Verification of Random Behaviours

S. Andova[1] and T.A.C. Willemse[2]

[1] Department of Computer Science,
Twente University, P.O. Box 217,
7500 AE Enschede, The Netherlands
suzana@cs.utwente.nl
[2] Department of Mathematics and Computer Science,
Eindhoven University of Technology, P.O. Box 513,
5600 MB Eindhoven, The Netherlands
T.A.C.Willemse@tue.nl

Abstract. We introduce abstraction in a probabilistic process algebra. The process algebra can be employed for specifying processes that exhibit both probabilistic and non-deterministic choices in their behaviours. Several rules and axioms are identified, allowing us to rewrite processes to less complex processes by removing redundant internal activity. Using these rules, we have successfully conducted a verification of the *Concurrent Alternating Bit Protocol*. The verification shows that after abstraction of internal activity, the protocol behaves as a buffer.
keywords: Process Algebra, Probabilistic Systems, Verification, Abstraction

1 Introduction

Formal methods have been used extensively for the verification of complex systems. Depending on the formalism that is used, specific parts of the system can be described in full detail, whereas other parts are abstracted from. This sometimes leaves a large gap between the specification that is investigated, and the real-life system that is used in practice. Therefore, much research is committed to investigating how new concepts fit into an existing theory. In this paper, we investigate how the notion of abstraction fits in a theory that combines both probabilistic and non-deterministic choices.

The combination of probabilities and non-determinism in a setting without abstraction is well-studied, and various competing and complementary theories have been developed for this [15, 24, 16]. Being able to assign probabilities to a choice enables a designer to analyse not only the functional aspects of his system, but also non-functional aspects of his system, such as performance. Examples of inherently probabilistic systems are data-link protocols, where decisions in the protocol can be based on a probabilistic process, and fluctuations in bandwidth and noise (which can be modelled using probabilities) affect the quality of underlying communications channels.

Notice that a probabilistic choice is inherently different from a non-deterministic choice. The latter is still of use in the specification of systems if, for instance, it is the environment that determines the choice, or if quantifiable information about the resolution of a choice is unavailable.

The question of how to add the notion of abstraction to a framework that already includes non-determinism and probabilities remains unsettled. Thus far, studies have focused mainly on the combination of abstraction in frameworks for fully probabilistic processes or fully non-deterministic processes. In this paper, we approach the problem from an axiomatic point of view. We discuss the theory pBPA_τ, which is a process algebra, based on the theory BPA (see e.g. [6]). The process algebra pBPA_τ deals with probabilistic choice, non-deterministic choice and abstraction. We identify several verification rules that are needed for coping with abstraction in systems containing both types of choices. We explain the intuition behind these rules by means of two small examples, and we discuss the current state of the art on this subject in some detail.

This paper is structured as follows. Section 2 introduces the basic concepts for the language pBPA. In Section 3, we introduce the concept of abstraction to the language pBPA and we discuss several axioms and rules for dealing with abstraction in this setting. Section 4 discusses work that is related to the work we describe in this paper. We illustrate the applicability of our rules for abstraction by sketching how they can be applied in the verification of the *Concurrent Alternating Bit Protocol* [28]. Section 6 summarises the results described in this paper.

2 Probabilistic process algebra

Process algebras provide the means for studying behaviours of (often reactive) processes by investigating the ways in which these can be constructed from basic operators and constants. The interplay between the operators, constants and processes is typically described by axioms. Examples of well-known process algebras are CCS [21] and ACP [6].

Here, we discuss a process algebra called pBPA [2, 4], which extends a subtheory of ACP, called BPA [6], with capabilities for reasoning about random behaviours of processes. In pBPA, random behaviours of processes are captured by means of a probabilistic choice operator. This binary operator models a choice between two processes, based on a probability distribution. The implications of adding this operator to the theory of BPA are studied extensively in [2, 4]. In the remainder of this section, we present the theory of pBPA.

Let A denote a set of (observable) atomic actions. The syntax for our Probabilistic Process Algebra pBPA is given by the following rules:

$$S ::= \delta \mid a \mid S \cdot S \mid S + S \mid S \leftthreetimes_\pi S$$

for $a \in A$ and $\pi \in (0, 1)$. Variables p, q, r range over the set of all expressions of pBPA. We distinguish a special subset of process expressions, viz. *dynamic processes*. These are defined by the following rules:

$$D ::= \delta \mid a \mid D \cdot S \mid D + D$$

Intuitively, dynamic processes are processes for which all probabilistic choices have been resolved, i.e. they are "ready" to perform an activity or they deadlock.

We briefly discuss the intended meaning of the operators and constants of pBPA. The constant δ stands for a process which with probability 1 cannot perform any transitions. A process a performs with probability 1 an activity that is modelled by action a. The binary operators $+$ and $\cdot$ model the classical notions of non-deterministic choice (also called *alternative composition*) and sequential composition, respectively. The above-mentioned operators are all part of the process algebra BPA. The probabilistic extension of BPA lies in the addition of the binary operator $\leftthreetimes_\pi$. A probabilistic process $p \leftthreetimes_\pi q$ behaves as process p with probability π and as process q with probability $1-\pi$. Remark that the algebra pBPA we discuss here is a *concrete* process algebra, i.e. the notion of abstraction is not included in the algebra.

The presence of both the non-deterministic choice operator $+$ and the probabilistic choice operator $\leftthreetimes_\pi$ enables one to specify processes that exhibit both probabilistic and non-deterministic behaviours. However, using this facility is not entirely trivial: both operators model the choice between two processes, but the moment of choice and the resolution of the choice are different. When modelling a real-life system, it is important to know how a particular choice is made. For instance, the choice can be made by the system itself (by e.g. performing an internal action) or it can be left to the environment to decide. Further, one should decide whether the available quantitative information about the possible alternatives is sufficient to define a probability distribution. Lastly, the order in which choices are made should be taken into account. Namely, process expression $(x + y) \leftthreetimes_\pi z$ may represent a process that first resolves the probabilistic choice between $x + y$ and z and later resolves the non-deterministic choice between x and y if $x + y$ has been chosen in the previous choice. But it can be given another interpretation, namely, the process first resolves the non-deterministic choice between x and y and later, depending on the result, it chooses probabilistically between x and z or y and z.

The approach to non-deterministic choice, taken in the framework of ACP and ACP-like process algebras, is that this type of choice is resolved at the same moment when one of the two alternatives performs its first action. The approach taken in our algebra pBPA agrees with this. The probabilistic choice is resolved differently from the non-deterministic choice. It is assumed that this type of choice is made *before* the first action occurs, but the exact moment is not known. That is, there is an internal behaviour of the process which determines the outcome of the choice $p \leftthreetimes_\pi q$, which takes place *before* p or q performs any action. The outcome of this choice cannot be influenced by the environment (it is made internally), but it can be observed.

The intuition we sketched thus far is formalised by a set of axioms (see Table 1). All but one of the listed axioms, that characterise the operators $+$ and $\cdot$ and the constant δ are the axioms that can be expected from the theory BPA. The exception is axiom AA3, which we will explain shortly. Axioms A1 and A2 express that a non-deterministic choice is both commutative and associative, i.e. all alternatives are treated equally. Axiom A6 shows that the constant δ is a neutral element for $+$. Axiom A4 shows that sequential composition is right-distributive over $+$, and axiom A7 shows that δ is a left-zero for sequential composition.

The approach for dealing with the interplay between a non-deterministic choice and a probabilistic choice, taken in pBPA, is that the probabilistic choice is resolved

Table 1. Axioms of pBPA.

A1	$x + y$	$= y + x$	PrAC1	$x \leftthreetimes_\pi y$	$= y \leftthreetimes_{1-\pi} x$
A2	$(x + y) + z$	$= x + (y + z)$	PrAC2	$x \leftthreetimes_\pi (y \leftthreetimes_\rho z)$	$= (x \leftthreetimes_{\frac{\pi}{\pi+\rho-\pi\rho}} y) \leftthreetimes_{\pi+\rho-\pi\rho} z$
AA3	$a + a$	$= a$	PrAC3	$x \leftthreetimes_\pi x$	$= x$
A4	$(x + y) \cdot z$	$= x \cdot z + y \cdot z$	PrAC4	$(x \leftthreetimes_\pi y) \cdot z$	$= x \cdot z \leftthreetimes_\pi y \cdot z$
A5	$x \cdot (y \cdot z)$	$= (x \cdot y) \cdot z$	PrAC5	$(x \leftthreetimes_\pi y) + z$	$= (x + z) \leftthreetimes_\pi (y + z)$
A6	$x + \delta$	$= x$			
A7	$\delta \cdot x$	$= \delta$			

first. Thus, the probabilistic choice has priority over a non-deterministic choice. This is expressed by axiom PrAC5, which states that the probabilistic choice operator right-distributes over a non-deterministic choice. Axiom PrAC1 states that for a process $p \leftthreetimes_\pi q$, the probability π actually applies to the left-hand side process. The remaining axioms relating to the operator $\leftthreetimes_\pi$ are self-explanatory.

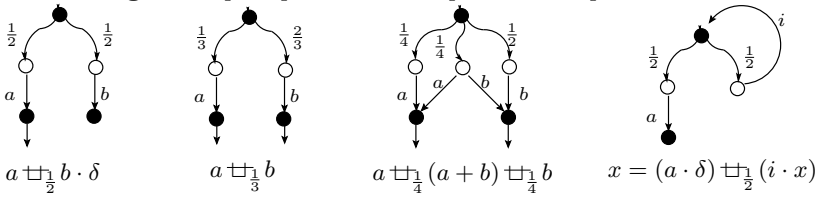
More interesting is axiom AA3, which is a restricted version of the usual axiom $p + p = p$ of BPA, saying that $+$ is idempotent for arbitrary processes. In the setting with probabilities, this law only applies to *dynamic* processes. This can be seen as follows. Consider process $a \leftthreetimes_{\frac{1}{2}} \delta$. Clearly, this is not a dynamic process. It can execute action a with probability $\frac{1}{2}$ and it deadlocks with probability $\frac{1}{2}$. Then, by axiom $PrAC5$ we obtain $(a \leftthreetimes_{\frac{1}{2}} \delta) + (a \leftthreetimes_{\frac{1}{2}} \delta) = ((a + a) \leftthreetimes_{\frac{1}{2}} (a + \delta)) \leftthreetimes_{\frac{1}{2}} ((a + \delta) \leftthreetimes_{\frac{1}{2}} (\delta + \delta))$. Using the other axioms, this can be reduced to $a \leftthreetimes_{\frac{3}{4}} \delta$. This process clearly differs from the process $a \leftthreetimes_{\frac{1}{2}} \delta$.

In addition, we consider guarded recursive specifications. Let V be a set of process variables. A guarded recursive specification is a set of equations of the form $X = s_X(V)$, where X is a variable from V and s is a possibly open pBPA term containing variables from V. Using guarded recursive specifications (see e.g. [6] for a definition of *guardedness*), infinite processes can be defined.

For a better understanding we will often present processes graphically by means of directed graphs, using the following convention: $\bullet \overset{\pi}{\rightsquigarrow} \circ$ represents a probabilistic step with probability π from a probabilistic node (represented by $\bullet$) to a dynamic node (represented by $\circ$). An action step is denoted $\circ \xrightarrow{a} \bullet$ representing that in a dynamic node the execution of an action a leads to a probabilistic node. A (dynamic) node that terminates is marked by a dangling outgoing edge. In Figure 1 we give several examples of this notation.

Apart from the axiomatic perspective, sketched in this section, an operational perspective of the theory of pBPA exists. In [4], an operational semantics of pBPA is provided, using Plotkin-style rules (see [23]). It is shown that this operational semantics, together with strong probabilistic bisimulation à la Larsen-Skou [20] as its equivalence relation, provides a model for the theory of pBPA. We do not give an exposition of this alternative semantics, as we restrict ourselves to axiomatic reasoning in this paper.

Fig. 1. Graph representation of probabilistic processes.



3 Abstraction and verification rules

In the previous section, we set out to explain the basics of our *concrete* probabilistic process algebra. In this section, we extend on this algebra by adding mechanisms for abstraction. Abstraction is modelled using a dedicated constant τ. Observable actions can be hidden by renaming them to τ using a unary τ_I operator (axiomatised by the rules of Table 2), where $I \subseteq A$ contains all actions to be hidden. We require that τ is not part of this set of actions A. Unlike ordinary atomic actions, τ cannot be *observed*

Table 2. Axioms for hiding of actions in $I \subseteq A$.

TI0	$\tau_I(\tau)$	$= \tau$	
TI1	$\tau_I(a)$	$= a$	if $a \notin I$
TI1'	$\tau_I(a)$	$= \tau$	if $a \in I$
TI2	$\tau_I(x + y)$	$= \tau_I(x) + \tau_I(y)$	
TI3	$\tau_I(x \cdot y)$	$= \tau_I(x) \cdot \tau_I(y)$	
TI4	$\tau_I(x \leftthreetimes_\pi y)$	$= \tau_I(x) \leftthreetimes_\pi \tau_I(y)$	

directly, but in some cases, its presence can be inferred. For example, in the (dynamic) process $b + \tau \cdot a$, the unobservable action τ can be detected, since this process has a state in which action a is inevitable and action b is no longer offered. However, some unobservable actions in a process can be removed without it affecting the (observable parts of the) behaviour of a process, e.g. process $a \cdot \tau \cdot b$ is considered indistinguishable from process $a \cdot b$. Such redundant τ actions are also called *inert* τ actions. In standard process algebras, there are two rules that allow us to remove redundant τ actions (see Table 3).

Table 3. Axioms for abstraction for non-probabilistic processes x, y and z.

B1	$x \cdot \tau$	$= x$
B2	$x \cdot (\tau \cdot (y + z) + z)$	$= x \cdot (y + z)$

While axiom B1 is still valid in a setting with probabilities, it is not immediately clear whether axiom B2 should still hold. We postulate that, whenever process z is a dynamic process, axiom B2 holds. For, no matter what x and y are, after termination of process x we are offered to proceed according to process z or silently perform action τ and still we are offered process z. The condition on process z deserves some extra explanation. Suppose $z = a \leftthreetimes_\pi b$. Then after termination of process x, the right-hand side of axiom B2 can be written as $y + (a \leftthreetimes_\pi b)$, which is equivalent to process $(y + a) \leftthreetimes_\pi (y + b)$. The reader is invited to check that, while in this process, there is at most a probability of π to perform an action a in case y cannot perform a, the left-hand side process $\tau \cdot (y + (a \leftthreetimes_\pi b)) + (a \leftthreetimes_\pi b)$ has a larger probability of performing action a.

However, we argue that axiom B2 also holds whenever $y + z = y$, i.e. all activity, described by process z is also captured by process y. Remark that this does not imply that process z is a dynamic process (i.e. we do not necessarily have $z = z + z$), as the following example suggests. For $y = a+b$ and $z = a \leftthreetimes_\pi b$, process $y+z = (a+b)+(a \leftthreetimes_\pi b)$ can be rewritten to process $(a+b+a) \leftthreetimes_\pi (a+b+b)$, using axiom PrAC5, which, using axioms PrAC3 and AA3 can be rewritten to process $a + b$. Thus, whenever $y + z = y$, we know that the behaviour of process z can be mimicked by the behaviour of process y (even if process z is a probabilistic process). Therefore, by executing an unobservable step, no behaviours are lost and the τ step is considered redundant, as is described by axiom PrB2'.

The theory pBPA_τ, which is the extension of pBPA with unobservable actions, is axiomatised using the axioms of pBPA and the axioms in Table 4.

As we will see, the axioms we discussed thus far do not allow us to deal with all redundant τ-steps.

Table 4. Axioms for abstraction for processes x, y and z.

PrB1	$x \cdot \tau = x$	
PrB2	$x \cdot (\tau \cdot (y + z) + z) = x \cdot (y + z)$	if $z = z + z$
PrB2'	$x \cdot (\tau \cdot (y + z) + z) = x \cdot (y + z)$	if $y = z + y$

In a setting without probabilities, the correctness of an implementation (with respect to a specification) often hinges on fairness assumptions about the resolution of non-deterministic choices. Algebraically, such fairness assumptions are described through the use of *Koomen's fair abstraction rules* (KFARs) [5]. These rules capture the idea that, due to an implicit fairness mechanism, in abstracting from a set of internal events (events that are "private" to a system), eventually an external event will be chosen. The eventuality arises as a result of an implicit fairness mechanism underlying a non-deterministic choice.

Interestingly, in the presence of probabilities, a fairness assumption for probabilistic choice is superfluous. By assigning a non-zero probability to every alternative in a probabilistic choice we can quantify the otherwise implicit fairness assumptions. Despite the fact that we do not need fairness assumptions to resolve a probabilistic choice,

additional verification rules for probabilistic processes are still needed to reason about systems in which internal activity and/or probability distributions over external events are involved

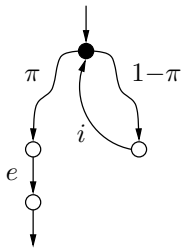


Fig. 2. Deterministic self-loop

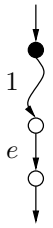


Fig. 3. Inevitability of action e.

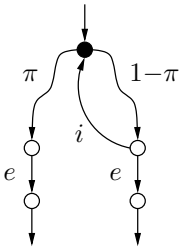


Fig. 4. Non-deterministic self-loop

For instance, consider process $x = e \leftthreetimes_\pi i \cdot x$ (see Fig. 2), where e is an external event and i is an internal event. It is rather unfortunate to find that the above process cannot be simplified further using the laws identified thus far. The intuition behind this process is that, with a chance of π, action e is executed immediately. With a probability of $\pi + \pi(1-\pi)$, action e is executed within two attempts, and, generalising, with a probability of $\sum_{0 \leq k < n} \pi(1-\pi)^k$, action e is executed within n attempts. Elementary mathematics teaches us that when n is sufficiently large, the probability of executing action e approaches 1, and thus the limit behaviour is that action e is executed with probability 1. Thus, the processes, depicted in Figs. 2 and 3 should be considered equivalent.

Although we do not have the illusion that we will cover all possible situations, we know the entanglement of probabilistic choice and non-deterministic choice runs deeper than the previous example illustrated. To exemplify, we consider process $x = e \leftthreetimes_\pi (e + i \cdot x)$ (see Fig. 4). This process is clearly different from the process depicted in Fig. 2. However, we argue that both processes have in fact the same solution when abstracting from their internal behaviours. Suppose the worst-case scenario, in which the internal action i takes precedence over external action e. Then, process $e \leftthreetimes_\pi (e+i \cdot x)$ behaves exactly as process $e \leftthreetimes_\pi i \cdot x$, i.e. action e is inevitably executed. But, in case the non-deterministic choice between the external action e and the internal action i is resolved fairly, action e is also executed inevitably, making this process indistinguishable from the processes depicted in Figs. 2 and 3.

Generalising these situations, we obtain the following three verification rules.

$$\frac{X = Z \leftthreetimes_\pi (Z+i \cdot X) \qquad Z = Z+Z \qquad i \in I}{\tau \cdot \tau_I(X) = \tau \cdot \tau_I(Z)} \tag{PVR}$$

$$\frac{X = Z \leftthreetimes_\pi (Y+i \cdot X) \qquad Z = Z+Y \qquad i \in I}{\tau \cdot \tau_I(X) = \tau \cdot \tau_I(Z \leftthreetimes_\pi (Z+i \cdot X))} \tag{PVR'}$$

$$\frac{X = Y \leftthreetimes_\pi i \cdot X \qquad i \in I}{\tau \cdot \tau_I(X) = \tau \cdot \tau_I(Y)} \tag{PVR"}$$

Remark 1. We use the paradigm of branching bisimulation [6], rather than weak equivalence [21, 6], as this notion of bisimulation allows for removing inert τ-steps, while respecting the branching structure of a process. In [6], for example, it is argued that this is not the case for weak bisimulation.

Remark 2. We aim at an axiomatic characterisation of (probabilistic) branching bisimulation for processes that can contain both probabilistic and non-deterministic behaviours. However, the operational characterisation of this equivalence relation has not yet been defined. Concerning the soundness of the axioms PrB1, PrB2 and PrB2' and the verification rules, it can be shown that probabilistic weak bisimulation [24] provides a model for these axioms. This is also to be expected, as for the non-probabilistic setting, this latter equality is already coarser than branching bisimulation, i.e. when we can derive that process p is branching bisimilar with process q, then we immediately know that process p is weakly bisimilar with process q. The converse does not always hold.

4 Related work

The combination of non-determinism and probabilities allows for the specification of systems that portray both types of choice. Analysing these specifications, however, is complicated exactly because of this combination. Several verification techniques have been proposed, inspired by their counterparts in the non-probabilistic case. On the one hand, *model checking* has been used to check whether the system satisfies desired (logical) properties with a certain probability, see e.g. [26, 14, 15, 1, 11, 9]. On the other hand different definitions of implementation relations, such as bisimulations [19, 12, 20, 13, 7, 22, 24, 10, 3], simulations and testing preorders [27, 17] have been proposed. For some of these relations an axiomatisation has been defined as well [12, 13, 20, 27, 15, 3]. The majority of this work focuses on concrete systems, i.e. they do not distinguish between internal actions and observable actions; both are treated as observable actions. Only a few attempts to formulate a method that abstracts away internal actions for probabilistic systems have been made. We classify them into two groups: those methods that consider only fully probabilistic systems, and those verification methods that consider systems with both types of behaviour, i.e. non-deterministic and probabilistic. The latter methods are more complicated and still consumes a lot of research effort.

In [7] the authors give a definition of weak bisimulation for fully probabilistic processes. The relation is inspired by Milner's weak bisimulation [21]. Instead of the weak transition $\Rightarrow$ from [21], the probabilities to reach states via sequences of internal transitions are computed. The authors define a probabilistic branching bisimulation and prove that it coincides with the weak equivalence. An attempt to axiomatise this relation is made in [8].

In his thesis, C.-C. Jou [18] defines a probabilistic branching bisimulation for fully probabilistic finite systems. He also presents a sound and complete axiomatisation for this relation. This algebra is a probabilistic variant of Milner's SCCS [21] where the non-deterministic choice is replaced by a probabilistic choice operator. It is interesting to see that the axiomatisation includes a probabilistic variant of axiom B2, i.e. all non-determinism is replaced by probabilistic choice.

A version of a fully probabilistic process algebra with abstraction is defined in [3, 4]. The authors propose several verification rules that are inspired by the KFAR verification rules, known from non-probabilistic process algebras with abstraction, such as ACP_τ. The verification rules, discussed in this paper, appear in [3, 4] too, but there they are restricted to fully probabilistic processes. Additionally, the authors define a probabilistic branching bisimulation which is used to build a model for the axiomatisation.

In [10], the authors define a recursion free probabilistic process algebras with both probabilistic and non-deterministic choice. They are used to define a complete axiomatisation for several notions of (strong and weak) probabilistic bisimulations in the alternating [15, **?**] and the non-alternating model [24]. The set of proposed axioms for strong probabilistic bisimulation is a subset of the axioms we discussed in Section 2. Also, variants of axioms PrB1 and PrB2, introduced in Section 3, are used for axiomatising probabilistic observation congruence on the alternating model. The probabilistic observation congruence proposed in [24] and used in [10] can be taken as an equivalence relation to build a model for our verification rules and axioms in Section 2.

5 Application

The rules we discussed in Section 3 were conceived in an attempt to specify and verify larger communications protocols, such as a version of the *Sliding Window Protocol* (SWP) and the *Concurrent Alternating Bit Protocol* (CABP). It is the verification of the latter protocol we discuss in this section. Both protocols are well documented in the literature, but, as far as we can tell, no attempt has yet been made to verify probabilistic versions of these.

The main intention of this section is to give the reader an insight to our concept of dealing with non-determinism in probabilistic processes. We do not aim at giving a complete description of the protocol. We refer the reader to [28] for more details. The specifications we provide use two other operators, viz. *parallel composition* and *encapsulation*. The parallel operator $\|$ models the communications (synchronisations) of two processes and the interleavings of behaviours of two processes. Using an elimination theorem (see [4]), all guarded recursive equations containing the parallel operator can be rewritten to equivalent guarded recursive equations using only the operators from the algebra pBPA_τ. For a detailed treatment on parallelism in the probabilistic setting, we refer to [4]. The unary encapsulation operator ∂_H is in a sense a special renaming operator, renaming all actions from the set H to δ. It is often used to encapsulate atomic actions that are intended to synchronise such that their asynchronous execution is not permitted.

The CABP is a more complex variant of the well-known *Alternating Bit Protocol* (ABP). Both protocols are used to reliably send and receive data via an unreliable channel, using a system of *acknowledgements*, also sent via unreliable channels. While the ABP resends messages that have not been received correctly only after receiving a *negative acknowledgement*, the CABP continuously (re)sends the same message until it receives a *positive acknowledgement*, confirming a correct delivery of the datum. Due to this phenomenon many activities are executed in parallel.

5.1 Specification of the CABP

The specification we use for the CABP is based on the description, given in [28]. While in [28], the channels for transmitting data and acknowledgements are modelled using non-determinism, we shall use probabilities to model the behaviours of the channels. The CABP is described by six separate processes that need to communicate with one another. A sketch of the system is provided in Fig. 5. The numbers 1 through 8 represent the gates at which actions are communicated. Together, the six processes model the

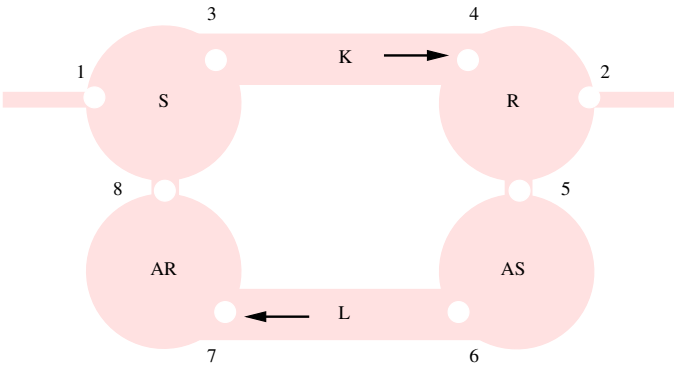


Fig. 5. Layout of the *Concurrent Alternating Bit Protocol*.

entire protocol. We represent the *sender* and *receiver* of data by process S and R, respectively. The sender and receiver of an acknowledgement are specified by processes AS and AR, respectively. In turn, the channel, carrying the data is represented by process K and the channel, carrying the acknowledgements is represented by process L.

We group these processes in two logical modules, viz. a module (I) responsible for transmitting and receiving data, containing processes S, K and R, and a module (II), responsible for transmitting and receiving acknowledgements, containing processes AS, L and AR. The CABP is then described by the parallel composition of these two modules.

Since the behaviour of both modules is quite similar, we here only investigate the behaviour of module I. The specifications for the sender, the receiver and the data channel are given in Table 5. The difference between our specification and the one, appearing in [28] is mainly in the definition of the channel, which, as we mentioned before, is modelled using the probabilistic choice operator. We briefly sketch the intuition behind the processes of module I. The set D denotes a finite set of data elements. The sender, modelled by process S uses action r_1 to receive a datum from its environment (the subscript 1 indicates that it is received via gate 1, see Fig. 5). This datum, augmented with a bit (for acknowledgement purposes), is then repeatedly sent using action s_3. Alternatively, an acknowledgement is received using action r_8. The receiver reads data (augmented with an acknowledgement bit) from the communication channel K using action r_4. A corrupted datum is represented by $\perp$ (a mechanism for detecting faulty data is assumed). If a non-corrupted, expected datum is received (i.e. it carries

Table 5. Specification of the sender, the receiver and the channel. We used the set Bit to denote the set $\{0,1\}$, and the type D represents the (finite) set of data. The probabilities π and ρ model the probability that a message is sent correctly and the probability that a message is corrupted. With a probability of $1-\pi-\rho$, the message is lost in a channel.

Sender:

$$\begin{aligned}
S &= RM(0)\\
RM(b{\in}Bit) &= \textstyle\sum_{d\in D}\ r_1(d)\cdot SF(d,b)\\
SF((d,b){\in}D{\times}Bit) &= s_3(d,b)\cdot SF(d,b) + r_8(ac)\cdot RM(1-b)
\end{aligned}$$

Receiver:

$$\begin{aligned}
R &= RF(0)\\
RF(b{\in}Bit) &= \textstyle\sum_{d\in D}\ r_4(d,b)\cdot RS(d,b)\\
&\quad + \textstyle\sum_{d\in D}\ r_4(d,1-b)\cdot RF(b) + r_4(\bot)\cdot RF(b)\\
RS((d,b){\in}D{\times}Bit) &= s_2(d)\cdot s_5(ac)\cdot RF(1-b)
\end{aligned}$$

Channel:

$$\begin{aligned}
K &= \textstyle\sum_{(d,b)\in D\times Bit}\ r_3(d,b)\cdot K_s(d,b)\\
K_s((d,b){\in}D{\times}Bit) &= (s_4(d,b) \trianglelefteq_\pi s_4(\bot) \trianglelefteq_\rho k)\cdot K
\end{aligned}$$

---

the expected acknowledgement bit), it is sent to the environment via action s_2 and the acknowledgement sender is triggered via action s_5. Finally, the unreliable channel uses action r_3 for receiving a datum, action s_4 for sending a (possibly corrupted) datum and action k for losing a datum.

We distinguish between the external and the internal behaviour of module I. The communications between the actions of module I are defined using a (commutative and associative) function γ, given by $\gamma(r_3(d,b),s_3(d,b)) = c_3(d,b)$ and $\gamma(r_4(d,b),s_4(d,b)) = c_4(d,b)$ and $\gamma(r_4(\bot),s_4(\bot)) = c_4(\bot)$ for all data $(d,b){\in}D{\times}Bit$ (here, c_3 and c_4 are fresh actions). Process $\partial_H(S\|R\|K)$ then describes the internal behaviour of module I, where $H = \{s_3(d,b), r_3(d,b), s_4(d,b), r_4(d,b), s_4(\bot), r_4(\bot) \mid (d,b){\in}D{\times}Bit\}$ contains all actions that need to be blocked to enforce communication. The external behaviour of module I is the behaviour of module I *after* abstracting from the actions from the set $I = \{c_3(d,b), c_4(d,b), c_4(\bot), k \mid (d,b){\in}D{\times}Bit\}$ using the τ_I operator. The external behaviour of module I can thus be represented by process $\tau_I \circ \partial_H(S\|R\|K)$.

5.2 Verification of the CABP

Our aim is to rewrite the process, representing the external behaviour of module I to a simpler process, for which it is easier to check that it behaves the way it is meant to behave. Our verification is based on the verification conducted in [28]. For lack of space, we will not go into the full detail, but pick out several interesting bits of the verification. Apart from using the rules we gave in Section 3 and the use of axiomatic reasoning, we employ a technique called *language matching*, which is explained and used in [28].

Before proceeding with our verification, we first explain this latter technique in a few words.

Language Matching Modularisation of specifications is an important technique in tackling the complexity of describing complex systems. By specifying a system as the result of the cooperation between several processes, the specification problem is reduced to the specification problem of these separate processes. A good example is the CABP, (partly) described in the previous section. Though this is very desirable for specification, for verification purposes, this introduces additional complexity: only when put together, the separate processes produce the desired result, but in isolation, behaviours that cannot occur in the end (due to e.g. a different choice by the environment) are still present.

Language matching provides a means to tackle this problem. Basically, language matching builds on the idea that certain behaviours in a subprocess p will not be present in the overall system $\partial_H(x \| p)$, for some x. Consider for example process $\partial_{\{a,c,d\}}(x \| y)$, where $x = a \cdot x + c \cdot z$ and $y = d \cdot y$, and the actions a and d are meant to communicate to action b. Clearly, c is a redundant summand because it gets encapsulated, and it is not involved in communications. Therefore, the expected behaviour of this process is to show an infinite sequence of b actions. The intuition, now, is that actions often occur in a fixed sequence. In the CABP, for example we first expect to see action $r_1(d)$ and only then action $s_2(d)$ followed by a new action $r_1(d)$, etc. The language matching operator, denoted by $\triangle_Z$, where Z is a collection of traces, checks for a given process if the actions that occur are executed according to the order, specified by a trace in Z. If so, the remainder of the process is inspected (with a possibly different set of traces). If not, then the remainder of the process is apparently not of any interest to the overall behaviour of the system, and hence, it is replaced by the special symbol $\Re$, which is not present in the alphabet A. It is important to note that whenever the trace is chosen correctly, the processes, represented by the symbol $\Re$ are not reachable, and hence an equivalent specification, not containing the symbol $\Re$ can be given. However, if the trace is chosen incorrectly, the overall process will still have reachable subprocesses that are represented by the symbol $\Re$.

Analysis of the CABP We start with analysing the subprocess $\partial_{H'}(R \| K)$, where H' is the set $\{r_4(d,b), s_4(d,b), r_4(\bot), s_4(\bot) \mid (d,b) \in D \times Bit\}$. Removing parallelism using the elimination theorem we arrive at the description for this process, given in Table 6. Abstracting from the actions k, $c_4(\bot)$ and $c_4(d,b)$, we then arrive at the process that is depicted by Figure 6.

Closer inspection of this process graph reveals that there is a potential τ-loop, once we place process $\partial_{H'}(R \| K)$ in parallel with the sender, i.e. process S. For instance, all actions r_3, present in the graph, will need to communicate with actions s_3, producing action c_3, which is subsequently turned into a τ-action. Unfortunately, the sender itself can always receive an acknowledgement of the correct reception of a message, and thereby change the entire global state. However, in the overall behaviour of the CABP, the reception of an acknowledgement only happens when an acknowledgement has been sent earlier. Thus, to filter out the reception of the ill-timed acknowledgements, we

Table 6. Process $R\|K$ after encapsulation of the actions in set H'.

$$
\begin{array}{ll}
X_1(b{\in}Bit) & = \sum_{d\in D}\ r_3(d,b)\cdot X_3(b,d) + \sum_{d\in D}\ r_3(d,1-b)\cdot X_2(b,d)\\
X_2(b{\in}Bit, d{\in}D) & = (c_4(d,1-b) \between_{\pi} c_4(\bot) \between_{\rho} k)\cdot X_1(b)\\
X_3(b{\in}Bit, d{\in}D) & = c_4(d,b)\cdot X_4(b,d) \between_{\pi} c_4(\bot)\cdot X_1(b) \between_{\rho} k\cdot X_1(b)\\
X_4(b{\in}Bit, d{\in}D) & = \sum_{d'\in D}\ r_3(d',b)\cdot X_5(b,d,d')\\
 & \quad + \sum_{d'\in D}\ r_3(d',1-b)\cdot X_6(b,d,d')\\
 & \quad + s_2(d)\cdot X_9(b)\\
X_5(b{\in}Bit, d, d'{\in}D) & = s_2(d)\cdot X_7(b,d') + (\delta \between_{\pi+\rho} k)\cdot X_4(b,d)\\
X_6(b{\in}Bit, d, d'{\in}D) & = s_2(d)\cdot X_8(b,d') + (\delta \between_{\pi+\rho} k)\cdot X_4(b,d)\\
X_7(b{\in}Bit, d{\in}D) & = s_5(ac)\cdot X_2(1-b,d) + (\delta \between_{\pi+\rho} k)\cdot X_9(b)\\
X_8(b{\in}Bit, d{\in}D) & = s_5(ac)\cdot X_3(1-b,d) + (\delta \between_{\pi+\rho} k)\cdot X_9(b)\\
X_9(b{\in}Bit) & = \sum_{d\in D}\ r_3(d,b)\cdot X_7(b,d)\\
 & \quad + \sum_{d\in D}\ r_3(d,1-b)\cdot X_8(b,d)\\
 & \quad + s_5(ac)\cdot X_1(1-b)
\end{array}
$$

apply the language matching. In short, all alternatives in the specification for module I that are in traces that do not match the language containing concatenations of traces $r_1(d)\ s_2(d)\ s_5(ac)\ r_8(ac)$ for arbitrary $d{\in}D$ should be marked with the symbol $\Re$. We define $Z = \{r_1(d)\ s_2(d)\ s_5(ac)\ r_8(ac) \mid d{\in}D\}^*$, and subsequently study process $\triangle_{Z'} \circ \partial_H(SF(d,b)\|X_4(b,d))$, where $Z' = \{z \mid d{\in}D, r_1(d)z{\in}Z\}$ contains all traces of Z from which the first action $r_1(d)$ has been removed. We derive that

$$
\begin{array}{ll}
 & \triangle_{Z'} \circ \partial_H(SF(d,b)\|X_4(b,d))\\
= & \\
 & r_8(ac)\cdot\Re + s_2(d)\cdot U_1 + c_3(d,b)\cdot \triangle_{Z'} \circ \partial_H(SF(d,b)\|X_5(b,d,d))
\end{array}
$$

Process U_1 is not of our interest at this point. Note that after action $r_8(ac)$, the remainder of the process has become unimportant, as the action $r_8(ac)$ is not a valid option at this point in the overall behaviour. We continue, and derive that

$$
\begin{array}{ll}
 & \triangle_{Z'} \circ \partial_H(SF(d,b)\|X_5(b,d,d))\\
= & \\
 & r_8(ac)\cdot\Re + s_2(d)\cdot U_2 + (\delta \between_{\pi+\rho} k)\cdot \triangle_{Z'} \circ \partial_H(SF(d,b)\|X_4(b,d))
\end{array}
$$

Again, process U_2 is not of our interest at this point. Now, by hiding the communications that occur on gates 3 and 4, i.e. all communications with the channel K, we derive the following equality:

$$
\begin{array}{ll}
 & \tau\cdot\tau_I \circ \triangle_{Z'} \circ \partial_H(SF(d,b)\|X_4(b,d))\\
= & \\
 & \tau\cdot\quad (r_8(ac)\cdot\Re + s_2(d)\cdot\tau_I(U_1)\\
 & +\tau\cdot(r_8(ac)\cdot\Re + s_2(d)\cdot\tau_I(U_2) +\\
 & \qquad (\delta \between_{\pi+\rho}\tau)\cdot\tau_I \circ \triangle_{Z'} \circ \partial_H(SF(d,b)\|X_4(b,d))))
\end{array}
$$

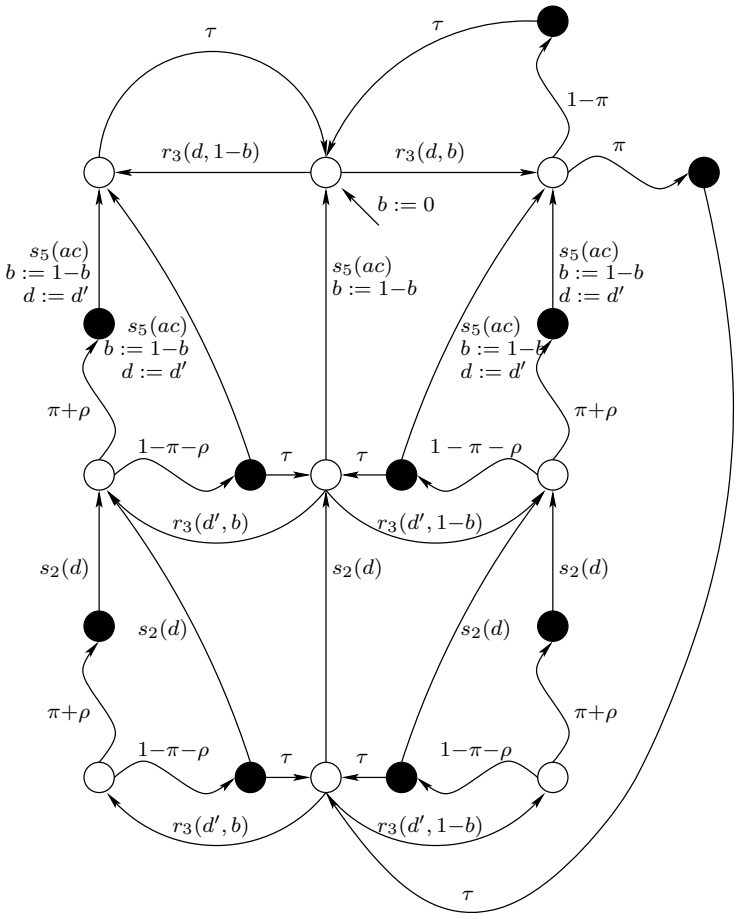


Fig. 6. Process $\partial_{H'}(R \| K)$ after abstraction from internal actions

It turns out that (by carrying the verification through to the end), we can prove that $\tau_I(U_1) = \tau_I(U_2)$, and hence, using axiom PrB2, we derive

$$\begin{aligned} & \tau \cdot \tau_I \circ \triangle_{Z'} \circ \partial_H(SF(d,b) \| X_4(b,d)) \\ = \\ & \tau \cdot (r_8(ac) \cdot \Re + s_2(d) \cdot \tau_I(U_2) + (\delta \trianglelefteq_{\pi+\rho} \tau) \cdot \tau_I \circ \triangle_{Z'} \circ \partial_H(SF(d,b) \| X_4(b,d))) \end{aligned}$$

This equation is in a form for which we can apply rule PVR, yielding the following solution to the equation:

$$\begin{aligned} & \tau \cdot \tau_I \circ \triangle_{Z'} \circ \partial_H(SF(d,b) \| X_4(b,d)) \\ = \\ & \tau \cdot (r_8(ac) \cdot \Re + s_2(d) \cdot \tau_I(U_2)) \end{aligned}$$

Repeatedly applying the rules and axioms in a fashion we just sketched, we can finally prove that the external behaviour of the CABP cannot be distinguished from the

behaviours of a one-place buffer. Let the set I' contain all activity over communication gates $3, 4, 5, 6, 7$ and 8 (i.e. all activity in and between modules I and II). Then, we can derive for the whole CABP that it behaves as a reliable buffer, i.e.

$$\begin{array}{l} \tau \cdot \tau_{I'} \circ \partial_{H''}(S \| K \| R \| AS \| L \| AR) \\ = \\ \tau \cdot \sum_{d \in D} \ r_1(d) \cdot s_2(d) \cdot \tau_{I'} \circ \partial_{H''}(S \| K \| R \| AS \| L \| AR) \end{array}$$

6 Summary

We discussed a concrete process algebra, called pBPA, that allows for the specification of systems that exhibit both probabilistic and non-deterministic behaviours. We discussed an extension of the framework of pBPA with capabilities for hiding internal activity of a process by renaming observable actions to an unobservable action τ. We furthermore presented a number of axioms that allow for the removal of *inert* unobservable actions, i.e. unobservable actions whose presence cannot be detected. Unfortunately, these axioms are not sufficient for dealing with real-life systems. Thus, analogously with rules such as *Koomen's Fair Abstraction Rules* (KFARs) [5], which are used in a setting *without* probabilities, we introduced a number of rules that proved useful in a setting *with* probabilities.

To illustrate the theory we developed in this paper, we conducted a verification of the *Concurrent Alternating Bit Protocol* [28], a variant of the well-studied *Alternating Bit Protocol* [6]. It turns out that the rules and the axioms we provided are indeed needed and sufficient to come to the desired result.

Several issues are left for future research. Obviously, identifying more verification rules will greatly improve the applicability of process algebraic reasoning. Finding a definition of probabilistic branching bisimulation in a setting with non-determinism and probabilities remains an important issue.

Acknowledgement

The authors would like to thank an anonymous reviewer for her/his detailed comments that helped to improve this paper.

References

1. R. Alur, C. Courcoubetis, D.L. Dill. Model-checking for probabilistic real-time systems. Automata, Languages and Programming: Proceedings of the 18th ICALP, LNCS 510, pp. 115-136, 1991.
2. S. Andova, *Process algebra with probabilistic choice (extended abstract)*, Proc. 5th International AMAST Workshop, ARTS'99, Bamberg, Germany, J.-P. Katoen, ed., LNCS 1601, Springer-Verlag, pp. 111-129, 1999. (Full version report CSR 99-12, Eindhoven University of Technology, 1999.)
3. S. Andova, J.C.M. Baeten *Abstraction in probabilistic process algebra*, Proc. Tools and Algorithms for the Construction and Analysis of Systems, 7th International Conference, TACAS 2001 Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2001 Genova, Italy, T. Margaria, Wang Yi, eds., LNCS 2031 Springer Verlag, pp. 204-219, 2001.

4. S. Andova, *Probabilistic process algebra*, Ph.D. thesis, Eindhoven University of Technology, 2002.
5. J.C.M. Baeten, J.A. Bergstra, J.W. Klop, *On the consistency of Koomen's fair abstraction rule*, Theor. Comp. Sci. 51, pp.129-176, 1987.
6. J.C.M. Baeten, W.P. Weijland, *Process algebra*, Cambridge University Press, 1990.
7. C. Baier, H. Hermanns, *Weak bisimulation for fully probabilistic processes*, Proc. CAV'97, O. Grumberg, ed., LNCS 1254, pp. 119-130, 1997.
8. C. Baier, H. Hermanns, *Weak bisimulation for fully probabilistic processes*, Techn. Bericht IMMD-VII/1-97, Universität Erlangen.
9. C. Baier, M. Kwiatkowska, *Automatic verification of liveness properties of randomized systems (extended abstract)*, Proc. 14th Annual ACM Symposium on Principles of Distributed Computing, Santa Barbara, California, 1997.
10. E. Bandini, R. Segala, *Axiomatizations for probabilistic bisimulation*, Proc. ICALP'01, F. Orejas, P.G. Spirakis, J. van Leeuwen, eds., Crete, Greece, LNCS 2076, Springer Verlag, pp. 370-381, 2001.
11. A. Bianco, L. de Alfaro, *Model Checking of Probabilistic and Nondeterministic Systems*, Proc. Foundation of Software Technology and Theoretical Computer Science, LNCS 1026, Springer-Verlag, pp. 499-513, 1995.
12. A. Giacalone, C.-C. Jou, S.A. Smolka *Algebraic reasoning for probabilistic concurrent systems*, Proc. IFIP TC2 Working Conference on Programming Concepts and Methods, 1990.
13. R. J. van Glabbeek, S.A. Smolka, B. Steffen, C.M.N. Tofts, *Reactive, generative and stratified models of probabilistic processes*, Proc. of 5th Annual IEEE Symp. on Logic in Computer Science, Philadelphia, PA, pp. 130-141, 1990.
14. H. Hansson, B. Jonsson, *A calculus for communicating systems with time and probabilities*, Proc. IEEE Real-Time Systems Symposium, IEEE Computer Society Press, pp 278-287, 1990.
15. H. Hansson, *Time and probability in formal design of distributed systems*, Ph.D. thesis, DoCS 91/27, University of Uppsala, 1991.
16. H. Hermanns, *Interactive Markov Chains: And the Quest for Quantified Quality*, LNCS 2428, Springer-Verlag, 2002.
17. B. Jonsson, W. Yi, *Compositional Testing Preorders for Probabilistic Processes*, Proc. LICS'95, pp. 431-443, 1995.
18. C.-C. Jou, *Aspects of probabilistic process algebra*, Ph.D.Thesis, State University of New York at Stony Brook, 1990.
19. C.-C. Jou, S.A. Smolka *Equivalences, congruences and complete axiomatizations for probabilistic processes*, Proc. CONCUR '90, LNCS 458, J.C.M. Baeten, J. W. Klop, eds. Springer Verlag, Amsterdam, pp. 367-383, 1990.
20. K.G. Larsen, A. Skou, *Bisimulation through probabilistic testing*, Information and Computation, 94:1-28, 1991.
21. R. Milner, *Communication and concurrency*, International Series in Computer Science, Prentice Hall, 1989.
22. A. Philippou, O. Sokolsky, I. Lee, *Weak bisimulation for probabilistic systems*, Proc. CONCUR'98, D. Sangiorgi, R. de Simone, eds., Nice, France, LNCS 1466, Springer Verlag, 1998.
23. G.D. Plotkin. *A structural approach to operational semantics*, Technical Report DIAMI FN-19, Computer Science Department, Aarhus University, 1981.
24. R. Segala, *Modeling and verification of randomized distributed real-time systems*, Ph.D. thesis, Massachusetts Institute of Technology, 1995.
25. R. Segala, N.A. Lynch, *Probabilistic simulations for probabilistic processes*, Nordic Journal of Computing, 2(2):250-273,1995.
26. M.Y. Vardi, *Automatic verification of probabilistic concurrent finite state programs*, Proc. of 26th Symp. on Foundations of Com. Sc., IEEE Comp. Soc. Press, pp. 327-338, 1985.
27. W. Yi, K. Larsen, *Testing probabilistic and nondeterministic processes*, Proc. Protocol, Specification, Testing, Verification XII, pp. 47-61, 1992.
28. J. van Wamel, *Verification techniques for elementary data types and retransmission protocols*, Ph.D. thesis, University of Amsterdam, 1995.