$$L_C(z) = \frac{L_A(z)}{L_B(z)} \tag{3.67}$$

Thus C is defined as the positive random variable which, when convolved with B, will give A.

This now allows us to express W_1 and W_2 in terms of X and Y:

$$W_1 = (Y|X < Y) \ominus (X|X < Y) \tag{3.68}$$

$$W_2 = (X|Y < X) \ominus (Y|Y < X) \tag{3.69}$$

Now, using the reduction equation (3.19) to preserve the out-transition from state 0, we can generate the ψ_1 distribution of figure 3.15.

$$\begin{aligned} L_{\psi_1}(z) &= \frac{\mathbb{P}(Y < X)L_{Y|Y<X}(z)}{1 - \mathbb{P}(X < Y)L_{X|X<Y}(z)L_{W_1}(z)} \\ &= \frac{\mathbb{P}(Y < X)L_{Y|Y<X}(z)}{1 - \mathbb{P}(X < Y)L_{Y|X<Y}(z)} \end{aligned} \tag{3.70}$$

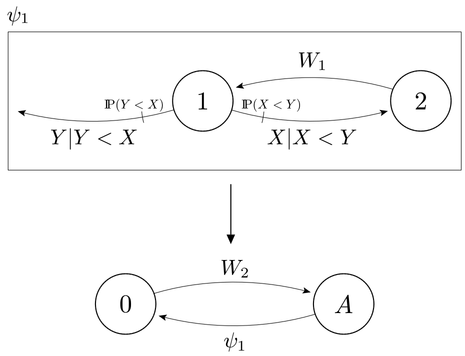

Fig. 3.15. Aggregating the states 1 and 2 to produce a new state A and an aggregated transition ψ_1.

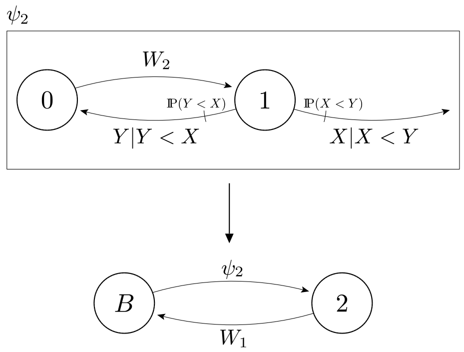

Fig. 3.16. Aggregating the states 0 and 1 to produce a new state B and an aggregated transition ψ_2.

The expectation of a distribution is easily obtained from its Laplace transform:

$$\mathbb{E}(A) = -\frac{\mathrm{d}}{\mathrm{d}z} L_A(z)\bigg|_{z=0} \tag{3.71}$$

This is all that is required to find out the stationary distribution of state 0, since the aggregated system of figure 3.15 is just the Cox-Miller Normal Form of section 3.2.6. Thus from equation (3.29), we obtain:

$$\pi_0 = \frac{\mathbb{E}(W_2)}{\mathbb{E}(W_2) + \mathbb{E}(\psi_1)} \tag{3.72}$$

Similarly, reducing the system to preserve state 2, we get the aggregated transition, ψ_2 in figure 3.16:

$$L_{\psi_2}(z) = \frac{\mathbb{P}(X < Y)L_{X|X<Y}(z)}{1 - \mathbb{P}(Y < X)L_{X|Y<X}(z)} \tag{3.73}$$

Now having obtained a second Cox-Miller Normal Form, π_2 follows:

$$\pi_2 = \frac{\mathbb{E}(W_1)}{\mathbb{E}(W_1) + \mathbb{E}(\psi_2)} \tag{3.74}$$

Finally:

$$\pi_1 = 1 - (\pi_0 + \pi_2) \tag{3.75}$$

M/G/1/2 Example

Specialising the G/G/1/2 result to use a Markovian arrival process simplifies some of the expressions. Importantly the end result can be checked using standard M/G/1/K queueing techniques.

Let the arrival process X and service process Y be defined as follows:

$$\begin{aligned} f_X(t) &= \lambda e^{-\lambda t} && (3.76) \\ f_Y(t) &= \alpha e^{-\alpha t} + \beta e^{-\beta t} - (\alpha + \beta)e^{-(\alpha+\beta)t} && (3.77) \end{aligned}$$

Now, we can specify the exact transition distributions of figure 3.14 and thus, from equations (3.72, 3.75, 3.74), the stationary distribution of the queue:

$$\begin{aligned} \pi_0 &= \alpha^2\beta^2(\alpha + \beta)(\alpha + \beta + 2\lambda)\chi && (3.78) \\ \pi_1 &= \alpha\beta\lambda(\alpha + \beta)((\alpha + \beta + \lambda)^2 - \alpha\beta)\chi && (3.79) \\ \pi_2 &= ((\alpha^2 + \alpha\beta + \beta^2)\lambda^2 + (\alpha + \beta)(2\alpha^2 + \alpha\beta + 2\beta^2)\lambda \\ &\quad + \alpha^4 + 2\alpha^3\beta + \alpha^2\beta^2 + 2\alpha\beta^3 + \beta^4)\chi && (3.80) \end{aligned}$$

where:

$$\begin{aligned}
\chi \;=\; & \Big((\alpha^2+\alpha\beta+\beta^2)\lambda^4 + 2(\alpha+\beta)(\alpha^2+\alpha\beta+\beta^2)\lambda^3 \\
& + (\alpha^2+\alpha\beta+\beta^2)(\alpha^2+3\alpha\beta+\beta^2)\lambda^2 \\
& + \alpha\beta(\alpha+\beta)(\alpha^2+3\alpha\beta+\beta^2)\lambda + \alpha^2\beta^2(\alpha+\beta)^2 \Big)^{-1} \qquad (3.81)
\end{aligned}$$

These results are identical to those obtained using embedded Markov chain techniques from traditional queueing theory [23, 66, 38].

Larger G/G/1/K Queues

So, can the same techniques be applied to larger general queueing systems? The answer, probably unsurprisingly, is no; not directly at least. To see why, we need to look at how the processes can interleave. Figure 3.17 shows an interleaving for a G/G/1/3 queue. The synchronising states (states from which both arrival and service distributions start simultaneously) are marked by concentric circles.

This is not the stochastic state space of other diagrams, it represents instead how the processes interleave by showing how many services (or arrivals) have occurred while an arrival (or service) has still to occur. It is a distinct diagram from that of the normal queue buffer representation. For example, observe that there are distinct instances of the state with one item in the queue: one instance when both the arrival and service processes start simultaneously and one when the service process Y is starting again (having just occurred) while the arrival process X has been running for a single service time. Whether the arrival process happens next depends now on a race between a fresh service process and a one-service-time-old arrival process.

However, as already stated, this is not a stochastic transition system either. The reason is that although this represents a consistent interleaving, the distributions of the individual edges are not necessarily the same each time that edge is traversed. Specifically, for the cycle between $2a$ and $1b$, each time

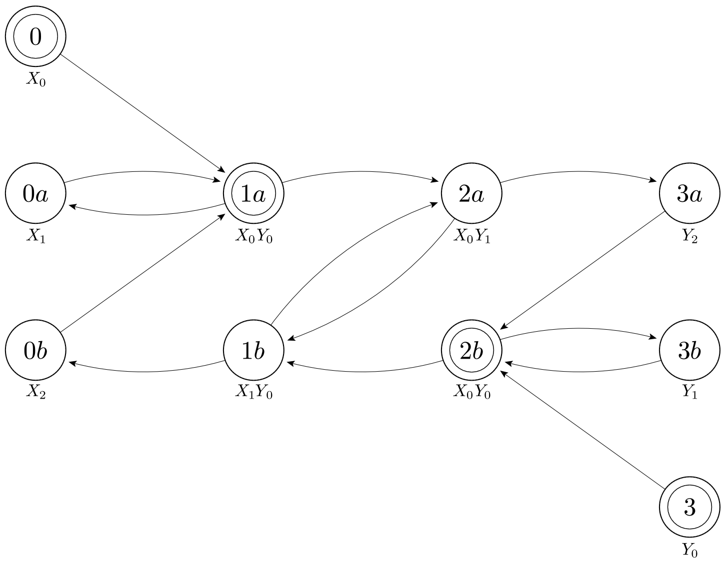

Fig. 3.17. The process interleaving for a G/G/1/3 queue, taking into account how many evolutions of the opposing process have occurred.

that cycle is traversed without first going through a synchronising state, a distinct distribution is sampled. This is where the memory of the system defeats us. Until a synchronising state is passed through[2], the distribution of subsequent transitions reflects the entire trace history of the system. So having a cycle between two non-synchronising states, in effect, generates an infinite stochastic state space.

To see this, let us assume that at some stage during operation a large number of cycles occur between $2a$ and $1b$ states. For this to happen the system must have alternating arrivals and services. The system starts with a service from state $2a$ and for simplicity we assign $g_1(X,Y)$ and $h_1(X,Y)$ as the form of distributions of the service and arrival processes which come out of the competitive choice at $2a$ the first time. The first few evolutions of this cycle are presented in table 3.3 where $j_n(X,Y) = g_n(X,Y) \ominus h_n(X,Y)$, $k_n(X,Y) = h_n(X,Y) \ominus g_n(X,Y)$.

[2]Synchronising states are marked with concentric circles. They represent states for which no memory of past evolution has to be maintained.

Tab. 3.3. The distributions of successive transitions along the $2a$, $1b$ cycle.

n	**Transition**	**Win**	**Service,** $g_n(X, Y)$	**Arrival,** $h_n(X, Y)$
1	$2a \longrightarrow 1b$	S	$g_1(X, Y)$	$h_1(X, Y)$
2	$1b \longrightarrow 2a$	A	$Y\|k_1(X, Y) < Y$	$k_1(X, Y)\|k_1(X, Y) < Y$
3	$2a \longrightarrow 1b$	S	$j_2(X, Y)\|j_2(X, Y) < X$	$X\|j_2(X, Y) < X$
4	$1b \longrightarrow 2a$	A	$Y\|k_3(X, Y) < Y$	$k_3(X, Y)\|k_3(X, Y) < Y$
5	$2a \longrightarrow 1b$	S	$j_4(X, Y)\|j_4(X, Y) < X$	$X\|j_4(X, Y) < X$
			$\vdots$	

In general:

$$\begin{aligned} g_{2n}(X, Y) &= Y|k_{2n-1}(X, Y) < Y \\ h_{2n}(X, Y) &= k_{2n-1}(X, Y)|k_{2n-1}(X, Y) < Y \\ g_{2n+1}(X, Y) &= j_{2n}(X, Y)|j_{2n}(X, Y) < X \\ h_{2n+1}(X, Y) &= X|j_{2n}(X, Y) < X \end{aligned} \tag{3.82}$$

As a result of this dependence on the previous distribution, no successive transitions along the same edge will carry the same distribution and the stochastic state space effectively has to encode the history of every competitive choice outcome, until a synchronisation state is reached again. At this stage the memory is reset.

What made the G/G/1/2 case tractable was that it passed through a synchronising state (the middle 1 state in figure 3.14) every two transitions. Hence there was a finite amount of history to encode in the distributions, and hence a finite stochastic state space.

3.3.3 Markovian Process Algebras

Markovian process algebras model only Markov processes. This allows them to take advantage of many useful simplifications, which effectively make general Markovian structures solvable.

It has already been shown that competitive choice can, in the generally-distributed case, be mapped on to the probabilistic branching of a stochastic transition system. Obviously competitive choice of Markov processes can be similarly mapped. The important difference to note from that of generally-distributed SPAs and generally-distributed queueing systems, is that, in a Markovian system, an expansion law exists for parallel composition.

For PEPA an expansion law is found in [52]; for MTIPP, one can be found in [47].

The expansion law for PEPA, for instance, maps the different components of the composition on to a finite Markovian state space consisting of competitive choices, in a similar fashion to CCS. We are in the luxurious position of saying that the hard work has therefore been done for us. This mapping represents a proof that the stochastic aggregation technique can be used directly to solve any Markovian process algebra.

So long as the parallel composition has been removed using the expansion law mentioned, we are able to map the components of a typical MPA on to its stochastic transition system loosely as follows:

$$(x, \lambda).P \quad \longrightarrow \quad \{X\}.P \tag{3.83}$$

$$(x, \lambda).P + (y, \mu).Q \quad \longrightarrow \quad \left[\frac{\lambda}{\lambda+\mu}\right]\{Z\}.P \oplus \left[\frac{\mu}{\lambda+\mu}\right]\{Z\}.Q \tag{3.84}$$

where $X \sim \exp(\lambda)$, $Y \sim \exp(\mu)$ and $Z \sim \exp(\lambda+\mu)$.

The immediate corollary of this is that an MPA no longer has to proceed via a Markov chain solution technique in order to obtain its stationary distribution. Since it can be mapped on to a finite stochastic transition system, the stochastic aggregation technique can be applied. An example of this is shown below.

Reward Structures

On the surface, the major benefit is that the stationary distribution can be generated in a state-wise fashion. This has a nice effect for calculating reward structures and performance measures, for MPAs. If a reward structure is sparse, i.e. it only rewards a few states, then the aggregation technique can be used to find only the stationary probabilities of the required states. Using traditional techniques, the entire model would have to be solved in order to calculate maybe only a few stationary values.

However, key to this is whether we can devise an algorithm which can generate a single element of the steady-state probability in less time than it would take to solve the CTMC. The proof method of section 3.2.7 is clearly not a good candidate for this. In fact, we suspect that the best we might be able to do is construct a matrix of Laplacian functions representing the transition distributions and then invert it in such a way as to get the overall path Laplacian function, representing the aggregated return path of the Cox-Miller Normal Form. Preliminary results indicate that this method would require inversion of an $n-1 \times n-1$ matrix for each of the n states. This represents a small practical saving per state over solving the complete CTMC, although the O-complexity is clearly the same.

A second property of the method is that it is a progressive technique. It will generate the distribution, a result at a time, rather than the entire solution in one go. This may well make it amenable to lazy implementation.

Example

As an example of solving a Markovian Process Algebra system, we examine the PEPA resource usage model of [52], defined in figure 3.18.

The cooperation is removed using the expansion law of [52], giving the state diagram of figure 3.19. In figure 3.20, states X_1 and X_3 have been aggregated to give state A_1 and states X_2 and X_3 have been aggregated to give state A_2. This first stochastic aggregation manœuvre reduces the transition system to

$$
\begin{aligned}
Process &\stackrel{\text{def}}{=} (use, r_1).(task, r_2).Process \\
Resource &\stackrel{\text{def}}{=} (use, r_3).(update, r_4).Resource \\
System &\stackrel{\text{def}}{=} Process \underset{\{use\}}{\bowtie} Resource
\end{aligned}
$$

Fig. 3.18. A resource usage model in PEPA.

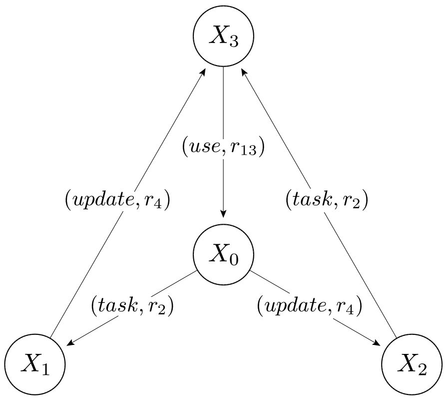

Fig. 3.19. The state space of the resource usage model.

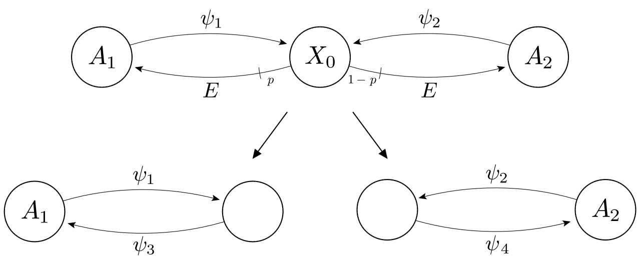

Fig. 3.20. Using stochastic aggregation on a PEPA model with cooperation.

the normal form for state X_0, where:

$$p = \frac{r_2}{r_2 + r_4} \tag{3.85}$$

$$L_E(z) = \frac{r_2 + r_4}{r_2 + r_4 + z} \tag{3.86}$$

$$L_{\psi_1}(z) = \frac{r_4 r_{13}}{(r_4 + z)(r_{13} + z)} \tag{3.87}$$

$$L_{\psi_2}(z) = \frac{r_2 r_{13}}{(r_2 + z)(r_{13} + z)} \tag{3.88}$$

in which $r_{13} = \min(r_1, r_3)$. Now we aggregate around states A_1 and A_2 to give:

$$L_{\psi_3}(z) = \frac{r_2(r_2+z)(r_{13}+z)}{(r_2+z)(r_{13}+z)(r_2+r_4+z) - r_2r_4r_{13}} \tag{3.89}$$

$$L_{\psi_4}(z) = \frac{r_4(r_4+z)(r_{13}+z)}{(r_4+z)(r_{13}+z)(r_2+r_4+z) - r_2r_4r_{13}} \tag{3.90}$$

A_1 and A_2 are now in Cox-Miller Normal Form and so we can get the steady-state probabilities, $\Pi(A_1)$ and $\Pi(A_2)$, from equation (3.29):

$$\Pi(A_1) = \frac{{r_2}^2(r_{13}+r_4)}{r_2r_4(r_2+r_4) + r_{13}({r_2}^2 + r_2r_4 + {r_4}^2)} \tag{3.91}$$

$$\Pi(A_2) = \frac{{r_4}^2(r_{13}+r_2)}{r_2r_4(r_2+r_4) + r_{13}({r_2}^2 + r_2r_4 + {r_4}^2)} \tag{3.92}$$

And now we can retrieve $\Pi(X_0) = 1 - (\Pi(A_1) + \Pi(A_2))$:

$$\Pi(X_0) = \frac{r_2r_4r_{13}}{r_2r_4(r_2+r_4) + r_{13}({r_2}^2 + r_2r_4 + {r_4}^2)} \tag{3.93}$$

This agrees with the result in Hillston 1994 [52, p.59] for the equivalent state X_1 which was obtained using Gaussian elimination. The other stationary probabilities are similarly obtained using stochastic aggregation:

$$\Pi(X_1) = \frac{{r_2}^2r_{13}}{r_2r_4(r_2+r_4) + r_{13}({r_2}^2 + r_2r_4 + {r_4}^2)} \tag{3.94}$$

$$\Pi(X_2) = \frac{{r_4}^2r_{13}}{r_2r_4(r_2+r_4) + r_{13}({r_2}^2 + r_2r_4 + {r_4}^2)} \tag{3.95}$$

$$\Pi(X_3) = \frac{r_2r_4(r_2+r_4)}{r_2r_4(r_2+r_4) + r_{13}({r_2}^2 + r_2r_4 + {r_4}^2)} \tag{3.96}$$

3.3.4 Component Analysis of a Stochastic Process Algebra Model

We now look at the possibility of a *component aggregation* analysis of a system, given that we now appreciate that a general interleaving analysis for a pre-emptive resume system is either going to be very hard or impossible.

So what do we mean by a component aggregation? This is similar to an idea of Bohnenkamp and Haverkort [10] which was based on stochastic bundle event structures. We wish to select a component of the system and alter it to add the aggregate stochastic effect of synchronisations and blocking from the other components. The concept differs, however, from that of [10] in so far as we do not attempt to regain any of the interleaving, instead preferring to aggregate over it completely. To explain this fully, we will demonstrate with an example using PEPA.

Before we attempt a worked example, we should understand what we are solving for and what kind of result we might expect to get back. At first sight it might appear as if we are obtaining some kind of product-form solution, working on a component-by-component level as we are. In fact this is not the case, instead what we will actually get is a derivative of the global model with much of the complexity from the interleaving being hidden by aggregate distributions. The effect of this means that we should not expect to see the whole stationary distribution, instead only parts of the distribution from states which have maintained their identity after the synchronisation information has been added.

This should not come as a surprise since obtaining a steady-state probability for a state involves having a model that isolates it, which for a synchronising action involves developing a proper interleaving. We have already seen that this is a hard task and indeed we think that this may be the best alternative in the absence of any way of handling such an interleaving.

$$C_0^0 \stackrel{\text{def}}{=} (a, \lambda).C_1^0 \qquad C_1^0 \stackrel{\text{def}}{=} (b, \mu).C_0^0$$

$$C_0^1 \stackrel{\text{def}}{=} (a, \top).C_1^1 \qquad C_1^1 \stackrel{\text{def}}{=} (c, \nu).C_0^1 \qquad S \stackrel{\text{def}}{=} C_0^0 \underset{\{a\}}{\bowtie} C_0^1$$

Fig. 3.21. A simple PEPA example for component aggregation

Worked Example in PEPA

We define the system of figure 3.21 for use with component aggregation.

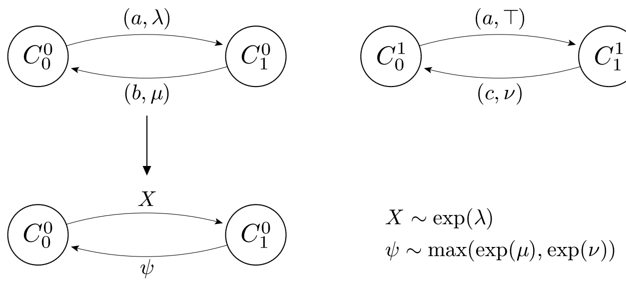

Fig. 3.22. Altering a component to reflect the stochastic effect of synchronisations.

Figure 3.22 shows the effect of incorporating the synchronisation into one of the components, C_0^0. The altered component, as expected, has the $\exp(\lambda)$ distribution attributed to its synchronised action as it was a passive synchronisation. However, we should also take into account the blocking effect of component C_0^1 after the a-transition has occurred. The a-transition can only re-enable globally after both components have enabled it locally—this means that both components must have completed their return b and c-transitions so the return edge takes on the maximum of the two associated distributions.

This is an interesting issue for process algebras, since the blocking issue is clearly a synchronisation in its own right, albeit an unspecified one; for all that process algebras pride themselves on specifying the *complete* observed behaviour, there is still some implicit behaviour in the paradigm.

Solving the altered component model in figure 3.22:

$$L_\psi(z) = \frac{\mu}{\mu+z} + \frac{\nu}{\nu+z} - \frac{\mu+\nu}{\mu+\nu+z} \tag{3.97}$$

$$\begin{aligned} \mathbb{E}(\psi) &= \frac{1}{\mu} + \frac{1}{\nu} - \frac{1}{\mu+\nu} \\ &= \frac{\mu^2+\mu\nu+\nu^2}{\mu\nu(\mu+\nu)} \end{aligned} \tag{3.98}$$

$$\Pi(C_0^0) = \frac{\mathbb{E}(X)}{\mathbb{E}(X)+\mathbb{E}(\psi)} = \frac{\mu\nu(\mu+\nu)}{\lambda(\mu^2+\mu\nu+\nu^2)+\mu\nu(\mu+\nu)} \tag{3.99}$$

Now, taking advantage of the fact that we have used a Markovian model, we can solve it using traditional PEPA techniques to check we have the same answer and see what we have not been able to obtain from the component solution.

We use the following state-mapping and display the interleaved state space of S in figure 3.23:

$$\begin{aligned} X_0 &\longrightarrow C_0^0 \underset{\{a\}}{\bowtie} C_0^1 \qquad & X_1 &\longrightarrow C_1^0 \underset{\{a\}}{\bowtie} C_1^1 \\ X_2 &\longrightarrow C_0^0 \underset{\{a\}}{\bowtie} C_1^1 \qquad & X_3 &\longrightarrow C_1^0 \underset{\{a\}}{\bowtie} C_0^1 \end{aligned}$$

This produces the following generator matrix, G:

$$G = \begin{pmatrix} -\lambda & \lambda & 0 & 0 \\ 0 & -(\mu+\nu) & \mu & \nu \\ \nu & 0 & -\nu & 0 \\ \mu & 0 & 0 & -\mu \end{pmatrix} \tag{3.100}$$

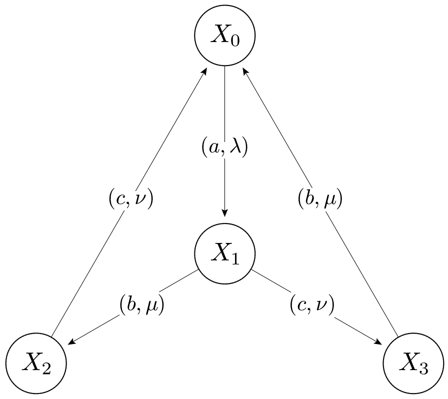

Fig. 3.23. The interleaved state space used to generate the Markov chain in PEPA.

which can be solved using Gaussian elimination to give:

$$\Pi(X_0) = 1 - \frac{\lambda(\mu^2+\mu\nu+\nu^2)}{\lambda(\mu^2+\mu\nu+\nu^2)+\mu\nu(\mu+\nu)} \tag{3.101}$$

$$\Pi(X_1) = \frac{\lambda\mu\nu}{\lambda(\mu^2+\mu\nu+\nu^2)+\mu\nu(\mu+\nu)} \tag{3.102}$$

$$\Pi(X_2) = \frac{\lambda\mu^2}{\lambda(\mu^2+\mu\nu+\nu^2)+\mu\nu(\mu+\nu)} \tag{3.103}$$

$$\Pi(X_3) = \frac{\lambda\nu^2}{\lambda(\mu^2+\mu\nu+\nu^2)+\mu\nu(\mu+\nu)} \tag{3.104}$$

We are pleased to note that equation (3.101) agrees with equation (3.99). Figure 3.23 shows that, as predicted, the states of the system which display the two different ways that the synchronisation can interleave are aggregated within our component model. Indeed in a generally-distributed synchronisation, we would not be able to display the finite global state space at all and we postulate that local modified components which give part of the steady-state solution are probably the best we can hope for.

In this instance, the modified components are identical, but it is easy to see that more complex systems would have different component solutions and would contribute different parts of the steady-state distribution.

As further research, it would be interesting to examine a general framework for this type of solution method. Although we have analysed a Markovian system here it would not have been difficult to tackle such a system with generally-distributed durations. It is possible that for heavily interdependent components the complexity of such a technique might be prohibitively high, but at the very least it might represent another tool in the arsenal to tackle both the state space explosion and generally-distributed processes.

3.4 Conclusion

In this chapter, we have shown that a class of generally-distributed systems can be solved. By restating a well-known aggregation technique in the language of stochastic process algebras, we can reduce any stochastic transition system to a normal form for a given state. These normal forms, in turn, can be solved by using the Cox-Miller Normal Form solution. For each state of the original system the stationary distribution can be recovered, in a progressive fashion. This extends a previous result [26] from a purely Markovian two-state example to a generally-distributed n-state solution technique.

We were able to present an analytical solution to a G/G/1/2 queueing system. Further, we demonstrated that the reason that larger queues were not directly soluble using this technique was that no finite representation existed as a stochastic transition system. Since Markovian process algebras have an expansion law for their concurrent operator, however, it is possible to write down a finite stochastic transition system for a general MPA system. This fact means that we are able to solve MPA systems without having to resort to Markov chains. This represents a small computational saving since it means that, in order to gain a few stationary probabilities, we no longer have to solve the entire system. We were also able to map a pre-emptive restart generally-distributed stochastic process algebra to stochastic transition systems, through our use of probabilistic operational semantics, defined in the annex 3.A.

The use of an aggregation technique means that we are no longer restricted to dealing in stationary distributions. The use of Laplace transforms has the benefit that we can easily derive the variance of the aggregated states as well as the mean. This will be useful for deriving reliability information in chapter 5. In the Markovian process algebra case, in the event that we require a probability distribution function, we note that there is a useful Laplace inversion formula by Harrison [39] which would allow us to invert any aggregate, non-cyclic path of a Markovian system.

The process of aggregation is not one which alters the stochastic behaviour of a system (unlike, for example, insensitivity), which is essential when we come to consider more than just steady-state distributions in chapter 5.

We completed the chapter with an example of a component-based analysis through component aggregation of a process algebra model which abstracted away from the complexity of the interleaving synchronisation. This may offer a performance model solution which avoids state space explosion problems and is capable of coping with generally-distributed evolutions. It may be possible to develop a general framework for steady-state solution of stochastic process algebras about such a technique; however this thesis is concerned predominantly with reliability rather than performance modelling at this stage. It is recognised, though, that a previous attempt at such a method [10] in a different stochastic framework had complexity difficulties and that this might well be a problem for this technique too. It is worth further investigation—especially in the area of bounding solutions for performance models.

Annex

3.A Probabilistic Operational Semantics of Stochastic Process Algebras

In this annex, we demonstrate the mapping of three stochastic process algebra models to their underlying stochastic transition systems. We do this by introducing a slightly modified form of operational semantics to that of Plotkin [82]. In order to express the probabilistic nature of the stochastic evolutions we need to employ a form of probabilistic operational semantics.

Operational semantics are usually presented as in figure 3.24. An individual rule is translated as: from the clause above the line, we can infer the resulting clause below the line, subject to the side conditions (if any).

The probabilistic operational semantics only extends the functional version by adding probabilities, for uncertain operation. So in figure 3.25, we interpret the competitive choice rule as: if $P \xrightarrow{(a,\lambda)} P'$ and $Q \xrightarrow{(b,\lambda)} Q'$ then we can infer that with probability $\lambda/(\lambda+\mu)$, $P+Q \xrightarrow{(a,\lambda+\mu)} P'$ or with probability $\mu/(\lambda+\mu)$, $P+Q \xrightarrow{(a,\lambda+\mu)} Q'$. This embodies perfectly the competitive nature of the race condition which is missing from the equivalent functional definition.

3.A.1 PEPA Operational Semantics

Figure 3.24 shows the traditional operational semantic definition of PEPA reproduced from [55], presented in the style of [82] for comparison.

Figure 3.25 shows the probabilistic operational semantic definition of PEPA, which can capture more of the behavioural features of the stochastic language, such as: the probabilistic nature of competitive choice, the actual interleaving of non-synchronising actions in the pre-emptive resume/restart cooperation and blocking cooperation. The active synchronisation model remains unchanged. The definitions of constant and hiding constructs are identical to those in figure 3.24.

3.A.2 MTIPP Operational Semantics

Figure 3.26 shows the probabilistic operational semantic definition of MTIPP. This is largely the same as for PEPA; but note the difference in the active synchronisation model, here called *scaling factor* synchronisation. This difference in Markovian synchronisation is the subject of further study in chapter 4.

3.A.3 Operational Semantics of a GDSPA

Figure 3.27 shows the probabilistic operational semantic definition for a generally-distributed stochastic process algebra with pre-emptive restart. As can be seen from the restart rule, the fact that a pre-empted process gets restarted means that a history of residual distributions does not need to be kept by the algebra. The stochastic results underlying this definition are taken from section 3.3.1.

This last set of semantics is effectively a generalisation of the previous Markovian semantics for PEPA and MTIPP. It summarises the internal workings of the MPAs in terms of the random variables as well as being able to model with general distributions.

The Last-to-Finish and First-to-Finish synchronisation models referred to in this semantics (figure 3.27) are the subject of chapter 4 and are defined in full in section 4.3.

Prefix

$$\frac{}{(a,\lambda).P \xrightarrow{(a,\lambda)} P}$$

Competitive Choice

$$\frac{P \xrightarrow{(a,\lambda)} P'}{P + Q \xrightarrow{(a,\lambda)} P'} \qquad \frac{Q \xrightarrow{(a,\mu)} Q'}{P + Q \xrightarrow{(a,\mu)} Q'}$$

Cooperation

$$\frac{P \xrightarrow{(a,\lambda)} P'}{P \underset{S}{\bowtie} Q \xrightarrow{(a,\lambda)} P' \underset{S}{\bowtie} Q} \quad a \notin S$$

$$\frac{Q \xrightarrow{(a,\mu)} Q'}{P \underset{S}{\bowtie} Q \xrightarrow{(a,\mu)} P \underset{S}{\bowtie} Q'} \quad a \notin S$$

$$\frac{P \xrightarrow{(a,\lambda)} P' \quad Q \xrightarrow{(a,\mu)} Q'}{P \underset{S}{\bowtie} Q \xrightarrow{(a,R)} P' \underset{S}{\bowtie} Q'} \quad a \in S$$

where $R = \frac{\lambda}{r_a(P)} \frac{\mu}{r_a(Q)} \min(r_a(P), r_a(Q))$

Hiding

$$\frac{P \xrightarrow{(a,\lambda)} P'}{P\backslash L \xrightarrow{(a,\lambda)} P'\backslash L} \quad a \notin L \qquad \frac{P \xrightarrow{(a,\lambda)} P'}{P\backslash L \xrightarrow{(\tau,\lambda)} P'\backslash L} \quad a \in L$$

Constant

$$\frac{P \xrightarrow{(a,\lambda)} P'}{A \xrightarrow{(a,\lambda)} P'} \quad A \stackrel{\mathrm{def}}{=} P$$

Fig. 3.24. Functional operational semantics for PEPA.

Prefix

$$\frac{}{(a,\lambda).P \xrightarrow{(a,\lambda)} P}$$

Competitive Choice

$$\frac{P \xrightarrow{(a,\lambda)} P' \quad Q \xrightarrow{(b,\mu)} Q'}{\begin{array}{l} \frac{\lambda}{\lambda+\mu} : P + Q \xrightarrow{(a,\lambda+\mu)} P' \\ \frac{\mu}{\lambda+\mu} : P + Q \xrightarrow{(b,\lambda+\mu)} Q' \end{array}}$$

Cooperation (Blocking)

$$\frac{P \xrightarrow{(a,\lambda)} P' \quad Q \xrightarrow{(b,\mu)} Q'}{P \underset{S}{\bowtie} Q \xrightarrow{(b,\mu)} P \underset{S}{\bowtie} Q'} \quad a \in S, b \notin S$$

Cooperation (Passive)

$$\frac{P \xrightarrow{(a,\lambda)} P' \quad Q \xrightarrow{(a,\top)} Q'}{P \underset{S}{\bowtie} Q \xrightarrow{(a,\lambda)} P' \underset{S}{\bowtie} Q'} \quad a \in S$$

Cooperation (Pre-emptive Restart/Resume)

$$\frac{P \xrightarrow{(a,\lambda)} P' \quad Q \xrightarrow{(b,\mu)} Q'}{\begin{array}{l} \frac{\lambda}{\lambda+\mu} : P \underset{S}{\bowtie} Q \xrightarrow{(a,\lambda+\mu)} P' \underset{S}{\bowtie} Q \\ \frac{\mu}{\lambda+\mu} : P \underset{S}{\bowtie} Q \xrightarrow{(b,\lambda+\mu)} P \underset{S}{\bowtie} Q' \end{array}} \quad a, b \notin S$$

Cooperation (Active)

$$\frac{P \xrightarrow{(a,\lambda)} P' \quad Q \xrightarrow{(a,\mu)} Q'}{P \underset{S}{\bowtie} Q \xrightarrow{(a,R)} P' \underset{S}{\bowtie} Q'} \quad a \in S$$

where $R = \frac{\lambda}{r_a(P)} \frac{\mu}{r_a(Q)} \min(r_a(P), r_a(Q))$

Fig. 3.25. Probabilistic operational semantics for PEPA.

Prefix

$$\frac{}{(a,\lambda).P \xrightarrow{(a,\lambda)} P}$$

Competitive Choice

$$\frac{P \xrightarrow{(a,\lambda)} P' \quad Q \xrightarrow{(b,\mu)} Q'}{\begin{array}{l} \frac{\lambda}{\lambda+\mu} : P + Q \xrightarrow{(a,\lambda+\mu)} P' \\ \frac{\mu}{\lambda+\mu} : P + Q \xrightarrow{(b,\lambda+\mu)} Q' \end{array}}$$

Synchronisation (Blocking)

$$\frac{P \xrightarrow{(a,\lambda)} P' \quad Q \xrightarrow{(b,\mu)} Q'}{P \,||_S\, Q \xrightarrow{(b,\mu)} P \,||_S\, Q'} \quad a \in S, b \notin S$$

Synchronisation (Pre-emptive Restart/Resume)

$$\frac{P \xrightarrow{(a,\lambda)} P' \quad Q \xrightarrow{(b,\mu)} Q'}{\begin{array}{l} \frac{\lambda}{\lambda+\mu} : P \,||_S\, Q \xrightarrow{(a,\lambda+\mu)} P' \,||_S\, Q \\ \frac{\mu}{\lambda+\mu} : P \,||_S\, Q \xrightarrow{(b,\lambda+\mu)} P \,||_S\, Q' \end{array}} \quad a, b \notin S$$

Synchronisation (Scaling factor)

$$\frac{P \xrightarrow{(a,\lambda)} P' \quad Q \xrightarrow{(a,\mu)} Q'}{P \,||_S\, Q \xrightarrow{(a,\lambda\mu)} P' \,||_S\, Q'} \quad a \in S$$

Fig. 3.26. Probabilistic operational semantics for MTIPP.

Prefix

$$\frac{}{(a,X).P \xrightarrow{(a,X)} P}$$

Competitive Choice

$$\frac{P \xrightarrow{(a,X)} P' \quad Q \xrightarrow{(b,Y)} Q'}{\begin{array}{l}\mathbb{P}(X<Y) : P+Q \xrightarrow{(a,X\,|\,X<Y)} P' \\ \mathbb{P}(Y<X) : P+Q \xrightarrow{(b,Y\,|\,Y<X)} Q'\end{array}}$$

Synchronisation (Blocking)

$$\frac{P \xrightarrow{(a,X)} P' \quad Q \xrightarrow{(b,Y)} Q'}{P \,||_S\, Q \xrightarrow{(b,Y)} P \,||_S\, Q'} \quad a \in S, b \notin S$$

Synchronisation (Pre-emptive Restart)

$$\frac{P \xrightarrow{(a,X)} P' \quad Q \xrightarrow{(b,Y)} Q'}{\begin{array}{l}\mathbb{P}(X<Y) : P \,||_S\, Q \xrightarrow{(a,X\,|\,X<Y)} P' \,||_S\, Q \\ \mathbb{P}(Y<X) : P \,||_S\, Q \xrightarrow{(b,Y\,|\,Y<X)} P \,||_S\, Q'\end{array}} \quad a, b \notin S$$

Synchronisation (Last-to-Finish)

$$\frac{P \xrightarrow{(a,X)} P' \quad Q \xrightarrow{(a,Y)} Q'}{P \,||_S\, Q \xrightarrow{(a,\max(X,Y))} P' \,||_S\, Q'} \quad a \in S$$

Synchronisation (First-to-Finish)

$$\frac{P \xrightarrow{(a,X)} P' \quad Q \xrightarrow{(a,Y)} Q'}{P \,||_S\, Q \xrightarrow{(a,\min(X,Y))} P' \,||_S\, Q'} \quad a \in S$$

Fig. 3.27. Probabilistic operational semantics for a generally-distributed stochastic process algebra with pre-emptive restart.

Chapter 4

Synchronisation in SPAs

4.1 Introduction

Markovian process algebras approximate their model of synchronising events in order to preserve their Markovian nature. This chapter investigates synchronisation models in a stochastic context. We introduce and define two distinct synchronisation models, Last-to-Finish synchronisation and First-to-Finish synchronisation (originally presented in [15, 16]). We demonstrate that MPAs, in trying to emulate the first of these models, Last-to-Finish synchronisation, have to make a necessary approximation to the real model.

Through analysis and examples it is demonstrated how the Markovian approximation of Last-to-Finish synchronisation affects the accuracy of the performance model. We compare results of four MPA models (including MTIPP [47] and PEPA [55]) with an analytic solution to see which systems provide better real-life synchronisations. This also allows us to suggest tactics for using combinations of synchronisation models to get a more reliable solution.

In focussing on this issue we accomplish half of our aim in this thesis, that of understanding how to use current methodologies in a reliable fashion. It is essential that the theoretical and practical advances that have been made in MPAs are made full use of by getting a better physical understanding of the end results. After all, it is in these purely Markovian systems that the best opportunity exists for solving large state-space stochastic models. This is partly as a result of MPAs having the most mature toolsets and partly because they are a simpler paradigm than the more general stochastic process algebras.

4.2 Synchronisation in MPAs

Purely Markovian process algebras have the following definition, as set out in chapter 2:

$$P \ ::= \ (a, \lambda).P \ \Big| \ P + P \ \Big| \ P \,||_S\, P \ \Big| \ P/L \ \Big| \ A \tag{4.1}$$

The issue lies in the $P \,||_S\, Q$ (or $P \underset{S}{\bowtie} Q$ in PEPA notation), which represents an interaction between components of the algebra. As we will see in sections 4.4.1 and 4.4.2 the intention is that this interaction should reflect the progress of the slower component. Equation (4.8) shows that this quantity is not exponential so some form of approximation has to be made, as recognised in [53]:

> ... there is an assumption that the interaction, or shared action, will have an exponentially distributed duration, in the same way as the individual actions of the agents. This assumption is a pragmatic one, although we will see ... that it is not necessarily a justified one.

In section 4.3, we examine common types of synchronisation that occur in software engineering. Section 4.4 discusses the models used by MTIPP and

PEPA and presents an analysis on how they differ from the real-world synchronisation models. Section 4.5 discusses possible synchronisation models for MPAs which allow more accurate performance statistics to be generated. Finally, a worked example is presented in section 4.6 which serves to compare MTIPP and PEPA with each other and with the suggested models of section 4.5.

Later stochastic process algebras have rectified this approximation by extending the class of distributions that can be modelled so that the slower component in an interaction is capable of being represented precisely. This can be done minimally by introducing immediate transitions to supplement the Markovian transitions, as in IMC [43] and EMPA [8], for example. Other algebras employ phase-type distributions which are also closed under this form of synchronisation, for instance, Stochastic Causality-Based Process Algebra [19] and El-Rayes et al's recent $\mathrm{PEPA}_{\mathrm{ph}}^{\infty}$ [29].

However, many example systems have been constructed in MTIPP and especially PEPA, because of their ease-of-use [62, 11, 56, 31]. In addition, two well developed tools have been constructed around the two algebras, TIPPtool [46] and PEPA Workbench [33]. So this analysis can give both extra accuracy information about existing models and confidence to a modeller wishing to use the full power of synchronisation in the two systems. It should also be noted that many of the recent generally-distributed algebras either do not yet have general analytic solution techniques or do not have mature tools capable of solving the larger systems.

4.3 Real-world Synchronisation Models

In this section, we distinguish between passive synchronisation, as described in section 4.3.1, and active synchronisations, where both or all the constituent processes make a contribution to the activity of the synchronisation. Active synchronisations are classified in sections 4.3.2–4.3.5.

4.3.1 Client-Server

This is often referred to as passive synchronisation, where one of the processes (the client) idles while its request is dealt with by the server process. The client takes no active part in the activity of the synchronisation, so the duration of the combined event is completely dependent on the server process.

Most Markovian process algebras can deal with this type of model (certainly PEPA and MTIPP have syntactic or numerical tricks for handling such synchronisations). It should be noted that the distinction between active and passive synchronisations is a physical one since in fact they represent limiting cases of the synchronisation models below.

4.3.2 First-to-Finish

The First-to-Finish (FTF) synchronisation of two processes X and Y states that the synchronisation can terminate on successful completion of either X or Y, whichever occurs first. Thus FTF synchronisation follows the minimum time duration of X and Y.

An example of this might be a distributed code-breaking effort between processors, where the first processor to find the correct key brings the entire effort to a close. No further processing is required as no further information is required.

This interaction can only be used in particular circumstances. For a searching problem as above, it requires that the search-space be completely divisible between processes, without any subsequent inter-process communication. Furthermore, there must only be one solution, such that when found it is immediately known that the remaining processes are undertaking a fruitless search.

Stochastic Details

For general distributions X and Y, we define:

$$\begin{aligned} F_{X \overset{\|}{\text{FTF}} Y}(t) &= F_{\min(X,Y)}(t) \\ &= \mathbb{P}(\min(X,Y) \leq t) \\ &= 1 - \mathbb{P}(\min(X,Y) > t) \\ &= 1 - \mathbb{P}(X > t)\mathbb{P}(Y > t) \\ &= 1 - \overline{F}_X(t)\overline{F}_Y(t) \qquad (4.2) \end{aligned}$$

In the case of n synchronising processes $X_1, \ldots, X_n$:

$$F_{X_1 \overset{\|}{\text{FTF}} \cdots \overset{\|}{\text{FTF}} X_n}(t) = 1 - \prod_{i=1}^{n} \overline{F}_{X_i}(t) \qquad (4.3)$$

If X and Y are exponentially distributed processes with rates λ and μ, then the well known result for First-to-Finish synchronisation of X and Y is generated:

$$f_{X \overset{\|}{\text{FTF}} Y}(t) = f_{\min(X,Y)}(t) = (\lambda + \mu)e^{-(\lambda+\mu)t} \qquad (4.4)$$

This is significant since it tells us that the FTF synchronisation of two exponential distributions is itself an exponential distribution, as shown in figure 4.1. Also, if $\lambda \to 0$ in equation (4.4), equivalent to X having infinite mean duration, then we recover the passive client-server model, described in section 4.3.1.

The First-to-Finish synchronisation for exponential distributions has the following mean and variance properties:

$$\mathbb{E}(\min(X,Y)) = \frac{1}{\lambda + \mu} \qquad \mathbf{Var}(\min(X,Y)) = \frac{1}{(\lambda+\mu)^2} \qquad (4.5)$$

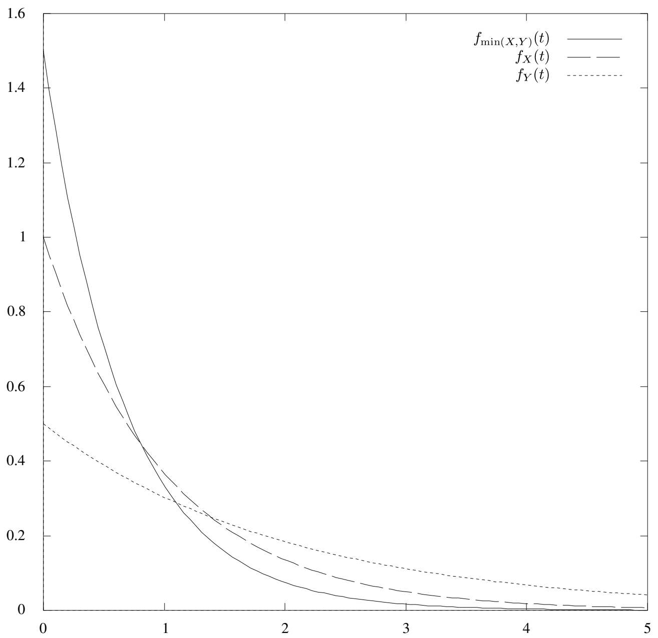

Fig. 4.1. The distribution of a First-to-Finish synchronisation of two exponential transitions.

4.3.3 Last-to-Finish

The Last-to-Finish (LTF) synchronisation of two processes X and Y only allows the synchronisation to terminate when both X and Y have completed. LTF synchronisation follows the maximum sojourn time of X and Y.

To use the searching analogy again: in a distributed search-effort, where there is an unknown number of solutions to be found, each process must search all of its allocated search-space to ensure that all solutions are identified. Thus the overall search time is governed by the longest running process.

In general, this type of synchronisation requires all participants to have completed before allowing the process to continue. It is widely used in parallel algorithms because it reduces algorithmic complexity, despite the fact that it can introduce considerable delay. The searching example given above required a LTF synchronisation for functional correctness, however most instances of LTF synchronisation do not require it for correct operation. In fact, most synchronisations use the LTF model to ensure that a parallel system is in a known state, for areas of program execution which require a safety condition.

Stochastic Details

For general distributions X and Y, we define:

$$\begin{aligned}
F_{X \overset{\parallel}{\text{LTF}} Y}(t) &= F_{\max(X,Y)}(t) \\
&= \mathbb{P}(\max(X,Y) \leq t) \\
&= \mathbb{P}(X \leq t)\mathbb{P}(Y \leq t) \\
&= F_X(t)F_Y(t) \qquad (4.6)
\end{aligned}$$

In the case of n synchronising processes $X_1, \ldots, X_n$:

$$F_{X_1 \overset{\parallel}{\text{LTF}} \cdots \overset{\parallel}{\text{LTF}} X_n}(t) = \prod_{i=1}^{n} F_{X_i}(t) \qquad (4.7)$$

If X and Y are exponentially distributed processes with parameters λ and μ, then the similarly well known result for Last-to-Finish synchronisation of X and Y is:

$$f_{X \overset{\parallel}{\text{LTF}} Y}(t) = f_{\max(X,Y)}(t) = \lambda e^{-\lambda t} + \mu e^{-\mu t} - (\lambda + \mu)e^{-(\lambda+\mu)t} \qquad (4.8)$$

This is the crux, for the reason that the LTF synchronisation of two exponential processes is not itself an exponential distribution, also shown in figure 4.2. Frequently, Markovian process algebras have to model Last-to-Finish synchronisations, because they are such a common form of synchronisation. However they can only do so with a single exponential distribution as a *best guess*. It is exactly this forced approximation and its consequences that we set out to investigate in the course of this chapter. Also note that by letting $\mu \to \infty$ in equation (4.8), equivalent to Y being instantaneous, we recover the passive client-server model, described in section 4.3.1.

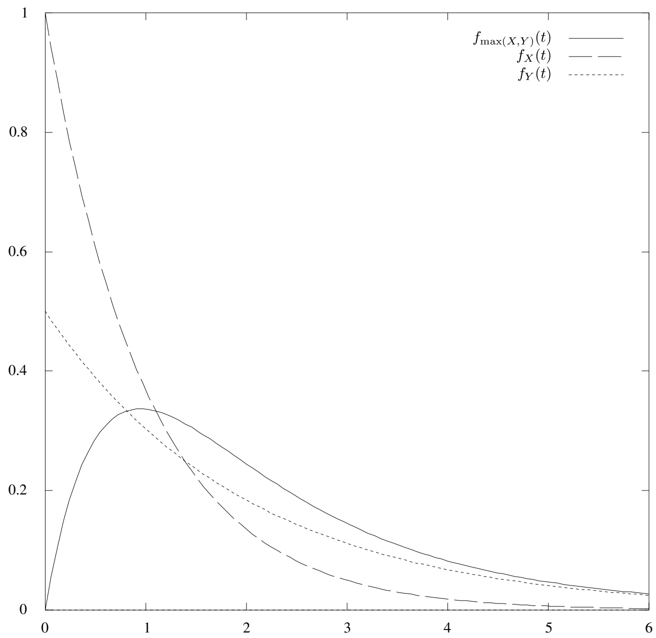

Fig. 4.2. The distribution of a Last-to-Finish synchronisation of two exponential transitions.

The Last-to-Finish synchronisation for exponential distributions has the following mean and variance properties:

$$\mathbb{E}(\max(X,Y)) = \frac{1}{\lambda} + \frac{1}{\mu} - \frac{1}{\lambda+\mu} \tag{4.9}$$

$$\mathbf{Var}(\max(X,Y)) = \frac{1}{\lambda^2} + \frac{1}{\mu^2} - \frac{3}{(\lambda+\mu)^2} \tag{4.10}$$

4.3.4 N-to-Finish

FTF and LTF synchronisations are not the only two possibilities. There are many possible synchronisations which lie between the two. These intermediate synchronisations can be roughly aggregated into an N-to-Finish class. Specifically, this is a system where N of M processes are required to finish before the synchronisation is complete. This class of cases clearly lies between the two FTF and LTF synchronisations and will not be considered explicitly in this chapter. Instead we suggest using the FTF and LTF models as bounding limits to these more esoteric synchronisations.

Real instances of these synchronisations do exist. In safety-critical computing a critical function might be managed by three or more controlling programs running in parallel. When a change of state is required, a single arbiter program will count the votes of the controlling programs and go with the majority decision. Because of the complexity of implementation, these forms of synchronisation are rarely encountered.

There is no obvious two-process version of this model, but in the event that a synchronisation with performance characteristics between that of FTF and LTF is required, we will not discard the notion.

4.3.5 Other Models

We now argue intuitively that there are no other synchronisation models outside the bounds of FTF and LTF synchronisation. Consider a two-process synchronisation which performs slower than a Last-to-Finish model. By modelling the overall sojourn time of the synchronisation as a third process and incorporating this, via another LTF synchronisation, into the original synchronisation, a Last-to-Finish model can be recovered.

Similarly, consider a two-process synchronisation which performs faster than the First-to-Finish synchronisation. By representing the faster model as a third process and incorporating its performance into the synchronisation, a First-to-Finish synchronisation is recovered.

Summarising our approach to synchronisation models: synchronisation is at best First-to-Finish and at worst Last-to-Finish. Passive synchronisations can be encompassed within this framework too: by considering them as either a Last-to-Finish model where the client terminates instantly or a First-to-Finish model where the client never terminates.

4.4 Comparison with MPA Synchronisations

In this section, we compare the synchronisations of two Markovian process algebras (PEPA and MTIPP) to the real-world LTF and FTF synchronisations, outlined in section 4.3.

4.4.1 PEPA

Introduction

The PEPA synchronisation model selects the exponential distribution with the largest mean to represent the evolution of the overall synchronisation. Passive synchronisations have an alternative syntactic representation. In [53], the selection of the active synchronisation style is described as follows:

> The definition ... is chosen to reflect the idea that the rate at which such a joint action can occur will be limited by the rate of the slowest participant.

Clearly, PEPA is emulating a Last-to-Finish strategy with its synchronisation model. However, given that it is restricted to using exponential distributions, this prevents it from using an exact LTF synchronisation, and so it has to settle for an approximation. The details of the PEPA cooperation have been described in detail in section 2.4.3, but in essence the synchronisation takes the minimum rate parameter of its constituent processes.

Stochastic Details

If X is distributed with exponential parameter λ and Y is distributed with exponential parameter μ, then the PEPA model (shown in figure 4.3 against the equivalent LTF synchronisation) has the following distribution:

$$X \underset{\text{PEPA}}{\|} Y \sim \exp(\min(\lambda, \mu)) \tag{4.11}$$

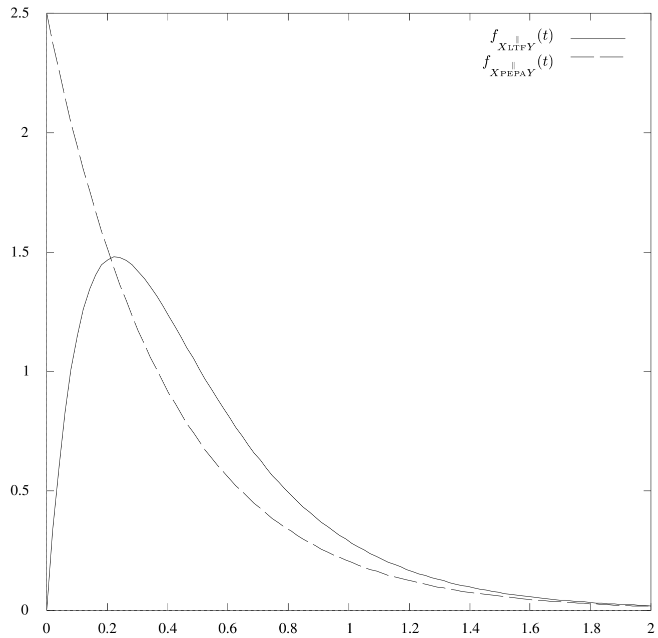

Fig. 4.3. The distribution of a Last-to-Finish synchronisation and the PEPA approximating cooperation.

This model has the following mean and variance properties for $0 < \lambda < \mu$:

$$\mathbb{E}(X \overset{\|}{\text{PEPA}} Y) = \frac{1}{\lambda} \qquad \mathbf{Var}(X \overset{\|}{\text{PEPA}} Y) = \frac{1}{\lambda^2} \tag{4.12}$$

4.4.2 MTIPP

Introduction

The MTIPP synchronisation model generates an exponential distribution with parameter equal to the product of its component parameters. [47] makes this statement about this choice of synchronisation model:

> From a stochastic point of view this ensures that the resulting distribution remains proportional to its origins which allows [us] to model [the] scaling influence of certain components.

In [53], this is described as a generalisation of the client-server model of interaction. So if either participating rate parameter is 1, then the standard passive synchronisation model is recovered. For other values of the rate parameters the picture is less clear: one value can be seen as a basic level of service which the server delivers, and the other can be taken as a *scaling factor* of that basic rate of service. One problem with this view, apart from not distinguishing syntactically between the client and server, is that it contains no concept of the maximum possible server performance. That is, the client could be modelled to request a service at a rate which the server could not supply.

This is only an interpretation of the model. The primary reason for choosing the model was almost certainly because of the useful algebraic properties of multiplication. The automatic inheritance of commutativity, associativity and distributivity across addition make it an ideal choice. These properties allow for the construction of a congruence relation across the algebra which in turn allows compositional features of the SPA to be exploited.

In order for PEPA to achieve the same properties ($\min(\cdot\,,\cdot)$ is not distributive across addition), the actual operator had to use *apparent rates* and renormalise the result ([52, 53, 54] and section 2.4.3 for a detailed description).

Stochastic Details

If X is distributed with exponential parameter λ and Y is distributed with exponential parameter μ, then the MTIPP model (shown in figure 4.4 against the equivalent LTF synchronisation) has the following distribution:

$$X \underset{\text{TIPP}}{\|} Y \sim \exp(\lambda\mu) \tag{4.13}$$

This model has the following mean and variance properties:

$$\mathbb{E}(X \underset{\text{TIPP}}{\|} Y) = \frac{1}{\lambda\mu} \qquad \mathbf{Var}(X \underset{\text{TIPP}}{\|} Y) = \frac{1}{(\lambda\mu)^2} \tag{4.14}$$

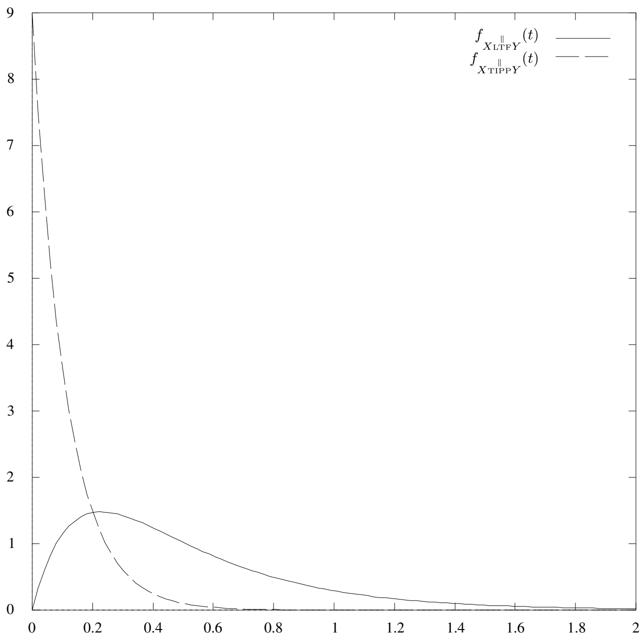

Fig. 4.4. The distribution of a Last-to-Finish synchronisation and the MTIPP approximating synchronisation.

4.4.3 Comparing MTIPP and PEPA Synchronisations

In the following sections, we summarise the stochastic properties of MTIPP and PEPA in comparison with the standard FTF and LTF models (details in the annex section 4.A). It is useful to compare expected synchronisation times, in each case, as this gives us an indication of how the mean performance predictions of the MPA models will differ from the real-world behaviour.

PEPA

For PEPA, the synchronisation model turns out to be static relative to the LTF and FTF models. From section 4.A, for all $0 < \lambda < \mu$, we have:

$$\mathbb{E}(X \overset{\|}{\text{FTF}} Y) < \mathbb{E}(X \overset{\|}{\text{PEPA}} Y) < \mathbb{E}(X \overset{\|}{\text{LTF}} Y) \tag{4.15}$$

Clearly PEPA is modelling a valid synchronisation and one closer to LTF than FTF (as is obvious from its choice of min$(\cdot, \cdot)$ operator). However, the fact that it underestimates the synchronisation mean will have a large effect on the steady-state solution of the CTMC. By underestimating the mean or overestimating the rate, PEPA will tend to overestimate the performance or capacity of a system. This is further borne out by the worked example in section 4.6.

MTIPP

For MTIPP, the synchronisation is not at all static compared to the LTF and FTF models. Summarising the detailed results in section 4.A:

$\mathbb{E}(X \underset{\text{TIPP}}{\parallel} Y) < \mathbb{E}(X \underset{\text{FTF}}{\parallel} Y)$ for $0 < \lambda < \mu$ iff:

$$1 + \frac{\lambda}{\mu} < \lambda < \infty \tag{4.16}$$

$\mathbb{E}(X \underset{\text{FTF}}{\parallel} Y) < \mathbb{E}(X \underset{\text{TIPP}}{\parallel} Y) < \mathbb{E}(X \underset{\text{LTF}}{\parallel} Y)$ for $0 < \lambda < \mu$ iff either:

$$\begin{cases} 1 < \lambda < 2 & \text{and} \quad \lambda - 1 < \frac{\lambda}{\mu} < 1 \\ \frac{2}{3} < \lambda < 1 & \\ 0 < \lambda < \frac{2}{3} & \text{and} \quad 0 < \frac{\lambda}{\mu} < \frac{1}{2}\left[-1 + \sqrt{\frac{1+3\lambda}{1-\lambda}}\right] \end{cases} \tag{4.17}$$

$\mathbb{E}(X \underset{\text{LTF}}{\parallel} Y) < \mathbb{E}(X \underset{\text{TIPP}}{\parallel} Y)$ for $0 < \lambda < \mu$ iff:

$$0 < \lambda < \frac{2}{3} \quad \text{and} \quad \frac{1}{2}\left[-1 + \sqrt{\frac{1+3\lambda}{1-\lambda}}\right] < \frac{\lambda}{\mu} < 1 \tag{4.18}$$

Since MTIPP's synchronisation model varies over the whole spectrum of other models, it is difficult to generalise its effect on performance modelling. However, it is clear that it would be possible to construct a system which either underestimated or overestimated the performance across a particular synchronisation, according to the choice of parameters. To maintain the criterion discussed in section 4.3.5, MTIPP should not have to stray outside the bounds provided by FTF and LTF synchronisation. Given this requirement, the allowable parameter space for MTIPP needs to be restricted by the ranges specified in equation (4.17). MTIPP's relative performance characteristics are compared with the other models in the worked example in section 4.6.

4.5 Alternative Synchronisation Strategies

The aim of this section is to suggest methods which accurately approximate a Last-to-Finish model. In one case, this involves maintaining the exponential distribution and suggesting an alternative synchronisation rate, and in the other two instances we suggest methods which use and augment the MTIPP and PEPA models. By following these different models through to the steady-state solution stage in section 4.6, we can then make qualitative statements about the errors that the MPA approximations introduce.

4.5.1 Mean-preserving Synchronisation

The result of solving the CTMC for an MPA system is the steady-state distribution for that system, that is, a description of the long-run average behaviour.

One possible solution would be to design an exponential synchronisation with the same mean as the LTF synchronisation. In this way, the correct average behaviour for each synchronisation is propagated into the CTMC. So we need to find the exponential distribution with mean (from equation (4.9)):

$$\frac{1}{\lambda} + \frac{1}{\mu} - \frac{1}{\lambda + \mu} \tag{4.19}$$

That is, we require a random variable $Z \sim \exp(\gamma)$ such that:

$$\frac{1}{\gamma} = \frac{1}{\lambda} + \frac{1}{\mu} - \frac{1}{\lambda + \mu} \tag{4.20}$$

Thus:

$$\gamma = \frac{\lambda\mu(\lambda+\mu)}{(\lambda+\mu)^2 - \lambda\mu} \tag{4.21}$$

In the notation of [53], we would write:

$$R = \lambda \otimes \mu = \frac{\lambda\mu(\lambda+\mu)}{(\lambda+\mu)^2 - \lambda\mu} \tag{4.22}$$

It should be noted that this operator is neither associative nor distributive over addition, so it is not possible to construct a congruence and make use of system decomposition in solving the model. Its relative performance characteristics are compared with the other models in the worked example in section 4.6.

4.5.2 Using PEPA to Bound the LTF Synchronisation

Unlike MTIPP, PEPA maintains the same mean characteristics in its synchronisation relative to the LTF and FTF models, for all parameter values (annex 4.A). That is, from equation (4.15), for all $0 < \lambda < \mu$:

$$\mathbb{E}(X \underset{\text{FTF}}{\parallel} Y) < \mathbb{E}(X \underset{\text{PEPA}}{\parallel} Y) < \mathbb{E}(X \underset{\text{LTF}}{\parallel} Y)$$

A possible approach for system design would be to design a dual synchronisation model to that of PEPA, whose mean is always greater than that of the LTF model. In doing this, we aim to try and bound the performance of the real synchronisation model with PEPA above and what we now call PEPA* below. So for all $0 < \lambda < \mu$, we require:

$$\mathbb{E}(X \underset{\text{FTF}}{\parallel} Y) < \mathbb{E}(X \underset{\text{PEPA}}{\parallel} Y) < \mathbb{E}(X \underset{\text{LTF}}{\parallel} Y) < \mathbb{E}(X \underset{\text{PEPA*}}{\parallel} Y) \tag{4.23}$$

If we define PEPA* to have synchronisation[1]:

$$X \underset{\text{PEPA*}}{\parallel} Y \sim \exp\left(\frac{\lambda\mu}{\lambda+\mu}\right) \tag{4.24}$$

[1] The reason that we have assigned the new synchronisation the label PEPA*, is that, coincidentally, it used to be the old synchronisation model for PEPA [53].

Now:

$$\mathbb{E}(X \underset{\text{PEPA*}}{||} Y) = \frac{1}{\lambda} + \frac{1}{\mu} \tag{4.25}$$

and since for all $\lambda, \mu > 0$:

$$\frac{1}{\lambda} + \frac{1}{\mu} - \frac{1}{\lambda + \mu} < \frac{1}{\lambda} + \frac{1}{\mu} \tag{4.26}$$

$$\Rightarrow \mathbb{E}(X \underset{\text{LTF}}{||} Y) < \mathbb{E}(X \underset{\text{PEPA*}}{||} Y) \tag{4.27}$$

So PEPA* matches the criterion. Now, in order to obtain bounds for the LTF synchronisation we can derive solutions with the PEPA model and with the PEPA* model.

It should be noted that, as stated, the PEPA* operator is not distributive across addition (it is associative). It is possible that a similar finesse to the one used on the PEPA operator involving apparent rates can be performed on this operator to make it distributive, however we are not aware of one.

4.5.3 Restricted MTIPP Synchronisation

Unlike PEPA, MTIPP does not have a *stable* synchronisation model; by which we mean that, for certain parameter values, it can have a mean less than or greater than the means of the LTF or FTF synchronisation (section 4.4.3). If, therefore, we require MTIPP to model the FTF or LTF synchronisation by forcing the mean of the MTIPP synchronisation to equal that of the real-world model, what restrictions does this place on MTIPP?

Thm. 2. $\mathbb{E}(X \underset{TIPP}{||} Y) = \mathbb{E}(X \underset{FTF}{||} Y)$ *iff, for* $0 < \lambda < \mu$*, both:*

$$1 < \lambda < 2 \quad \textit{and} \quad \mu = \frac{\lambda}{\lambda - 1} \textit{ such that } \infty > \mu > 2 \tag{4.28}$$

Proof. Assuming $0 < \lambda < \mu$ and thus $0 < \lambda/\mu < 1$:

$$\mathbb{E}(X \overset{||}{\text{TIPP}} Y) = \mathbb{E}(X \overset{||}{\text{FTF}} Y) \tag{4.29}$$

$$\Leftrightarrow \frac{1}{\lambda\mu} = \frac{1}{\lambda + \mu} \tag{4.30}$$

$$\Leftrightarrow \begin{cases} \mu = \frac{\lambda}{\lambda - 1} \\ 1 < \lambda < 2 \end{cases} \tag{4.31}$$

■

Thm. 3. $\mathbb{E}(X \overset{||}{\mathit{TIPP}} Y) = \mathbb{E}(X \overset{||}{\mathit{LTF}} Y)$ *iff, for* $0 < \lambda < \mu$, *both:*

$$0 < \lambda < \frac{2}{3} \quad \textit{and} \quad \mu = \frac{1}{2}\left[(1-\lambda) + \sqrt{(1-\lambda)(1+3\lambda)}\right]$$
$$\textit{such that } 1 > \mu > \frac{2}{3} \tag{4.32}$$

Proof. Assuming $0 < \lambda < \mu$ and thus $0 < \lambda/\mu < 1$:

$$\mathbb{E}(X \overset{||}{\text{TIPP}} Y) = \mathbb{E}(X \overset{||}{\text{LTF}} Y) \tag{4.33}$$

$$\Leftrightarrow \frac{1}{\lambda} + \frac{1}{\mu} - \frac{1}{\lambda + \mu} = \frac{1}{\lambda\mu} \tag{4.34}$$

$$\Leftrightarrow \frac{\lambda}{\mu} = \frac{1}{2}\left[-1 \pm \sqrt{1 + \frac{4\lambda}{1-\lambda}}\right] \tag{4.35}$$

$$\Leftrightarrow \begin{cases} \mu = \frac{1}{2}\left[(1-\lambda) + \sqrt{(1-\lambda)(1+3\lambda)}\right] \\ 0 < \lambda < \frac{2}{3} \end{cases} \tag{4.36}$$

■

So now for these specific ranges of λ (the smaller rate) the value of μ is fixed and a synchronisation can take place for which the mean behaviour is identical to that of LTF or FTF synchronisation. This is not a general panacea however, since there is no guarantee that the synchronisations in a model will have the appropriate rates. Indeed, it is easy to construct an LTF synchronisation of two legitimate LTF synchronisations which cannot then be modelled in MTIPP.

4.6 A Worked Example

In order to compare the different synchronisation models, we construct a test example (defined in equations (4.37–4.41)) which is designed to demonstrate relative performance of an LTF transition across a synchronisation. The LTF synchronisation occurs between the components U and V and is combined in Q. The component R is designed not to affect the stochastic properties of the synchronisation at all, hence the use of the passive $\top$ symbol.

$$P \stackrel{\text{def}}{=} Q \underset{\text{LTF}\{token\}}{\|} R \tag{4.37}$$

$$Q \stackrel{\text{def}}{=} U \underset{\text{LTF}\{token\}}{\|} V \tag{4.38}$$

$$U \stackrel{\text{def}}{=} (token, \alpha).U \tag{4.39}$$

$$V \stackrel{\text{def}}{=} (token, \beta).V \tag{4.40}$$

$$R \stackrel{\text{def}}{=} (token, \top).\underbrace{(ok, \gamma).R}_{R'} + (notok, \gamma).\underbrace{(token, \top).R}_{R''} \tag{4.41}$$

4.6.1 Markovian Model

In this model, we compete the evolution of the *token* with the evolution of the *notok* action (figure 4.5). If the system generates the *token* first then it also generates an *ok* action, before returning to its initial position. If it does not produce a *token*, it means that the *notok* has won the race condition; the token is subsequently executed to return the system to its initial position.

The design motivation behind this example is that, for each case, we can then compare the ratio of normal operation (a *token* followed by an *ok*) to error-handling operation (a *notok* then a *token*) in the steady-state. For each of the Markovian cases, where we assume an exponential synchronisation (figure 4.5), we solve the resulting Markov chain and obtain the relevant stationary distribution for that system. We do this for each of PEPA, MTIPP, the mean-preserving synchronisation and PEPA* in section 4.6.4.

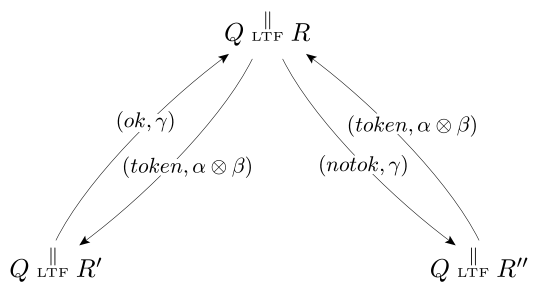

Fig. 4.5. The transition diagram for the Markovian process algebra system of equations (4.37–4.41). The transition rate of the synchronisation is expressed as $\alpha \otimes \beta$ to represent a generic MPA synchronisation rate.

4.6.2 General Model

Now we have to construct the real-life model which takes into account the non-Markovian Last-to-Finish synchronised transition. This has to be done with a deal of care since there are a number of implicit features of the Markovian model which have to be made explicit in the general case.

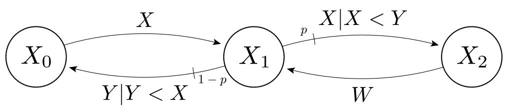

Fig. 4.6. Transition diagram for the generally distributed synchronisation, $X \sim \exp(\gamma)$ and $Y \sim \max(A, B)$ where $A \sim \exp(\alpha)$, $B \sim \exp(\beta)$. $p = \mathbb{P}(X < Y)$.

The stochastic state space is represented in figure 4.6. To identify the states of the model from now on, we use the following state mapping:

$$\begin{aligned} X_0 &\longleftrightarrow Q \overset{\parallel}{_{\text{LTF}}} R' \\ X_1 &\longleftrightarrow Q \overset{\parallel}{_{\text{LTF}}} R = P \\ X_2 &\longleftrightarrow Q \overset{\parallel}{_{\text{LTF}}} R'' \end{aligned} \tag{4.42}$$

The main issue that presents itself at this stage is whether the general model shown in figure 4.6 uses a pre-emptive restart or pre-emptive resume strategy. Stochastic process algebras have in the past used pre-emptive restart with resampling [52]. However, in the case of Markovian process algebras the distinction is a fine one since the memoryless nature of the evolutions in MPAs means that the strategies are identical stochastically. It is only when working with general distributions that the choice of pre-emptive strategy becomes important. In this case the transition from state X_2 to X_1 marked in figure 4.6 by the random variable W is the only one affected by the choice of strategy.

In either case, whether we take into account the elapsed time of a concurrent process or resample, the techniques of chapter 3 will give us a steady-state solution with: $W = Y$ for pre-emptive restart or $W = (Y|X < Y) \ominus (X|X < Y)$ for pre-emptive resume.

This latter case is exactly the same as the M/G/1/2 queue case that was studied in chapter 3.

4.6.3 Comparative Solutions

In commenting on this work, people have been divided about whether we should be comparing Markovian systems to a general solution which employs pre-emptive restart or pre-emptive resume. To be complete in our analysis, we have therefore made comparisons with both pre-emptive restart and pre-emptive resume general solutions.

Now we can define ψ_{S} (for a model S) to represent the ratio of normal operation to error-handling operation. This can be seen as the ratio of time spent in state X_0 to time spent in state X_2 (since every time the system generates an *ok* action it must have gone through X_0 to do so and similarly when the system produces a *notok* it must then go through X_2). Alternatively, we could consider the ratio of the number of times the system goes through state X_0 to the number of times it goes through X_2.

These are different quantities and will be considered separately in the following analysis. We call ψ_{S}^t the time ratio of error to correct behaviour and ψ_{S}^n the number-of-transitions ratio.

$$\psi_{\mathrm{S}}^{t} = \frac{\Pi(X_0)}{\Pi(X_2)} \tag{4.43}$$

$$\psi_{\mathrm{S}}^{n} = \frac{\Pi(X_0)}{\mathbb{E}(X)} \Bigg/ \frac{\Pi(X_2)}{\mathbb{E}(W)} \tag{4.44}$$

where W, from figure 4.6, represents the random variable which is affected by the choice of pre-emptive strategy. This gives us four general model ratios: $\psi^n_{\mathrm{RESTART}}$, ψ^n_{RESUME}, $\psi^t_{\mathrm{RESTART}}$ and ψ^t_{RESUME}.

Now for each of the above approximate Markovian models, we have a proper yardstick by which to judge their accuracy in Last-to-Finish synchronisation. The general model is solved in section 4.6.5 and the comparative results are presented in section 4.6.6.

4.6.4 Markovian Solution

We can solve this system generally for all MPAs of the MTIPP and PEPA variety. Figure 4.5 shows the transition diagram for the general Markovian model. By setting $\delta = \alpha \otimes \beta$, the synchronisation operator, we obtain the Markov chain generator matrix, G:

$$G = \begin{pmatrix} -\gamma & \gamma & 0 \\ \delta & -(\delta+\gamma) & \gamma \\ 0 & \delta & -\delta \end{pmatrix} \tag{4.45}$$

Using Gaussian elimination, we find the stationary probability vector $\Pi(X_i)$ such that $\Pi G = 0$ and $\sum_i \Pi(X_i) = 1$. We obtain:

$$\Pi(X_0) = \frac{\delta^2}{\gamma^2+\delta\gamma+\delta^2} \tag{4.46}$$

$$\Pi(X_1) = \frac{\delta\gamma}{\gamma^2+\delta\gamma+\delta^2} \tag{4.47}$$

$$\Pi(X_2) = \frac{\gamma^2}{\gamma^2+\delta\gamma+\delta^2} \tag{4.48}$$

Now from section 4.6.3:

$$\psi^t_{\text{S}} = \frac{\Pi(X_0)}{\Pi(X_2)} = \left(\frac{\delta}{\gamma}\right)^2 \tag{4.49}$$

Since in the Markovian case, $W \sim \exp(\delta)$:

$$\psi^n_{\text{S}} = \frac{\Pi(X_0)}{\mathbb{E}(X)} \bigg/ \frac{\Pi(X_2)}{\mathbb{E}(W)} = \frac{\delta}{\gamma} = \sqrt{\psi^t_{\text{S}}} \tag{4.50}$$

As mentioned before, we do not have to treat pre-emptive restart and pre-emptive resume differently for Markovian systems since they are observationally the same.

PEPA

For $\delta = \min(\alpha, \beta) = \alpha$ for $0 < \alpha < \beta$ WLOG, equations (4.49) and (4.50) give:

$$\psi^t_{\text{PEPA}} = \left(\frac{\alpha}{\gamma}\right)^2 \qquad \psi^n_{\text{PEPA}} = \frac{\alpha}{\gamma} \tag{4.51}$$

MTIPP

For $\delta = \alpha\beta$, equations (4.49) and (4.50) give:

$$\psi^t_{\text{TIPP}} = \left(\frac{\alpha\beta}{\gamma}\right)^2 \qquad \psi^n_{\text{TIPP}} = \frac{\alpha\beta}{\gamma} \tag{4.52}$$

Mean-preserving Synchronisation

For:

$$\delta = \frac{\alpha\beta(\alpha+\beta)}{(\alpha+\beta)^2 - \alpha\beta} \tag{4.53}$$

Equations (4.49) and (4.50) give:

$$\psi^t_{\text{MEAN}} = \left(\frac{\alpha\beta(\alpha+\beta)}{\gamma(\alpha^2+\alpha\beta+\beta^2)}\right)^2 \qquad \psi^n_{\text{MEAN}} = \frac{\alpha\beta(\alpha+\beta)}{\gamma(\alpha^2+\alpha\beta+\beta^2)} \tag{4.54}$$

PEPA*

For:

$$\delta = \frac{\alpha\beta}{\alpha+\beta} \tag{4.55}$$

Equations (4.49) and (4.50) give:

$$\psi^t_{\text{PEPA*}} = \left(\frac{\alpha\beta}{\gamma(\alpha+\beta)}\right)^2 \qquad \psi^n_{\text{PEPA*}} = \frac{\alpha\beta}{\gamma(\alpha+\beta)} \tag{4.56}$$

4.6.5 Analytic Solution

General Pre-emptive Resume Model

In the pre-emptive resume case, we observe that our example system has an underlying stochastic transition system of an M/G/1/2 queueing system. Thus using standard queueing techniques from [26] and [66] or our aggregation technique from chapter 3, we can obtain the precise analytical solution for the steady-state distribution. In the queueing case, we choose the general service time distribution to be the maximum of the two exponential distributions with parameters α and β, as in equation (4.8).

We obtain:

$$\Pi(X_0) = \frac{\chi}{\chi+\gamma\rho} \tag{4.57}$$

$$\Pi(X_1) = \frac{1-\chi}{\chi+\gamma\rho} \tag{4.58}$$

$$\Pi(X_2) = 1-\frac{1}{\chi+\gamma\rho} \tag{4.59}$$

where:

$$\rho = \frac{1}{\alpha}+\frac{1}{\beta}-\frac{1}{\alpha+\beta} \tag{4.60}$$

$$\chi = \frac{\alpha}{\alpha+\gamma}+\frac{\beta}{\beta+\gamma}-\frac{\alpha+\beta}{\alpha+\beta+\gamma} \tag{4.61}$$

Now, as before we can calculate ψ^t_{S} and ψ^n_{S} using equations (4.43) and (4.44):

$$\psi^t_{\text{RESUME}} = \frac{\Pi(X_0)}{\Pi(X_2)} = \frac{\chi}{\chi + \gamma\rho - 1} \tag{4.62}$$

$$\psi^n_{\text{RESUME}} = \frac{\Pi(X_0)}{\mathbb{E}(X)} \Big/ \frac{\Pi(X_2)}{\mathbb{E}(W)} = \frac{\alpha\beta(\alpha+\beta+2\gamma)}{\gamma\sigma} \tag{4.63}$$

where $\sigma = (\alpha+\beta+\gamma)^2 - \alpha\beta$.

General Pre-emptive Restart Model

Using the stochastic aggregation technique from chapter 3 with the appropriate value of W (figure 4.6) for pre-emptive restart, we obtain:

$$\Pi(X_0) = \frac{\alpha^2\beta^2(\alpha+\beta)(\alpha+\beta+2\gamma)}{\phi} \tag{4.64}$$

$$\Pi(X_1) = \frac{\alpha\beta\gamma(\alpha+\beta)\sigma}{\phi} \tag{4.65}$$

$$\Pi(X_2) = \frac{\gamma^2(\alpha^2+\alpha\beta+\beta^2)\sigma}{\phi} \tag{4.66}$$

where $\sigma = (\alpha+\beta+\gamma)^2 - \alpha\beta$ as before, and:

$$\begin{aligned}\phi &= \alpha^2\beta^2(\alpha+\beta)(\alpha+\beta+2\gamma) + \alpha\beta\gamma(\alpha+\beta)\sigma + \gamma^2(\alpha^2+\alpha\beta+\beta^2)\sigma \\ &= (\alpha^2+\alpha\beta+\beta^2)\gamma^4 + (\alpha+\beta)(2\alpha^2+3\alpha\beta+2\beta^2)\gamma^3 \\ &\quad + (\alpha^4+4\alpha^3\beta+7\alpha^2\beta^2+4\alpha\beta^3+\beta^4)\gamma^2 \\ &\quad + \alpha\beta(\alpha+\beta)(\alpha^2+3\alpha\beta+\beta^2)\gamma + \alpha^2\beta^2(\alpha+\beta)^2\end{aligned} \tag{4.67}$$

Now, as before we can calculate ψ^t_{S} and ψ^n_{S} using equations (4.43) and (4.44):

$$\psi^t_{\text{RESTART}} = \frac{\Pi(X_0)}{\Pi(X_2)} = \frac{\alpha^2\beta^2(\alpha+\beta)(\alpha+\beta+2\gamma)}{\gamma^2(\alpha^2+\alpha\beta+\beta^2)\sigma} \tag{4.68}$$

$$\psi^n_{\text{RESTART}} = \frac{\Pi(X_0)}{\mathbb{E}(X)} \Big/ \frac{\Pi(X_2)}{\mathbb{E}(W)} = \frac{\alpha\beta(\alpha+\beta+2\gamma)}{\gamma\sigma} \tag{4.69}$$

This coincidence, the fact that $\psi^n_{\text{RESTART}} = \psi^n_{\text{RESUME}}$, is explained by considering an alternative analysis of the physical meaning of the number-of-transitions ratio, defined in section 4.6.3. The quantity ψ^n_{S} returns the ratio of the number of times the system passes down one branch (X_1 to X_0) as opposed to going down the other (X_1 to X_2). This can be more simply expressed as the ratio of the probability that the system goes one way to the probability that it goes the other. Since the probabilities at this point are identical for each pre-emptive strategy, the quantities ψ^n_{RESTART} and ψ^n_{RESUME} will also be identical.

4.6.6 Model Comparisons

General Results

We are looking for general statements that can be made comparing ψ_{S} with ψ^t_{RESTART}, ψ^t_{RESUME} and[2] ψ^n_{RESTART}. The first result to note is:

$$\psi^t_{\text{RESTART}} < \psi^t_{\text{RESUME}} \tag{4.70}$$

This is because the restart strategy means that longer is spent in the error branch of the system and therefore the overall ratio of time spent doing normal operation will be lower. What does this mean when we come to make comparisons with ψ_{S} for other models? It gives us a larger margin within which to aim, when judging whether a Markovian model produces a good approximation to reality. (Figures 4.7–4.10 show various graphical comparisons). Given that it is a point of debate whether the Markovian models are trying to reproduce pre-emptive restart or pre-emptive resume strategies, it seems reasonable to compare the Markovian systems with both.

Similarly, the following properties are true of our various metrics, for all α, β, γ:

$$\psi^t_{\text{RESTART}} < \psi^t_{\text{MEAN}} \qquad \psi^t_{\text{RESTART}} < \psi^t_{\text{PEPA}} \qquad \psi^t_{\text{RESUME}} < \psi^t_{\text{PEPA}} \tag{4.71}$$

[2] Since $\psi^n_{\text{RESTART}} = \psi^n_{\text{RESUME}}$, we only consider ψ^n_{RESTART} from now on when discussing the relative number metric.

Also:

$$\psi^n_{\text{RESTART}} < \psi^n_{\text{MEAN}} \qquad \psi^n_{\text{RESTART}} < \psi^n_{\text{PEPA}} \tag{4.72}$$

The PEPA results are consistent with predictions from the expectation results of section 4.4.3. It would seem intuitive to infer that other examples which simulate Last-to-Finish synchronisation will also have similar qualitative results.

As we hoped, our mean-preserving synchronisation lies between PEPA and PEPA*, that is for all α, β, γ:

$$\psi^t_{\text{PEPA}*} < \psi^t_{\text{MEAN}} < \psi^t_{\text{PEPA}} \tag{4.73}$$

and:

$$\psi^n_{\text{PEPA}*} < \psi^n_{\text{MEAN}} < \psi^n_{\text{PEPA}} \tag{4.74}$$

However, unfortunately, it is not true to say that our suggested bounding model PEPA* always underestimates performance, since neither $\psi^t_{\text{PEPA}*}$ nor $\psi^n_{\text{PEPA}*}$ is consistently less than any of the analytically obtained ratios. In fact, it turns out that no Markovian synchronisation model can always underestimate any of ψ^t_{RESTART}, ψ^t_{RESUME} or ψ^n_{RESTART}. This statement has more general consequences and leads to:

Thm. 4. *There does not exist any Markovian synchronisation which can consistently underestimate (or equal) the correct performance model of an LTF synchronisation for all possible systems.*

Proof. We use the example defined in this section as a counterexample to the hypothesis: "There exists a Markovian synchronisation which can consistently underestimate (or equal) the correct performance model of an LTF synchronisation for all possible systems". To achieve this, we have to demonstrate a contradiction for all three of the pre-emptive restart timed, pre-emptive resume timed and pre-emptive restart number-of-transitions ratios. In fact, the proofs for each of ψ^t_{RESUME} and ψ^n_{RESTART} are identical in structure and method to the one below, where we consider the pre-emptive restart timed ratio, ψ^t_{RESTART}.

For any MPA model S, equation (4.49) gives:

$$\psi^t_{\text{S}} = \left(\frac{\delta_s(\alpha,\beta)}{\gamma}\right)^2 \tag{4.75}$$

From equation (4.68), we can derive:

$$\psi^t_{\text{RESTART}} = \left(\frac{\alpha\beta}{\gamma}\right)^2 \frac{(\alpha+\beta)(\alpha+\beta+2\gamma)}{R(\alpha,\beta,\gamma)} \tag{4.76}$$

where:

$$R(\alpha,\beta,\gamma) = (\alpha^2+\alpha\beta+\beta^2)[\gamma^2+2(\alpha+\beta)\gamma+(\alpha^2+\alpha\beta+\beta^2)] \tag{4.77}$$

We require $\psi^t_{\text{S}} < \psi^t_{\text{RESTART}}$ for all α, β, γ. So WLOG by setting $\delta_s(\alpha,\beta) = \Delta_s^{-1}(\alpha,\beta)$, we get equation (4.78).

$$\underbrace{(\alpha\beta)^2(\alpha+\beta)\Delta_s^2(\alpha,\beta)(\alpha+\beta+2\gamma)}_{(*)} - R(\alpha,\beta,\gamma) > 0 \tag{4.78}$$

However, $(*)$ can only be $O(\gamma)$, independent of the choice of $\Delta_s(\alpha,\beta)$, whereas $R(\alpha,\beta,\gamma)$ is $O(\gamma^2)$. Thus LHS of equation (4.78) < 0 for some α,β,γ.

■

So, our worked example acts as a counterexample and we have shown quite a powerful result. It shows that no Markovian process algebra can always underestimate the performance of a system. We have shown this for both pre-emptive restart and pre-emptive resume and two different definitions of performance.

This result is almost certainly a consequence of the original Markovian assumption of all MPAs.

However, all is not lost for PEPA*. Examination of the algebra indicates that, for many parameter values, PEPA* will still underestimate the correct

Tab. 4.1. Some example values of ψ^t_{RESTART} and ψ^t_{S} for various Markovian models.

α	β	γ	ψ^t_{RESTART}	$\psi^t_{\text{PEPA*}}$	ψ^t_{MEAN}	ψ^t_{PEPA}	ψ^t_{TIPP}
0.5	1.0	0.7	0.292	0.227	0.375	0.510	0.510
1.1	7.2	0.7	2.313	1.858	2.372	2.469	128.013
5.0	92.0	0.7	50.696	45.896	50.736	51.020	431,836.7
0.5	2.0	5.1	0.00410	0.00615	0.00872	0.00961	0.0384

Tab. 4.2. Some example values of ψ^t_{RESUME} and ψ^t_{S} for various Markovian models.

α	β	γ	ψ^t_{RESUME}	$\psi^t_{\text{PEPA*}}$	ψ^t_{MEAN}	ψ^t_{PEPA}	ψ^t_{TIPP}
0.5	1.0	0.7	0.338	0.227	0.375	0.510	0.510
1.1	7.2	0.7	2.406	1.858	2.372	2.469	128.013
5.0	92.0	0.7	50.981	45.896	50.736	51.020	431,836.7
0.5	2.0	5.1	0.00431	0.00615	0.00872	0.00961	0.0384

performance, as intended and as suggested by the mean properties of the PEPA* synchronisation. In the next section, the numerical results indicate that this is indeed the case.

Numerical Evidence

For specific parameter values, we can see how the performance measures vary in the stationary case (tables 4.1, 4.2 and 4.3). In the graphs presented, we concentrate on varying the smaller parameter, α, of the synchronisation relative to γ, since the exponential distribution with the smaller parameter will tend to dictate the behaviour of the LTF synchronisation.

Figures 4.7 and 4.11 show that, probably for large relative values of α and β to γ, PEPA and PEPA* still provide good bounding models for the ana-

Tab. 4.3. Some example values of $\psi^n_{\mathrm{RESTART}}$ and ψ^n_{S} for various Markovian models.

α	β	γ	$\psi^n_{\mathrm{RESTART}}$	$\psi^n_{\mathrm{PEPA*}}$	ψ^n_{MEAN}	ψ^n_{PEPA}	ψ^n_{TIPP}
0.5	1.0	0.7	0.477	0.476	0.612	0.714	0.714
1.1	7.2	0.7	1.502	1.363	1.540	1.571	11.314
5.0	92.0	0.7	7.117	6.775	7.123	7.143	657.143
0.5	2.0	5.1	0.0439	0.0784	0.0934	0.0980	0.196

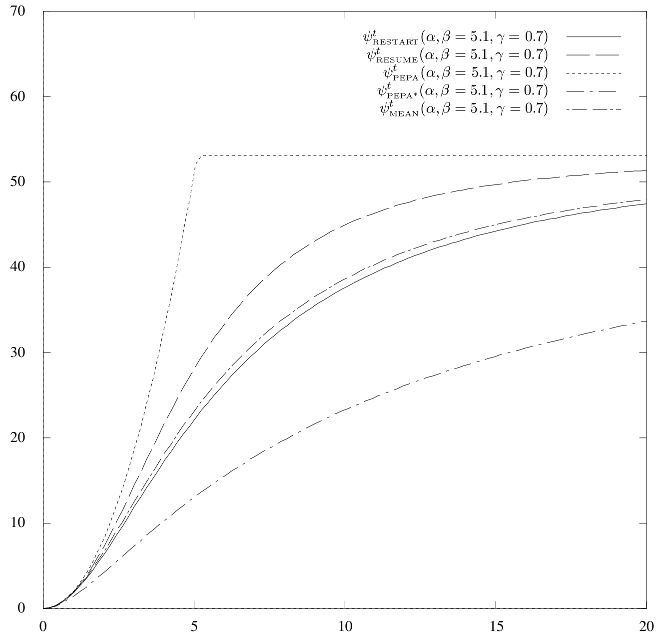

Fig. 4.7. Time ratio: PEPA and PEPA* bound the analytic solution with the mean-preserving model lying in between.

lytic result. Also, satisfyingly, the mean-preserving Markovian model comes closest to the actual solution.

Figures 4.8 and 4.12 demonstrate where PEPA* ceases to become a lower bound for $\psi^t_{\mathrm{RESTART}}$, ψ^t_{RESUME} or $\psi^n_{\mathrm{RESTART}}$ respectively. Here, all the Markovian models overestimate the performance of the system, but still shadow it closely with the result that the absolute error is small.

Figures 4.9 and 4.13 and some of the more eccentric results in table 4.1 show that MTIPP can rather overestimate the correct performance value. How-

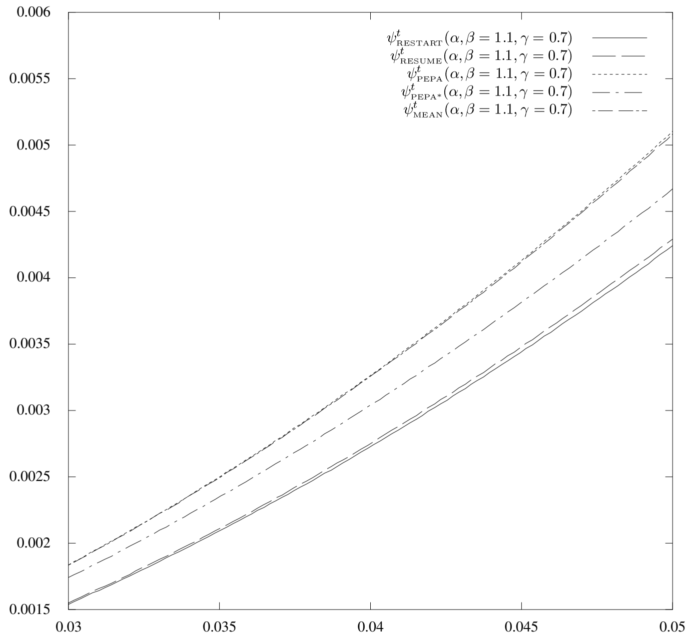

Fig. 4.8. Time ratio: all consistent MPAs overestimate the analytic solution for these low value parameters, but the absolute error is still small.

ever, if we restrict the parameters to the ranges specified in equation (4.32), MTIPP's predictive accuracy can be greatly enhanced (as demonstrated in figures 4.10 and 4.14). Not unexpectedly, in the same figures, plotting the mean-preserving Markovian case completely overlaid the MTIPP model; this confirms that MTIPP has indeed taken on the mean of the LTF synchronisation for this restricted set of parameter values.

4.7 Conclusion

For a long time, it has been understood that popular Markovian process algebras approximated their model of synchronisation in order to maintain their Markovian property. Until now, it has not been understood how this approximation affected the accuracy of their performance model. We have been able to map the effect of this approximation through to the final performance measures for a number of different Markovian process algebras. In

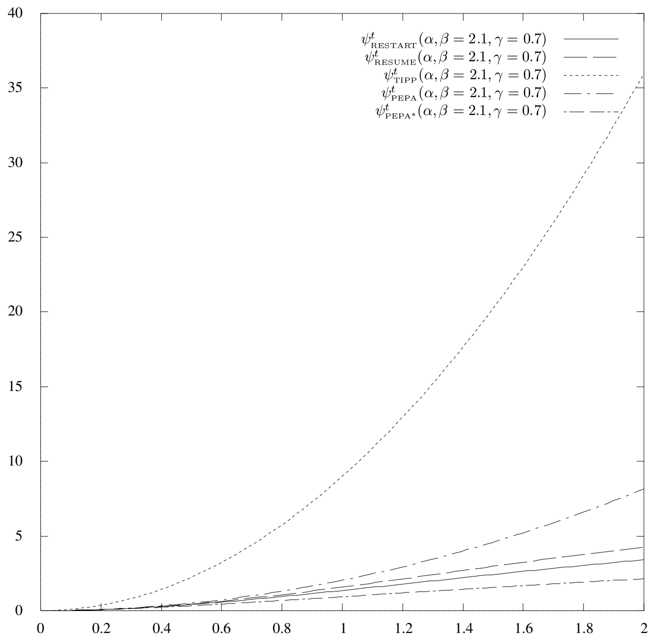

Fig. 4.9. Time ratio: MTIPP can overestimate the analytic solution considerably in some circumstances.

doing so, we have been able to demonstrate how these different MPAs can be used to gain a useful and accurate performance model for a system.

This is specifically useful for modellers using the MPA tools for PEPA and MTIPP, since the full expressiveness of the synchronisation can be used with the understanding of how reliable the resulting performance figures will be.

In developing our test model for comparing synchronisations, we were keen to produce an analytical solution which met all the possible physical interpretations of the Markovian description. Hence, we produced both solutions for pre-emptive restart and pre-emptive resume strategies, which would differ in the general case. Further, for both of these cases we calculated two different metrics for defining relative performance across synchronisation. What is striking about the results we have obtained is that qualitatively all the graphs, tables and theorems have produced identical relative results, and this gives us a significant measure of confidence in the conclusions we draw from them.

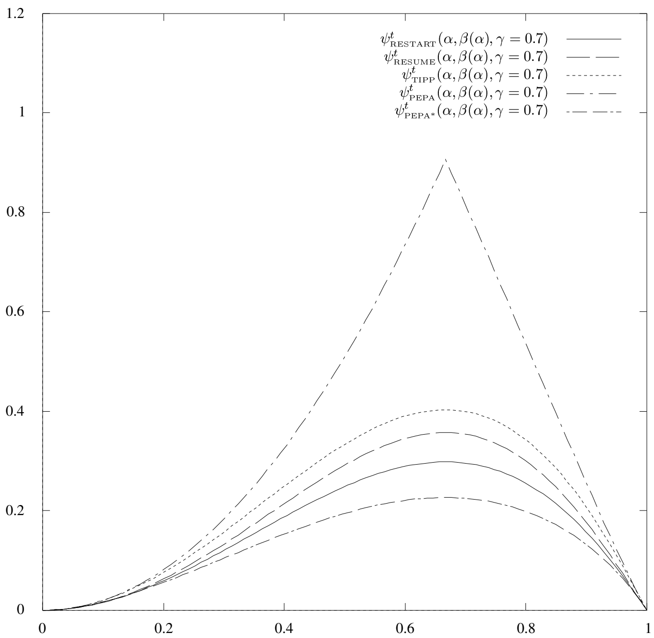

Fig. 4.10. Time ratio: if the parameter values are restricted to the ranges in equation (4.32), then MTIPP can simulate LTF synchronisation with reasonable accuracy.

4.7.1 Summary

In this chapter, we have developed real-world synchronisation models and shown how they can be described in the field of Stochastic Process Algebras. We find two forms of general synchronisation model which can usefully act as bounds to other synchronisation models, namely Last-to-Finish (LTF) and First-to-Finish (FTF) synchronisation.

Since most of the current work on SPAs involves specialising to the Markovian case, we have examined in detail how these general models of synchronisation compare to the Markovian models used in MTIPP and PEPA. It is interesting to note that most stochastic process algebras implement a form of Last-to-Finish synchronisation; only Stark's Probabilistic I/O Automata [92] (a Markovian process algebra), to our knowledge, uses a First-to-Finish strategy. From what we have seen, no stochastic process algebra implements multiple synchronisation strategies, which would clearly be appropriate for sufficient

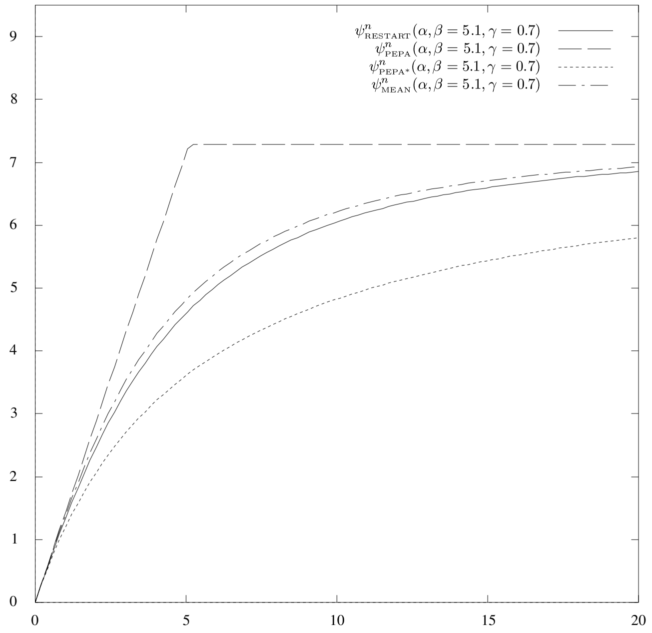

Fig. 4.11. Number-of-transitions ratio: PEPA and PEPA* bound the analytic solution with the mean-preserving model lying in between.

modelling expressiveness. We rectify this situation in chapter 5, where we define a stochastic process algebra with both FTF and LTF synchronisation models.

We have investigated three ways of improving the accuracy of predicted performance in Markovian process algebras:

Mean-preserving Synchronisation By attempting to replicate the mean of an individual general LTF synchronisation as an exponential distribution, we hope to improve the accuracy of the long term performance model. The numerical evidence suggests that this is the best of all the MPA models in consistently coming closest to an accurate solution. This is not perhaps surprising, as this strategy of replacing a general distribution by an exponential with the same mean is used in conjunction with insensitivity in generally-distributed systems (discussed in chapter 3). Although there are usually specific conditions for using this technique so that it preserves the steady-state distribution, we have shown that even when these conditions are not met, it still provides a very good approximation to the correct solution.

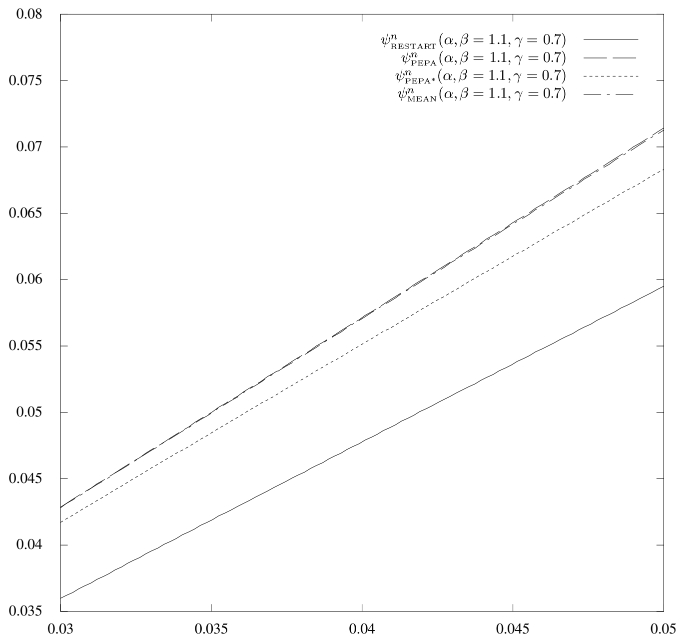

Fig. 4.12. Number-of-transitions ratio: all consistent MPAs overestimate the analytic solution for these low value parameters, but the absolute error is still small.

Bounded PEPA Modelling We make use of the fact that PEPA consistently overestimates model performance and attempt to define a dual model, PEPA*, which similarly underestimates the LTF model. This was very successful for frequent occurrences of a synchronised event (large values of ψ_{S}), where PEPA and PEPA* bounded the correct solution almost equally. However for rare synchronised events (ψ_{S} small), both PEPA and PEPA* overestimated the performance measure, but in both cases only by very small absolute values.

Restricted MTIPP Synchronisation By restricting MTIPP's *scaling factor* concept to specific ranges of parameters we can emulate LTF or FTF synchronisation very effectively. In the case of LTF synchronisation, we obtain as good a set of results with restricted MTIPP as we do with the mean-preserving model.

We have also been able to show that it is impossible to design a Markovian synchronisation which consistently underestimates or equals the performance

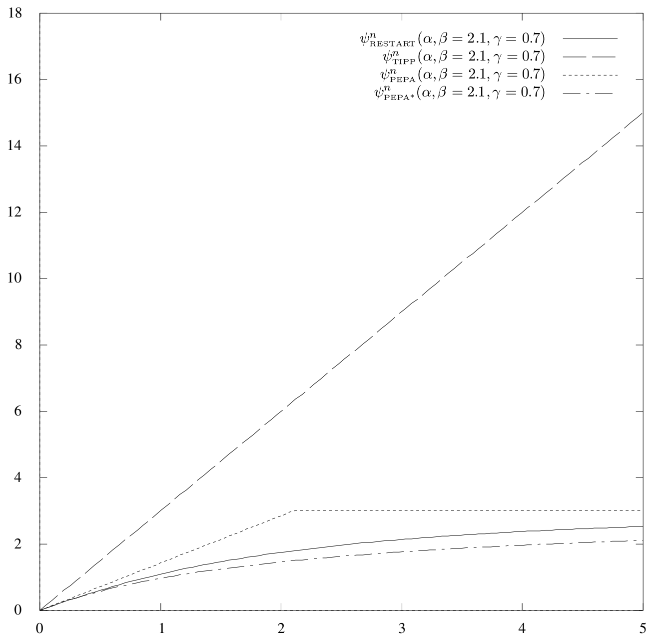

Fig. 4.13. Number-of-transitions ratio: MTIPP can overestimate the analytic solution considerably in some circumstances.

measure of the correct LTF synchronisation for all systems. However, despite this, we have defined PEPA*, which underestimates for a large proportion of the parameter space, and this may have useful implications.

PEPA

A good engineering solution to the performance modelling problem could well be to use a combination of PEPA, PEPA* and the mean-preserving model: PEPA and PEPA* to bound the correct model (in the right circumstances) and the mean-preserving model to compare to the average of the PEPA and PEPA* predictions. For the first time, it will also be possible to define a variance for the accuracy of the performance measures using these three values. This measure of model reliability could be very useful for designers of real-time systems or safety-critical systems who need to know how well they can trust their performance modelling paradigm.

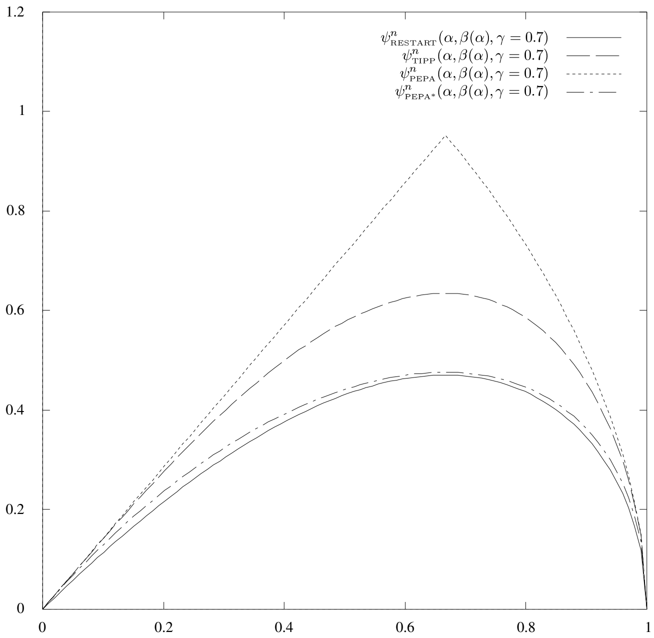

Fig. 4.14. Number-of-transitions ratio: if the parameter values are restricted to the ranges in equation (4.32), then MTIPP can simulate LTF synchronisation with reasonable accuracy.

MTIPP

It is clear from our analysis and results that MTIPP was not really designed to model LTF synchronisation, and if not used carefully can produce startling performance results. The restricted MTIPP model (section 4.5.3) gives a very good approximation but may well turn out to be too restrictive to use in practice, given that it effectively reduces two dimensions of freedom in setting a synchronisation model to one.

4.7.2 General Remarks

Markovian process algebras are fairly severe modelling approximations to reality. It has been assumed in this chapter, as with the MPAs themselves, that the atomic actions are in fact Markovian. In analytical terms, clearly a generally-distributed process algebra would be an obvious solution, however there is a paucity of techniques and tools for obtaining performance figures for large-scale systems. In the absence of such techniques and while Markovian process algebra research into faster solution techniques continues to flourish, an understanding of how to use MPAs in a reliable way can only be a good thing.

Annex

4.A Comparative Stochastic Properties

In this annex, comparisons are made between the different synchronisation methods. This provides the analytical background for the results in section 4.4.3. We demonstrate what restrictions (if any) are imposed on the parameter space for certain properties to be true. In the proofs, only meaningful roots of quadratic equations are presented. In all the theorems stated below $X \sim \exp(\lambda)$, $Y \sim \exp(\mu)$ and $0 < \lambda < \mu$:

4.A.1 PEPA vs First-to-Finish

Thm. 5. $\mathbb{E}(X \overset{||}{_{FTF}} Y) < \mathbb{E}(X \overset{||}{_{PEPA}} Y)$ *for all* $0 < \lambda < \mu$.

Proof. Assuming $0 < \lambda < \mu$:

$$\mathbb{E}(X \overset{||}{_{\text{FTF}}} Y) < \mathbb{E}(X \overset{||}{_{\text{PEPA}}} Y) \tag{4.79}$$

$$\Leftrightarrow \frac{1}{\lambda + \mu} < \frac{1}{\lambda} \tag{4.80}$$

$$\Leftrightarrow \frac{1}{\lambda} > 0 \tag{4.81}$$

■

Thm. 6. $\mathbf{Var}(X \overset{||}{_{FTF}} Y) < \mathbf{Var}(X \overset{||}{_{PEPA}} Y)$ *for all* $0 < \lambda < \mu$.

Proof. Assuming $0 < \lambda < \mu$:

$$\mathbf{Var}(X \overset{||}{_{\text{FTF}}} Y) < \mathbf{Var}(X \overset{||}{_{\text{PEPA}}} Y) \tag{4.82}$$

$$\Leftrightarrow \frac{1}{(\lambda + \mu)^2} < \frac{1}{\lambda^2} \tag{4.83}$$

$$\Leftrightarrow \mu(\mu + 2\lambda) > 0 \tag{4.84}$$

$$\Leftrightarrow \mu > 0 \tag{4.85}$$

■

4.A.2 MTIPP vs First-to-Finish

Thm. 7. $\mathbb{E}(X \overset{\|}{_{FTF}} Y) < \mathbb{E}(X \overset{\|}{_{TIPP}} Y)$ *for* $0 < \lambda < \mu$*, iff:*

$$0 < \lambda < \frac{\lambda}{\mu} + 1 \tag{4.86}$$

Proof. Assuming $0 < \lambda < \mu$ and thus $0 < \lambda/\mu < 1$:

$$\mathbb{E}(X \overset{\|}{_{\text{FTF}}} Y) < \mathbb{E}(X \overset{\|}{_{\text{TIPP}}} Y) \tag{4.87}$$

$$\Leftrightarrow \frac{1}{\lambda + \mu} < \frac{1}{\lambda\mu} \tag{4.88}$$

$$\Leftrightarrow \lambda < \frac{\lambda}{\mu} + 1 \tag{4.89}$$

■

Thm. 8. $\mathbf{Var}(X \overset{\|}{_{FTF}} Y) < \mathbf{Var}(X \overset{\|}{_{TIPP}} Y)$ *for* $0 < \lambda < \mu$*, iff:*

$$0 < \lambda < \frac{\lambda}{\mu} + 1 \tag{4.90}$$

Proof. Assuming $0 < \lambda < \mu$ and using $\delta = \lambda/\mu$ so that $0 < \delta < 1$:

$$\mathbf{Var}(X \overset{\|}{_{\text{FTF}}} Y) < \mathbf{Var}(X \overset{\|}{_{\text{TIPP}}} Y) \tag{4.91}$$

$$\Leftrightarrow \frac{1}{(\lambda + \mu)^2} < \frac{1}{(\lambda\mu)^2} \tag{4.92}$$

$$\Leftrightarrow \delta^2 + 2\delta + (1 - \lambda^2) > 0 \tag{4.93}$$

$$\Leftrightarrow \delta > -1 + \lambda \tag{4.94}$$

■

4.A.3 PEPA vs Last-to-Finish

Thm. 9. $\mathbb{E}(X \overset{||}{_{PEPA}} Y) < \mathbb{E}(X \overset{||}{_{LTF}} Y)$ *for all* $0 < \lambda < \mu$.

Proof. Assuming $0 < \lambda < \mu$:

$$\mathbb{E}(X \overset{||}{_{\text{PEPA}}} Y) < \mathbb{E}(X \overset{||}{_{\text{LTF}}} Y) \tag{4.95}$$
$$\Leftrightarrow \frac{1}{\lambda} + \frac{1}{\mu} - \frac{1}{\lambda+\mu} > \frac{1}{\lambda} \tag{4.96}$$
$$\Leftrightarrow \lambda > 0 \tag{4.97}$$

■

Thm. 10. $\mathbf{Var}(X \overset{||}{_{LTF}} Y) < \mathbf{Var}(X \overset{||}{_{PEPA}} Y)$ *for* $0 < \lambda < \mu$, *iff:*

$$0 < \frac{\lambda}{\mu} < \sqrt{3} - 1 \tag{4.98}$$

Proof. Assuming $0 < \lambda < \mu$ and $\delta = \lambda/\mu$ so that $0 < \delta < 1$:

$$\mathbf{Var}(X \overset{||}{_{\text{LTF}}} Y) < \mathbf{Var}(X \overset{||}{_{\text{PEPA}}} Y) \tag{4.99}$$
$$\Leftrightarrow \frac{1}{\lambda^2} + \frac{1}{\mu^2} - \frac{3}{(\lambda+\mu)^2} < \frac{1}{\lambda^2} \tag{4.100}$$
$$\Leftrightarrow (\lambda+\mu)^2 - 3\mu^2 < 0 \tag{4.101}$$
$$\Leftrightarrow 0 < \delta < \sqrt{3} - 1 \tag{4.102}$$

■

4.A.4 MTIPP vs Last-to-Finish

Thm. 11. $\mathbb{E}(X \overset{||}{_{LTF}} Y) < \mathbb{E}(X \overset{||}{_{TIPP}} Y)$ *for* $0 < \lambda < \mu$, *iff both:*

$$0 < \lambda < \frac{2}{3} \quad \textit{and} \quad \frac{1}{2}\left[-1 + \sqrt{\frac{1+3\lambda}{1-\lambda}}\right] < \frac{\lambda}{\mu} < 1 \tag{4.103}$$

Proof. Assuming $0 < \lambda < \mu$ and using $\delta = \lambda/\mu$ so that $0 < \delta < 1$:

$$\mathbb{E}(X \mathbin{\underset{\text{LTF}}{\|}} Y) < \mathbb{E}(X \mathbin{\underset{\text{TIPP}}{\|}} Y) \tag{4.104}$$

$$\Leftrightarrow \frac{1}{\lambda} + \frac{1}{\mu} - \frac{1}{\lambda+\mu} < \frac{1}{\lambda\mu} \tag{4.105}$$

$$\Leftrightarrow (\lambda - 1)\delta^2 + (\lambda - 1)\delta + \lambda < 0 \tag{4.106}$$

$$\Leftrightarrow \delta > \frac{1}{2}\left[-1 + \sqrt{\frac{1+3\lambda}{1-\lambda}}\right] \tag{4.107}$$

However, this root of δ is only meaningful for $\delta < 1$, so combining the two inequalities gives the additional condition, $0 < \lambda < 2/3$.

■

Thm. 12. $\mathbf{Var}(X \mathbin{\underset{\mathit{LTF}}{\|}} Y) < \mathbf{Var}(X \mathbin{\underset{\mathit{TIPP}}{\|}} Y)$ *for* $0 < \lambda < \mu$, *iff both:*

$$0 < \lambda\mu < \frac{4}{5} \quad \textit{and} \quad 4 < \gamma < \frac{(2\beta+1) + \sqrt{16\beta^2 + 4\beta + 1}}{2\beta} \tag{4.108}$$

where $\gamma = (\lambda+\mu)^2/\beta$, $\beta = \lambda\mu$.

Proof. Using $\delta = \lambda/\mu$, $\alpha = (\lambda+\mu)^2$, $\beta = \lambda\mu$, $\gamma = \alpha/\beta = \delta + 2 + \delta^{-1}$ and assuming $0 < \lambda < \mu$ and thus $0 < \delta < 1$ and $\gamma > 4$:

$$\mathbf{Var}(X \mathbin{\underset{\text{LTF}}{\|}} Y) < \mathbf{Var}(X \mathbin{\underset{\text{TIPP}}{\|}} Y) \tag{4.109}$$

$$\Leftrightarrow \frac{1}{\lambda^2} + \frac{1}{\mu^2} - \frac{3}{(\lambda+\mu)^2} < \frac{1}{(\lambda\mu)^2} \tag{4.110}$$

$$\Leftrightarrow \alpha^2 - 2\alpha\beta - 3\beta^2 - \alpha < 0 \tag{4.111}$$

$$\Leftrightarrow \gamma^2 - \frac{2\beta+1}{\beta}\gamma - 3 < 0 \tag{4.112}$$

$$\Leftrightarrow \gamma < \frac{(2\beta+1) + \sqrt{16\beta^2 + 4\beta + 1}}{2\beta} \tag{4.113}$$

However, this root of γ is only meaningful for $\gamma > 4$, so combining the two inequalities gives the additional condition, $0 < \beta < 4/5$.

■

4.A.5 MTIPP vs PEPA

Thm. 13. $\mathbb{E}(X \underset{TIPP}{\parallel} Y) < \mathbb{E}(X \underset{PEPA}{\parallel} Y)$ *for* $0 < \lambda < \mu$, *iff* $\mu > 1$.

Proof. Assuming $0 < \lambda < \mu$:

$$\mathbb{E}(X \underset{\text{TIPP}}{\parallel} Y < \mathbb{E}(X \underset{\text{PEPA}}{\parallel} Y) \tag{4.114}$$

$$\Leftrightarrow \frac{1}{\lambda\mu} < \frac{1}{\lambda} \tag{4.115}$$

$$\Leftrightarrow \mu > 1 \tag{4.116}$$

■

Thm. 14. $\mathbf{Var}(X \underset{TIPP}{\parallel} Y) < \mathbf{Var}(X \underset{PEPA}{\parallel} Y)$ *for* $0 < \lambda < \mu$, *iff* $\mu > 1$.

Proof. Assuming $0 < \lambda < \mu$:

$$\mathbf{Var}(X \underset{\text{TIPP}}{\parallel} Y) < \mathbf{Var}(X \underset{\text{PEPA}}{\parallel} Y) \tag{4.117}$$

$$\Leftrightarrow \frac{1}{(\lambda\mu)^2} < \frac{1}{\lambda^2} \tag{4.118}$$

$$\Leftrightarrow \mu > 1 \tag{4.119}$$

■

Chapter 5

Reliability of Models in SPAs

5.1 Introduction

Reliability is often seen as a cut-and-dried property. A computer model is proposed, a reliability criterion is defined and the model is analysed against that criterion to see whether it is or is not reliable. By defining a system with a stochastic process algebra we have the opportunity to model the uncertainty that, in reality, pervades today's communication networks and distributed systems.

In this chapter, we concentrate on reliable models rather than the paradigm itself. What seriously hinders stochastic modelling and simulation today is the explosion in variance that can easily overtake a model and make it analytically useless. If a system has large variations in temporal behaviour, these will impinge on other systems with which it communicates, causing them in turn to be unpredictable (if they are not specifically designed to expect high variance communication). The whole system will take longer to converge to any sort of equilibrium behaviour, if it ever converges at all. This has a direct corollary: it will be longer before any steady-state analysis has any bearing on physical reality.

Systems which have small variation are therefore much easier to use from a safety-critical point of view—their behaviour is nicely bounded and will not cause other components to behave erratically.

Our brief in this chapter, then, is to investigate variance-management techniques[1] at the design stage of a system from a stochastic process algebra perspective. We construct a simple stochastic process algebra which embodies the properties that we have identified as being important to modelling expressively and reliably (using multiple synchronisation types and modelling with general distributions). We develop a simple framework for interpreting SPAs in terms of reliable behaviour and variance. We go on to demonstrate how this property changes under combination of the SPA operators.

By targeting variance reduction in a modelling environment such as stochastic process algebras, we aim to give the modeller a set of tools by which they can know how to reduce variance and increase global system predictability. We go some way towards that goal in this chapter, but also demonstrate some of the issues that make it difficult to achieve in general.

We finish by analysing two examples in terms of reliability: a searching problem (section 5.5.1 and originally presented in [13]) and a model of a telephone communication system (section 5.5.2). Finally, we explore what can currently be achieved with reliable modelling techniques and SPAs, informed by the results of this chapter.

5.2 Definitions of Reliability

As we have seen in chapter 2, from an SPA point-of-view reliability is a slightly alien concept. It is generally considered a hard enough problem to

[1]This typically, but not always, means reducing variance. In the event that variance needs to be increased in a system, we will see that this can be achieved easily.

obtain average behaviour for a stochastically defined system without having to worry about obtaining other measures. As a result, it is difficult to see how reliability could be extracted from current solution techniques which are centred around finding steady-state probabilities.

In a performance context, knowing how much a system can vary from its mean performance is an essential practical consideration. Or indeed, to put it the other way round, if the variation of behaviour is very large then having only the mean figure for a sojourn is probably not much use for analysis.

So, how to define reliability in the context of a stochastically defined system? If we consider a finite path in a stochastic transition system, we can aggregate the random variables that make up the path and obtain a distribution for the entire path [12]. This distribution will have a mean, a variance and all the higher moments; as a starting point the variance will be a useful reliability statistic to gather. A path with large variance will slow down a system's convergence to stationary behaviour, whereas a path with small variance will make the system more predictable (or deterministic).

However, if we are able to derive the distribution completely for such a path, this represents a better measure of the path's reliability. If the distribution were asymmetric, for instance, this would be evident from the distribution but not reflected in the variance.

In the course of this chapter, we will look at both aggregate distributions and variances. We will discover that general statements about reliability can be made according to how systems are constructed. In particular, the type of operator and modelling strategy used can dramatically affect the predicted reliability of a system.

5.2.1 Real-Time Systems

For systems for which reliability is important, average long-term behaviour is not the primary issue: hazards may occur or functionality may be at risk if individual instances of operation fall outside certain bounds. We want to be able to reason about local or transient behaviour as well as operation in the long term.

For instance, if we have a time-out race between an exponentially distributed action, $X \sim \exp(10)$, and a deterministic cut-off, $Y \sim \delta(10)$, if the time-out, Y, occurs first the system might well go into some kind of failure mode or error handler or may just deadlock. We need to be able to ascertain the probability of the system going into such a state every time it goes into this time-out:

$$P(X > Y) = P(X > 10) = \overline{F}_X(10) = e^{-100} \tag{5.1}$$

Such vanishingly small probabilities might form the basis of a safety case or an industrial standard for a component or hazard situation.

In MTL [72] (a real-time modelling algebra), typically, such local behaviour is looked at by examining the interval between two events where each event in the system is given a fixed duration. Similarly in TIS [90] (another real-time modelling language), for example, specific amounts of time are allocated to actions of a model. Verification takes the form of asking whether intervals of operation take less than or greater than a hard bound.

In a stochastic system we can use the notion of stochastic aggregation [12] to produce a similar, but stochastically defined, result. Given a distribution for a particular interval, how might reliability be defined on it? Various TIS papers [90, 89, 100] have a categorisation of reliable probabilistic operation, which can basically be summarised into two categories:

1. **One-tailed test:** $P(X < a) = p$ or $P(X > b) = q$
2. **Two-tailed test:** $P(a < X < b) = p$

where a, b, p and q would be specified for necessary reliable behaviour. These are easily attainable from the cumulative distribution function of an aggregate distribution. However, given that TIS itself specifies exact durations, it is not able to obtain these quantities on a given path—it has to settle for obtaining deterministic or hard bounds on path execution.

5.2.2 Variance in Stochastic Transition Systems

A second idea is to use the aggregation method again, but this time calculate the variance and then, easily, the coefficient of variation, C_X:

$$C_X = \frac{\sqrt{\mathbf{Var}(X)}}{\mathbb{E}(X)} \tag{5.2}$$

This is particularly attractive since it not only gives an idea of how "concentrated" or "diffuse" a distribution might be (which is a good measure of determinism) but also allows us to aggregate in Laplace-space and generate the moments from the Laplace transform of the aggregate distribution [97]. Thus an inversion of a Laplace transform and an integration to the cumulative distribution function are avoided.

Let us consider the following stochastic transition system (similar to that from chapter 3) where each transition has a random variable, X, attributed to its execution:

$$P \quad ::= \quad \mathbf{0} \,\Big|\, \{X\}.P \,\Big|\, [m]P \oplus [1-m]P \tag{5.3}$$

This represents a stochastic component where either transitions occur, taking a random amount of time, or a probabilistic branch occurs with probability m for one path and $(1-m)$ for the other.

Just by considering the variance of certain stochastic paths, we can gain some interesting insight. Take the following reliability function, V_0, which measures the variance of a terminating path, $V_0 : P \longrightarrow \mathbb{R}$:

$$V_0(\mathbf{0}) = 0 \tag{5.4}$$

$$V_0(\{X\}.P) = \mathbf{Var}(X) + V_0(P) \tag{5.5}$$

$$\begin{aligned} V_0([m]P \oplus [1-m]Q) = {} & mV_0(P) + (1-m)V_0(Q) \\ & + m(1-m)(E_0(P) - E_0(Q))^2 \end{aligned} \tag{5.6}$$

where $E_0 : P \longrightarrow \mathbb{R}$ represents the mean time of path execution:

$$E_0(\mathbf{0}) = 0 \tag{5.7}$$

$$E_0(\{X\}.P) = \mathbb{E}(X) + E_0(P) \tag{5.8}$$

$$E_0([m]P \oplus [1-m]Q) = mE_0(P) + (1-m)E_0(Q) \tag{5.9}$$

where $\mathbb{E}(X)$ is the mean, as before, and $\mathbf{Var}(X)$ is the variance of a random variable X. Equation (5.5) is essentially a result of the convolution formula and shows that a finite system which is described by nothing but a sequence of events will have a monotonically increasing variance. Equation (5.6) represents the formula for a probabilistic branch [97] and shows that this is where variance reduction might be achieved within a stochastic system.

This is the premise of the chapter: many systems are largely sequential and this leads to the variance explosion problem, when modelling or simulating systems. The fact that probabilistic branching (and thus other constructs like competitive choice and synchronisations, which, from chapter 3, we know can be expressed in terms of probabilistic branching [16]) can reduce variance, means that there is a possibility of reducing this explosion. We will demonstrate that this is so in the next section.

5.3 A General Stochastic Process Algebra

5.3.1 Definition

If we are to take a process algebra perspective on reliability in models of communicating systems, then we clearly need a process algebra to work on. As we have seen from our discussion of chapter 3, traditional analysis of generally distributed stochastic process algebras (obtaining stationary distributions) can prove very difficult. However, finding stationary distributions is of limited usefulness from a reliability modelling point of view.

This gives us an opportunity to define other analysis techniques on a generally distributed system which we can make relevant to reliability modelling and hopefully tractable at the same time.

Define the following system, for $p = 1 - q$, $0 < p, q < 1$:

$$\begin{aligned} P \quad ::= \quad & \mathbf{0} \;\Big|\; X \;\Big|\; (a, F_A(t)).P \;\Big|\; [p]P \oplus [q]P \\ & \Big|\; P + P \;\Big|\; P \underset{\mathrm{FTF}_S}{\parallel} P \;\Big|\; P \underset{\mathrm{LTF}_S}{\parallel} P \end{aligned} \tag{5.10}$$

This is just an approach with which to put our ideas in a stochastic process algebra context, it is not intended to be a complete modelling algebra, at this stage of development.

Defining the component prefix using the cumulative distribution function comes from the original TIPP paper [37]. A cumulative distribution function is one way of uniquely defining a distribution, whereas a probability distribution function can have a countably infinite number of values at which it differs from a second probability distribution function and they will still represent the same stochastic event.

We also require the Laplacian path function $L_1 : P \longrightarrow 2^{\mathsf{LAP}}$ for LAP, a Laplacian function type, defined in figure 5.1 and 5.2. It is completely defined for the simpler operators in figure 5.1. For simplicity L_1 is only defined on active synchronisations in figure 5.2 as we shall only require single evolution results in section 5.3.2.

Using L_1, we will demonstrate, in our path variance definition of V_1, how the higher level combinators (First-to-Finish and Last-to-Finish synchronisation and competitive choice) can be rewritten stochastically in terms of probabilistic branching.

5.3.2 Variance Metric under SPA Combination

We define V_1 just as we did for V_0 on the stochastic transition system of section 5.2.2. $V_1 : P \longrightarrow \mathbb{R}$ to represent the combined variance of a path

Nil

$$L_1(\mathbf{0}) = 1$$

Prefix

$$\begin{aligned} & L_1((a, F_X(t)).E) \\ &= L_X(z) L_1(E) \end{aligned}$$

Probabilistic Branching

$$\begin{aligned} & L_1([p]E \oplus [1-p]F) \\ &= pL_1(E) + (1-p)L_1(F) \end{aligned}$$

Competitive Choice

$$\begin{aligned} & L_1((a, F_X(t)).E + (b, F_Y(t)).F) \\ &= L_1([\mathbb{P}(X < Y)](a, F_{X|X<Y}(t)).E \\ &\qquad \oplus [\mathbb{P}(Y < X)](b, F_{Y|Y<X}(t)).F) \end{aligned}$$

Fig. 5.1. The Laplacian path function definition, L_1, for sequential and choice combinators.

of the stochastic system. We use the definition of L_1 to obtain the Laplace transform of the distribution of the effect of each combinator in each case. We now attempt to get a general expansion for the path variance, V_1, for each combinator.

Let $V_1(\mathbf{0}) = 0$ and $V_1((x, F_X(t)).\mathbf{0}) = \mathbf{Var}(X)$ in the analysis below.

First-to-Finish Synchronisation

$$\begin{aligned} & L_1((a, F_X(t)).E \underset{\text{FTF}_S}{\parallel} (a, F_Y(t)).F) \\ = \quad & L_1((a, F_Z(t)).(E \underset{\text{FTF}_S}{\parallel} F)) \quad \text{if } a \in S \text{ where} \end{aligned}$$

$$L_Z(z) = L_1((a, F_X(t)).\mathbf{0} + (b, F_Y(t)).\mathbf{0})$$

Last-to-Finish Synchronisation

$$\begin{aligned} & L_1((a, F_X(t)).E \underset{\text{LTF}_S}{\parallel} (a, F_Y(t)).F) \\ = \quad & L_1((a, F_Z(t)).(E \underset{\text{LTF}_S}{\parallel} F)) \quad \text{if } a \in S \text{ where} \end{aligned}$$

$$\begin{aligned} L_Z(z) = L_1([\mathbb{P}(X > Y)](a, F_{X|X>Y}(t)).\mathbf{0} \\ \oplus [\mathbb{P}(Y > X)](b, F_{Y|Y>X}(t)).\mathbf{0}) \end{aligned}$$

Fig. 5.2. The Laplacian path function definition, L_1, for synchronisation combinators.

Prefix

For a path $A \stackrel{\text{def}}{=} (a, F_X(t)).(b, F_Y(t)).\mathbf{0}$:

$$V_1(A) = V_1((a, F_X(t)).(b, F_Y(t))) \tag{5.11}$$

This is a convolution operation on the random variables X and Y which make up the prefix. Selecting our convolved variable W such that:

$$L_W(z) = L_1(A) = L_X(z)L_Y(z) \tag{5.12}$$

such that:

$$
\begin{aligned}
\mathbb{E}(W) &= -\frac{\mathrm{d}}{\mathrm{d}z} L_X(z) L_Y(z) \Big|_{z=0} \\
&= -L_Y(0) \frac{\mathrm{d}}{\mathrm{d}z} L_X(z) \Big|_{z=0} - L_X(0) \frac{\mathrm{d}}{\mathrm{d}z} L_Y(z) \Big|_{z=0} \\
&= L_Y(0)\mathbb{E}(X) + L_X(0)\mathbb{E}(Y) \\
&= \mathbb{E}(X) + \mathbb{E}(Y) \qquad (5.13)
\end{aligned}
$$

since $L_X(0) = \int_0^\infty f_X(t)\,\mathrm{d}t = 1$ for a probability distribution function of a random variable, X. Now similarly:

$$
\begin{aligned}
\mathbb{E}(W^2) &= \frac{\mathrm{d}^2}{\mathrm{d}z^2} L_X(z) L_Y(z) \Big|_{z=0} \\
&= L_Y(0) \frac{\mathrm{d}^2}{\mathrm{d}z^2} L_X(z) \Big|_{z=0} + 2 \frac{\mathrm{d}}{\mathrm{d}z} L_X(z) \Big|_{z=0} \frac{\mathrm{d}}{\mathrm{d}z} L_Y(z) \Big|_{z=0} \\
&\quad + L_X(0) \frac{\mathrm{d}^2}{\mathrm{d}z^2} L_Y(z) \Big|_{z=0} \\
&= \mathbb{E}(X^2) + 2\mathbb{E}(X)\mathbb{E}(Y) + \mathbb{E}(Y^2) \qquad (5.14)
\end{aligned}
$$

So:

$$
\begin{aligned}
\mathbf{Var}(W) &= \mathbb{E}(X^2) + 2\mathbb{E}(X)\mathbb{E}(Y) + \mathbb{E}(Y^2) - (\mathbb{E}(X) + \mathbb{E}(Y))^2 \\
&= (\mathbb{E}(X^2) - \mathbb{E}^2(X)) + (\mathbb{E}(Y^2) - \mathbb{E}^2(Y)) \\
&= \mathbf{Var}(X) + \mathbf{Var}(Y) \qquad (5.15)
\end{aligned}
$$

Now we can define our variance metric across a process algebra prefix operation:

$$
\begin{aligned}
V_1(A) &= V_1((a, F_X(t)).(b, F_Y(t)).\mathbf{0}) \\
&= V_1((a, F_X(t))) + V_1((b, F_Y(t))) \\
&= \mathbf{Var}(X) + \mathbf{Var}(Y) \qquad (5.16)
\end{aligned}
$$

which justifies the original statement of the result in V_0 of equation 5.5.

Probabilistic Branching

For a path $A \stackrel{\text{def}}{=} [p](a, F_X(t)).\mathbf{0} \oplus [q](b, F_Y(t)).\mathbf{0}$, where W represents the stochastic aggregation of the system A:

$$L_W(z) = L_1(A) = pL_X(t) + qL_Y(t) \tag{5.17}$$

$$\mathbb{E}(W) = p\mathbb{E}(X) + q\mathbb{E}(Y) \tag{5.18}$$

$$\mathbb{E}(W^2) = p\mathbb{E}(X^2) + q\mathbb{E}(Y^2) \tag{5.19}$$

$$\begin{aligned}
\mathbf{Var}(W) &= p\mathbb{E}(X^2) + q\mathbb{E}(Y^2) - p^2\mathbb{E}^2(X) - q^2\mathbb{E}(Y) \\
&\quad - 2pq\mathbb{E}(X)\mathbb{E}(Y) \\
&= p\mathbf{Var}(X) + pq\mathbb{E}^2(X) + q\mathbf{Var}(Y) + pq\mathbb{E}^2(Y) \\
&\quad - 2pq\mathbb{E}(X)\mathbb{E}(Y) \\
&= p\mathbf{Var}(X) + q\mathbf{Var}(Y) + pq(\mathbb{E}(X) - \mathbb{E}(Y))^2 \qquad (5.20)
\end{aligned}$$

So we define V_1 over probabilistic branching as follows:

$$\begin{aligned}
V_1(A) &= V_1([p](a, F_X(t)).\mathbf{0} \oplus [q](b, F_Y(t)).\mathbf{0}) \\
&= p\mathbf{Var}(X) + q\mathbf{Var}(Y) + pq(\mathbb{E}(X) - \mathbb{E}(Y))^2 \qquad (5.21)
\end{aligned}$$

which again justifies V_0, equation (5.6).

Competitive Choice and First-to-Finish Synchronisation

From the definition of L_1, we know that, stochastically speaking, competitive choice and First-to-Finish synchronisation are the same. They, of course, differ in operational terms since, as the first process terminates, the competitive choice operator terminates the slower process whereas the synchronisation operator pre-empts the other process but thereafter allows it to continue. In both cases, though, the time taken is equal to that of the faster process.

Thus for:

$$A \stackrel{\text{def}}{=} (a, F_X(t)).\mathbf{0} + (b, F_Y(t)).\mathbf{0} \tag{5.22}$$

$$B \stackrel{\text{def}}{=} (a, F_X(t)).\mathbf{0} \underset{\text{FTF}\{a\}}{||} (a, F_Y(t)).\mathbf{0} \tag{5.23}$$

$V_1(A) = V_1(B)$ since $L_1(A) = L_1(B)$. Now, directly from the definition of L_1, we can say:

$$\begin{aligned} V_1(A) &= \mathbb{P}(X < Y)\mathbf{Var}(X|X < Y) + \mathbb{P}(Y < X)\mathbf{Var}(Y|Y < X) \\ &\quad + \mathbb{P}(X < Y)\mathbb{P}(Y < X)(\mathbb{E}(X|X < Y) - \mathbb{E}(Y|Y < X))^2 \end{aligned} \tag{5.24}$$

This does not seem to be reducible to an expression involving just terms $\mathbf{Var}(X)$, $\mathbf{Var}(Y)$ and $\mathbb{E}(X)$, $\mathbb{E}(Y)$ in general. Thus in the next section we investigate synchronisations with specific common modelling distributions.

Last-to-Finish Synchronisation

From our stochastic definition of LTF synchronisation, L_1, we can use the probabilistic branch result to give us a general formula for a Last-to-Finish synchronisation, $A \stackrel{\text{def}}{=} (a, F_X(t)).\mathbf{0} \underset{\text{LTF}\{a\}}{||} (a, F_Y(t)).\mathbf{0}$ and so:

$$\begin{aligned} V_1(A) &= \mathbb{P}(X > Y)\mathbf{Var}(X|X > Y) + \mathbb{P}(Y > X)\mathbf{Var}(Y|Y > X) \\ &\quad + \mathbb{P}(X > Y)\mathbb{P}(Y > X)(\mathbb{E}(X|X > Y) - \mathbb{E}(Y|Y > X))^2 \end{aligned} \tag{5.25}$$

Similarly, this does not appear to be reducible to an expression involving just terms $\mathbf{Var}(X)$, $\mathbf{Var}(Y)$ and $\mathbb{E}(X)$, $\mathbb{E}(Y)$ in general, and so we are forced to derive conclusions from specific distribution examples in section 5.4.

5.4 Specific Distributions under SPA Combination

In the following section we look at how specific distributions are affected by stochastic process algebra combinators, like competitive choice and Last-to-Finish synchronisation. We will examine variance properties of some of the simpler distributions in combinations:

Dirac delta function which represents deterministic behaviour in a stochastic environment. It is used in models of the ATM protocol to represent a deterministic service process in a switch. Also, more generally, it can represent a deterministic time-out of a late process.

Exponential distribution which, although arguably overused in stochastic modelling, does have real-world applicability. As we will see a naïvely implemented algorithm can easily have an exponentially distributed time-to-finish. Also, as we are not necessarily interested solely in computer systems, but in process modelling in general, many naturally occurring phenomena can occur with an exponential distribution.

Uniform distribution represents bounded random events. It is an easy way of expressing an approximation to an event which, it is known, must occur between two time values.

The study of these distributions is motivated by their practical application, discussed above, and, practically speaking, because they are easier to analyse than some of the more complex multi-parameter distributions.

We cannot claim any kind of complete taxonomy with the analysis in the sections to come. However, we hope to motivate discussion of specific distributions or sets of distributions which will be of real modelling value if some closed-form solution could be obtained. Ultimately this is probably one of the main goals of reliability analysis using stochastic processes and stochastic process algebras—the ability to be able to say that using a specific distribution set under a specific combinator, the reliability and predictability of an overall system will be improved. Obviously from a modelling perspective the broader that set of distributions, the more descriptive the paradigm and the more useful it can be. Ideally, if general reliability statements could be made about, say, a combination of Coxian stage-type distributions, then that would give us a great deal of descriptive power, but as we will see, even for something as simple as relative variance, making general statements about even simple distributions is quite difficult.

The analysis below considers the effect on path variance of specific combinations of commonly occurring distributions. Specifically, we consider a primary distribution which is then combined with a second distribution under a particular discipline; we then compare the overall variance of the event with that of the primary distribution.

The difficulty of obtaining general variance results for these more complex combinators and indeed for any more complex distribution types is very much a limiting factor of this analysis. However, we feel that, given the physical importance of the distributions under consideration and the obvious usefulness of the combinators, including these more specialised results is still of use to the modeller and does add to the understanding of the subject.

The analysis for each pair of distributions takes three parts: the first extracts the variance of the combined distribution; the second, where possible, extracts any general analytic conclusions that can be drawn from comparing the overall path variance with that of the primary distribution; and finally, the third includes a plot of differential variance ranging over a key parameter.

For the higher level combinators below, we use the stochastic results for $\min(X, Y)$ and $\max(X, Y)$ in terms of cumulative density functions, from chapter 4. They are identical to the expressions we used in L_1 to obtain a general case (as proved in annex 5.B), but were easier to obtain specific distribution results from. We also refer forward to the annex 5.A, in order to use the single-distribution-specific results.

For the analysis below, let $G_X(t) = \int_0^t F_X(u)\,\mathrm{d}u$.

5.4.1 Competitive Choice and First-to-Finish Synchronisation

Exponential v Deterministic

Let $X \sim \exp(\lambda)$, an exponential distribution, and $Y \sim \delta(\nu)$ in a competitive choice or First-to-Finish synchronisation. W represents the overall stochastic event and we calculate $\mathbb{E}(W)$ and $\mathbf{Var}(W)$ below:

$$\begin{aligned}
\overline{F}_W(t) &= \overline{F}_X(t)\overline{F}_Y(t) \\
&= e^{-\lambda t}H(\nu - t) \\
f_W(t) &= \lambda e^{-\lambda t}H(\nu - t) + \lambda e^{-\lambda t}\delta(\nu - t)
\end{aligned} \tag{5.26}$$

Now, we want to use equations (5.115) and (5.117) to obtain $\mathbb{E}(W)$ and $\mathbf{Var}(W)$, so we evaluate the function $G_X(\nu)$:

$$\begin{aligned}
G_X(t) &= \int_0^t 1 - e^{-\lambda u}\,\mathrm{d}u \\
G_X(\nu) &= \nu - \frac{1}{\lambda}(1 - e^{-\lambda\nu})
\end{aligned} \tag{5.27}$$

$$\begin{aligned}
\mathbb{E}(W) &= \nu - G_X(\nu) \\
&= \frac{1}{\lambda}(1 - e^{-\lambda\nu})
\end{aligned} \tag{5.28}$$

$$\begin{aligned}
\mathbf{Var}(W) &= 2\int_0^\nu G_X(t)\,\mathrm{d}t - {G_X}^2(\nu) \\
&= \frac{1}{\lambda^2}(1 - 2\lambda\nu e^{-\lambda\nu} - e^{-2\lambda\nu})
\end{aligned} \tag{5.29}$$

As discussed in the introduction to this section, one physical interpretation of the deterministic transition is as a time-out on a concurrent process. By introducing this time-out event, we can establish what effect it has had on the variance in comparison to that of the original exponential event.

If $\mathbf{Var}(W) < \mathbf{Var}(X)$ then:

$$e^{-\lambda\nu}(2\lambda\nu + e^{-\lambda\nu}) > 0 \tag{5.30}$$

$$\Leftrightarrow e^{-\lambda\nu} > 0 \tag{5.31}$$

which is true for all $\lambda, \nu > 0$. Thus:

$$\mathbf{Var}(W) \leq \mathbf{Var}(X) \tag{5.32}$$

This tells us that a deterministic time-out process in a First-to-Finish synchronisation against an exponential distribution will always reduce the variance of the overall transition.

To show this reduction in variance numerically, we plot $\mathbf{Var}(W) - \mathbf{Var}(X)$ for fixed exponential parameter, $\lambda = 0.2$, and varying deterministic parameter, $0 \leq \nu \leq 25$, in figure 5.3.

Exponential v Exponential

For $X \sim \exp(\lambda)$ and $Y \sim \exp(\mu)$, we can take the results for exponential First-to-Finish synchronisation directly from chapter 4. As usual W represents the overall process:

$$F_W(t) = 1 - e^{-(\lambda+\mu)t} \tag{5.33}$$

$$\mathbb{E}(W) = \frac{1}{\lambda + \mu} \tag{5.34}$$

$$\mathbf{Var}(W) = \frac{1}{(\lambda + \mu)^2} \tag{5.35}$$

Now comparing the variance of W to the primary distribution, say X.

$$\frac{1}{(\lambda + \mu)^2} < \frac{1}{\lambda^2} \tag{5.36}$$

which is true for all $\lambda, \mu > 0$.

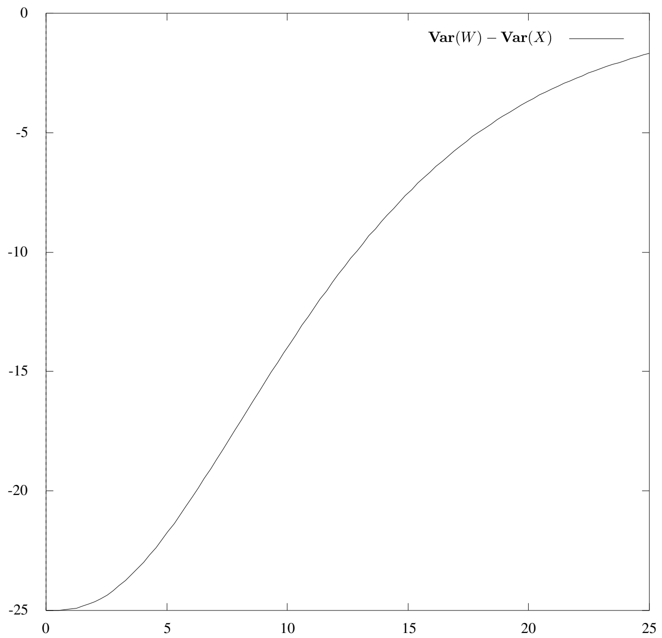

Fig. 5.3. The variance reduction of a First-to-Finish synchronisation for deterministic vs exponential.

So $\mathbf{Var}(W) < \mathbf{Var}(X)$ for all λ, μ. So a First-to-Finish or competitive choice combination of exponential distributions will always reduce variance.

To show this reduction in variance numerically, we plot $\mathbf{Var}(W) - \mathbf{Var}(X)$ for fixed parameter, $\lambda = 0.2$, and varying mean, $0 \leq 1/\mu \leq 20$, in figure 5.4.

Uniform v Deterministic

Let $X \sim U[a, b]$ where $a = \mu - \lambda$ and $b = \mu + \lambda$ and $Y \sim \delta(\nu)$. As before, for simplicity we would like to make use of the results of equations (5.115) and (5.117). To do so we require $G_X(t)$; the calculations are slightly more involved for the multiple cases of the uniform distribution, so we will state the results without intermediate steps in this section:

$$G_X(t) = \begin{cases} 0 & t < a \\ (t-a)^2/4\lambda & a \leq t < b \\ t - \mu & b \leq t \end{cases} \tag{5.37}$$

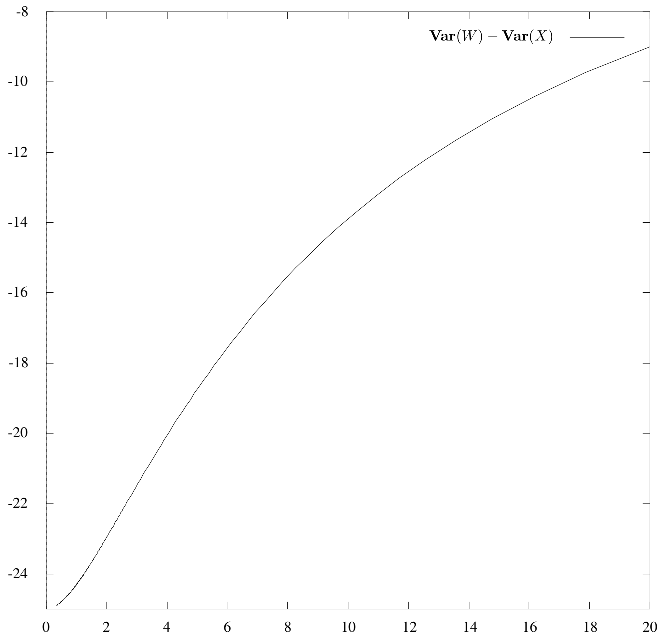

Fig. 5.4. The variance reduction of a First-to-Finish synchronisation for exponential vs exponential.

We can now use this in equation (5.115) to give us the mean of the FTF synchronisation with Y. As ever, the overall stochastic behaviour is captured in the variable W:

$$\mathbb{E}(W) = \begin{cases} \nu & \nu < a \\ \nu - (\nu - a)^2/4\lambda & a \leq \nu < b \\ \mu & b \leq t \end{cases} \tag{5.38}$$

Similarly the variance is obtained using equation (5.117):

$$\mathbf{Var}(W) = \begin{cases} 0 & \nu < a \\ (5\lambda + 3\gamma)(\nu - a)^3/48\lambda^2 & a \leq \nu < b \\ \lambda^2/3 & b \leq t \end{cases} \tag{5.39}$$

where $\gamma = \mu - \nu$.

As before, taking the uniform distribution as the primary distribution of the combination, we compare $\mathbf{Var}(W)$ with $\mathbf{Var}(X)$. Take $\mathbf{Var}(W) = R\,\mathbf{Var}(X)$, so for the deterministic time-out Y to reduce the variance in general, we are looking for $R < 1$. Now $\mathbf{Var}(X) = \lambda^2/3$ so:

$$R = \begin{cases} 0 & \nu < a \\ (5\lambda + 3\gamma)(1 - \gamma/\lambda)^3/16\lambda & a \leq \nu < b \\ 1 & b \leq t \end{cases} \tag{5.40}$$

The only case to be considered is for $a \leq \nu < b$, so we let $\gamma = \alpha\lambda$ so that $-1 < \alpha \leq 1$. Now:

$$\underbrace{(1-\alpha)^3(3\alpha+5)}_{f(\alpha)} \leq 16 \tag{5.41}$$

Since $f(-1) = 16$, we now require to show that the gradient of f is always negative for $-1 < \alpha \leq 1$ and we have shown $R \leq 1$ for all parameter values.

$$\frac{\mathrm{d}}{\mathrm{d}\alpha} f(\alpha) = -12(1-\alpha)^2(1+\alpha) < 0 \quad \text{for all } -1 < \alpha \leq 1 \tag{5.42}$$

Hence the result that $\mathbf{Var}(W) \leq \mathbf{Var}(X)$, and in physical terms that First-to-Finish or competitive choice combination of a uniform distribution with a deterministic event will reduce or maintain the variance of the original uniform distribution.

To show this reduction in variance numerically, we plot $\mathbf{Var}(W) - \mathbf{Var}(X)$ for fixed parameters, $\mu = 5$, $\lambda = 3.5$ and varying deterministic parameter, $0 \leq \nu \leq 10$, in figure 5.5.

Uniform v Exponential

Let $X \sim U[a, b]$ where $a = \mu - \rho$ and $b = \mu + \rho$ and $Y \sim \exp(\lambda)$. For simplicity we would like to make use of the results of equations (5.121) and (5.123). To do so we require $L'_\psi(\lambda)$ and $L''_\psi(\lambda)$, where:

$$\psi(t) = f_X(t) - \lambda F_X(t) \tag{5.43}$$

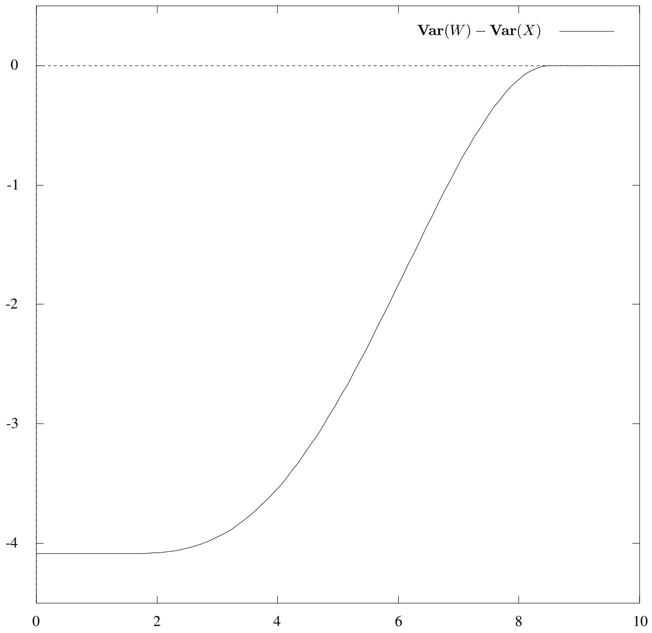

Fig. 5.5. The variance reduction of a First-to-Finish synchronisation for uniform vs deterministic.

$$\psi(t) = \begin{cases} 0 & t < a \\ (1 + \lambda a - \lambda t)/2\rho & a \le t < b \\ -\lambda & b \le t \end{cases} \tag{5.44}$$

$$L'_\psi(\lambda) = \frac{e^{-\lambda a} - e^{-\lambda b}}{2\rho\lambda^2} \tag{5.45}$$

$$L''_\psi(\lambda) = \frac{1}{\lambda^2}(e^{-\lambda a} + e^{-\lambda b}) - \frac{2 + \lambda\mu}{\rho\lambda^3}(e^{-\lambda a} - e^{-\lambda b}) \tag{5.46}$$

Thus from equation (5.121) we get:

$$\mathbb{E}(W) = \frac{1}{\lambda} - \frac{e^{-\lambda a} - e^{-\lambda b}}{2\rho\lambda^2} \tag{5.47}$$

We obtain $\mathbf{Var}(W)$ from equation (5.123) although it is not displayed here for reasons of sanity. Perhaps not surprisingly, we were not able to make any general analytical conclusions about whether the synchronisation reduced the variance. We do use $\mathbf{Var}(W)$ however for demonstrating numerically the variance reduction properties of the FTF synchronisation or competitive choice of exponential and uniform evolutions.

In this case it is not obvious which distribution should be the primary distribution i.e. which distribution obviously reduces variance, so we plot both $\mathbf{Var}(W) - \mathbf{Var}(X)$ and $\mathbf{Var}(W) - \mathbf{Var}(Y)$ in figures 5.6 and 5.7.

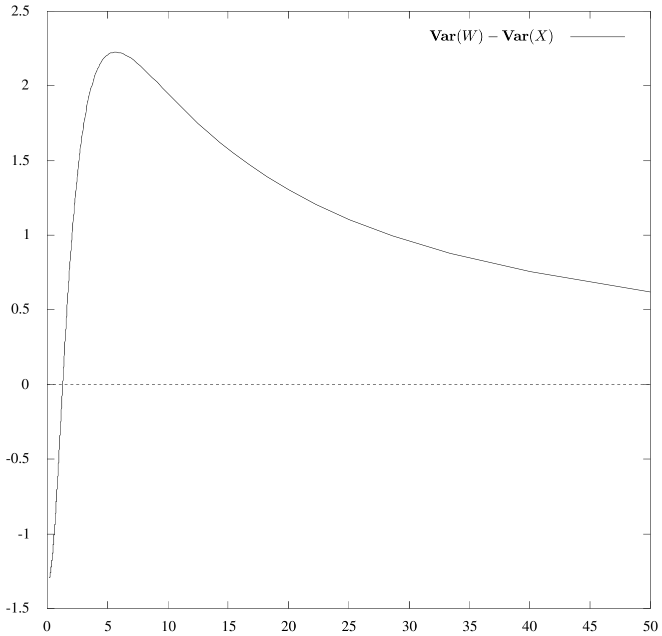

Fig. 5.6. The variance reduction of a First-to-Finish synchronisation for exponential vs uniform.

For $\mathbf{Var}(W) - \mathbf{Var}(X)$, figure 5.6, we fix the uniform distribution, X, with $\mu = 5$ and $\rho = 2$, while varying the mean of the exponential distribution Y over $0 < 1/\lambda \leq 50$.

For $\mathbf{Var}(W) - \mathbf{Var}(Y)$, figure 5.7, we fix the exponential distribution, Y, with parameter $\lambda = 0.2$ and the width of the uniform distribution, X, with $\rho = 5$, while varying the mean of the uniform distribution over $5 < \mu \leq 30$.

From the graphs we see that it is the uniform distribution X, when combined in FTF or competitive choice with Y, which consistently reduces the variance of the overall transition (figure 5.7). We can intuitively see this by recognising

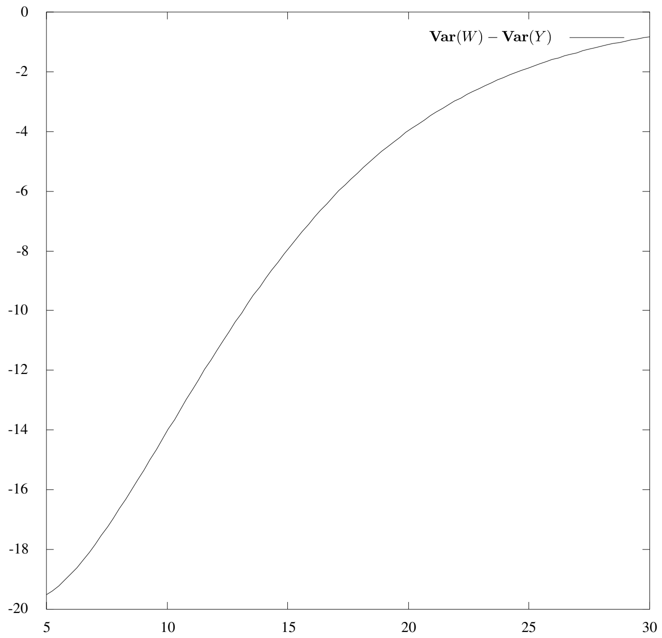

Fig. 5.7. The variance reduction of a First-to-Finish synchronisation for uniform vs exponential.

that Y does not have an infinite non-zero domain, whereas the exponential distribution does and so we might have anticipated that it would *improve* on the variance of the exponential distribution.

5.4.2 Last-to-Finish Synchronisation

Exponential v Deterministic

Let $X \sim \exp(\lambda)$, an exponential distribution, and $Y \sim \delta(\nu)$ in a Last-to-Finish synchronisation. W represents the overall stochastic event and we calculate $\mathbf{Var}(W)$ below:

$$\begin{aligned} F_W(t) &= F_X(t)F_Y(t) \\ &= (1-e^{-\lambda t})H(t-\nu) \\ f_W(t) &= \lambda e^{-\lambda t}H(t-\nu) + (1-e^{-\lambda t})\delta(t-\nu) \end{aligned} \tag{5.48}$$

Now, we want to use equations (5.125) and (5.127) to obtain $\mathbb{E}(W)$ and $\mathbf{Var}(W)$, so we use the function $G_X(\nu)$ from equation (5.27):

$$G_X(\nu) = \nu - \frac{1}{\lambda}(1 - e^{-\lambda\nu}) \tag{5.49}$$

$$\begin{aligned} \mathbb{E}(W) &= \frac{1}{\lambda} + G_X(\nu) \\ &= \nu + \frac{1}{\lambda}e^{-\lambda\nu} \end{aligned} \tag{5.50}$$

$$\begin{aligned} \mathbf{Var}(W) &= \mathbf{Var}(X) + 2\left(\nu - \frac{1}{\lambda}\right) G_X(\nu) - 2\int_0^\nu G_X(t)\,\mathrm{d}t - {G_X}^2(\nu) \\ &= \frac{1}{\lambda^2}(2 - e^{-\lambda\nu})e^{-\lambda\nu} \end{aligned} \tag{5.51}$$

By introducing this deterministic event, we can establish what effect it has had on the variance in comparison to that of the original exponential event. If $\mathbf{Var}(W) < \mathbf{Var}(X)$ then:

$$2e^{-\lambda\nu} - e^{-2\lambda\nu} < 1 \tag{5.52}$$

$$\Leftrightarrow (e^{-\lambda\nu} - 1)^2 > 0 \tag{5.53}$$

$$\Leftrightarrow e^{-\lambda\nu} \neq 1 \tag{5.54}$$

which is true for all $\lambda, \nu > 0$ and equality when $\nu = 0$, on the assumption that λ cannot be zero. Thus:

$$\mathbf{Var}(W) < \mathbf{Var}(X) \tag{5.55}$$

This tells us that a deterministic process in a Last-to-Finish synchronisation against an exponential distribution will always reduce the variance of the overall transition.

To show this reduction in variance numerically, we plot $\mathbf{Var}(W) - \mathbf{Var}(X)$ for fixed exponential parameter, $\lambda = 0.2$, and varying deterministic parameter, $0 \leq \nu \leq 20$, in figure 5.8.

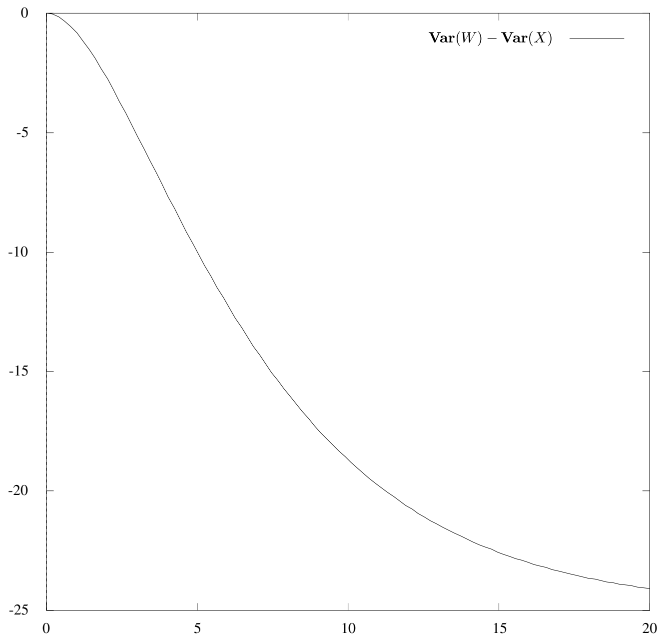

Fig. 5.8. The variance reduction of a Last-to-Finish synchronisation for deterministic vs exponential.

Exponential v Exponential

For $X \sim \exp(\lambda)$ and $Y \sim \exp(\mu)$, we can take the results for exponential Last-to-Finish synchronisation directly from chapter 4. As usual W represents the overall process:

$$\begin{aligned} F_W(t) &= (1 - e^{-\lambda t})(1 - e^{-\mu t}) \\ f_W(t) &= \lambda e^{-\lambda t} + \mu e^{-\mu t} - (\lambda + \mu) e^{-(\lambda+\mu)t} \end{aligned} \tag{5.56}$$

$$\mathbb{E}(W) = \frac{1}{\lambda} + \frac{1}{\mu} - \frac{1}{\lambda + \mu} \tag{5.57}$$

$$\mathbf{Var}(W) = \frac{1}{\lambda^2} + \frac{1}{\mu^2} - \frac{3}{(\lambda + \mu)^2} \tag{5.58}$$

Now comparing $\mathbf{Var}(W)$ with, say, $\mathbf{Var}(X)$ to obtain any variance reduction conditions: we initially assume that $\mathbf{Var}(W) < \mathbf{Var}(X)$ to see whether we get a condition or a contradiction and this gives:

$$\frac{1}{\lambda^2} + \frac{1}{\mu^2} - \frac{3}{(\lambda+\mu)^2} < \frac{1}{\lambda^2} \tag{5.59}$$

Substituting $\delta = \lambda/\mu$ gives:

$$\delta^2 + 2\delta - 2 < 0 \tag{5.60}$$

which has the solution:

$$-1 - \sqrt{3} < \delta < -1 + \sqrt{3} \tag{5.61}$$

Now given $\lambda, \mu > 0$, so $\delta > 0$, thus $\mathbf{Var}(W) < \mathbf{Var}(X)$ if $0 < \lambda/\mu < \sqrt{3} - 1$.

So under these parametric constraints, a Last-to-Finish synchronisation of exponential distributions will reduce the overall variance.

To show this reduction in variance numerically, we plot $\mathbf{Var}(W) - \mathbf{Var}(X)$ for fixed parameter, $\lambda = 0.2$, and varying mean, $0 \leq 1/\mu \leq 10$, in figure 5.9.

As predicted, this confirms that the variance reduction only occurs for a specific range of μ; in this case given by the formula:

$$\frac{1}{\mu} < \underbrace{5(\sqrt{3} - 1)}_{=3.660} \tag{5.62}$$

This particular case is interesting because of the existence of a turning point in the curve $\mathbf{Var}(W) - \mathbf{Var}(X)$. This implies that there will be a point of maximum variance reduction for exponential/exponential LTF synchronisation. Let $\mu = r\lambda$:

$$\begin{aligned} \Delta V &= \mathbf{Var}(W) - \mathbf{Var}(X) \\ &= \frac{1}{r^2\lambda^2} - \frac{3}{\lambda^2(1+r)^2} \end{aligned} \tag{5.63}$$

We require to find:

$$\frac{\mathrm{d}}{\mathrm{d}r}\Delta V = 0 \tag{5.64}$$

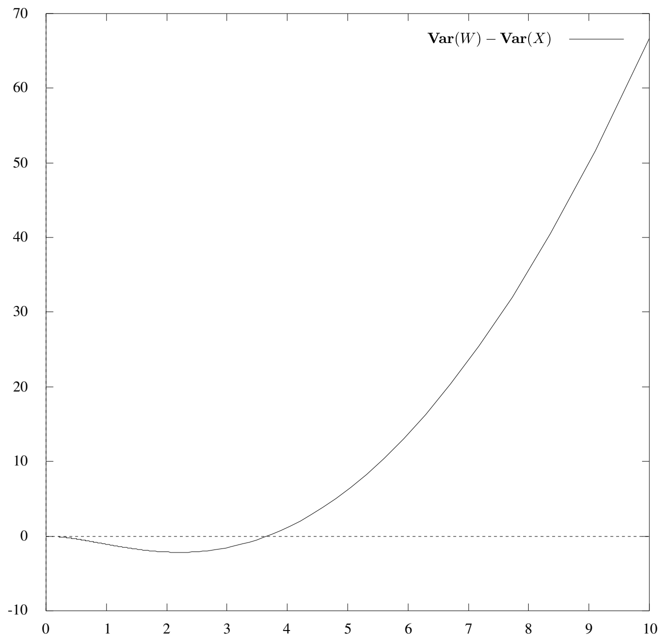

Fig. 5.9. The variance reduction of a Last-to-Finish synchronisation for exponential vs exponential.

which gives the following cubic:

$$2r^3 - 3r^2 - 3r - 1 = 0 \tag{5.65}$$

and gives the only real solution $r = 2.2611667$.

Again, this result concurs with figure 5.9 where we can see that the point of maximum variance reduction occurs at $1/\mu = (0.2 \times 2.2611667)^{-1} = 2.211$.

Uniform v Deterministic

Let $X \sim U[a, b]$ where $a = \mu - \lambda$ and $b = \mu + \lambda$ and $Y \sim \delta(\nu)$. As before, for simplicity we would like to make use of the results of equations (5.125) and (5.127). To do so we require $G_X(t)$; the calculations are slightly more involved for the multiple cases of the uniform distribution, so again we will state the results without intermediate steps in this section. From equation (5.37):

$$G_X(t) = \begin{cases} 0 & t < a \\ (t-a)^2/4\lambda & a \le t < b \\ t - \mu & b \le t \end{cases} \tag{5.66}$$

We can now use this in equation (5.125) to give us the mean of the LTF synchronisation with Y. As ever, the overall stochastic behaviour is captured in the variable W:

$$\mathbb{E}(W) = \begin{cases} \mu & \nu < a \\ \mu + (\nu - a)^2/4\lambda & a \leq \nu < b \\ \nu & b \leq t \end{cases} \tag{5.67}$$

Similarly, the variance is obtained using equation (5.127):

$$\mathbf{Var}(W) = \begin{cases} \lambda^2/3 & \nu < a \\ (5\lambda - 3\gamma)(\lambda + \gamma)^3/48\lambda^2 & a \leq \nu < b \\ 0 & b \leq t \end{cases} \tag{5.68}$$

where $\gamma = \mu - \nu$.

As before, taking the uniform distribution as the primary distribution of the combination, we compare $\mathbf{Var}(W)$ with $\mathbf{Var}(X)$. Take $\mathbf{Var}(W) = R\,\mathbf{Var}(X)$; for the deterministic Y to reduce the variance in general, we are looking for $R < 1$. Now $\mathbf{Var}(X) = \lambda^2/3$ so:

$$R = \begin{cases} 1 & \nu < a \\ (5\lambda - 3\gamma)(\lambda + \gamma)^3/16\lambda^4 & a \leq \nu < b \\ 0 & b \leq t \end{cases} \tag{5.69}$$

The only case to be considered is for $a \leq \nu < b$, so we let $\gamma = \alpha\lambda$ so that $-1 < \alpha \leq 1$. Now:

$$\underbrace{(1 + \alpha)^3(5 - 3\alpha)}_{f(\alpha)} \leq 16 \tag{5.70}$$

Since $f(1) = 16$, we now require to show that the gradient of f is always positive for $-1 \leq \alpha < 1$ and we have shown $R \leq 1$ for all parameter values.

$$\frac{\mathrm{d}}{\mathrm{d}\alpha} f(\alpha) = 12(1 + \alpha)^2(1 - \alpha) > 0 \quad \text{for all } -1 \leq \alpha < 1 \tag{5.71}$$

Hence result that $\mathbf{Var}(W) \leq \mathbf{Var}(X)$ and in physical terms that Last-to-Finish combination of a uniform distribution with a deterministic event will reduce or maintain the variance of the original uniform distribution.

To show this reduction in variance numerically, we plot $\mathbf{Var}(W) - \mathbf{Var}(X)$ for fixed parameters, $\mu = 5$, $\lambda = 3.5$ and varying deterministic parameter, $0 \leq \nu \leq 10$, in figure 5.10.

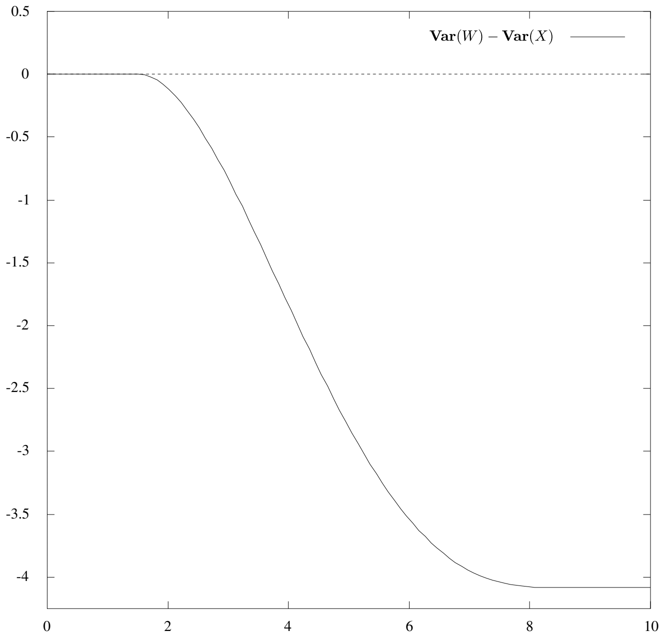

Fig. 5.10. The variance reduction of a Last-to-Finish synchronisation for uniform vs deterministic.

Uniform v Exponential

Let $X \sim U[a, b]$ where $a = \mu - \rho$, $b = \mu + \rho$ and $Y \sim \exp(\lambda)$. For simplicity we would like to make use of the results of equations (5.131) and (5.133). To do so we use $L'_\psi(\lambda)$ and $L''_\psi(\lambda)$ from equations (5.43) and (5.44):

$$\psi(t) = f_X(t) - \lambda F_X(t) \tag{5.72}$$

$$\psi(t) = \begin{cases} 0 & t < a \\ (1 + \lambda a - \lambda t)/2\rho & a \leq t < b \\ -\lambda & b \leq t \end{cases} \tag{5.73}$$

$$L'_\psi(\lambda) = \frac{e^{-\lambda a} - e^{-\lambda b}}{2\rho\lambda^2} \tag{5.74}$$

$$L''_\psi(\lambda) = \frac{1}{\lambda^2}(e^{-\lambda a} + e^{-\lambda b}) - \frac{2+\lambda\mu}{\rho\lambda^3}(e^{-\lambda a} - e^{-\lambda b}) \tag{5.75}$$

Thus from equation (5.131) we get:

$$\mathbb{E}(W) = \mu + \frac{e^{-\lambda a} - e^{-\lambda b}}{2\rho\lambda^2} \tag{5.76}$$

We obtain $\mathbf{Var}(W)$ from equation (5.133) although it is not displayed here for brevity. Perhaps not surprisingly, we were not able to make any general analytical conclusions about whether the synchronisation reduced the variance. We do use $\mathbf{Var}(W)$ however for demonstrating numerically the variance reduction properties of the LTF synchronisation of exponential and uniform evolutions.

In this case it is not obvious which distribution should be the primary distribution, i.e. which distribution obviously reduces variance, so we plot both $\mathbf{Var}(W) - \mathbf{Var}(X)$ and $\mathbf{Var}(W) - \mathbf{Var}(Y)$ in figures 5.11 and 5.12.

For $\mathbf{Var}(W) - \mathbf{Var}(X)$, figure 5.11, we fix the uniform distribution, X, with $\mu = 5$ and $\rho = 2$, while varying the mean of the exponential distribution Y over $0 < 1/\lambda \leq 10$.

For $\mathbf{Var}(W) - \mathbf{Var}(Y)$, figure 5.12, we fix the exponential distribution, Y, with parameter $\lambda = 0.2$ and the width of the uniform distribution, X, with $\rho = 5$, while varying the mean of the uniform distribution over $5 < \mu \leq 30$.

From the graphs we see that, as with the FTF case, it is the uniform distribution X, when combined in an LTF synchronisation with Y, which consistently reduces the variance of the overall transition (figure 5.7).

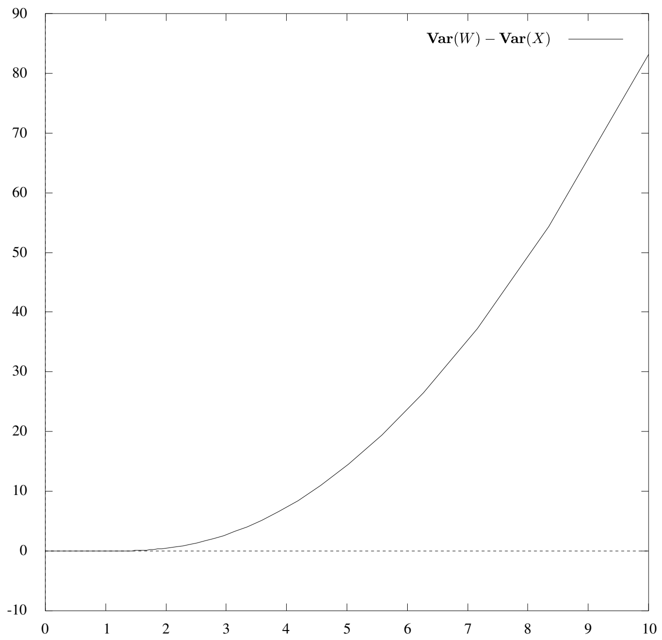

Fig. 5.11. The variance reduction of a Last-to-Finish synchronisation for exponential vs uniform.

5.5 Worked Examples

5.5.1 Search Algorithm

As an example of how to improve the reliability of a system, we consider a search algorithm. The goal is to design a second algorithm which, when combined in a First-to-Finish strategy, would improve the reliability of the overall search. In this case, we consider *improved reliability* to mean decreased variance.

As an initial gambit we devise a naïve algorithm for searching an array of size N. We make the following assumptions about the search:

1. the data in the array is unordered

2. fetching any element from the array is achieved in constant time, μ

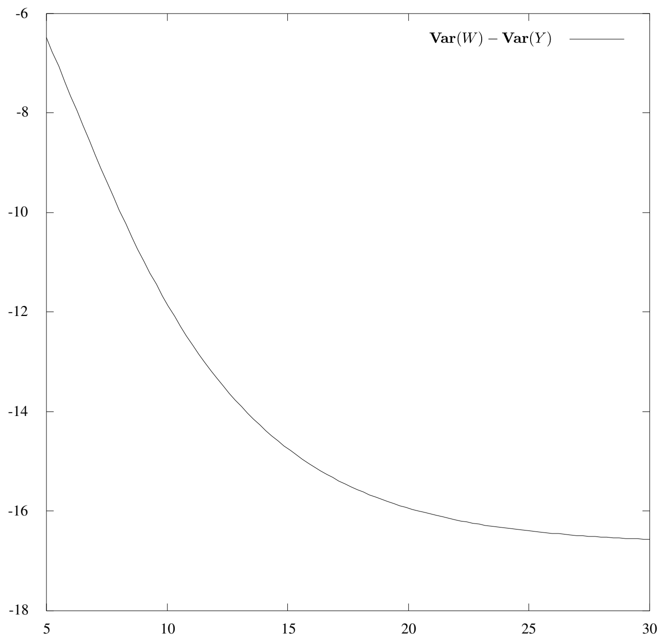

Fig. 5.12. The variance reduction of a Last-to-Finish synchronisation for uniform vs exponential.

3. there is only one element to find in the array and once it is found, the search can be terminated.

The naïve algorithm simply randomly selects an element from the array and then terminates if it is the correct element. No attempt is made to mark the element as searched or remove the element from the array if it is not the correct element; another random selection is made and the process continues.

The second algorithm is simply a sequential search through the array; on finding the correct element, it too terminates immediately. This has the obvious benefit of terminating successfully after at most time $N\mu$.

The algorithms are described in process algebra terms in figure 5.13.

$$
\begin{aligned}
S^N_{random} &\stackrel{\text{def}}{=} (search, F_X(t)).T^N_{random}\\
T^N_{random} &\stackrel{\text{def}}{=} \left[\frac{N-1}{N}\right](fail, F_Y(t)).S^N_{random} \oplus \left[\frac{1}{N}\right](found, F_Y(t)).\mathbf{0}
\end{aligned}
$$

$$
\begin{aligned}
S^N_{linear} &\stackrel{\text{def}}{=} (search, F_X(t)).T^N_{linear}\\
T^N_{linear} &\stackrel{\text{def}}{=} \left[\frac{N-1}{N}\right](fail, F_Y(t)).S^{N-1}_{linear} \oplus \left[\frac{1}{N}\right](found, F_Y(t)).\mathbf{0}\\
S^1_{linear} &\stackrel{\text{def}}{=} (found, F_Y(t)).\mathbf{0}
\end{aligned}
$$

where $X \sim \delta(\mu)$ and $Y \sim \delta(0)$

$$
Search \stackrel{\text{def}}{=} S^N_{random} \parallel_{\mathrm{FTF}\{found\}} S^N_{linear}
$$

Fig. 5.13. Process algebra descriptions of the search algorithms.

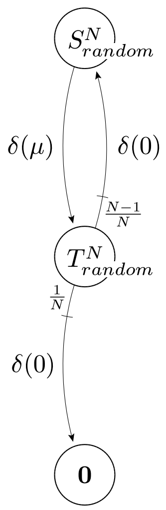

Fig. 5.14. Stochastic transition system for random search algorithm.

Analysis of Random Search Algorithm

Let T represent the amount of time spent searching for the element (as shown in figure 5.14), such that:

$$L_T(\omega) = L_1(S^N_{random}) \tag{5.77}$$

$$L_T(\omega) = \frac{e^{-\mu\omega}}{N-(N-1)e^{-\mu\omega}} \tag{5.78}$$

$$L'_T(\omega) = -\frac{\mu(N-1)e^{-2\mu\omega}}{(N-(N-1)e^{-\mu\omega})^2} - \frac{\mu e^{-\mu\omega}}{N-(N-1)e^{-\mu\omega}} \tag{5.79}$$

$$\begin{aligned} L''_T(\omega) \quad = \quad & \frac{2\mu^2(N-1)e^{-2\mu\omega} + \mu^2(N-1)e^{-2\mu\omega}}{(N-(N-1)e^{-\mu\omega})^2} \\ & + \frac{2\mu^2(N-1)^2e^{-3\mu\omega}}{(N-(N-1)e^{-\mu\omega})^3} + \frac{\mu^2 e^{-\mu\omega}}{N-(N-1)e^{-\mu\omega}} \end{aligned} \tag{5.80}$$

$$\mathbb{E}(T) = \mu N \tag{5.81}$$

$$\mathbb{E}(T^2) = (2N-1)N\mu^2 \tag{5.82}$$

$$\mathbf{Var}(T) = N(N-1)\mu^2 \tag{5.83}$$

Now looking at the coefficient of variation:

$$C_T = \frac{\sqrt{N(N-1)}\mu}{N\mu} = \sqrt{\frac{N-1}{N}} \tag{5.84}$$

Thus $C_T \longrightarrow 1$ as $N \longrightarrow \infty$. Given that the exponential distribution is the only distribution which has a coefficient of variation of 1, we approximate T by an exponential distribution with parameter λ:

$$\lambda = \frac{1}{N\mu} \tag{5.85}$$

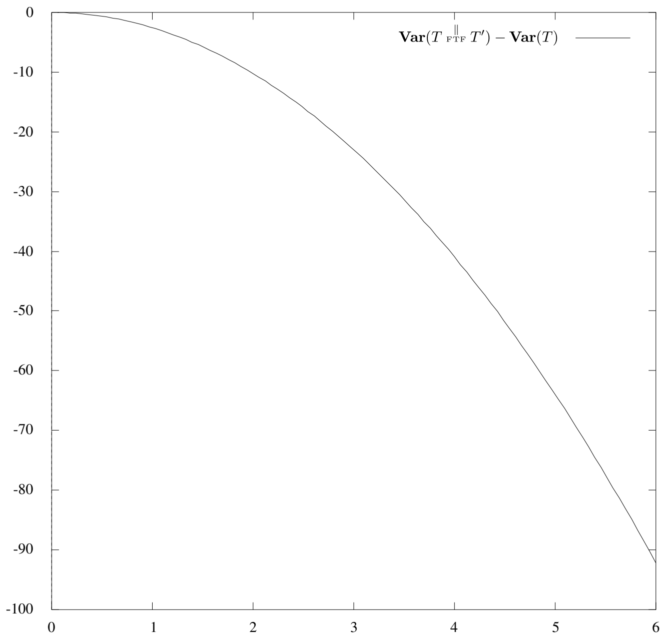

Fig. 5.15. ΔV_1 against $\mathbb{E}(T')$ for fixed size array, $N = 5$: the introduction of the linear search produces a large variance reduction.

Analysis of Sequential Search Algorithm

We can immediately write down an expression for the overall evolution of the sequential search, given that after r stages:

1. The next element is fetched from the array, incurring a delay of μ.
2. The search then terminates with probability $\frac{1}{N-r}$ if the element is the target, or continues with probability $\frac{N-r-1}{N-r}$.
3. The search proceeds to the $(r+1)$th stage. In the event that the search reaches the Nth stage, then the Nth element is going to be the target with probability 1, so the search terminates before the array ends.

Let T' represent the overall evolution of the sequential search, such that:

$$L_{T'}(\omega) = L_1(S^N_{linear}) \tag{5.86}$$

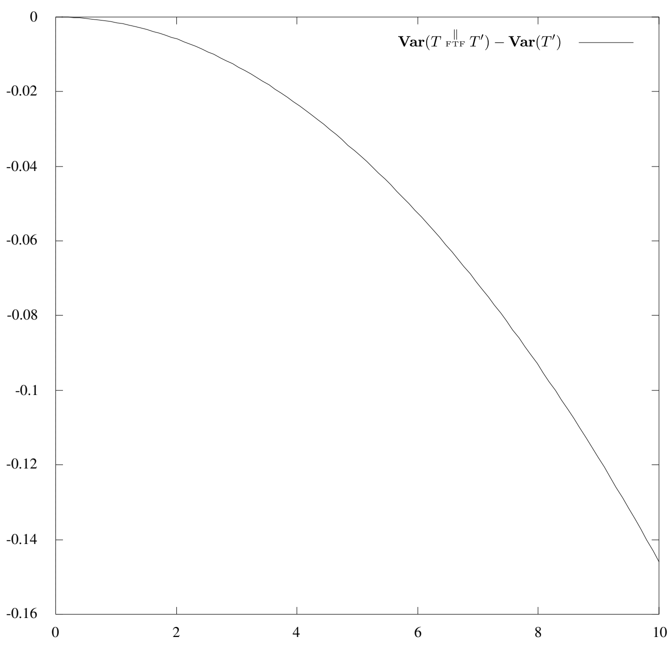

Fig. 5.16. ΔV_2 against $\mathbb{E}(T)$ for fixed size array, $N = 5$: the introduction of the random search produces a small variance reduction.

Thus:

$$L_{T'}(\omega) = e^{-\mu\omega}\left(\frac{1}{N} + \frac{N-1}{N}e^{-\mu\omega}\left(\frac{1}{N-1} + \frac{N-2}{N-1}e^{-\mu\omega}\left(\frac{1}{N-2} + \cdots\left(\frac{1}{2} + \frac{1}{2}e^{-\mu\omega}\right)\cdots\right)\right)\right) \tag{5.87}$$

which simplifies to:

$$L_{T'}(\omega) = \frac{1}{N}\sum_{r=1}^{N} e^{-r\mu\omega} \tag{5.88}$$

$$\mathbb{E}(T') = \frac{\mu}{N}\sum_{r=1}^{N} r = \frac{N+1}{2}\mu \tag{5.89}$$

$$\mathbb{E}(T'^2) = \frac{\mu^2}{N}\sum_{r=1}^{N} r^2 = \frac{1}{6}(2N+1)(N+1)\mu^2 \tag{5.90}$$

$$\mathbf{Var}(T') = \frac{1}{12}(N+1)(N-1)\mu^2 \tag{5.91}$$

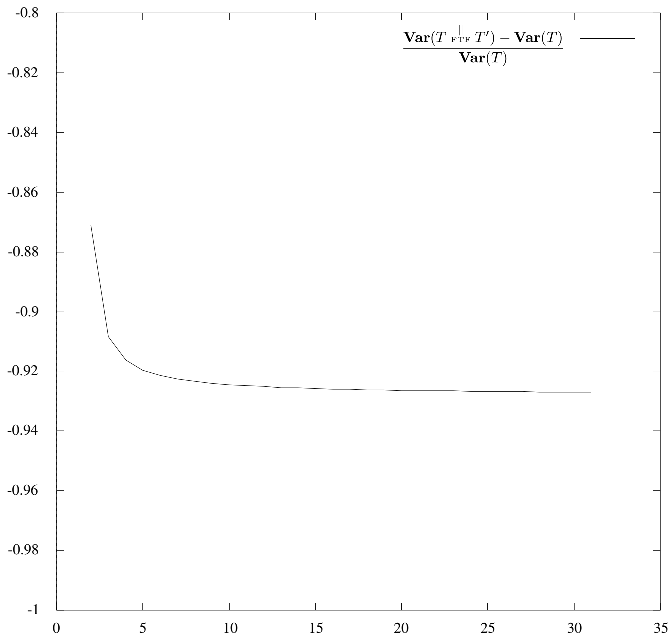

Fig. 5.17. rV_1 against $\mathbb{E}(T')$ for variable size array: relative to the original random search, the linear search reduces the variance by up to 93% for large N.

Analysis of Concurrent Search Process

We now require to combine the two strategies under a First-to-Finish synchronisation. The improvement in variance can then be calculated.

$$\overline{F}_{T \overset{\|}{\text{FTF}} T'}(t) = \frac{1}{N} e^{-\lambda t} \sum_{r=1}^{N} H(r\mu - t) \tag{5.92}$$

$$f_{T \overset{\|}{\text{FTF}} T'}(t) = \frac{1}{N} \sum_{r=1}^{N} (\lambda e^{-\lambda t} H(r\mu - t) + e^{-\lambda t} \delta(t - r\mu)) \tag{5.93}$$

This leads to the following results for the overall FTF synchronisation process:

$$\mathbb{E}(T \overset{\|}{\text{FTF}} T') = \frac{1}{N\lambda} \sum_{r=1}^{N} (1 - e^{-\lambda r \mu}) \tag{5.94}$$

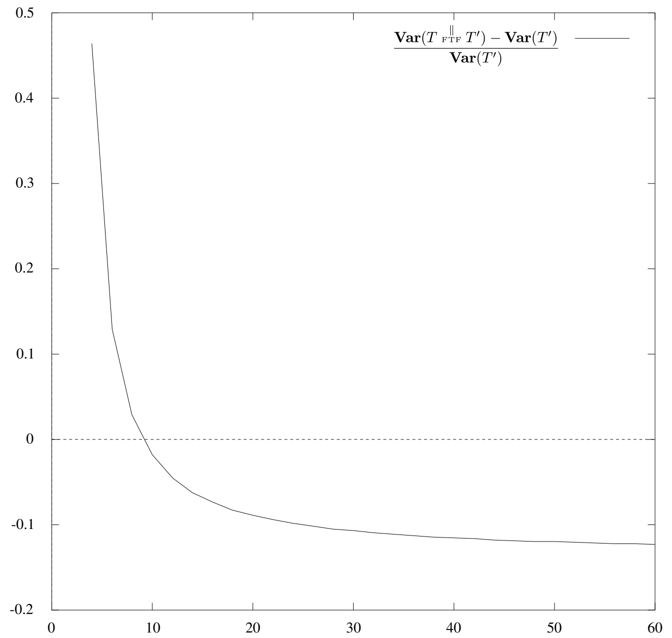

Fig. 5.18. rV_2 against $\mathbb{E}(T)$ for variable size array: relative to the original linear search, the random search still reduces the variance by up to 12% for large N.

$$\mathbf{Var}(T \overset{\|}{_{\text{FTF}}} T') = \frac{2}{N\lambda^2}\sum_{r=1}^{N}(1 - (1 + \lambda r\mu)e^{-\lambda r\mu}) \tag{5.95}$$

We now calculate:

$$\Delta V_1 = \mathbf{Var}(T \overset{\|}{_{\text{FTF}}} T') - \mathbf{Var}(T) \tag{5.96}$$

$$\Delta V_2 = \mathbf{Var}(T \overset{\|}{_{\text{FTF}}} T') - \mathbf{Var}(T') \tag{5.97}$$

These are plotted (figures 5.15, 5.16) against the means of T' and T respectively to show how much delay needs to be introduced into the synchronising process to produce a given change in variance. These clearly show that the synchronisation is successful in reducing the variance, although it can be seen that the random algorithm has far less effect on the variance of the linear algorithm (figure 5.16).

In order to establish relative improvement in variance, we need to see how ΔV_1 and ΔV_2 change relative to the original variances, $\mathbf{Var}(T)$ and $\mathbf{Var}(T')$.

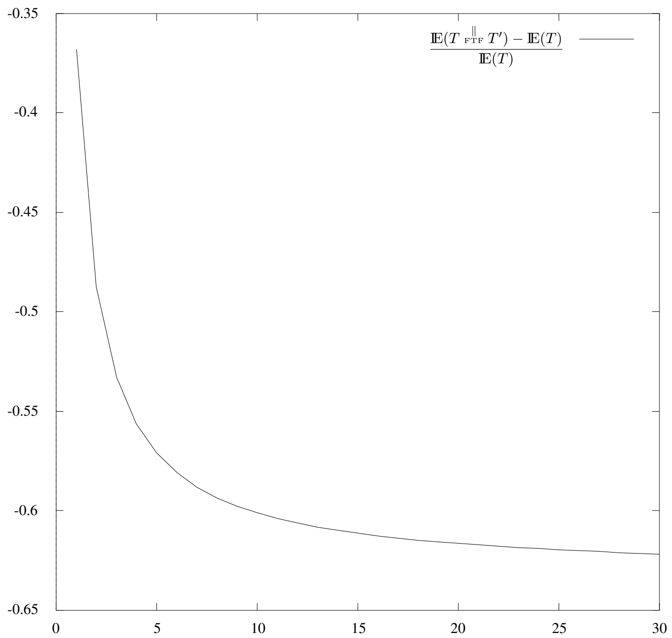

Fig. 5.19. rE_1 against N: the mean reduces by up to 62% when the linear search is combined with the random search, for large N.

To this end, figures 5.17, 5.18 show:

$$rV_1 = \frac{\Delta V_1}{\mathbf{Var}(T)} \text{ and } rV_2 = \frac{\Delta V_2}{\mathbf{Var}(T')} \tag{5.98}$$

Given that $\mathbf{Var}(\cdot) \geq 0$, we have $rV_N \geq -1$ with $rV_N = -1$ indicating a deterministic evolution and a 100% reduction in variance.

Figure 5.17 represents the reduction of variance relative to the random search, given the introduction of the linear search; it can be seen that for increasing size of array (which linearly increases the size of $\mathbb{E}(T')$), the variance is reduced by approaching 93%. Figure 5.18 shows the reduction of variance relative to the linear search, given the introduction of the random search as a synchronising process; even here, despite the fact that the random algorithm has a much larger variance, its introduction will still reduce the overall variance by approaching 12% for large N. This demonstrates our assertion that, in this case at least, a concurrent algorithm can be employed to improve the reliability of the overall system.

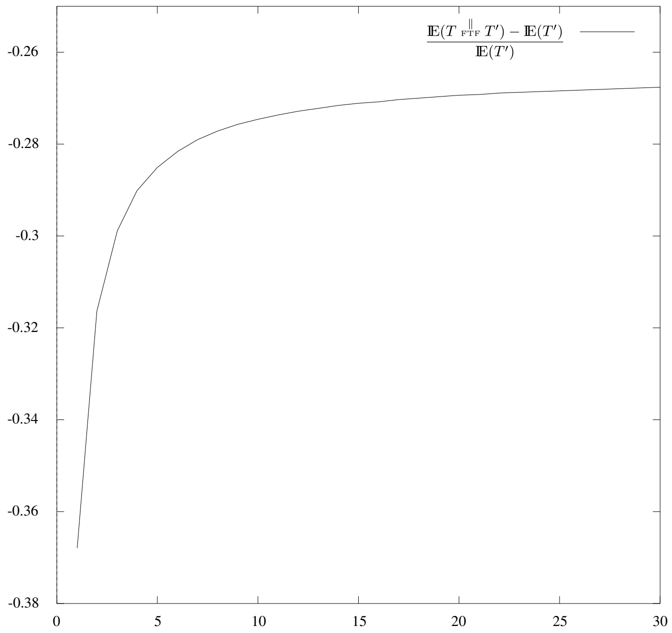

Fig. 5.20. rE_2 against N: the mean reduces by 27% when the random search is combined with the linear search, for large N.

We produce similar relative mean plots in figures 5.19 and 5.20, where:

$$rE_1 = \frac{\mathbb{E}(T \overset{||}{_{\text{FTF}}} T') - \mathbb{E}(T)}{\mathbb{E}(T)} \tag{5.99}$$

$$rE_2 = \frac{\mathbb{E}(T \overset{||}{_{\text{FTF}}} T') - \mathbb{E}(T')}{\mathbb{E}(T')} \tag{5.100}$$

These plots show that the means are reduced as well through such an interaction, however an instance could be envisaged when a modeller might be prepared to put up with larger average operation in return for a more reliable system.

5.5.2 Morse Code via Telephone

Another example, this time in the area of communications, might be provided by trying to implement a simple Morse code system using a ringing telephone—the idea being that the phone is never answered and the information is conveyed only by the length of time it rings. There is inherent uncertainty in this problem, as the person or protocol doing the ringing can have no idea whether the phone at the other end is actually ringing or importantly how long it has been ringing for.

Although this source of error could be mitigated by error-correcting codes, the efficiency of the overall protocol could be substantially improved if the underlying transport mechanism has a smaller chance of error.

System Description

To describe such a process, we need two parallel processes to represent the calling process (R in figure 5.21) and the network connecting process (C in figure 5.21). Thus *ring* taking random time X represents the amount of time that the protocol spends waiting after dialling the number; this can be varied according to whether the protocol is dialling a dot or a dash. The *connect* action, taking time Y, is enabled simultaneously with the *ring* event and represents the amount of time the network takes to connect the call, and thus the amount of time before the receiving phone starts to ring. The stochastic transition system for the protocol is shown in figure 5.22.

As can be seen from figure 5.22, we use the fact that we are working with a pre-emptive resume strategy to give us the key transition[2] $(X|Y < X) \ominus (Y|Y < X)$. This is the residual time left by the *ring* action after it was pre-empted by the *connect* action and therefore, crucially, represents the time that the phone actually rings at the receiver's end.

[2]For some formulations of this problem, we discovered that we cannot obtain a well-defined independent distribution representing the residual $(X|Y < X)$ process. Further research is needed to understand when the $\ominus$ operator can be used legitimately.

$$
\begin{aligned}
R &\stackrel{\text{def}}{=} (ring, F_X(t)).R \\
C &\stackrel{\text{def}}{=} (connect, F_Y(t)).C \\
S &\stackrel{\text{def}}{=} (ring, \top).(reset, F_Z(t)).S \\
&\quad + (connect, \top).(ring, \top).(reset, F_Z(t)).S \\
P &\stackrel{\text{def}}{=} (R \,||\, C) \underset{\text{LTF}\{ring,connect\}}{||} S
\end{aligned}
$$

where $X \sim \exp(\lambda)$, $Y \sim \exp(\mu)$ and $Z \sim \delta(\nu)$

Fig. 5.21. A process algebra description of Morse code over a ringing phone.

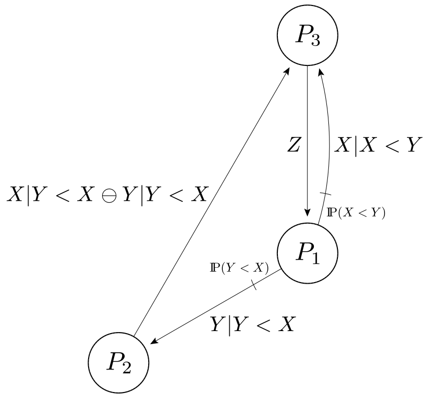

Fig. 5.22. Stochastic transition system for the Morse code protocol.

Reliability Analysis

There are three issues to be investigated in the system in figure 5.21. The first is the variance of the residual of the *ring* action; this will help ensure that for a given choice of length of ring, the actual ring heard is fairly tightly distributed. The second is a one-tailed test on the residual distribution which establishes what probability of error there is in getting a ring of dot length confused with a ring of dash length. The third issue is the probability of missing a dot or dash in transmission.

In the system of figure 5.22, the exponential interleaving allows us to say without calculation that $(X|Y < X) \ominus (Y|Y < X) \sim \exp(\lambda)$. Thus the variance of the residual is simply $1/\lambda^2$. This is really only an indication and tells us that, for small values of λ, there might well be a possibility of confusion of a dash with a dot, but not vice-versa.

In actual fact we have two *types* of system, one for which a dot is being transmitted where $X = X_1 \sim \exp(\lambda_1)$ and one for which a dash is being transmitted, $X = X_2 \sim \exp(\lambda_2)$. If we set $\lambda_1 = k\mu$ for a dot, $\lambda_2 = k\mu/2$ for a dash and the threshold for deciding which is which at the receiving end as γ, then we see that:

$$\mathbb{P}(\text{dot is not sent}) = p_1 = \mathbb{P}(X_1 < Y) = \frac{k}{k+1} \tag{5.101}$$

$$\mathbb{P}(\text{dash is not sent}) = p_2 = \mathbb{P}(X_2 < Y) = \frac{k}{k+2} \tag{5.102}$$

$$\mathbb{P}(\text{dot is misreceived as a dash}) = p_3 = \overline{F}_{X_1}(\gamma) = e^{-k\gamma\mu} \tag{5.103}$$

$$\mathbb{P}(\text{dash is misreceived as a dot}) = p_4 = F_{X_2}(\gamma) = 1 - e^{-k\gamma\mu/2} \tag{5.104}$$

Now for simplicity in the error-correcting coding layer that would sit above this protocol, it would be desirable for the transmission error probabilities to be the same, i.e. $p_3 = p_4$. Thus:

$$e^{-k\gamma\mu} = 1 - e^{-k\gamma\mu/2} \tag{5.105}$$

giving:

$$e^{-k\gamma\mu/2} = \frac{\sqrt{5}-1}{2} \tag{5.106}$$

$$\gamma = \frac{2}{k\mu} \ln\left(\frac{2}{\sqrt{5}-1}\right) \tag{5.107}$$

So:

$$\mathbb{P}(\text{transmission error}) = \frac{3-\sqrt{5}}{2} = 0.382 \tag{5.108}$$

which is independent of μ or k. This gives us the rather interesting result that we can reduce k arbitrarily to lower the non-transmission probabilities, p_1 and p_2, and providing the receiving end knows the values of k and μ in use, a constant transmission error of 38.2% can be predicted and factored into the coding algorithm.

The trade-off now becomes one of reliability to performance. Given a particularly low value of k for increased reliability, the bandwidth of the data link would obviously be correspondingly low.

Using stochastic aggregation of chapter 3, we obtain the following steady-state distribution:

$$\Pi(P_1) = \frac{\lambda}{(\lambda+\mu)(1+\lambda\nu)} \tag{5.109}$$

$$\Pi(P_2) = \frac{\mu}{(\lambda+\mu)(1+\lambda\nu)} \tag{5.110}$$

$$\Pi(P_3) = \frac{\lambda\nu}{1+\lambda\nu} \tag{5.111}$$

From this, we can obtain a bandwidth for the system. We deduce that the system transmits one bit for every cycle of the system, $P_1 \longrightarrow P_2 \longrightarrow P_3$. So if we can calculate how long the system takes to cycle in the steady-state then we will be able to calculate the bandwidth.

From chapter 3, we know that we can interpret the steady-state distribution as representing the proportion of time that the system spends in a given state. We know that in state P_3, the system will spend precisely ν seconds. Thus, we can deduce that the system will take:

$$\frac{1}{\mathbb{P}(Y<X)}\frac{\nu}{\Pi(P_3)} = \frac{\lambda+\mu}{\mu}\frac{1+\lambda\nu}{\lambda} \tag{5.112}$$

seconds to cycle round $P_1 \longrightarrow P_2 \longrightarrow P_3$, once it reaches its steady-state. Thus the bandwidth, B_P, is just the inverse of this figure in bits/second:

$$B_P = \frac{\lambda\mu}{(1+\lambda\nu)(\lambda+\mu)} \tag{5.113}$$

In figure 5.23, we plot, for dots and dashes, B_P, $1 - p_1$ and $1 - p_2$ against $1/k$, and see that for decreasing k the bandwidth tails off as predicted, while the probabilities of successful reception improve.

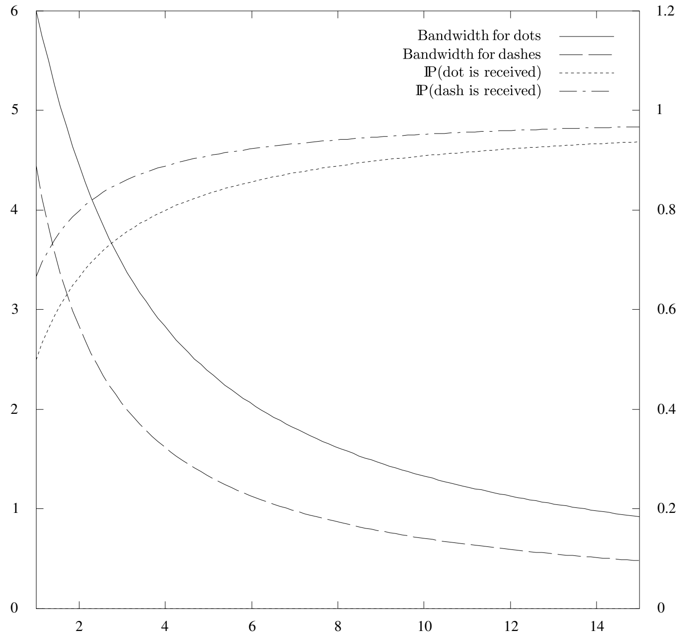

Fig. 5.23. The bandwidth of dots and dashes in bits per minute, and the equivalent probabilities of reception, for decreasing k.

To obtain at least a 99% probability of reception for a system with $1/\mu = 4$ and $\nu = 1$, say to the US, the bandwidth would be 0.1496 bits per minute for both dots and dashes. So perhaps the phone companies do not need to be particularly worried, though as a covert channel, maybe the NSA do!

Further analysis could be done with other calling distributions for R in figure 5.21. For instance an improvement in the error probabilities could certainly be made if a more condensed distribution such as an Erlang-k or a deterministic distribution were used, however the probabilistic error analysis would be correspondingly harder.

5.6 Conclusion

In this chapter, we have discussed the difference between average behaviour or performance and variant behaviour or reliability. Performance in stochastic systems is usually obtained from stationary distributions and represents long-run state probabilities. As such, it automatically discounts variation or short-term extreme behaviour.

What has been attempted here, therefore, is to show that by identifying a key transition or set of transitions, and by employing techniques to reduce the variance of such a transition, the reliability (i.e. determinism) of the whole system can be improved.

5.6.1 Equivalences for Model Checking

In summarising this chapter, we are keen to demonstrate how such a set of properties can be used to extend reliability analysis systematically and not necessarily just on a system by system basis.

Traditionally in process algebras this is achieved by constructing a bisimilarity, a relation that allies two processes. Such a relationship, once formally stated, can be systematically tested for and used at different levels of the system design process. Two models might display the same properties (if one is looking to replace the other), or a model and a specification (used for model refinement and invariant proof) or even design and specification (checking compliance to a formal standard perhaps).

Unfortunately we are in a dilemma, since although we have shown what reliability might mean in a stochastic setting, our examples demonstrate that it can be quite a model specific attribute. In what way model specific? Insofar as the modeller needs to know which parts of the model are reliability critical and for which parts it is not so important.

This is at once an advantage and a disadvantage. Computationally, we have seen that reliability need not necessarily be anything like as hard a quantity to calculate as, say, a steady-state distribution. This is lucky since general steady-state distribution solutions of generally-distributed systems are analytically beyond the current state-of-the-art. Reliability tends to be path specific and need not necessarily involve consideration of the whole system. Depending on how the system has been constructed in a process algebra, the reliability of a key part of the model may only involve a few local transitions.

This brings us to the disadvantage however, the fact that reliability analysis still requires quite a well informed modeller. The construction of the model is the first hurdle, and if done at too high a level of detail or looking at the wrong communication model abstraction, can make analysis prohibitively hard. Our experience with constructing the telephone example tells us that if the wrong type of model is constructed then reliability analysis can be almost as hard as steady-state analysis. Further, a fundamental understanding of the process is required to understand which parts of the system are reliability critical. This latter point is not necessarily a drawback though, since modelling is, after all, carried out to understand a system's operation better. Experience will clearly be needed at the modelling level.

We postulate the existence of some equivalences or bisimulations of generally-distributed models which might be of use. At the performance level, Larsen and Skou's probabilistic bisimulation [73] considered probabilistic flux through a Markovian process algebra; this clearly does not have a direct extension to a generally-distributed environment. However the intention is to equate systems which have the same long-term behaviour, so we might suggest a performance equivalence which equates generally-distributed systems with the same steady-state distribution. To allow model refinement and specification checking, we could allow systems to be stochastically aggregated into a form where they might be checked.

On the reliability side, we suggest two path reliability equivalences which operate at the local level, identified earlier. The first would be to equate paths of a system which have the same stochastic aggregation, that is that their aggregated passage time is the same. Finally, a weaker path equivalence would be to equate paths with the same mean and variance. On a similar basis, a reliability preorder could be established between paths which have the same mean but differing variance; the smaller the variance the more reliable the critical path.

These tools could then be applied to the relevant parts of a stochastically specified model to test for reliability constraints. Clearly more work is required to make these first attempts at generalising stochastic reliability more concrete.

Annex

5.A General Variance Results

In the following annex we include some general mean and variance results for specific process algebra combinators, where one of the distributions is left unspecified. Some of these results were used for direct calculation of the individual distribution results in section 5.4, others are included for completeness and reference.

For the analysis below, let $G_X(t) = \int_0^t F_X(u)\,\mathrm{d}u$.

5.A.1 Competitive Choice and First-to-Finish Synchronisation

Deterministic v General

Let X be a general distribution and $Y \sim \delta(\nu)$ in a competitive choice or First-to-Finish synchronisation. W represents the overall stochastic event and we calculate $\mathbb{E}(W)$ and $\mathbf{Var}(W)$ below:

$$\begin{aligned}
\overline{F}_W(t) &= \overline{F}_X(t)\overline{F}_Y(t) \\
&= F_X(t)H(\nu - t) \\
f_W(t) &= f_X(t)H(\nu - t) + \overline{F}_X(t)\delta(\nu - t) && (5.114)
\end{aligned}$$

$$\begin{aligned}
\mathbb{E}(W) &= \int_0^\nu t f_X(t)\,\mathrm{d}t + \nu\overline{F}_X(\nu) \\
&= [tF_X(t)]_0^\nu - \int_0^\nu F_X(t)\,\mathrm{d}t + \nu\overline{F}_X(\nu) \\
&= \nu - G_X(\nu) && (5.115)
\end{aligned}$$

$$\begin{aligned}
\mathbb{E}(W^2) &= \int_0^{\nu} t^2 f_X(t)\,\mathrm{d}t + \nu^2 \overline{F}_X(\nu) \\
&= [t^2 F_X(t)]_0^{\nu} - 2\int_0^{\nu} t F_X(t)\,\mathrm{d}t + \nu^2 \overline{F}_X(t) \\
&= \nu^2 - 2\nu G_X(\nu) + 2\int_0^{\nu} G_X(t)\,\mathrm{d}t
\end{aligned} \tag{5.116}$$

$$\mathbf{Var}(W) = 2\int_0^{\nu} G_X(t)\,\mathrm{d}t - {G_X}^2(\nu) \tag{5.117}$$

Exponential v General

Let X be a general distribution and $Y \sim \exp(\lambda)$ in a competitive choice or First-to-Finish synchronisation. W represents the overall stochastic event and we calculate $\mathbb{E}(W)$ and $\mathbf{Var}(W)$ below:

$$\begin{aligned}
\overline{F}_W(t) &= \overline{F}_X(t)\overline{F}_Y(t) \\
&= F_X(t)e^{-\lambda t} \\
f_W(t) &= \lambda e^{-\lambda t} + f_X(t)e^{-\lambda t} - F_X(t)\lambda e^{-\lambda t}
\end{aligned} \tag{5.118}$$

$$\begin{aligned}
L_W(\omega) &= L_Y(\omega) + \int_0^{\infty} (f_X(t) - \lambda F_X(t))e^{-(\lambda+\omega)t}\,\mathrm{d}t \\
&= L_Y(\omega) + L_{\psi}(\lambda + \omega)
\end{aligned} \tag{5.119}$$

where:

$$\psi(t) = f_X(t) - \lambda F_X(t) \tag{5.120}$$

Now we can calculate the mean and variance:

$$\mathbb{E}(W) = \frac{1}{\lambda} - L'_{\psi}(\lambda) \tag{5.121}$$

$$\mathbb{E}(W^2) = \frac{2}{\lambda^2} + L''_{\psi}(\lambda) \tag{5.122}$$

Thus:

$$\mathbf{Var}(W) = \frac{1}{\lambda^2} + L''_{\psi}(\lambda) + \frac{2}{\lambda}L'_{\psi}(\lambda) - (L'_{\psi}(\lambda))^2 \tag{5.123}$$

5.A.2 Last-to-Finish Synchronisation

Deterministic v General

Let X be a general distribution with mean μ and $Y \sim \delta(\nu)$ in a Last-to-Finish synchronisation. W represents the overall stochastic event and we calculate $\mathbb{E}(W)$ and $\mathbf{Var}(W)$ below:

$$\begin{aligned} F_W(t) &= F_X(t)F_Y(t) \\ &= F_X(t)H(t-\nu) \\ f_W(t) &= f_X(t)H(t-\nu) + F_X(t)\delta(t-\nu) \end{aligned} \tag{5.124}$$

$$\begin{aligned} \mathbb{E}(W) &= \int_\nu^\infty t f_X(t)\,\mathrm{d}t + \nu F_X(\nu) \\ &= \mu + \nu F_X(\nu) - \int_0^\nu t f_X(t)\,\mathrm{d}t \\ &= \mu + G_X(\nu) \end{aligned} \tag{5.125}$$

$$\begin{aligned} \mathbb{E}(W^2) &= \int_\nu^\infty t^2 f_X(t)\,\mathrm{d}t + \nu^2 F_X(\nu) \\ &= \mathbb{E}(X^2) + \nu^2 F_X(\nu) - \int_0^\nu t^2 f_X(t)\,\mathrm{d}t \\ &= \mathbb{E}(X^2) + 2\nu G_X(\nu) - 2\int_0^\nu G_X(t)\,\mathrm{d}t \end{aligned} \tag{5.126}$$

$$\mathbf{Var}(W) = \mathbf{Var}(X) + 2(\nu-\mu)G_X(\nu) - 2\int_0^\nu G_X(t)\,\mathrm{d}t - {G_X}^2(\nu) \tag{5.127}$$

Exponential v General

Let X be a general distribution with mean μ and $Y \sim \exp(\lambda)$ in a Last-to-Finish synchronisation. W represents the overall stochastic event and we calculate $\mathbb{E}(W)$ and $\mathbf{Var}(W)$ below:

$$\begin{aligned} F_W(t) &= F_X(t)F_Y(t) \\ &= F_X(t)(1-e^{-\lambda t}) \\ f_W(t) &= f_X(t)(1-e^{-\lambda t}) + F_X(t)\lambda e^{-\lambda t} \end{aligned} \tag{5.128}$$

$$\begin{aligned} L_W(\omega) &= -\int_0^\infty (f_X(t) - \lambda F_X(t))e^{-(\lambda+\omega)t}\,\mathrm{d}t + \int_0^\infty f_X(t)e^{-\omega t}\,\mathrm{d}t \\ &= L_X(\omega) - L_\psi(\lambda+\omega) \end{aligned} \tag{5.129}$$

where:

$$\psi(t) = f_X(t) - \lambda F_X(t) \tag{5.130}$$

Now we can calculate the mean and variance:

$$\mathbb{E}(W) = \mathbb{E}(X) + L'_\psi(\lambda) \tag{5.131}$$

$$\mathbb{E}(W^2) = \mathbb{E}(X^2) - L''_\psi(\lambda) \tag{5.132}$$

Thus:

$$\mathbf{Var}(W) = \mathbf{Var}(X) - L''_\psi(\lambda) - 2\mathbb{E}(X)L'_\psi(\lambda) - (L'_\psi(\lambda))^2 \tag{5.133}$$

5.B Minimum and Maximum Distributions of Random Variables

We want to demonstrate that the conditional expansion of the maximum and minimum distributions as described by competitive choice, First-to-Finish synchronisations and Last-to-Finish synchronisations (in section 5.3 and used in derivations in figures 5.1 and 5.2) are identical to the cumulative distribution function versions used in chapter 4 (equations (4.2) and (4.6)).

5.B.1 Maximum Distribution

The maximum distribution is used for Last-to-Finish synchronisation. We claim that for a random variable $W = \max(X, Y)$:

$$\begin{aligned} F_W(t) &= \mathbb{P}(X > Y)F_{X|X>Y}(t) + \mathbb{P}(Y > X)F_{Y|Y>X}(t) \\ &= F_X(t)F_Y(t) \end{aligned} \tag{5.134}$$

Thus:

$$\begin{aligned} F_W(t) &= \mathbb{P}(X > Y)F_{X|X>Y}(t) + \mathbb{P}(Y > X)F_{Y|Y>X}(t) \\ &= \mathbb{P}(X \le t, X > Y) + \mathbb{P}(Y \le t, Y > X) \\ &= \int_0^t F_Y(u)f_X(u) + F_X(u)f_Y(u)\,\mathrm{d}u \\ &= \int_0^t \frac{\mathrm{d}}{\mathrm{d}u}\left[F_X(u)F_Y(u)\right]\,\mathrm{d}u \\ &= \left[F_X(u)F_Y(u)\right]_0^t \end{aligned} \tag{5.135}$$

Since $F_X(0)F_Y(0) = 0$:

$$F_W(t) = F_X(t)F_Y(t) \tag{5.136}$$

■

5.B.2 Minimum Distribution

The minimum distribution is used for both competitive choice and First-to-Finish synchronisation. We claim that for a random variable $W = \min(X, Y)$[3]:

$$\begin{aligned} F_W(t) &= \mathbb{P}(X < Y)F_{X|X<Y}(t) + \mathbb{P}(Y < X)F_{Y|Y<X}(t) \\ &= 1 - \overline{F}_X(t)\overline{F}_Y(t) \end{aligned} \tag{5.137}$$

[3] This repeats a result from chapter 3, equations (3.59)–(3.60), displayed here for completeness and ease-of-reference.

Thus:

$$\begin{aligned}
F_W(t) &= \mathbb{P}(X < Y) F_{X|X<Y}(t) + \mathbb{P}(Y < X) F_{Y|Y<X}(t) \\
&= \mathbb{P}(X \leq t, X < Y) + \mathbb{P}(Y \leq t, Y < X) \\
&= \int_0^t \overline{F}_Y(u) f_X(u) + \overline{F}_X(u) f_Y(u)\,\mathrm{d}u \\
&= \int_0^t -\frac{\mathrm{d}}{\mathrm{d}u}\left[\overline{F}_X(u)\overline{F}_Y(u)\right]\,\mathrm{d}u \\
&= \left[-\overline{F}_X(u)\overline{F}_Y(u)\right]_0^t \qquad (5.138)
\end{aligned}$$

Since $\overline{F}_X(0)\overline{F}_Y(0) = 1$:

$$F_W(t) = 1 - \overline{F}_X(t)\overline{F}_Y(t) \qquad (5.139)$$

■

Chapter 6

Conclusions

6.1 Overview

In this thesis, we have examined reliable modelling in the context of stochastic process algebras. This took the form of improved treatment of generally-distributed stochastic processes, improved accuracy of performance measures in Markovian process algebras, and finally, reduced model variance for greater predictability and determinism within systems.

6.2 Summary

6.2.1 Introduction and Motivation

In chapter 1, we set out our reasoning and motivation for considering time in addition to functional operation. In modelling temporal behaviour, we reason that we are observing the full behaviour of a system. Thus we are able to make statements about the reliability of operation of the system which are time dependent and not just functionality dependent.

This train of reasoning is not a particularly new one, but in fact underpins the development of all the timed and stochastic process algebras, introduced in chapter 2. Here, we have two goals: to demonstrate the progressive refinement of process algebras from the purely functional to the stochastic; and also to show what stochastic process algebras are currently good at doing and where their analytical shortcomings might lie.

We identify that for Markovian process algebras, the solution techniques are well developed for obtaining steady-state solutions of large Markov processes. However, as the distributions get more general, so the underlying techniques become less capable of dealing with larger models. In all cases, though, our main complaint was that the solution techniques concentrated on the average behaviour of the steady-state solution and thus ignored any of the tail-end behaviour which might cause problems such as network *jitter*.

We also noted that the two longest standing Markovian process algebras, which have been studied and honed the most, make an approximation in a key area of the process algebra model: that of synchronisation.

Thus, we develop our approach for the thesis: a dual strategy to aid the development of reliable models. The first strand, to investigate methods of analysing generally-distributed processes which maintain the integrity of both the model and distribution. By this we mean, a method which does not alter the model, as insensitivity analysis does, and does not discard tail-end behaviour, as in steady-state analysis. This motivates our work in chapter 3, in which we develop the technique of stochastic aggregation.

6.2.2 Synchronisation

Our second strand of attack was implemented in chapter 4, where we examine the issue of synchronisation in stochastic process algebras and the Markovian process algebras, MTIPP and PEPA, in particular. In recognising that generally-distributed analysis techniques are going to be affected by space and time complexity issues, we would like to improve the reliability of the performance figures that these two algebras were able to produce. We gain an understanding of the different approximations that MTIPP and PEPA made in synchronisation and how they affected their respective performance figures.

Now we can use the advanced tools (such as PEPA Workbench [33] and TIPPtool [46]) and solution techniques that exist in the Markovian area (such as quasi-reversibility [40], and quasi-separability [93] and similar product-form solutions). We can obtain efficient solutions to a given model and, crucially, still understand how reliable this solution is likely to be. We may even be in a position to improve the accuracy of the technique by running multiple model versions with some of the different synchronisation strategies suggested in chapter 4.

6.2.3 Reliability Modelling

Having developed the stochastic aggregation method of chapter 3, we were keen to see what tail-end behaviour we could model, having kept the distribution information intact. We quickly realised that one of the key areas of reliable modelling is that of keeping the variance small over critical paths of execution. We saw that, if modelling with too diffuse a distribution, there is a tendency to get a variance explosion problem; generated primarily when using the sequential prefix combinator in stochastic process algebras. Most of the chapter is then dedicated to examining which combinators and which distribution types are good at reducing variance. We illustrated the point with a searching algorithm which had its variance considerably improved by the use of an appropriate combination with another technique.

We finished with the insight that reliable modelling and reliability in stochastic systems is a very diverse and problem-specific entity: in the same way that performance modelling talks about issues such as bandwidth, resource utilisation and throughput. However, we envisage that progress will be best made by formally expressing concepts such as variance reduction or tail-test invariants in bisimulations. Only when this has happened formally can we systematically model with these reliability goals in mind. However, in the end, it still requires the modeller to know what quantity is critically important to the problem and how that might be best translated into a stochastic test on an SPA model.

6.3 Specific Results

6.3.1 Stochastic Aggregation and Probabilistic Semantics

In chapter 3, we show how stochastic aggregation can be used to solve some generally-distributed processes for their steady-state distributions. We also demonstrate how it can be used to solve CTMCs, through analysing Markovian process algebra models, in a progressive fashion with a small saving in complexity terms. Through an example, we suggest how stochastic aggregation might be used to analyse systems in a component-wise fashion: aggregating the behaviour of other system components into a single variable delay for a particular component, and thus solving a structurally simpler model.

We also presented a form of probabilistic operational semantics, which act as a direct mapping on to the stochastic transition systems of chapter 3. This allowed us to specify probabilistically two Markovian process algebras (PEPA and MTIPP), whereas previously they had only been specified using the traditional functional operational semantics. We also defined a generally-distributed, pre-emptive restart stochastic process algebra with two synchronisation models, FTF and LTF (figure 3.27).

6.3.2 Synchronisation Classification and Reliable Markovian Modelling

Chapter 4 coined the ideas of First-to-Finish and Last-to-Finish synchronisation. It demonstrated that FTF synchronisation could be implemented precisely in Markovian process algebras whereas LTF could not. In approximating LTF synchronisation, we showed that MTIPP was not consistent in its relative prediction, whereas PEPA would tend to overestimate performance figures. We showed that, in fact, no Markovian process algebra could consistently underestimate our performance metrics, but nonetheless came up with a model, PEPA*, which was effective at usually underestimating performance. We were therefore able to suggest using both PEPA and PEPA* to obtain a more reliable estimate for the synchronisation. We analysed a third synchronisation operator, the mean-preserving synchronisation, which was essentially an approximate insensitive transition and came closest to the true solution. Finally, we managed to restrict the parameter space of the MTIPP synchronisation, so that it would emulate the mean-preserving synchronisation.

6.3.3 Classification of Variance Reducing Operators and Distributions

In chapter 5, we identified increasing variance as a problem that would make systems unreliable and simulation meaningless. We take the view that, if stochastic process algebra modelling is carried out at the design stage of a project, then this variance explosion can be mitigated. To achieve this we were able to provide an analysis of which distributions, and under which combinators, would reduce variance. We also provided useful metrics for measuring this variance reduction.

6.4 Further Investigation

There are two main areas of investigation that we would like to explore further: that of improved generally-distributed stochastic analysis, which would allow representation of a wider class of processes; and also a more systematic approach to reliability modelling.

6.4.1 Component Aggregation

We view as particularly interesting the development of the component aggregation technique, demonstrated in chapter 3 and also [10]. The problem area of stochastic modelling and analysis is that of pre-emptive resume systems. While we were able to provide steady-state analysis of general pre-emptive restart systems (section 3.3.1), we were also able to show exactly why an interleaving expansion approach could not be used for pre-emptive resume systems (section 3.3.2)[1]. If the process of component aggregation could be refined to deal with cyclically dependent components then it might well be possible to obtain analysis of generally-distributed pre-emptive resume systems which is currently not possible. The fact that this would be embodied in a stochastic process algebra environment rather than say, a queueing system environment, would also automatically give it greater applicability.

[1] We wonder whether the non-existence of a finite interleaving for even the simplest of interacting systems (a G/G/1/3 queue, for example) might not have wider consequences outside of the field of stochastic modelling. Simulation of such interacting processes inevitably involves synchronising with the internal clock of the system that the simulation is running on; this synchronisation would clearly make the state-space finite again and may well destroy the equivalence with the system being simulated. What consequences does this have for simulation of a system which is completely constructed of such simple interactions, for example neural simulations?

6.4.2 Reliability through Feature Modelling

Given that issues of reliability are quite diverse and model-specific, we see the progression of this area in the development of a reliability description language. We see the presentation of desirable features and properties of a model through a form of modal calculus, as presented by Gilmore [35, 36], as a particularly interesting technique. This approach might well lead to a form of temporal model checking which could be automated and incorporated into a tool.

Another approach, maybe more expressive but correspondingly less systematic, involves testing stochastic properties of a model with small synchronising fragments of stochastic process algebra. These so-called *stochastic process algebra probes* [27] would passively measure key properties (sequences of observable actions) and, through the analysis of the combined model-probe system, it would be ascertained whether a property had been met or not.

Bibliography

[1] Ajmone Marsan, M., Balbo, G., and Conte, G. *Performance Models of Multiprocessor Systems.* MIT Press Series in Computer Systems. The MIT Press, 1986.

[2] Ajmone Marsan, M., Bianco, A., Cimminiera, L., Sisto, R., and Valenzano, A. A LOTOS extension for the performance analysis of distributed systems. *IEEE/ACM Transactions on Networking 2*, 2 (April 1994), 151–165.

[3] Ajmone Marsan, M., and Chiola, G. On Petri nets with deterministic and exponentially distributed firing times. In *APN'87, Advances in Petri Nets, Proceedings of the 7th European Workshop on Applications and Theory of Petri Nets* (Oxford, June 1987), G. Rozenberg, Ed., vol. 266 of *Lecture Notes in Computer Science*, Springer-Verlag, pp. 132–145.

[4] Ajmone Marsan, M., Conte, G., and Balbo, G. A class of generalized stochastic Petri nets for the performance evaluation of multiprocessor systems. *ACM Transactions on Computer Systems 2*, 2 (May 1984), 93–122.

[5] Allen, A. O. *Probability, Statistics and Queueing Theory with Computer Science Applications*, 2nd ed. Academic Press, 1990.

[6] Baeten, J. C. M., and Bergstra, J. A. Real time process algebra. *Formal Aspects of Computing 3*, 2 (April-June 1991), 142–188.

[7] Bergstra, J. A., and Klop, J. W. Algebra for communicating processes with abstraction. *Journal of Theoretical Computer Science 37*, 1 (May 1985), 77–121.

[8] Bernardo, M. *Theory and Application of Extended Markovian Process Algebra.* PhD thesis, Department of Computer Science, University of Bologna, February 1999.

[9] Bernardo, M., and Gorrieri, R. Extended Markovian Process Algebra. In *CONCUR'96, Proceedings of the 7th International Conference on Concurrency Theory* (Pisa, August 1996), U. Montanari and V. Sassone, Eds., vol. 1119 of *Lecture Notes in Computer Science*, Springer-Verlag, pp. 315–330.

[10] Bohnenkamp, H. C., and Haverkort, B. R. Stochastic event structures for the decomposition of stochastic process algebra models. In Hillston and Silva [57], pp. 25–39.

[11] Bowman, H., Bryans, J. W., and Derrick, J. Analysis of a multimedia stream using stochastic process algebras. In Priami [84], pp. 51–69.

[12] Bradley, J. T., and Davies, N. J. An aggregation technique for analysing some generally distributed stochastic processes. CSTR Technical Report CSTR–99–003, Department of Computer Science, University of Bristol, Bristol BS8 1UB, UK, March 1999.

[13] Bradley, J. T., and Davies, N. J. Measuring improved reliability in stochastic systems. In *UKPEW'99, Proceedings of the 15th Annual UK Performance Engineering Workshop* [14], pp. 121–130.

[14] Bradley, J. T., and Davies, N. J., Eds. *Proceedings of the 15th Annual UK Performance Engineering Workshop* (Bristol, July 1999),

CSTR Technical Report, Department of Computer Science, University of Bristol, Research Press. CSTR–99–007, ISBN 0 9524027 8 5.

[15] Bradley, J. T., and Davies, N. J. Reliable performance modelling with approximate synchronisations. In Hillston and Silva [57], pp. 99–118.

[16] Bradley, J. T., and Davies, N. J. Reliable performance modelling with approximate synchronisations. CSTR Technical Report CSTR–99–002, Department of Computer Science, University of Bristol, Bristol BS8 1UB, UK, February 1999.

[17] Bravetti, M., Bernardo, M., and Gorrieri, R. GSMPA: A core calculus with generally distributed durations. UBLCS Report UBLCS–98–6, Department of Computer Science, University of Bologna, 40127 Bologna, Italy, June 1998.

[18] Bravetti, M., Bernardo, M., and Gorrieri, R. Towards performance evaluation with general distributions in process algebras. In Sangiorgi and de Simone [88], pp. 405–422.

[19] Brinksma, E., Katoen, J.-P., Langerak, R., and Latella, D. A Stochastic Causality-based Process Algebra. In Gilmore and Hillston [34], pp. 553–565.

[20] Buchholz, P. Markovian process algebra: Composition and equivalence. In Ribaudo [86], pp. 11–30.

[21] Chen, L., Anderson, S., and Moller, F. A timed calculus of communicating systems. LFCS Report ECS–LFCS–90–127, Department of Computer Science, University of Edinburgh, Edinburgh EH9 3JZ, UK, December 1990.

[22] Clark, G. Stochastic process algebra structure for insensitivity. In Hillston and Silva [57], pp. 63–82.

[23] COHEN, J. W. *The Single Server Queue*, vol. 8 of *Applied Mathematics and Mechanics*. North-Holland, 1969.

[24] COX, D. R. The analysis of non-Markovian stochastic processes by the inclusion of supplementary variables. *Proceedings of the Cambridge Philosophical Society 51*, 3 (1955), 433–441.

[25] COX, D. R. A use of complex probabilities in the theory of stochastic processes. *Proceedings of the Cambridge Philosophical Society 51*, 3 (1955), 313–319.

[26] COX, D. R., AND MILLER, H. D. *The Theory of Stochastic Processes*. Methuen, 1965.

[27] DAVIES, N. J. Stochastic process algebra probes. Personal communication, January 1999.

[28] DIRAC, P. A. M. Bakerian lecture: The physical interpretation of quantum mechanics. In *Proceedings of the Royal Society of London* (1942), vol. 180 of *Series A—Maths and Physics*, The Royal Society, pp. 1–40.

[29] EL-RAYES, A., KWIATKOWSKA, M., AND NORMAN, G. Solving infinite stochastic process algebra models through matrix-geometric methods. In Hillston and Silva [57], pp. 41–62.

[30] FELLER, W. *An Introduction to Probability Theory and its Applications*, 3rd ed., vol. 1. John Wiley, 1968.

[31] FOURNEAU, J. M., KLOUL, L., AND VALOIS, F. Performance modelling of hierarchical cellular networks using PEPA. In Hillston and Silva [57], pp. 139–154.

[32] GIBBONS, A. *Algorithmic Graph Theory*. Cambridge University Press, 1985.

[33] Gilmore, S., and Hillston, J. The PEPA workbench: A tool to support a process algebra-based approach to performance modelling. In *Proceedings of the 7th International Conference on Modelling Techniques and Tools for Computer Performance Evaluation* (Vienna, May 1994), G. Haring and G. Kotsis, Eds., vol. 794 of *Lecture Notes in Computer Science*, Springer-Verlag, pp. 353–368.

[34] Gilmore, S., and Hillston, J., Eds. *PAPM'95, Proceedings of the 3rd International Workshop on Process Algebra and Performance Modelling* (Edinburgh, June 1995), vol. 38(7) of *Special Issue: The Computer Journal*, CEPIS. ISSN 0010–4620.

[35] Gilmore, S., and Hillston, J. Feature interaction in PEPA. In Priami [84], pp. 17–26.

[36] Gilmore, S., and Hillston, J. A feature construct for PEPA. In Bradley and Davies [14], pp. 225–236.

[37] Götz, N., Herzog, U., and Rettelbach, M. TIPP—Introduction and application to protocol performance analysis. In *Formale Beschreibungstechniken für verteilte Systeme* (Munich 1993), H. König, Ed., FOKUS, Saur-Verlag.

[38] Gross, D., and Harris, C. M. *Fundamentals of Queueing Theory*, 3rd ed. John Wiley & Sons, 1998.

[39] Harrison, P. G. Laplace transform inversion and passage-time distributions in Markov processes. *Journal of Applied Probability 27* (1990), 74–87.

[40] Harrison, P. G., and Hillston, J. Exploiting quasi-reversible structures in Markovian process algebra models. In Gilmore and Hillston [34], pp. 510–520.

[41] Henderson, W. Insensitivity and reversed-Markov processes. *Advances in Applied Probability 15* (1983), 752–768.

[42] Henderson, W., and Lucic, D. Aggregation and disaggregation through insensitivity in stochastic Petri nets. *Performance Evaluation 17* (1993), 91–114.

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

[44] Hermanns, H., Herzog, U., and Hillston, J. Stochastic process algebras—A formal approach to performance modelling. Tutorial, Department of Computer Science, University of Edinburgh, Edinburgh EH9 3JZ, UK, 1996.

[45] Hermanns, H., and Lohrey, M. Priority and maximal progress are completely axiomatisable. In Sangiorgi and de Simone [88], pp. 237–252.

[46] Hermanns, H., and Mertsiotakis, V. A stochastic process algebra based modelling tool. In *UKPEW'95, Proceedings of the 11th UK Performance Engineering Workshop on Computer and Telecommunication Systems* (Liverpool, September 1995), M. Merabti, M. Carew, and F. Ball, Eds., Liverpool John Moores University, Springer.

[47] Hermanns, H., and Rettelbach, M. Syntax, semantics, equivalences, and axioms for MTIPP. In Herzog and Rettelbach [49], pp. 69–88.

[48] Hermanns, H., Rettelbach, M., and Weiss, T. Formal characterisation of immediate actions in SPA with nondeterministic branching. In Gilmore and Hillston [34], pp. 530–541.

[49] Herzog, U., and Rettelbach, M., Eds. *PAPM'94, Proceedings of the 2nd International Workshop on Process Algebra and Performance Modelling* (Regensberg, July 1994), Arbeitsberichte des IMMD, Universtät Erlangen-Nürnberg.

[50] Hillston, J. PEPA: A performance enhanced process algebra. In *Process Algebra and Performance Modelling* (Edinburgh, May 1993),

J. Hillston and F. Moller, Eds., CSR Technical Report, Department of Computer Science, University of Edinburgh, pp. 21–29.

[51] Hillston, J. PEPA: Performance enhanced process algebra. CSR Technical Report CSR–24–93, Department of Computer Science, University of Edinburgh, Edinburgh EH9 3JZ, UK, March 1993.

[52] Hillston, J. *A Compositional Approach to Performance Modelling.* PhD thesis, Department of Computer Science, University of Edinburgh, Edinburgh EH9 3JZ, UK, 1994. CST–107–94.

[53] Hillston, J. The nature of synchronisation. In Herzog and Rettelbach [49], pp. 49–68.

[54] Hillston, J. Compositional Markovian modelling using a process algebra. In *Proceedings of the 2nd International Workshop on Numerical Solution of Markov Chains* (Raleigh, January 1995), Kluwer Academic Press.

[55] Hillston, J. *A Compositional Approach to Performance Modelling*, vol. 12 of *Distinguished Dissertations in Computer Science.* Cambridge University Press, 1996. ISBN 0 521 57189 8.

[56] Hillston, J., and Kloul, L. Investigating an on-line auction system using PEPA. In Bradley and Davies [14], pp. 143–153.

[57] Hillston, J., and Silva, M., Eds. *PAPM'99, Proceedings of the 7th International Workshop on Process Algebra and Performance Modelling* (Zaragoza, September 1999), Centro Politécnico Superior de la Universidad de Zaragoza, Prensas Universitarias de Zaragoza. ISBN 84 7733 513 3.

[58] Hillston, J., and Thomas, N. Product form solution of a class of PEPA models. In *IPDS'98, Proceedings of the IEEE International Computer Performance and Dependability Symposium* (Durham NC, September 1998), IEEE, pp. 152–161.

[59] Hillston, J., and Thomas, N. A syntactical analysis of reversible PEPA models. In Priami [84], pp. 37–49.

[60] Hoare, C. A. R. Communicating sequential processes. *Communications of the ACM 21*, 8 (August 1978), 666–677.

[61] Hoare, C. A. R. *Communicating Sequential Processes.* PHI Series in Computer Science. Prentice Hall, 1985. ISBN 0 13 153289 8.

[62] Holton, D. R. W. A PEPA specification of an industrial production cell. In Gilmore and Hillston [34], pp. 542–551.

[63] Howard, R. A. *Dynamic Programming and Markov Processes.* John Wiley, 1960.

[64] Howard, R. A. *Dynamic Probabilistic Systems: Markov Models*, vol. 1 of *Series in Decision and Control.* John Wiley & Sons, 1971.

[65] Howard, R. A. *Dynamic Probabilistic Systems: Semi-Markov and Decision Processes*, vol. 2 of *Series in Decision and Control.* John Wiley & Sons, 1971.

[66] Keilson, J. The ergodic queue length distribution for queueing systems with finite capacity. *Journal of the Royal Statistical Society B*, 28 (1966), 190–201.

[67] Kelly, F. P. *Reversibility and Stochastic Networks.* Wiley Series in Probability and Mathematical Statistics. John Wiley & Sons, 1979.

[68] Kemeny, J. G., and Snell, J. L. *Finite Markov Chains.* Van Nostrand, 1960.

[69] Kendall, D. G. Some problems in the theory of queues. *Journal of the Royal Statistical Society B–13*, 2 (1951), 151–185.

[70] Kleinrock, L. *Queueing Systems, Volume I: Theory.* John Wiley & Sons, 1975.

[71] Kleinrock, L. *Queueing Systems, Volume II: Computer Applications.* John Wiley & Sons, 1976.

[72] Koymans, R. Specifying real-time properties with Metric Temporal Logic. *Real Time Systems 2*, 4 (1990), 255–290.

[73] Larsen, K. G., and Skou, A. Bisimulation through probabilistic testing. *Information and Computation 94* (1991), 1–28.

[74] Lindemann, C. *Performance Modelling with Deterministic and Stochastic Petri Nets.* John Wiley & Sons, 1998.

[75] Matthes, J. Zur Theorie der Bedienungsprozesse. In *Transactions of the 3rd Prague Conference on Information Theory, Statistical Decision Functions and Random Processes* (1962), pp. 513–528.

[76] Milner, R. *A Calculus of Communicating Systems*, vol. 92 of *Lecture Notes in Computer Science.* Springer-Verlag, 1980.

[77] Milner, R. Calculi for synchrony and asynchrony. *Journal of Theoretical Computer Science 25* (1983), 267–310.

[78] Milner, R. *Communication and Concurrency.* PHI Series in Computer Science. Prentice Hall, 1989. ISBN 0 13 115007 3.

[79] Moller, F., and Tofts, C. M. N. A temporal calculus of communicating systems. LFCS Report ECS–LFCS–89–104, Department of Computer Science, University of Edinburgh, Edinburgh EH9 3JZ, UK, December 1989.

[80] Molloy, M. K. Performance analysis using stochastic Petri nets. *IEEE Transactions on Computers 31*, 9 (September 1982), 913–917.

[81] Petri, C. A. *Communications with Automata.* Technical report RADC–TR–65–377, New York, January 1966.

[82] Plotkin, G. D. A structured approach to operational semantics. Technical Report DAIMI FM–19, Department of Computer Science, Aarhus University, DK–8000 Aarhus C, Denmark, 1981.

[83] PRIAMI, C. A stochastic π-calculus with general distributions. In Ribaudo [86], pp. 41–57.

[84] PRIAMI, C., Ed. *PAPM'98, Proceedings of the 6th International Workshop on Process Algebra and Performance Modelling* (Nice, September 1998), Università Degli Studi di Verona.

[85] RETTELBACH, M. Probabilistic branching in Markovian process algebras. In Gilmore and Hillston [34], pp. 590–599.

[86] RIBAUDO, M., Ed. *PAPM'96, Proceedings of the 4th International Workshop on Process Algebra and Performance Modelling* (Torino, July 1996), CLUT. ISBN 8 879 92120 7.

[87] RUMSEWICZ, M., AND HENDERSON, W. Insensitivity with age-dependent routing. *Advances in Applied Probability 21* (1989), 398–408.

[88] SANGIORGI, D., AND DE SIMONE, R., Eds. *International Conference on Concurrency Theory* (Nice, September 1998), vol. 1466 of *Lecture Notes in Computer Science*, Springer-Verlag.

[89] SCHIEFERDECKER, I. Verifying performance oriented properties of TIS specifications. *Journal of Mathematical Modelling and Simulation in Systems Analysis* (October 1993).

[90] SCHIEFERDECKER, I., AND WOLISZ, A. Verifying deterministic performance bounds of communication protocols. In *Third International Workshop on Responsive Computer Systems* (Lincoln, New Hampshire, October 1993).

[91] SERENO, M. Towards a product form solution for stochastic process algebras. In Gilmore and Hillston [34], pp. 622–632.

[92] STARK, E. W., AND PEMMASANI, G. Implementation of a compositional performance analysis algorithm for probabilistic I/O automata. In Hillston and Silva [57], pp. 3–24.

[93] Thomas, N. Extending quasi-separability. In Bradley and Davies [14], pp. 131–142.

[94] Thomas, N., and Gilmore, S. Applying quasi-separability to Markovian process algebras. In Priami [84], pp. 27–36.

[95] Tofts, C. M. N. Temporal ordering for concurrency. LFCS Report ECS–LFCS–88–49, Department of Computer Science, University of Edinburgh, Edinburgh EH9 3JZ, UK, April 1988.

[96] Tofts, C. M. N. Timing concurrent processes. LFCS Report ECS–LFCS–89–103, Department of Computer Science, University of Edinburgh, Edinburgh EH9 3JZ, UK, December 1989.

[97] Trivedi, K. S. *Probability and Statistics with Reliability, Queuing and Computer Science Applications.* Prentice-Hall, 1982.

[98] Watson, E. J. *Laplace Transforms and Applications.* Van Nostrand Reinhold, 1981.

[99] Wilson, H. J., and Bradley, J. T. A note on the proof of reduction of biconnected digraphs to normal forms. CSTR Technical Report CSTR–99–009, Department of Computer Science, University of Bristol, Bristol BS8 1UB, UK, September 1999.

[100] Wolisz, A., Schieferdecker, I., and Walch, M. An integrated approach to the design of communication protocols. In *Fourth Workshop on Future Trends of Distributed Computer Systems* (Lisbon 1993).

Index