ELSEVIER

Theoretical Computer Science 152 (1995) 219–249

Theoretical Computer Science

Probabilistic communicating processes

Karen Seidel[1]

Oxford University Computing Laboratory, Programming Research Group, Wolfson Building, Parks Road, Oxford OX1 3QD, UK

Received January 1994; revised November 1994
Communicated by M. Nivat

Abstract

We explore the suitability of two semantic spaces as a basis for a probabilistic variant of the language of Communicating Sequential Processes (*CSP*), so as to provide a formalism for the specification and proof of correctness of probabilistic algorithms. The two spaces give rise to two sublanguages, each of which is characterised by an algebraic axiomatisation which is shown to be sound and complete for finite processes.

In the first semantics, processes are defined as probability measures on the space of infinite traces and operators are defined as functions (mostly transformations) of probability measures. The advantage of this semantics is that it is simple and good for reasoning about probabilistic properties such as self-stabilisation or fairness of random algorithms. The disadvantage is that neither external choice nor parallel composition other than fully synchronised parallel composition can be defined in this semantics.

This problem is solved in the second model which is based on the space of conditional probability measures of infinite traces. This model leads to a set of proof rules for the deterministic properties of probabilistic algorithms, but it is more difficult to use in the analysis of probabilistic properties.

The last part of the paper shows how the two models are related and how this can be exploited to combine their advantages and get around their disadvantages. This is illustrated by the example of a self-stabilising tokenring.

1. Introduction

The language of Communicating Sequential Processes (*CSP*) [9] provides a mathematical formalism for the specification of distributed systems. Its main advantages are an effective treatment of concurrency, support for algebraic reasoning and a concept of refinement. We would like to extend these advantages to the specification and proof of correctness of probabilistic algorithms such as the self-stabilising tokenring

[1] Supported by a grant from the Science and Engineering Research Council.

SSDI 0304-3975(94)00286-X

due to [8] (which we will treat as an example). Also, there are some properties such as fairness or the absence of infinite overtaking which cannot easily be specified in *CSP*, but are expressed very naturally in probabilistic terms. For instance, a process which performs a choice over and over again can be said to be fair if asymptotically it chooses each alternatives with equal frequency. It is a standard probability-theoretic result that this would be true if each choice was made randomly with uniform probability. A weaker definition of fairness would require only that the probability of possible choice being overlooked forever from some point onwards in zero. (This is the notion of *extreme fairness* introduced by [16].) This could again be achieved by using probabilistic choice.

Our aim therefore is to introduce probabilistic choice into *CSP*, in such a way as to preserve the behaviour of the usual constructs of *CSP*. This paper investigates the possibility of doing so with two particular semantics in which processes are defined as (conditional) probability measures on the space of infinite traces and operators are defined as functions (mostly transformations) of (conditional) probability measures. We chose to work with infinite rather than finite traces because many probabilistic considerations such as asymptotic behaviours involve taking limits to infinity which cannot be expressed in terms of finite traces.

The paper is organised as follows: Section 2 presents a model in which processes are defined as probability measures on the space of infinite traces and operators as functions of probability measures. We call this the independent model, because in this model any choice of action is made independently of the environment. The section is divided into several parts, namely the syntax and axiomatic characterisation of a language called $PCSP_0$, notation and probability theory needed for the semantics, the semantics itself, and a part which shows that the axiomatisation is sound and complete for finite processes. Section 3 presents the second model, which we call conditional, because a choice of action may be conditioned on what the environment offers. This section is divided similarly to Section 2. In Section 4 we present a specification and proof of correctness of a self-stabilising tokenring as an example of how to use the formalism we developed. The last section contains a survey of related work and conclusions.

2. The independent model

2.1. Syntax and axioms

The syntax of $PCSP_0$ is a subset of *CSP*, except for the probabilistic choice operator. Mutual recursion can be added to this set in the standard way [5]. Informally, the intended behaviour of the constructs of $PCSP_0$ is as follows. Let Σ denote the (non-empty, finite or countable alphabet, or set of actions, of the system which is to be modelled.

$STOP$	deadlocks immediately, i.e. never does anything.
$a \to P$	first performs action a and then behaves as process P.
X	is a process variable, needed for proper treatment of recursion.
$P \,{}_p\!\sqcap\, Q$	behaves as P with probability p and as Q with probability $1-p$.
$P \parallel Q$	lockstep parallel composition of two processes P and Q which synchronise on every action.
$P \backslash B$	behaves as P without the actions in B (where $B \subseteq \Sigma$)
$f(P)$	performs the events of P renamed as determined by the injection $f: \Sigma \to \Sigma$
$\mu X.P$	a process in which every occurrence of X in P represents a recursive process invocation.

Formally, the behaviour denoted by these constructs is characterised by the set of axioms given below. We write $P \equiv Q$ if the terms P and Q are equivalent in the sense that they denote the same behaviour. Equational properties stated as laws are derivable from the axioms.

Probabilistic choice is commutative, associative and idempotent in the sense that identically behaved branches can be replaced by a single branch of that behaviour, but with the sum of the probabilities of the individual branches. It also has the special property that 0-probability branches are redundant.

A1 $P \,{}_p\!\sqcap\, P \equiv P$
A2 $Q \,{}_0\!\sqcap\, P \equiv P$
A3 $P \,{}_p\!\sqcap\, Q \equiv Q \,{}_{1-p}\!\sqcap\, P$
A4 $(P \,{}_{p/(1-q)}\!\sqcap\, Q) \,{}_{1-q}\!\sqcap\, R \equiv (R \,{}_{q/(1-q)}\!\sqcap\, Q) \,{}_{1-p}\!\sqcap\, P$
A5 $a \to (P \,{}_p\!\sqcap\, Q) \equiv (a \to P) \,{}_p\!\sqcap\, (a \to Q)$

We will use $\bigsqcap_{i \in I} p_i, P_i$ as a generalised form of probabilistic choice, in which I is a finite set of indices, the p_i are probabilities, and P_i process terms. The reason why we insist on I being finite is that all the choice axioms need only be stated for binary choice. If we allowed probabilistic choice with infinitely many branches, the axioms would have to be much more complicated. For instance, the idempotence axiom would be that two processes offering probabilistic choice are equal if there is a mapping from the branches of one to the other such that related branches behave identically and their probabilities add up to the same value. The premise of this axiom could only be asserted by taking recourse to set theory, so the axiomatisation would be only relatively complete, that is up to the point where it relied on set theory.

The axiomatic characterisation of the non-probabilistic constructs of $PCSP_0$ is the same as in standard CSP.
Axioms of parallel composition:

A6 $P \parallel Q \equiv Q \parallel P$
A7 $P \parallel STOP \equiv STOP$
A8 $(P \,{}_p\!\sqcap\, Q) \parallel R \equiv P \parallel R \,{}_p\!\sqcap\, Q \parallel R$
A9 $(a \to P) \parallel (a \to Q) \equiv a \to (P \parallel Q)$
A10 $a \neq b \Rightarrow (a \to P) \parallel (b \to Q) \equiv STOP$

Associativity of parallel composition can be proved from the axioms:

L1 $P \| (Q \| R) \equiv (P \| Q) \| R$

We use $\|_{i \in I} P_i$ as a general form of parallel composition.
Axioms of hiding:

A11 $STOP\backslash B \equiv STOP$

A12 $(a \to P)\backslash B \equiv \begin{cases} a \to P\backslash B & \text{if } a \notin B \\ P\backslash B & \text{otherwise} \end{cases}$

A13 $(P \,{}_p\!\sqcap Q)\backslash B \equiv P\backslash B \,{}_p\!\sqcap Q\backslash B$

Laws of hiding:

L2 $P\backslash \Sigma \equiv STOP$

L3 $P\backslash \{\} \equiv P$

L4 $(P\backslash B)\backslash C \equiv P\backslash B \cup C$

Axioms of renaming:

A14 $f(STOP) \equiv STOP$

A15 $f(a \to P) \equiv f(a) \to f(P)$

A16 $f(P \,{}_p\!\sqcap Q) \equiv f(P) \,{}_p\!\sqcap f(Q)$

Laws of renaming:

L5 $f(g(P)) \equiv (f \circ g)P$

L6 $f(P \| Q) \equiv f(P) \| f(Q)$

L7 $f(P\backslash B) \equiv f(B)\backslash f(B)$

Recursion Axiom:

A17 $\mu X \bullet P \equiv P[(\mu X \bullet P)/X]$

where $[a/x]$ denotes syntactic substitution of a for x. Axiom A17 shows that a recursively denoted process must be a fixed point of a recursive equation. Provided a fixed point exists and is unique, this justifies the use of recursive equations as process definitions and allows us to write, for example, $P = a \to P$ as an alternative to $P \triangleq \mu X \bullet a \to X$.

In $PCSP_0$, as in other models of *CSP*, a sufficient condition for the existence of a unique fixed point is that every recursive process invocation is guarded. However, perhaps surprisingly, unguarded recursions may also have a unique fixed point, namely if the following is true: Let P, Q be terms possibly containing the variable X and define the *contraction coefficient* c as follows:

$$c(STOP) = 0$$

$$c(X) = 1$$

$$c(a \to P) = 1/2$$

$$c(P \,{}_p\!\sqcap Q) = p \cdot c(P) + (1 - p) \cdot c(Q)$$

$$c(P \| Q) = c(P) + c(Q)$$

If $c(P) < 1$ then $\mu X \bullet P$ has a unique fixed point. For example, if $p < 1$ then

$$c(X \,{}_p\!\sqcap a \to X) = p \cdot 1 + (1 - p) \cdot (1/2) \cdot < 1.$$

In fact, $\mu X \bullet X \,{}_p\!\sqcap a \to X$ denotes the same process as $\mu X \bullet a \to X$ which performs a forever. This rule was arrived at by defining a metric on the semantic space on which $PCSP_0$ is based, and by showing that an expression denotes a contraction mapping with regard to this metric if the above condition holds. The proof is given [20], but is omitted here for reasons of space.

2.2. Semantics

2.2.1. Notation

Sequences of actions are called *traces*. The following is an informal summary of the notation we use for traces and operations on traces. (For the formal definitions see [9].) The notation is used both for finite and infinite traces unless otherwise stated.

$\langle\,\rangle$	the empty trace,
$\langle a \rangle$	the trace containing only a,
ts	concatenation of traces t and s (where t finite),
$\#t$	the length of a trace t ($\#t = \infty$ if t infinite),
$t_n, n < \#t$	$(n+1)$th element of a trace t (the first element is always t_0),
$t \restriction n$	restriction of a trace t to its first n actions,
$t \restriction B$	restriction of a trace t to actions in the set B,
$t \downarrow B$	the number of elements of B contained in t,
$t < u$	t is a proper prefix of u,
$t^n, n : \mathbf{N}$	a finite trace t repeated n times ($t^0 = \langle\,\rangle$),
t^ω	a finite trace t repeated infinitely many times,
$t/n, n \leqslant \#t$	t after n, i.e. t with its first n steps removed.

We introduce a special 'unobservable' action τ to encode as infinite traces with a tail $\langle\tau\rangle^\omega$ all finite traces after which a process may terminate. We write Σ_+ as shorthand for $\Sigma \cup \{\tau\}$. Let Ω denote the set of infinite traces of visible events and of infinite traces with a tail of τ's: $\Omega \triangleq \Sigma^\omega \cup \{t\langle\tau\rangle^\omega \mid t \in \Sigma^*\}$. The restriction function $\restriction$ adds a tail of τ's where $\restriction$ procedures a finite trace:

$$\forall z \in \Omega \cdot z \restriction B \triangleq \begin{cases} z \restriction B & \text{if } z \downarrow B = \infty \\ (z \restriction B)\langle\tau\rangle^\omega & \text{otherwise.} \end{cases}$$

Given a trace $t \in \Sigma^\ddagger$, let $S(t) \triangleq \{u : \Omega \mid u > t\}$ denote the set of infinite traces which are extensions of t. If t consists of a single element a we leave out the round brackets and write $S\langle a \rangle$. Note that the only trace leading on from a τ is the tail of τ's: $S(t\langle\tau\rangle) = \{t\langle\tau\rangle^\omega\}$. Also if $t \in \Sigma^*$ then $S(t\langle\tau\rangle)$ can be expressed as a difference of sets with τ-free prefixes:

$$S(t\langle\tau\rangle) = S(t) - \bigcup_{e \neq \tau} S(t\langle e \rangle).$$

2.2.2. Probability theory

We will need the following definitions and results from probability theory [2, 21] for the semantics of $PCSP_0$.

A *probability space* consists of a set Ω of points, a σ-field $\mathscr{F}$ defined on Ω and a probability measure P defined on $\mathscr{F}$. A *σ-field* is a family of sets on Ω which contains Ω and is closed under the formation of complements as well as finite and countable unions. A *probability measure* P on a σ-field $\mathscr{F}$ is a function $P: \mathscr{F} \to \mathbf{R}$ which satisfies the following conditions:

1. $\forall A \in \mathscr{F} \cdot 0 \leqslant PA \leqslant 1$,
2. $P\{\} = 0$, $P\Omega = 1$,
3. if $(A_n)_{n:\mathbf{N}}$ forms a disjoint sequence of $\mathscr{F}$-sets then $P\bigcup_n A_n = \sum_n P_n$.

Given two spaces $(\Omega, \mathscr{F})$ and $(\Omega', \mathscr{F}')$, a function $f: \Omega \to \Omega'$ is said to be *measurable* $\mathscr{F}/\mathscr{F}'$ if for all sets $A \in \mathscr{F}'$ the inverse image $f^{-1}A$ is an element of $\mathscr{F}$. Given a measure P on $(\Omega, \mathscr{F})$ and an $\mathscr{F}/\mathscr{F}'$-measurable function f we can transform P into a measure P' on $(\Omega', \mathscr{F}')$ by setting $P'A \triangleq Pf^{-1}A$ for any set $A \in \mathscr{F}'$. P' is called the measure *induced* by f. Most of the operators in our language are defined using transformation functions.

If $(\Omega', \mathscr{F}') = (\mathbf{R}, \mathscr{R})$, i.e. the real line with the σ-field generated by the open intervals, then f is called a *random variable*. If the range of f is a finite set of points, f is called a *simple function* or *simple random variable* and can be uniquely written in the form

$$f = \sum_{i=1}^{n} a_i I_{A_i},$$

where $\{a_i \mid 0 \leqslant i \leqslant n\}$ is the range of f, $A_i = f^{-1}a_i$, and I_{A_i} is the *indicator function*, defined as

$$I_A u \triangleq \begin{cases} 1 & \text{if } u \in A \\ 0 & \text{otherwise.} \end{cases}$$

In connection with product measures it will prove useful to use the notation of integrals. Integration of a simple function f with respect to a measure P is defined by

$$\int f(u) P(\mathrm{d}u) = \sum_{i=1}^{n} a_i Pf^{-1}a_i.$$

The definition of the integral of an arbitrary non-negative measurable function $f: \Omega \to \mathbf{R}$ is based on the fact that every such function is the limit of a monotonic increasing sequence of simple functions.

$$\int f(u) P(\mathrm{d}u) = \sup\left\{ \int s(u) P(\mathrm{d}u) \,\middle|\, s \leqslant f,\ s \text{ a simple function} \right\}.$$

Let $A \times B$ denote the Cartesian product of two sets: $A \times B \triangleq \{(u, v) \mid u \in A \wedge v \in B\}$. Given two probability spaces $(\Omega_X, \mathscr{F}_X, P_X)$ and $(\Omega_Y, \mathscr{F}_Y, P_Y)$ we construct the product space $(\Omega_{XY}, \mathscr{F}_{XY}, (P_X \times P_Y))$ as follows. The set Ω_{XY} consists of the pairs of points in $\Omega_X \times \Omega_Y$. The σ-field $\mathscr{F}_{XY}$ is generated by the *measurable rectangles* which are sets of

the form $A \times B$ where $A \in \mathscr{F}_X$ and $B \in \mathscr{F}_Y$. These sets have probability

$$(P_X \times P_Y)A \times B \triangleq P_X A P_Y B.$$

The probability of general $E \in \mathscr{F}_{XY}$ can be evaluated as $(P_X \times P_Y)E = \int P_X E_y P_Y(\mathrm{d}y)$ where $E_y = \{x \mid (x, y) \in E\}$.

2.2.3. The semantic function $[\![\ ]\!]$

For the semantics of $PCSP_0$ let PM denote the space of probability measures on $(\Omega, \mathscr{F})$ where $\mathscr{F}$ is the σ-field generated by the sets of infinite traces with a common prefix. The semantic function $[\![\ ]\!]: PCSP_0 \to PM$ maps $PCSP_0$ terms to probability measures.

Let A be any set in $\mathscr{F}$.

The semantics of $STOP$ is the point measure which gives probability 1 to the trace of unobservable actions and probability 0 to everything else:

$$[\![STOP]\!]A \triangleq \begin{cases} 1 & \text{if } \langle\tau\rangle^\omega \in A \\ 0 & \text{otherwise.} \end{cases}$$

The semantics of prefixing is defined as a tranformation of measures, based on the function $prefix_a: \Omega \to \Omega$.

$$[\![a \to P]\!]A \triangleq [\![P]\!]\, prefix_a^{-1} A \quad \text{where } \forall u \in \Omega \cdot prefix_a(u) = \langle a \rangle u$$

So, for example, for non-empty $t \in \Sigma_\ddagger^*$ the set $S(t)$ has inverse image $prefix_a^{-1} S(t) = S(t/t1)$ if $t_0 = a$, and $\{\}$ otherwise. Therefore, as expected,

$$[\![a \to P]\!]S(t) = \begin{cases} [\![P]\!]S(t/1) & \text{if } t_0 = a \\ 0 & \text{otherwise.} \end{cases}$$

The semantics of probabilistic choice is the weighted average of the probabilities assigned by its branches.

$$[\![P\ {}_p\sqcap\ Q]\!]A \triangleq p[\![P]\!]A + (1-p)[\![Q]\!]A.$$

The semantics of $P \backslash B$ is defined as a transformation of the measure denoted by P, induced by the function $hide_B: \Omega \to \Omega$ which removes all actions in B from the traces:

$$[\![P \backslash B]\!] \triangleq [\![P]\!]\, hide_B^{-1} A \quad \text{where } \forall u \in \Omega \cdot hide_B(u) = u \restriction B^c$$

In simple parallel composition two processes synchronise on every action that is performed. We would expect the probability that the parallel system $P \parallel Q$ performs an action to be the product of the probabilities with which the components P and Q perform this action. So it seems natural to define the measure for $P \parallel Q$ as a transformation of the product measure $([\![P]\!] \times [\![Q]\!])$. This transformation uses a function $par: \Omega \times \Omega \to \Omega$ which maps a pair of traces to the longest trace up to which

they agree. If that is a finite trace it adds a tail of unobservable actions. This reflects the fact that for the parallel system to perform an infinite trace u both component processes must perform u. If the component processes set out to perform traces which differ after n steps the parallel system will deadlock at that point.

$$[\![P \parallel Q]\!] A \triangleq ([\![P]\!] \times [\![Q]\!])\ par^{-1} A$$

$$\text{where } \forall u, v \in \Omega \cdot par(u, v) = \begin{cases} u & \text{if } u = v \\ (u \downarrow n) \langle \tau \rangle^{\omega} & \text{if } u \downarrow n = v \downarrow n \wedge u_n \neq v_n. \end{cases}$$

Let $f: \Sigma \to \Sigma$ be an injection. Lifting f to traces turns it into a transformation function, which can be used for renaming.

$$[\![f(P)]\!] A \triangleq [\![P]\!] f^{-1} A.$$

For the semantics of the recursion operator we define an ordering on the space PM by saying that a process P is below a process Q if the probability of Q performing any visible event is always higher than that of P:

$$P \sqsubseteq Q \triangleq \forall t \in \Sigma^* \bullet PS(t) \leqslant QS(t).$$

Let P be a term possibly containing the free variable X. Let $M(X, P)$ be the mapping on PM represented by P if X is bound to the argument of M

$$[\![\mu X \bullet P]\!] \triangleq \text{the least fixed point of } M(X, P) \text{ in } (PM, \sqsubseteq).$$

This semantics is well-defined only if we can establish that the least fixed point exists. By the Knaster–Tarski Fixed Point Theorem [22], if $(PM, \sqsubseteq)$ is a complete semi-lattice and $F: PM \to PM$ is continuous, then F has a least fixed point within PM.

Lemma 2.1. *$(PM, \sqsubseteq)$ is a complete partial order.*

Proof. $(PM, \sqsubseteq)$ is a complete partial order if PM has a least element under the ordering and every directed set $\Delta \in PM$ has a least upper bound in PM. The least element is the process $[\![STOP]\!]$, since $\forall t \in \Sigma^* \bullet [\![STOP]\!] S(t) = 0$. This least upper bound $P_{\sqcup}$ is the process such that $\forall t \in \Sigma^* \bullet [\![P_{\sqcup}]\!] S(t) = \text{lub}\{DS(t) \mid D \in \Delta\}$. □

Lemma 2.2. *$(PM, \sqsubseteq)$ forms a complete semi-lattice.*

Proof. Since $(PM, \sqsubseteq)$ is a cpo it only remains to show that every subset of PM has a greatest lower bound $P_{\sqcap}$ in PM. This can be constructed by taking $P_{\sqcap}$ such that $\forall t \in \Sigma^* \bullet [\![P_{\sqcap}]\!] S(t) = \text{glb}\{DS(t) \mid D \in \Delta\}$. □

Lemma 2.3. *Let P be a recursion-free term possibly containing the free variable X. Then $M(X, P)$ is continuous.*

Proof. We must establish that for Δ a directed set in $(PM, \sqsubseteq)$

$$M(X, P) \bigsqcup_{D\in\Delta} D = \bigsqcup_{D\in\Delta} M(X, P)D.$$

This is true if and only if $\forall t\in\Sigma^* \bullet M(X, P)(\bigsqcup_{D\in\Delta} D)S(t) = \mathrm{lub}_{D\in\Delta} M(X, P)D\,S(t)$. We give a proof by induction. The continuity of atomic processes, i.e. $[\![STOP]\!]$ and process variable X is trivial. For prefixing suppose that $M(X, P)$ is continuous. Then $\forall t\in\Sigma^*$

$$\begin{aligned} M(X, a\to P) \bigsqcup_{D\in\Delta} DS(t) &= M(X, P) \bigsqcup_{D\in\Delta} D\ prefix_a^{-1} S(t) \\ &= \mathrm{lub}_{D\in\Delta}\ M(X, P)D\ prefix_a^{-1} S(t) \\ &= \mathrm{lub}_{D\in\Delta}\ M(X, a\to P)D\,S(t) \end{aligned}$$

Continuity of probabilistic choice derives from the fact that taking the least upper bound distributes through addition. For all the other operators continuity follows because the inverse image of $S(t)$ under the transformation function can be expressed as a disjoint union of sets with fixed prefixes, so that taking the lub is possible because of the inductive hypothesis. □

Thus if P is recursion-free then $[\![\mu X \bullet P]\!]$ is well defined. Moreover, the least fixed point of $M(X, P)$ is given by the limit $\bigsqcup_{n>0} M(X, P)^n([\![STOP]\!])$. Since taking the lub preserves continuity, the result extends to terms which contain recursion.

2.3. *Soundness and completeness*

We now show that the axioms in Section 2.1 are sound and complete for finite processes. The proofs that the axioms are sound, i.e. consistent with the semantics, follow from the semantic definitions. The proofs for the distributivity of ${}_p\sqcap$ are all very similar to the following, which shows that prefixing distributes through probabilistic choice (A5). For all sets $A\in\mathscr{F}$

$$\begin{aligned} [\![a\to(P\ {}_p\sqcap Q)]\!]A &= [\![P\ {}_p\sqcap Q]\!]\ prefix_a^{-1} A \\ &= p[\![P]\!]\ prefix_a^{-1} A + (1-p)[\![Q]\!]\ prefix_a^{-1} A \\ &= p[\![a\to P]\!]A + (1-p)[\![a\to Q]\!]A \\ &= [\![(a\to P)\ {}_p\sqcap (a\to Q)]\!]A \end{aligned}$$

Algebraic equivalences between the other operators follow from the fact that different combinations of transformation functions are the same; obviously if $f\circ g = g\circ f$ then the measure induced by $f\circ g$ is the same as the measure induced by $g\circ f$. For example it is easy to check that $par\circ(prefix_a, prefix_a) = prefix_a\circ par$, which

proves the soundness of A9 (one of the axioms defining synchronisation between parallel processes). The remaining axioms define the effect of the operators on *STOP*. These can be proved sound by considering the effect of the transformation function on $\langle\tau\rangle^\omega$. For instance, A7 (any process in parallel with *STOP* deadlocks) is proved sound by

$$\begin{aligned}(\llbracket P \parallel STOP\rrbracket)A &= (\llbracket P\rrbracket \times \llbracket STOP\rrbracket)\,par^{-1}A\\ &= (\llbracket P\rrbracket \times \llbracket STOP\rrbracket)\,par^{-1}A \cap (\Omega \times \{\langle\tau\rangle^\omega\})\\ &\quad \text{since } (\llbracket P\rrbracket \times \llbracket STOP\rrbracket)(\Omega \times \{\langle\tau\rangle^\omega\}) = 1\\ &= \begin{cases} 1 & \text{if } \langle\tau\rangle^\omega \in A\\ 0 & \text{otherwise}\end{cases}\\ &\quad \text{since } \langle\tau\rangle^\omega \in A \Leftrightarrow (\Omega \times \{\langle\tau\rangle^\omega\}) \subseteq par^{-1}A\\ &= \llbracket STOP\rrbracket A.\end{aligned}$$

To show that the axiomatisation is complete we must show that whenever an assertion is true it is provable.

Let $PCSP_0^F$ be the subset of $PCSP_0$ consisting only of non-recursive terms with no free variables.

Definition 2.4. A $PCSP_0^F$ term is in *normal form* (NF) if it is *STOP* or if it is of the form $\bigsqcap_{a\in S} p_a, P_a$ where $\{\} \subset S \subseteq \Sigma_+$, $p_a > 0$ and

$$P_a = \begin{cases} STOP & \text{if } a = \tau\\ a \to P'_a, \text{ where } P'_a \text{ in NF} & \text{otherwise.}\end{cases}$$

Lemma 2.5. *Every $PCSP_0^F$ term is provably equivalent to a process in normal form.*

We will prove this by induction on the depth of terms. For P a $PCSP_0^F$ term, define

$$\begin{aligned} d(STOP) &= 0\\ d(a \to P) &= d(P) + 1\\ d(P \sqcap_p Q) &= d(P) + d(Q) + 1\\ d(P \parallel Q) &= d(P)\cdot d(Q) + 1\\ d(P\backslash B) &= d(P) + 1\\ d(f(P)) &= d(P) + 1\end{aligned}$$

Note that for a term in normal form we have $d(\bigsqcap_{a\in S} p_a, P_a) = \sum_{a\in S} d(P_a) + \#S - 1$.

Proof. For the base case note that the only process of depth 0 is $STOP$ and in normal form already. As inductive hypothesis take

$$d(P) = n \Rightarrow \exists Q \cdot Q \text{ in NF} \wedge d(Q) \leqslant d(P) \wedge \vdash Q \equiv P$$

We present only the case of parallel composition, since the checks for the other operators are straightforward. Suppose that $d(P \parallel Q) = n + 1$. Then $d(P) \leqslant n$ and $d(Q) \leqslant n$ and we can find equivalent normal form processes P_{NF} and Q_{NF}. If either of these is $STOP$ then by A7 we have $P \parallel Q \equiv STOP$ and we are done. Otherwise they must be of the form $P_{NF} = \bigsqcap_{a \in I} p_a, P_a$ and $Q_{NF} = \bigsqcap_{a \in J} q_a, Q_a$. Then

$$\begin{aligned} P \parallel Q &\equiv \left(\bigsqcap_{a \in I} p_a, P_a\right) \parallel Q \\ &\equiv \bigsqcap_{a \in I} p_a, (P_a \parallel Q) \quad \text{by A8} \end{aligned}$$

We can expand P_a to get

$$P_a \parallel Q = \begin{cases} STOP \parallel Q & \text{if } a = \tau \\ (a \to P_a') \parallel Q & \text{otherwise.} \end{cases}$$

By A7, $STOP \parallel Q \equiv STOP$. Also

$$a \to P_a' \parallel Q \equiv (a \to P_a') \parallel \left(\bigsqcap_{e \in J} q_e, Q_e \right)$$

$$\equiv \begin{cases} STOP & \text{if } a = \tau \\ a \to (P_a' \parallel Q_a')_{\,qa} \sqcap STOP & \text{otherwise} \end{cases} \quad \text{by A8–A10}$$

By hypothesis we have $d(P_a' \parallel Q_a') < d(P_{NF} \parallel Q_{NF}) \leqslant d(P \parallel Q)$. So we can find a normal form process R_a' which is equivalent to $P_a' \parallel Q_a'$. Thus we can write

$$\bigsqcap_{a \in I} p_a, (P_a \parallel Q) \equiv \bigsqcap_{a \in I} p_a, R_a$$

where

$$R_a = \begin{cases} STOP & \text{if } = \tau \\ a \to R_{a\,q_a} \sqcap STOP & \text{otherwise.} \end{cases}$$

Using associativity and distributivity of probabilistic choice, this can be transformed to the normal form process

$$\bigsqcap_{a \in K} r_a, R_a \quad \text{where } K = \{e\} \text{ if } I = J = \{e\}, (I \cap J) \cup \{\tau\} \text{ otherwise,}$$

$$r_a = p_a . q_a \text{ if } a \in K - \tau,\ r_\tau = 1 - \sum_{a \neq r} r_a \text{ and}$$

$$R_a = \begin{cases} STOP & \text{if } a = \tau \\ a \to R_a' & \text{otherwise.} \end{cases}$$

It only remains to show that this process has depth less than or equal to $d(P \parallel Q)$. Suppose that $\#I \geqslant 2$ and $\#J \geqslant 2$.

$$d\left(\prod_{a\in K} p_a, R_a\right) = \sum_{a\in K} d(R_a) + \#K - 1$$
$$\leqslant \sum_{a\in(I\cap J)-\{\tau\}} d(R_a) + \#(I\cap J)$$

where the last step takes account of a possible R_τ. For $a \neq \tau$ we can substitute

$$\begin{aligned} d(R_a) &= d(R'_a) + 1 \\ &\leqslant d(P'_a \parallel Q'_a) + 1 \\ &= d(P'_a).d(Q'_a) + 1 \\ &\leqslant d(P'_a).d(Q_a) \\ &\leqslant d(P_a).(d(Q) - \#J + 1) \end{aligned}$$

Thus

$$\begin{aligned} &\sum_{a\in(I\cap J)-\{\tau\}} d(R_a) + \#(I\cap J) \\ &\leqslant (d(Q) - \#J + 1) \sum_{a\in(I\cap J)-\{\tau\}} d(P_a) + \#(I\cap J) \\ &\leqslant (d(Q) - \#J + 1)(d(P) - \#I + 1) + \#(I\cap J) \\ &\leqslant d(Q)\cdot d(P) \end{aligned}$$

where the last step follows because $(\#J - 1)d(P) \geqslant d(P) \geqslant \#I \geqslant \#(I\cap J)$. The case where $\#I = 1$ or $\#J = 1$ can be treated similarly. □

It is likely that the axiomatisation could be made complete with regard to recursive terms by adding axioms about the ordering between processes, i.e. by stating that the ordering is transitive and anti-symmetric, that *STOP* is the least element and that all the operators are monotonic, as well as an axiom that a process P is below a process Q if all finite approximations of P are below Q. However, we have not checked this.

2.4. Examples

We now give examples to show how the semantics of $PCSP_0$ enables us to reason about properties of probabilistic processes. Among the most interesting properties of probabilistic processes are their asymptotic behaviours. For instance, very often a probabilistic process cannot be guaranteed to terminate within any finite number of

steps, but can be guaranteed to terminate eventually, with probability 1. Our first example shows how the algebraic laws of $PCSP_0$ can be used to transform two parallel coin-flipping processes into a sequential process, and hence to calculate the probability that they will deadlock eventually. For a property such as termination it is perhaps a disadvantage if it holds only asymptotically, but for others, such as fairness, it may be exactly what is required. Our second example shows how fairness can be expressed as a predicate on infinite traces, and its probability evaluated. The third example concerns a property which can only be expressed in probabilistic terms, namely the asymptotic frequency of an action.

Two parallel coin-flippers

In this example we examine two parallel coin-tossing processes and show that they will eventually deadlock. This also demonstrates that as in *CSP*, parallel composition in $PCSP_0$ is not idempotent. A coin-tossing process can be written as $P = hd \to P \,{}_{1/2}\sqcap\, tl \to P$. From the axioms it follows that in parallel with itself it behaves as

$$\begin{aligned} P \parallel P &\equiv (hd \to P \,{}_{1/2}\sqcap\, tl \to P) \parallel P \\ &\equiv ((hd \to P) \parallel P) \,{}_{1/2}\sqcap\, ((tl \to P) \parallel P) \\ &\equiv ((hd \to P \parallel P) \,{}_{1/2}\sqcap\, STOP) \,{}_{1/2}\sqcap\, ((tl \to P \parallel P) \,{}_{1/2}\sqcap\, STOP) \\ &\equiv STOP \,{}_{1/2}\sqcap\, ((hd \to P \parallel P) \,{}_{1/2}\sqcap\, tl \to P \parallel P) \end{aligned}$$

Let $A_k = \bigcup_{t \in \Sigma^*} S(t\langle\tau\rangle)$ be the set of traces such that $P \parallel P$ deadlocks after k steps. Then $[\![P \parallel P]\!] A_k = (1/2)^{k+1}$. Therefore the total probability of deadlock is $[\![P \parallel P]\!] \bigcup_{k=0}^{\infty} A_k = \sum_{k=0}^{\infty} [\![P \parallel P]\!] A_k = 1$.

Fairness

Let $P = \bigsqcap_{0 \leqslant n < N} p_n(a_n \to P)$ where the a_n are distinct and $p_n > 0$. We will show that P is fair in the sense that the set of traces of P which contain every a_n infinitely often has probability 1, i.e.

$$\forall 0 \leqslant n < N \cdot [\![P]\!] \limsup_i \{u \mid u_i = a_n\} = 1.$$

This means that the probability that from some point onwards one branch is overlooked forever is zero.

The proof relies on the second Borel–Cantelli lemma, which we quote from [2]: If (A_n) is a sequence of independent events and $\sum_n PA_n$ diverges then $P(\limsup_n A_n) = 1$. (Note that "event" here means "a set of points in a probability space", not to be confused with "event" as a synonym for action.)

Let A_i be the set of traces whose $(i+1)$th element is a_n: $A_i \triangleq \{u \mid u_i = a_n\} = \bigcup_{s \in \Sigma^i} S(s\langle a_n\rangle)$. For all $s \in \Sigma^i$ we have $[\![P]\!] S(s\langle a_n\rangle) = p_n [\![P]\!] S(s)$. Also

$\sum_{s\in\Sigma^i}[\![P]\!]S(s) = 1$. Hence

$$[\![P]\!]A_i = \sum_{s\in\Sigma^i} [\![P]\!]S(s\langle a_n\rangle) = p_n.$$

Similarly, it can be shown that if $i \neq j$ then any A_i, A_j are independent, i.e. $P(A_i \mid A_j) = p_n$. Consider the set $\limsup_i A_i = \bigcap_{i=1}^{\infty} \bigcup_{k=i}^{\infty} A_k$ consisting of all the traces which contain a_n infinitely often. Then A_i are independent and the sum $\sum_i [\![P]\!]A_i = \sum_i p_n$ diverges. Therefore by the Borel–Cantelli lemma $[\![P]\!](\limsup_i A_i) = 1$. Hence P is fair.

Asymptotic frequency of an action

An action a occurs with *asymptotic frequency* l in an infinite trace u if the ratio of occurrences of a to the length of successively longer prefixes of u tends towards the limit l:

$$\lim_{n\to\infty} \frac{(u\downarrow n)\downarrow\{a\}}{n} = l.$$

We say that the process P performs a with asymptotic frequency l if the probability of the traces in which a occurs with asymptotic frequency l is 1:

$$P\left\{u \,\middle|\, \lim_{n\to\infty} \frac{(u\downarrow n)\downarrow\{a\}}{n} = l\right\} = 1.$$

In the following we show that the process which we considered in the previous example not only performs every branch infinitely often, but with asymptotic frequency p_n

Consider the set $R(i,j)$ of traces which contain a run of j actions other than a_n after the ith a_n.

$$R(0,j) \triangleq \{u\downarrow(u\mid j)\downarrow\{a_n\} = 0 \wedge u_j = a_n\}$$

$$R(i,j) \triangleq \{t\langle a_n\rangle s\langle a_n\rangle u \mid \#s = j \wedge s\downarrow\{a_n\} = 0 \wedge t\downarrow\{a_n\} = i-1\} \quad i > 0.$$

Then $[\![P]\!]R(0,j) = p_n(1-p_n)^j$ and for $i > 0$

$$[\![P]\!]R(i,j) = \sum_{k=0}^{\infty} \binom{i+k-1}{k} (1-p_n)^i p_n^k (1-p_n)^j p_n = (1-p_n)^j p_n.$$

Thus $[\![P]\!]R(i,j)$ is independent of i. Similarly, it can be shown that the probabilities of two different runs are independent of each other. Let V_i be a random variable which records the number of actions other than a_n in the ith run, that is $V_i(u) = \sum_j j I_{R(i,j)}(u)$. The sequence (V_i) is a sequence of independent, identically distributed random variables, each of which has expected value $(1-p_n)p_n$. This translates into an expected ratio of the number of a_n's to the length of each run of $1/(E(V)+1) = p_n$. By the strong

law of large numbers (cf. [2]) this is the same as the asymptotic ratio for all runs. Hence as required

$$[\![P]\!]\left\{u \,\middle|\, \lim_{n\to\infty} \frac{(u\downarrow n)\downarrow\{a\}}{i} = p_n\right\} = 1.$$

3. The conditional model

In *CSP* the term $e:E\to P_e$ denotes a process which offers *deterministic* or *external* choice. Such a process will participate in whatever action e its environment offers, as long as e is in E, and then behave like P_e. If the environment offers an action outside E the system will deadlock. This behaviour cannot be described in $PCSP_0$, because the probability with which a $PCSP_0$-process decides what to do is always independent of its environment. External choice requires a notion of dependence or conditioning on the environment. We formalise this notion by defining a process as a conditional probability measure. The idea is that if a process P is offered a sequence $y\in\Omega$ by its environment, we know the probability with which it performs a set $A\in\mathscr{F}$. We use this to define the semantics of a second language, *PCSP*, which differs from $PCSP_0$ in that it contains operators for external choice and alphabetised parallel composition, but lacks the hiding operator.

3.1. Syntax and axioms

The syntax of *PCSP* contains the following constructs:

$$P ::= STOP \mid a\to P \mid P_p\sqcap Q \mid e:E\to P_e \mid P\,\Box\, Q \mid$$
$$P\,\|\,Q \mid P_B\|_C Q \mid f(P) \mid \mu X\bullet P$$

Operators which are present in both languages behave in the same way, and all the $PCSP_0$ axioms except the hiding axioms (A11–A13) also hold in *PCSP*. The additional constructs in *PCSP* behave as follows.

The expression $e:E\to P_e$ (where $E\subseteq\Sigma$) denotes the process which, when offered an action e in E, performs e with probability 1 and then behaves like P_e conditioned on the second and further steps of the environment. When offered an action outside E the process deadlocks. The axioms of external choice are:

A18 $e:\{\}\to P_e \equiv STOP$
A19 $e:\{a\}\to P_a \equiv a\to P_a$
A20 $e:E\to(P_e\,{}_p\sqcap Q_e) \equiv (e:E\to P_e)\,{}_p\sqcap(e:E\to Q_e)$.

The general choice operator $\Box$ denotes external choice between processes rather than actions. It is more restricted than in other models in *CSP*, because without a concept of non-deterministic choice, the choice between identical initial actions cannot be defined, so in *PCSP* external choice is defined only between processes with

disjoint sets of possible first steps. The possible first steps of a process can be determined syntactically, as follows:

$$fs(STOP) = \{\}$$

$$fs(X) = \{\}$$

$$fs(a \to P) = \{a\}$$

$$fs(e{:}E \to P_e) = E$$

$$fs(P \sqcap_p Q) = fs(P) \cup fs(Q)$$

$$fs(P \,\Box\, Q) = fs(P) \cup fs(Q)$$

$$fs(P \parallel_B^C Q) = (fs(P) \cap (B - C)) \cup (fs(P) \cap fs(Q) \cap B \cap C) \cup (fs(Q) \cap (C - B))$$

$$fs(f(P)) = \{f(e) \mid e \in fs(P)\}$$

$$fs(\mu X \bullet P) = fs(P)$$

The axioms of general choice are:

A21 $P \,\Box\, STOP \equiv P$.
A22 $P \,\Box\, (Q \sqcap_p R) \equiv (P \,\Box\, Q) \sqcap_p (P \,\Box\, R)$.
A23 $(e{:}E \to P_e) \,\Box\, (e{:}D \to Q_e) \equiv e{:}E \cup D \to R_e$ where

$$R_e \equiv \begin{cases} P_e & \text{if } e \in E \\ Q_e & \text{if } e \in D. \end{cases}$$

Associativity can be proved as a law:

L8 $(P \,\Box\, Q) \,\Box\, R \equiv P \,\Box\, (Q \,\Box\, R)$.

Let B and C be two sets of actions such that $\tau \in B \subseteq \Sigma_+$, $\tau \in C \subseteq \Sigma_+$. We write $P \parallel_B^C Q$ to denote the parallel composition of two processes P and Q such that P can only perform actions in B, Q can only perform actions in C, and P and Q synchronise on actions in the intersection $B \cap C$.

A24 $P \parallel_B^C Q \equiv Q \parallel_B^C P$.
A25 $(P \sqcap_p Q) \parallel_B^C R \equiv P \parallel_B^C R \sqcap_p Q \parallel_B^C R$.
A26 Let E and D be sets of visible actions. Then

$$(e{:}E \to P_e) \parallel_B^C (d{:}D \to Q_d) \equiv g{:}G \to P' \parallel_B^C Q'$$

where

$$E' = E \cap B,\ D' = D \cap C$$

$$G = (E' \cap D') \cup (E' - C) \cup (D' - B)$$

$$P' = P_g \text{ if } g \in E',\ P \text{ otherwise}$$

$$Q' = Q_g \text{ if } g \in D',\ Q \text{ otherwise.}$$

Laws:

L9 $(P_B \| _C Q) _{B \cup C} \| _D R \equiv P_B \| _{C \cup D} (Q_C \| _D (R)$.

L10 $a \in B \cap C \Rightarrow$

$$(a \to P)_B \| _C (a \to Q) \equiv a \to (P_B \| _C Q).$$

L11 $a \in B \cap C, b \in B - C \Rightarrow$

$$(b \to P)_B \| _C (a \to Q) \equiv b \to (P_B \| _C a \to Q).$$

L12 $a, b \in B \cap C, a \neq b \Rightarrow$

$$(b \to P)_B \| _C (a \to Q) \equiv STOP.$$

L13 $b \in B - C, c \in C - B \Rightarrow$

$$(b \to P)_B \| _C (c \to Q) \equiv b \to (P_B \| _C c \to Q) \,\square\, c \to ((b \to P)_B \| _C Q).$$

Additional renaming axioms:

A27 $f(e : E \to P_e) \equiv e : f(E) \to f(P)_e$.
A28 $f(P_B \| _C Q) \equiv f(P)_{f(B)} \| _{f(C)} f(Q)$.

3.2. Semantics

We will model dependence on an environment using conditional probability measures, which are defined as follows:

Definition 3.1. A conditional probability measure (cpm) is a function of two parameters, $P : \mathscr{F} \times \Omega \to [0, 1]$, such that
* for fixed $y \in \Omega$ and varying $A \in \mathscr{F}$, $P(A, y)$ is a probability measure and
* for fixed $A \in \mathscr{F}$ and varying $y \in \Omega$, $P(A, y)$ is a $\mathscr{F}$-measurable random variable.

Given functions f and $g \subseteq f^{-1}$ on infinite traces we can define a cpm P' as a transformation of a cpm P by setting $P'(A, z) \triangleq P(f^{-1}A, gz)$. Products and linear combinations of conditional probability measures can be defined in the same way as for probability measures.

Let CM be the space of conditional probability measures. We use round brackets $(\!|\ |\!)$ for the semantic function which defines the meaning of $PCSP$ terms:

$$(\!|\ |\!) = PCSP \to CM.$$

All definitions of the semantics of $PCSP$ are for any $\mathscr{F}$-set A and trace $y \in \Omega$.

The process $STOP$ deadlocks no matter what the environment offers and is therefore constant with respect to y:

$$(\!|STOP|\!)(A, y) \triangleq I_A \langle \tau \rangle^{\omega}.$$

If the environment offers an $a \in \Sigma$, the probability of $a \to P$ performing a set A is the probability of P performing $prefix_a^{-1} A$, which depends on the second and further actions offered by the environment. If the environment does not offer a, then $a \to P$ behaves like $STOP$. We therefore define

$$(\!|a \to P|\!)(A, y) \triangleq I_{S\langle a\rangle}(y)(\!|P|\!)(prefix_a^{-1} A, y/1) + I_{S\langle a\rangle^c}(y)(\!|STOP|\!)(A, y).$$

The semantics of external choice is a generalised form of the semantics of prefixing:

$$(\!|e : E \to P_e|\!)(A, y)$$
$$\triangleq \sum_{e \in E} I_{S\langle e\rangle}(y)(\!|P_e|\!)(prefix_e^{-1} A, y/1) + \sum_{e \notin E} I_{S\langle e\rangle}(y)(\!|STOP|\!)(A, y).$$

The semantics of probabilistic choice is the weighted sum of the conditional probabilities of the component processes.

$$(\!|P \sqcap_p Q|\!)(A, y) \triangleq p(\!|P|\!)(A, y) + (1 - p)(\!|Q|\!)(A, y).$$

The semantics of $P \,\square\, Q$ is a choice between conditional probabilities depending on whether the trace offered by the environment is in set $S_1 = \{u \mid u_0 \in fs(P)\}$ or in $S_2 = \{u \mid u_0 \in fs(Q)\}$.

$$(\!|P \,\square\, Q|\!)(A, y) \triangleq I_{S_1}(y)(\!|P|\!)(A, y) + I_{S_2}(y)(\!|Q|\!)(A, y) + I_{S_1^c \cap S_2^c}(\!|STOP|\!)(A, y).$$

Parallel composition works in the same way as before, that is the probability that the system performs a sequence of actions is the product of the probabilities with which the components take part. It is defined as a transformation of the product of the component cpm's, based on the function $cpar_{B,C,z} : \Omega \times \Omega \to \Omega$ which merges two sequences x and y as far as possible in accordance with the sequence offered by the environment. If the system is offered a trace $z \in \Omega$, then the component P is affected only by the steps in z which are elements of B, i.e. it behaves as P given $z \restriction B$.

$$(\!|P \,{}_B\|_C\, Q|\!)(A, z) \triangleq \int (\!|P|\!)((cpar_{B,C,z}^{-1} A)_y, z \restriction B)(\!|Q|\!)(dy, z \restriction C)$$

where $\forall x, y \in \Omega$

$$cpar_{B,C,z}(x, y) = \begin{cases} z & \text{if } z \downharpoonright B \leqslant x \wedge z \downharpoonright C \leqslant y \\ (z \downharpoonright n)\langle\tau\rangle^\omega & \text{if } (z \downharpoonright n) \downharpoonright B < x \wedge (z \downharpoonright n) \downharpoonright C < y \\ & \quad \wedge ((z \downharpoonright n + 1) \downharpoonright B \not< x \vee (z \downharpoonright n + 1) \downharpoonright C \not< y) \end{cases}$$

Simple parallel composition is the special case of alphabetised parallel composition which is synchronised on all actions: $P \| Q \equiv P \,{}_{\Sigma_+}\|_{\Sigma_+}\, Q$. We cannot define hiding as a transformation of cpm's because there is no function contained in $hide^{-1}$, which could be used to transform the trace offered by the environment. However, we can

define relabelling, since the relabelling functions $f: \Sigma \to \Sigma$ is injective.

$$(\!|f(P)|\!)(A, z) \triangleq (\!|P|\!)(f^{-1}A, f^{-1}z).$$

For the semantics of recursion for cpm's we take the same approach as in section 2. First we define an ordering on CM which is similar to the ordering on PM:

$$P \sqsubseteq Q \;\Leftrightarrow\; \forall u \in \Omega, \forall t \in \Sigma^* \bullet P(S(t), u) \leqslant Q(S(t), u).$$

Let $M(X, P)$ be the mapping on CM represented by P if X is bound to the argument of M

$$[\![\mu X \bullet P]\!] \triangleq \text{the least fixed point of } M(X, P) \text{ in } (CM, \sqsubseteq).$$

3.3. *Soundness and completeness*

The soundness of the axioms for $PCSP$ can be proved by appealing to the semantic definitions. These proofs are not difficult, but tedious, so we include only one in the appendix. To establish completeness we appeal again to a normal form for non-recursive processes, this time

Definition 3.2. A $PCSP^F$ term is in normal form if it is $STOP$ or if it is of the form

$\bigsqcap_{E \in I} p_E, P_E$ where I is a non-empty finite subset of $\mathbf{P}\Sigma$

$p_E > 0$, and

$$P_E = \begin{cases} STOP & \text{if } E = \{\} \\ e: E \to P_{E,e}, \text{ where } P_{E,e} \text{ is in NF} & \text{otherwise.} \end{cases}$$

Lemma 3.3. Every $PCSP^F$ term is provably equivalent to a process in normal form.

This lemma can again be proved by induction on the depth of terms, which is now define as

$$d(STOP) = 0$$

$$d(\bigsqcap_{i \in I} p_i, P_i) = \sum_{i \in I} d(P_i) + \#I - 1$$

$$d(P \,{}_A\|_B\, Q) = d(P).d(Q) + 1$$

$$d(e: E \to P_e) = \sum_{e \in E} d(P_e) + 1$$

$$d(P \,\Box\, Q) = d(P) + d(Q) + 1$$

$$d(f(P)) = d(P) + 1$$

3.4. Proof rules

So far we have defined a process algebra and given support for algebraic reasoning, but sometimes it may be desirable to specify the properties of a system or algorithm using predicates upon process behaviours. The following set of proof rules provides a link between these two styles of specification.

We say that a process satisfies a predicate if it can be shown that the predicate holds of every possible behaviour of the process. In the traces model of *CSP*, a behaviour is just a finite trace and the semantics of a process is the set of all possible behaviours of that process. In the probabilistic model a behaviour is an infinite trace. The semantics of a process is a cpm which assigns a probability to every behaviour, so the set of possible behaviours could naively be defined as every behaviour with non-zero probability. However, a process like the coin-tossing process would then not have any possible behaviour, since every single infinite trace has probability zero. We therefore define

$$traces(P) \triangleq \{u \mid \forall n \geqslant 0 \bullet (\!| P |\!)(S(u \downarrow n), u) > 0\}.$$

From this definition the following explicit expressions for the traces of a process can be derived.

Lemma 3.4.

$$traces(STOP) = \{\langle\tau\rangle^{\omega}\}$$

$$traces(a \to P) = \{\langle\tau\rangle^{\omega}\} \cup \{\langle a\rangle u \mid u \in traces(P)\}$$

$$traces(e : E \to P_e) = \{\langle\tau\rangle^{\omega}\} \cup \{\langle e\rangle u \mid e \in E \wedge u \in traces(P_e)\}$$

$$traces(P \,{}_p\!\sqcap\, Q) = \begin{cases} traces(P) & \text{if } p = 1 \\ traces(P) \cup traces(Q) & \text{if } 0 < p < 1 \\ traces(Q) & \text{if } p = 0 \end{cases}$$

$$traces(P \,\Box\, Q) = traces(P) \cup traces(Q)$$

$$traces(P \,{}_B\|_C\, Q) = \{u \mid u \in (B \cup C)^{\omega} \wedge (u \restriction B) \in traces(P) \wedge (u \restriction C) \in traces(Q)\}$$

$$traces(f(P)) = \{f(u) \mid u \in traces(P)\}$$

$$traces(\mu X \bullet P) = \bigcup_{n \geqslant 0} traces(P^n) \text{ where } P^0 \equiv STOP \text{ and } P^{n+1} \equiv P[P^n/STOP]$$

We say that a process P satisfies a specification expressed as a predicate R with free variable u if R is true of every trace of P:

$$P \textbf{ sat } R \triangleq (u \in traces(P) \Rightarrow R(u)).$$

We now present an inference rule for each clause in the syntax of *PCSP*, expressing the properties of a process in terms of predicates with several components. For compound

processes, the antecedent of the rule will consist of component specifications for the component processes.

The definition of **sat** gives rise to the usual logical rules:

$$\frac{}{P \textbf{ sat } true} \qquad \frac{\begin{array}{c} P \textbf{ sat } R \\ P \textbf{ sat } T \end{array}}{P \textbf{ sat } (R \wedge T)} \qquad \frac{\begin{array}{c} P \textbf{ sat } R \\ R \Rightarrow T \end{array}}{P \textbf{ sat } T}$$

The process $STOP$ is unwilling to participate in any external activity. The first visible action performed by $a \rightarrow P$ must be a and the subsequent behaviour is that of P. So the inference rules for $STOP$ and $a \rightarrow P$ are

$$\frac{}{STOP \textbf{ sat } u = \langle\tau\rangle^{\omega}} \qquad \frac{P \textbf{ sat } R}{a \rightarrow P \textbf{ sat} (u = \langle\tau\rangle^{\omega}) \vee (u_0 = a \wedge R(u/1))}$$

These last two rules are special cases of the following:

$$\frac{\forall e \in E \cdot P_e \textbf{ sat } R_e}{e : E \rightarrow P_e \textbf{ sat} (u = \langle\tau\rangle^{\omega}) \vee (u_0 \in E \wedge R_{u_0}(u/1))}$$

Any behaviour of the choice $P \,\square\, Q$ must arise from either P or Q. The same is true of the probabilistic choice $P_p \sqcap Q$, unless $p = 1$ or $p = 0$. This gives rise to the inference rules

$$\frac{\begin{array}{c} P \textbf{ sat } R \\ Q \textbf{ sat } T \end{array}}{P \,\square\, Q \textbf{ sat} (R \vee T)} \qquad \frac{\begin{array}{c} P \textbf{ sat } R \\ Q \textbf{ sat } T \end{array}}{P_p \sqcap Q \textbf{ sat } (R \vee T)} \quad [0 < p < 1]$$

$$\frac{P \textbf{ sat } R}{P_1 \sqcap Q \textbf{ sat } R} \qquad \frac{Q \textbf{ sat } T}{P_0 \sqcap Q \textbf{ sat } T}$$

A parallel system $P_B \| _C Q$ preserves the properties which are satisfied by its components:

$$\frac{\begin{array}{c} P \textbf{ sat } R \\ Q \textbf{ sat } T \end{array}}{P_B \| _C Q \textbf{ sat } u \in (B \cup C)^{\omega} \wedge R(u \restriction B) \wedge T(u \restriction C)}$$

A recursive process satisfies a predicate R upon traces, if this is preserved by an unfolding of the recursion.

$$\frac{X \textbf{ sat } R \Rightarrow P \textbf{ sat } R}{\mu X \bullet P \textbf{ sat } R}$$

Most of these rules can be proved sound very easily. For example, to show that the rule for parallel composition is sound, we use a fact established by Lemma 3.4,

namely that

$$\forall u \bullet u \in traces(P_B \|_C Q)$$
$$\Rightarrow u \in (B \cup C)^{\omega} \wedge (u \restriction B) \in traces(P) \wedge (u \restriction C) \in traces(Q)$$

Together with the antecedents of the rule this implies the consequent. The soundness proof for recursion follows the proof steps given by [19, 5].

The proof rules are also complete in the sense that if every behaviour in $traces(P)$ meets predicate R then the proof rules are sufficient to prove that P **sat** R. The proof of completeness is analogous to the proofs of completeness given for the inference rules in other models [4]. It uses structural induction on the syntax of *PCSP* to establish that the inference rules are sufficient to ascertain the predicate $u \in traces(P)$. If a behavioural specification $R(u)$ holds of a process P then $u \in traces(P) \Rightarrow R(u)$ and the inference rule for weakening specifications can be used to complete the proof that P **sat** R.

3.5. *Safety and liveness*

Note that since every process behaves like *STOP* when offered $\langle \tau \rangle^{\omega}$ by the environment, i.e. $\forall P \bullet \langle \tau \rangle^{\omega} \in traces(P)$, it follows that P **sat** $R \Rightarrow R(\langle \tau \rangle^{\omega}) = true$. So typically R is a predicate which constrains what a process may do if it does anything at all. Such a constraint is called a *safety* property, as opposed to a *liveness* property which asserts that a process will do something.

If a safety property is violated, then at some finite point some undesired behaviour occurs which is irremediable. For instance, the statement that certain actions always happen in the same order constitutes a safety property because once the order has been violated, it cannot be restored by any later actions. By contrast, a liveness property can be satisfied at some point in the future no matter what happens initially. Typical liveness properties are fairness, asymptotic behaviours, starvation freedom and termination. These observations motivate the following definitions which we adopt from Alpern and Schneider [1].

Definition 3.5. A predicate R upon infinite sequences with free variable u represents a *safety property* if $\forall u : \Omega \cdot \neg R(u) \Rightarrow \exists n : \mathbb{N} \cdot S(u \downarrow n) \cap R = \{\}$.

Definition 3.6. A predicate R upon infinite sequences with free variable u represents a *liveness property* if $\forall t \in \Sigma^* \cdot S(t) \cap R \neq \{\}$.

The proof rules presented in the last section are most useful for safety properties. For liveness properties we have to assume that the environment does not block the progress of the system whose properties we are trying to prove, i.e. that the environment resolves every external choice on which the system depends, but accepts every

internal (that is: probabilistic choice) made by the system. It turns out that any process combined with such an environment can be modelled simply as a probability measure, rather than as a cpm.

In this section we identify a subset of *PCSP* which has a semantics both in *CM* and in *PM*. We show that the semantics $(\!|\ |\!)$ of a construct in this subset of the language is related to its semantics as given by $[\![\ ]\!]$ by a transformation function on traces. We then show that the assumption we make of the environment of a process to analyse its liveness properties results in a system that belongs to this subset of *PCSP*. So to analyse liveness properties we never have to consider cpm's, but only simple probability measures. Therefore the same techniques which we used in the earlier examples to prove liveness properties of the rather limited class of $PCSP_0$ processes can also be used to analyse processes in general.

First note that every probability measure can be used to induce a conditional probability measure.

Lemma 3.7. *If P is a probability measure in PM, then the function defined as*

$$Q(A, y) \triangleq P\ cond_y^{-1}\ A$$

where $cond_y(x) \triangleq par(x, y)$ *is a cpm.*

Proof. We know that *par* is measurable $(\mathscr{F} \times \mathscr{F})/\mathscr{F}$. Hence $cond_y$ is measurable $\mathscr{F}/\mathscr{F}$. So for fixed y, $P\ cond_y^{-1}\ A$ induces a probability measure. It remains to prove that for fixed $A \in \mathscr{F}$ the function $P\ cond_y^{-1}\ A$ is $\mathscr{F}$-measurable. Let $\mathscr{C}$ denote the class of sets such that for $C \in \mathscr{C}$ the function $P\ cond_y^{-1}\ C$ is $\mathscr{F}$-measurable. Suppose first that $C = S(t)$ where t is τ-free. Then

$$cond_y^{-1}\ S(t) = \begin{cases} S(t) & \text{if } y \in S(t) \\ \{\} & \text{otherwise.} \end{cases}$$

Therefore $P\ cond_y^{-1} S(t) = I_{S(t)}(y)\, PS(t)$, which is a simple random variable. If t is not τ-free, the value of $P\ cond_y^{-1} S(t)$ can be computed as the difference of the probabilities of τ-free traces, i.e. as a difference of simple random variables. So in this case, too, $P\ cond_y^{-1} S(t)$ is a simple random variable. So $\mathscr{C}$ contains all the sets with fixed prefixes. It is easily shown that $\mathscr{C}$ is closed under finite unions and countable intersections. Therefore it is a monotone class and hence $\mathscr{C} = \mathscr{F}$. □

So for every probability measure we can construct a corresponding cpm. However, what we really need is to identify when a cmp has a corresponding probability measure.

Lemma 3.8. A *PCSP*-process $(\!|\ P\ |\!)$ has a corresponding probability measure $[\![P]\!]$ if and only if $\forall n : \mathbf{N} \cdot \sum_{t \in \Sigma^n} (\!|\ P\ |\!)(S(t), t\langle\tau\rangle^\omega) \leqslant 1$.

Proof. If t is a τ-free trace of length n then $cond^{-1}_{t\langle\tau\rangle^\omega} S(t) = S(t)$. Therefore if $(\!|P|\!)$ is a *PCSP*-process with a corresponding probability measure $[\![P]\!]$ then

$$[\![P]\!]S(t) = [\![P]\!]\, cond^{-1}_{t\langle\tau\rangle^\omega} S(t)$$

$$= (\!|P|\!)(S(t), t\langle\tau\rangle^\omega).$$

This means that $\sum_{t\in\Sigma^n}(\!|P|\!)(S(t), (t\langle\tau\rangle^\omega) \leqslant 1$ or else $[\![P]\!]$ would not be a probability measure. □

The next lemma provides a rule by which the existence of a probability measure corresponding to a cpm can be checked syntactically rather than by recourse to the semantics.

Lemma 3.9. *If P is a PCSP-term containing only STOP, $\rightarrow$, ${}_p\sqcap$, $\|$ and possibly variables bound by recursion, then the semantics of P in PCSP and $PCSP_0$ are related by*

$$(\!|P|\!)(A, y) = [\![P]\!]\, cond_y^{-1} A.$$

The proof of this lemma is given by [20], but omitted here.

4. A self-stabilising tokenring

To demonstrate how *PCSP* may be used, we now give a formal specification and proof of liveness of a self-stabilising tokenring, which is due to [8]. Its purpose is to pass a token around a cyclically arranged group of processes. The process in possession of the token can execute some task without interference from any other process. For our purposes the nature of the task is immaterial. Each process is in one of two states; it alternately reads the state of its left-hand neighbour and passes its own state to its right-hand neighbour. Every process which is in the same state as its left-hand neighbour is said to have a token. A process without a token keeps its state, whereas a process with a token changes state with probability $\frac{1}{2}$. This causes the token to pass to the next process. The total number of processes must be odd, so that under normal conditions all neighbouring processes bar one pair are in different states.

We will prove this algorithm to be self-stabilising in the sense that whatever their initial states, the processes eventually reach a state where exactly one token exists (i.e. spurious tokens disappear). Informally, this can be explained as follows. Suppose there are two adjacent spurious tokens in the ring. This corresponds to three adjacent processes with identical states, two of which conclude that they have a token. Both tokens will disappear if only the left-most tokenholder decides to change state. Non-adjacent spurious tokens disappear, because they are bound to collide eventually.

We will also prove that the tokenring is live in the sense that each process is guaranteed to receive the token infinitely often.

The tokenring TR consists of an odd number, N say, of identical processes operating in parallel. They are numbered by an index i, $0 \leqslant i \leqslant N-1$. As is shown in Fig. 1 each process i. P communicates only with its immediate neighbours, namely $i \ominus 1.P$ and $i \oplus 1.P$ where $\oplus$ and $\ominus$ denote addition and subtraction modulo N. The channels between the processes are named only by their index. Communication between $i.P$ and $i \ominus 1.P$ occurs on channel i and communication between $i.P$ and $i \oplus 1.P$ occurs on channel $i \oplus 1$. The values communicated are booleans, so communication events are of the form $i.j$ where $j \in \{0, 1\}$.

$$TR = \|_{B_i} i.P \quad \text{where } 0 \leqslant i < N \text{ and } B_i = \{i.0, i.1, (i \oplus 1).0, (i \oplus 1).1\}.$$

Each process is parameterised by its state which may be either 0 or 1. The algorithm could start in any state, but for simplicity's sake we assume that every process starts out in state 0. The first process starts communication by outputting its initial state to its right-hand neighbour, i.e. on channel 1.

$$0.P = 1!0 \rightarrow 0.P_0$$

$$i.P = i.P_0 \qquad i > 0.$$

Afterwards, every process first asks for input from its left-hand neighbour, outputs its own state to its right-hand neighbour, and then decides whether or not to change state. If the state of its left-hand neighbour is different to its own, it does not have the token and keeps the same state. Otherwise it chooses to change state with probability $\frac{1}{2}$, which will transfer the token to its right-hand neighbour. For $m \in \{0, 1\}$, $0 \leqslant i < N$ define

$$i.P_m = i?l \rightarrow i \oplus 1!m \rightarrow \quad \text{if } l = m \text{ then } i.P_{0\ 1/2}\sqcap i.P_1 \text{ else } i.P_m$$

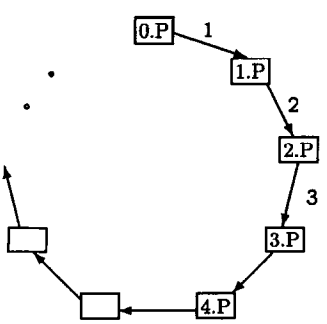

Fig. 1. A tokenring.

Using the laws of parallel composition we can show that TR is equivalent to the sequential process TS: Let $S \in \{0, 1\}^N$ denote the N-tuple of the states of the processes. For $0 \leqslant i < N$ define

$$TS = T(0, \{0\}^N)$$

$$\begin{aligned} T(i, S) = i.S(i) \to \text{if } & S(i \ominus 1) = S(i) \\ & \text{then } T(i \oplus 1, S[0/S(i)])_{1/2} \sqcap T(i \oplus 1, S[1/S(i)]) \\ & \text{else } T(i \oplus 1, S) \end{aligned}$$

where $S[b/S(i)]$ denotes the state S with its ith element overwritten with the value b. Since TS does not contain any external choice, it can be analysed as a probability measure rather than a cpm. We present a proof of correctness which is an alternative to the one given in [8]. The difference is that [8] starts from first principles, whereas the proof given here exploits some general results about finite Markov chains and shows that the invariant proved by [8], namely that the algorithm never increases the number of tokens, only needs to hold in the special case where the number of tokens is one.

Theorem 4.1. *The tokenring is self-stabilising and live.*

Proof. Let *chan* and *msg* be the obvious projection functions on communication events. Clearly communication in TS happens in rounds:

$$[\![TS]\!]\,\{u : \Omega \mid chan(u_n) = n \bmod N\} = 1.$$

After a communication on channel i only the state $S(i)$ is affected. Therefore in one round of communication each part of the state may change at most once. Let $A(k, s)$ be the set of traces such that the states communicated in round k are S:

$$A(k, S) \triangleq \{u : \Omega \mid \forall 0 \leqslant i < N \cdot msg(u_{kN+i}) = S(i)\}.$$

Let $t(S)$ be the number of tokens in the ring. If the total number of processes is odd, then at least one process must have a token.

The probability that the state of the tokenring is S' in round $k+1$ given that it was S in round k is

$$[\![TS]\!]\,(A(k+1, S') \mid A(k, S))$$

$$= \begin{cases} \frac{1}{2}^i & \text{if } t(S) = i \wedge \forall j \cdot S(j \ominus 1) = S(j) \Rightarrow S'(j) = S(j) \\ 0 & \text{otherwise.} \end{cases}$$

Since there are only finitely many states, and the transition probability from one to the next does not depend on any previous states the sets $A(k, S)$ form a finite Markov chain. From a one-token state, only two transitions are possible. Both are again

one-token states. So the set of states in which exactly one process has the token is a *closed* set in the sense that the transition probabilities from any element of this set to any element outside this set are all zero. All states outside this set are *transient* in the sense that the probability of eventual return to this state is strictly less than one. In a finite Markov chain the probability of staying forever in a set of transient states is zero [6]. So the tokenring will eventually end up in a state where exactly one process has the token, and from that point onwards the only other states it can visit are those where exactly one process has the token. Thus the tokenring is guaranteed to stabilise.

The closed set by itself represents a finite irreducible Markov chain, in which all states are persistent [6], i.e. all states are visited infinitely often. So the tokenring is live in the sense that every process is guaranteed to receive the token infinitely often. □

5. Related work and conclusions

5.1. Related work

There exist several formalisms for the specification of probabilistic processes, reflecting the variety of formal methods in general. Broadly speaking, all probabilistic languages define the semantics of choice and parallel composition in terms of sums and products of probabilities respectively. Differences arise in the treatment of external choice and unsynchronised parallel composition, as well as in the methods of defining fixed points and equivalences between processes.

Miguel et al. [14] have added probabilistic choice to LOTOS [10], with a semantics based on probabilities which are conditioned on experiments. Glabbeek et al. [7] present three models for PCCS, a probabilistic dialect of Milner's SCCS [15]. The semantics of these models are based on probabilistic labelled transition systems, which are state transition systems with probabilities attached to each branch. Differences between the models arise from the treatment of choice: in the 'reactive' model the probabilities for all transitions with the same action sum to 1, whereas in the 'generative' model the probabilities for all transitions sum to 1. The former can be understood as a mixture of internal and external choice, in the sense that the choice of action is made externally but the choice of transition with a given action is made internally. Parallel composition is defined as lockstep interleaving. Equivalence between processes is established by probabilistic bisimulation (due to Larsen and Skou [13]), which is an analog of strong bisimulation. This leads to very fine distinctions between processes; for instance it rules out the law of distributivity of probabilistic choice over prefixing.

Jou and Smolka [12] investigate weaker concepts of process equivalence for the generative model and present a sound and complete axiomatisation of finite serial processes in the generative model with respect to probabilistic bisimulation. The only

laws that hold for probabilistic choice are symmetry, associativity and idempotence and there are no laws for parallel composition.

Jones and Plotkin [11] present the semantics of a probabilistic programming language consisting of atomic commands, sequential composition, if-statements, while-loops, probabilistic choice and interleaving. The latter is parametrised on a probabilistic scheduler which decides, given a state, which process runs next. There is no construct for input or external choice.

Rao [12] presents a probabilistic extension to UNITY [3]. He introduces probabilistic assignment, which probabilistically chooses one of a finite list of possible expressions to assign to a variable. The probability with which an expression is chosen is arbitrary and cannot be made explicit so that an algorithm whose correctness depends on a specific probability cannot be proved correct. The only probabilistic property important for Rao is that in an infinite trace of executions of a probabilistic assignment each expression will be chosen infinitely often. He defines the weakest precondition of the probabilistic assignment as the one which holds of every branch. This enables him to extend the usual UNITY proof rules for safety properties to probabilistic programs. He then defines the weakest *probabilistic* precondition as one which must hold of at least one branch and uses it to develop a set of proof rules for liveness properties, which hold with probability 1. His approach is closest to our own in that he also uses infinite traces and constructs separate proof rules for safety and liveness properties.

5.2. *Conclusions*

We have given an axiomatisation and a denotational semantics for a probabilistic specification language called $PCSP_0$ which is based on *CSP*. In the semantics of $PCSP_0$ processes are defined as probability measures on infinite traces, and operators as transformations of measure or weighted sums of measures. Although this semantics is very different to the semantics of *CSP*, the axiomatisation is very similar, which is what we wanted. We have shown the axiomatisation to be sound and complete with respect to the semantics. We have given examples which show that $PCSP_0$ enables us to reason about important properties such as fairness and the asymptotic frequencies of actions.

The main disadvantage of $PCSP_0$, which limits its usefulness, is that it lacks the operators for general parallel composition and external choice. To solve this problem, we have given a semantics for a second model in terms of conditional probability measures. It contains operators for external choice and alphabetised parallel composition, but not for hiding. Like the first model it has a sound and complete axiomatisation of algebraic laws. Furthermore, we have given sound and complete proof rules to relate the process algebra to specifications defined in terms of predicates upon infinite traces. This has enabled us to reason about safety properties. To reason about liveness properties it is necessary to make some assumptions about the environment of a system, namely that the environment does not block the system and that it resolves

all the external choices on which the system depends. We have shown that given such an environment, the resulting system has a well-defined semantics both in the first and in the second model, which means the techniques we used to analyse liveness properties in the first model are also applicable to systems specified in the second model. We have demonstrated this in the example of a self-stabilising tokenring.

Appendix. Soundness of axiom A28

Proof. Let $(\!|P|\!) = (\!|e\colon E \to P_e|\!)$ and let $(\!|Q|\!) = (\!|d\colon D \to Q_d|\!)$. By definition

$$(\!|P_B \|_C Q|\!)(A, z) = \int (\!|P|\!)((cpar^{-1}_{B,C,z} A)_y, z \restriction B)(\!|Q|\!)(\mathrm{d}y, z \restriction C).$$

If $z_0 = e \in E \cap D$ then $(z \restriction B)_0 = e$ and $(z \restriction C)_0 = e$. Therefore in this case

$$\begin{aligned}
&(\!|P_B \|_C Q|\!)(A, z)\\
&\quad = \int (\!|e \to P_e|\!)((cpar^{-1}_{B,C,z} A)_w, z \restriction B)(\!|e \to Q_e|\!)(\mathrm{d}w, z \restriction C)\\
&\quad = \int (\!|P_e|\!)(prefix_e^{-1}(cpar^{-1}_{B,C,z} A)_{\langle e\rangle w}, (z \restriction B)/1)(\!|Q_e|\!)(\mathrm{d}w, (z \restriction C)/1)\\
&\quad = \int (\!|P_e|\!)(cpar_{B,C,z/1}(prefix_e^{-1} A)_w, (z/1) \restriction B)(\!|Q_e|\!)(\mathrm{d}w, (z/1) \restriction C)\\
&\quad = (\!|P_{e\,B} \|_C Q_e|\!)(prefix_e^{-1} A, z/1)\\
&\quad = (\!|e \to (P_{eB} \|_C Q_e)|\!)(A, z).
\end{aligned}$$

If $z_0 = e \in E - C$ then $(z \restriction B)_0 = e$ and $z \restriction C = (z/1) \restriction C$. Therefore

$$\begin{aligned}
&(\!|P_B \|_C Q|\!)(A, z)\\
&\quad = \int (\!|e \to P_e|\!)((cpar^{-1}_{B,C,z} A)_y, (z \restriction B))(\!|Q|\!)(\mathrm{d}y, z \restriction C)\\
&\quad = \int (\!|P_e|\!)(prefix_e^{-1}(cpar^{-1}_{B,C,z} A)_y, (z \restriction B)/1)(\!|Q|\!)(\mathrm{d}y, z \restriction C)\\
&\quad = \int (\!|P_e|\!)(cpar_{B,C,z/1}(prefix_e^{-1} A)^{-1} y, (z/1) \restriction B)(\!|Q|\!)(\mathrm{d}y, (z/1) \restriction C)\\
&\quad = (\!|P_{e\,B} \|_C Q|\!)(prefix_e^{-1} A, z/1)\\
&\quad = (\!|e \to (P_{e\,B} \|_C Q|\!)(A, z).
\end{aligned}$$

Similarly, if $z_0 = e \in D - B$ then

$$(\!|P_B \|_C Q|\!)(A, z) = (\!|e \to (P_B \|_C Q_e)|\!)(A, z).$$

If $z_0 \in (B - E) \cap C$ then $(z \restriction B)_0 = z_0$ and $(z \restriction C)_0 = z_0$. So

$$(\!|P_B \|_C Q|\!)(A, z) = \int (\!|STOP|\!)((cpar^{-1}_{B,C,z} A)_y, (z \restriction B))(\!|Q|\!)(\mathrm{d}y, z \restriction C).$$

Also $z_0 \in (B - E) \cap C \Rightarrow cpar_{B,C,z}(\langle\tau\rangle^\omega, y) = \langle\tau\rangle^w$ for all y. Therefore it follows that $cpar^{-1}_{B,C,z} A \supseteq \{\langle\tau\rangle^\omega\} \times \Omega \Leftrightarrow \langle\tau\rangle^\omega \in A$ and

$$(\!|P_B \|_C Q|\!)(A, z) = (\!|STOP|\!)(A, z).$$

The same is true if $z_0 \in (C - D) \cap B$. Finally, if $z_0 \in (D \cup C)^c$ then

$$\begin{aligned}(\!|P_B \|_C Q|\!)(A, z) &= \int (\!|STOP|\!)((cpar^{-1}_{B,C,z} A)_y, (z \restriction B))(\!|STOP|\!)(\mathrm{d}y, z \restriction C)\\ &= (\!|STOP|\!)(A, z)\end{aligned}$$

since $\{\langle\tau\rangle^\omega\} \times \{\langle\tau\rangle^\omega\} \in cpar^{-1}_{B,C,z} A \Leftrightarrow \langle\tau\rangle^\omega \in A$. Drawing all these cases together, we get

$$\begin{aligned}(\!|P_B \|_C Q|\!)(A, z) = &\sum_{e \in E \cap D} I_{S\langle e\rangle}(z)(\!|e \to (P_{e\,B} \|_C Q_e)|\!)(A, z)\\ &+ \sum_{e \in E - C} I_{S\langle e\rangle}(z)(\!|e \to (P_{e\,B} \|_C Q)|\!)(A, z)\\ &+ \sum_{e \in D - B} I_{S\langle e\rangle}(z)(\!|e \to (P_B \|_C Q_e)|\!)(A, z)\\ &+ \sum_{e \in \Sigma - G} I_{S\langle e\rangle}(z)(\!|STOP|\!)(A, z)\end{aligned}$$

where $G = (E \cap D) \cup (E - C) \cup (D - B)$. Thus as required

$$(\!|P_B \|_C Q|\!)(A, z) = (\!|g : G \to (P'_B \|_C Q')|\!)(A, z)$$

where P', Q' as defined in A28. □

References

[1] B. Alpern and F.B. Schneider, Defining liveness, *Inform. Process. Lett.* **21** (1985) 181–185.
[2] P. Billingsley, *Probability and Measure* (Wiley, New York, 1979).
[3] K.M. Chandy and J. Misra, *Parallel Program Design: A Foundation* (Addison-Wesley, Reading, MA, 1988).
[4] J. Davies, Specification and proof in real-time systems, Oxford University D. Phil Thesis, 1991.
[5] J. Davies and S. Schneider, Recursion induction for real-time processes, *Formal Aspects Computing*, to appear.

[6] W. Feller, *An Introduction to Probability Theory and its Applications*, Vol. I (Wiley, New York, 2nd ed. 1957).
[7] R.J. van Glabbeek, S.A. Smolka, B. Steffen and C. Tofts, Reactive, generative and stratified models of probabilistic processes, in: *IEEE Symp. on Logic in Computer Science*, Philadelphia, PA, USA (1990).
[8] T. Herman, Probabilistic self-stabilization, *Inform. Process. Lett.* **35** (1990) 63–67.
[9] C.A.R Hoare, *Communicating Sequential Processes* (Prentice-Hall, Englewood Cliffs, NJ, 1985).
[10] ISO, LOTOS: a formal description technique based on the temporal ordering of observational behaviour, IS 8807, TC97/SC21, 1989.
[11] C. Jones and G. Plotkin, A probabilistic powerdomain of evaluations, in: *Proc. 4th Ann. Symp. on Logic in Computer Science* (1989).
[12] C. Jou and S.A. Smolka, Equivalences, congruences, and complete axiomatizations for probabilistic processes, *Concur 90, Theories of Concurrency, Unification and Extension*, Lecture Notes in Computer Science, Vol. 458 (Springer, Berlin, 1990).
[13] K.G. Larsen and A. Skou, Bisimulation through probabilistic testing, in: *Proc. 16th ACM Symp. on Principles of Programming Languages*, Austin, TX (1989).
[14] LOTOS extended with probabilistic behaviours, *Formal Aspects Computing* **5**(3), (1993).
[15] R. Milner, *Communication and Concurrency* (Prentice-Hall, Englewood Cliffs, NJ, 1989).
[16] A. Pnueli, On the extremely fair treatment of probabilistic algorithms, in: *Proc. 15th Ann. Symp. on the Theory of Computing* (1983) 278–290.
[17] J.R. Rao, Reasoning about probabilistic parallel programs, *ACM Trans. Programming Languages Systems* **16**(3) (1994).
[18] G.M. Reed, A uniform mathematical theory for real-time distributed computing, Oxford University D. Phil Thesis, 1988.
[19] A.W. Roscoe, A mathematical theory of communicating processes, Oxford University D. Phil Thesis, 1982.
[20] K. Seidel, Probabilistic communicating processes, Oxford University Computing Laboratory, Technical Monograph PRG-102, 1992.
[21] A.N. Shiryayev, *Probability* (Springer, Berlin, 1984).
[22] A. Tarski, A lattice-theoretical fixpoint theorem and its applications, *Pacific J. Math.* **5** (1905) 285–309.