Programming Research Group

ARGUMENT DUPLICATION IN PROBABILISTIC CSP

Carroll Morgan, Annabelle McIver, Karen Seidel and JW Sanders

PRG-TR-11-95

Oxford University Computing Laboratory
Wolfson Building, Parks Road, Oxford OX1 3QD

Argument duplication in probabilistic CSP

Carroll Morgan, Annabelle McIver,
Karen Seidel and JW Sanders

October 25, 1995

Abstract

In [2] a space PCSP of probabilistic processes is constructed uniformly from the standard CSP failures-divergences model.

Laws of CSP are shown to be valid in PCSP also, provided they have no duplication on either side of $=$ or $\sqsubseteq$ separately: thus for example $A \sqcap B = B \sqcap A$ carries over to PCSP; but $A \sqcap A = A$ does not, because of the duplication on its left-hand side.

Here we propose two devices for retaining more of CSP's laws within PCSP: the first introduces a limited form of state; the second allows multiple occurrences of a term to be abstracted to a single syntactic location.

Either [2] or its published revision are prerequisites for understanding this report fully, although skimming Sec. 4 and App. A may give an idea of what has been achieved.

1 Introduction

The law $A \sqcap A = A$ fails in PCSP only when A is properly nondeterministic; and when that is so, the reason for its failure is the possibly independent resolution of probabilistic choice in the two copies. For example,

$$(B \;{}_{\frac{1}{2}}\!\oplus C) \sqcap (B \;{}_{\frac{1}{2}}\!\oplus C) \tag{1}$$

behaves like B only when *both* arguments of $\sqcap$ make that choice; since the two occurrences of ${}_{\frac{1}{2}}\oplus$ execute independently, that happens only 1/4 of the time (not 1/2).

Thus the problem is that ${}_{\frac{1}{2}}\oplus$ occurs only once in $B\ {}_{\frac{1}{2}}\oplus\ C$; but in (1) it occurs twice. That may be what we mean; but for example in the law

$$a \to P \parallel b \to Q \quad = \quad a \to (P \parallel b \to Q) \quad [] \quad b \to (a \to P \parallel Q)\ ,$$

the duplication of P (or Q) on the right does not indicate multiple copies of those processes; rather it represents alternative executions, and in some sense it is 'the same P' whichever way you go.

In this report we propose two methods for avoiding duplication where we really do not want it. The first is a special case, the duplication caused by the CSP convention that (2) is an abbreviation for (3), as explained below; in that case, a simple state mechanism suffices.

The second method concerns more general duplication.

Sections 4 and A present thoroughly worked-out examples of the two techniques, applied to a chain of probabilistic and, in the second case, nondeterministic buffers.

2 CSP with state

2.1 Variables and value-passing

Standard (non-probabilistic) CSP contains a *general external choice* construction

$$([]a\colon \alpha \cdot P(a))\ , \tag{2}$$

where the bound variable a represents an event drawn from a given non-empty set of events α: the process is the external choice between all the processes $P(a)$ in which occurrences of a have been replaced by (constants denoting) events in the set α. Thus if α were the set of events $\{a_1, \cdots, a_N\}$ we would regard (2) as an abbreviation for

$$P(a_1) [] \cdots [] P(a_N)\ . \tag{3}$$

One use of the construction (2) is to form a *generalised input*, by defining for *channel name c*

$$(c?a\colon \alpha \to P(a)) \quad := \quad ([]a\colon \alpha \cdot c.a \to P(a))\ . \tag{4}$$

Thus by (3) we could regard either side of (4) as an abbreviation for

$$c.a_1 \to P(a_1) \mathbin{[\!]} \cdots \mathbin{[\!]} c.a_N \to P(a_N)\ . \tag{5}$$

In PCSP however, the equivalence of $(c?a\colon \alpha \to P(a))$ and the expansion (5) is inconvenient. Recalling the example of [2, Section 14.1], we consider a probabilistic buffer over channels *left* (input) and *right* (output)

$$PB \quad := \quad (\mathit{left}?a\colon\{0,1\} \to (\mathit{right}!a \to PB \;{}_{0.5}{\oplus}\; \mathit{right}!\overline{a} \to PB))\ , \tag{6}$$

where $\overline{a}$ denotes the complement $1-a$ of a: the process PB appears to have probability 0.5 of transmitting each successive input without corruption. The equivalence above however would have us take the right-hand as an abbreviation for

$$\begin{array}{rlcl} & \mathit{left}.0 & \to & (\mathit{right}!0 \to PB \;{}_{0.5}{\oplus}\; \mathit{right}!1 \to PB) \\ [\!] & \mathit{left}.1 & \to & (\mathit{right}!1 \to PB \;{}_{0.5}{\oplus}\; \mathit{right}!0 \to PB)\ , \end{array} \tag{7}$$

whence distribution of the two occurrences of ${}_{0.5}{\oplus}$ gives us a four-branch probabilistic choice:

$$\begin{array}{ll} \mathit{left}.0 \to \mathit{right}!0 \to PB \mathbin{[\!]} \mathit{left}.1 \to \mathit{right}!1 \to PB & @\,0.25 \\ \mathit{left}.0 \to \mathit{right}!0 \to PB \mathbin{[\!]} \mathit{left}.1 \to \mathit{right}!0 \to PB & @\,0.25 \\ \mathit{left}.0 \to \mathit{right}!1 \to PB \mathbin{[\!]} \mathit{left}.1 \to \mathit{right}!1 \to PB & @\,0.25 \\ \mathit{left}.0 \to \mathit{right}!1 \to PB \mathbin{[\!]} \mathit{left}.1 \to \mathit{right}!0 \to PB & @\,0.25\ . \end{array} \tag{8}$$

(We use @ to indicate a probabilistic 'linear combination' of processes: it is easier to read in such cases than the equivalent ${}_p{\oplus}$ form.) The probability of correct transmission on the first step is then

$$(\mathit{left}?x \to \mathit{right}!x \to CHAOS) \quad \underset{\sim}{\sqsubset} \quad PB$$

which, as (8) shows, is clearly 0.25 rather than the 0.5 we expected.

The technical reason for the anomaly is the duplication of the probabilistic choice in (7), caused by the syntactic expansion of the generalised input. In the next section we discuss briefly how we might adjust our intuition so that we would, after all, expect (6) and (7) to differ; in the section following that we propose a modification to (standard) CSP that will indeed give (2) (and hence (6)) a more convenient meaning.

2.2 The dangers of duplication

Process (7) contains two distinct probabilistic choices that affect each input/output exchange. They are, as usual, resolved independently so that with probability (only) 0.25 do both sides of the external choice resolve to processes that put out a rather than $\overline{a}$ (taking a to be 0 or 1 as appropriate). In the remaining 0.75 of the cases at least one of the branches is prepared to put out the inverse of what was put in.

Thus an experimenter testing (7), or equivalently (8), with a succession of arbitrarily chosen inputs could be unlucky enough in (as many as) 0.75 of the cases to pick the input that leads to inverted output, and so in a demonic world someone expecting correct outputs with 0.5 probability would be disappointed. Similar reasoning applies to the probability of inverted output; so the best that can be said is that one expects correct output in somewhere between 0.25 and 0.75 of the cases (and for inverted output the figures are the same).

In (6) however there is apparently only one occurrence of the probabilistic choice ${}_{0.5}\oplus$, and our intuition would be strained indeed if we were not to expect correct (or inverted) output with probability 0.5. In the next section therefore we arrange that (6) and (8) are not equal — we remove (more generally) the equality of (2) and (3), at least at the probabilistic level.

2.3 CSP with state

Let us consider (4) and (5); the reason they are equal is that, without a state in which values can be passed, having separate branches is the only way of 'carrying the information' from the selection of initial event $c.a$ to the instance $P(a)$ that should follow it. Thus we introduce a state, proceeding as follows:

Definition 2.1 Let α be the finite and non-empty set of events from which our CSP processes are made, and let Var be a finite set of *variable names*. Then

$$\begin{array}{rcl} \text{State} & := & \text{Var} \to \alpha \\ \text{CSP}_s & := & \text{State} \to \text{CSP} \ . \end{array}$$

$\square$

Thus the state is a simple binding of variables to values (more like an environment, in denotational semantics), and the values are events.

The source (State) of the space CSP_s is finite (because both Var and α are), and so the *ipo* formed by taking the pointwise extension of the order on CSP remains algebraic: thus CSP_s is suitable for a replaying of the constructions of [2], originally applied to (the stateless) CSP.

Most operators in CSP_s can be carried over pointwise from the corresponding CSP operators. Thus, using CSP_s operators on the left and CSP operators on the right, we could define for arbitrary state s in State

$$(A \mathbin{[\!]} B)(s) \quad := \quad A(s) \mathbin{[\!]} B(s)\ .$$

We are however interested in the operators that do not carry over, and we now treat each of those in turn. The first is *substitution* of a value for a variable in a process, the analogue of syntactic value-passing in ordinary CSP:

Definition 2.2 For CSP_s process P, variable a, expression $E(s)$ a function of the state, and state s,

$$(P[a{:=}\,E])(s) \quad := \quad P(s[a{:=}\,E(s)])\ ,$$

where the substitution on the right is the usual 'overriding' of a function (here s) by an explicitly given pair (here $(a, E(s))$). $\square$

The *conditional* selects one of two processes depending on the value of a Boolean-valued expression (again a function of the state). Thus we have

Definition 2.3 For CSP_s processes P, Q, Boolean expression $B(s)$ a function of the state, and state s,

$$\begin{array}{rcll}(P \ \underline{\mathrm{if}}\ B \ \underline{\mathrm{else}}\ Q)(s) & := & P(s) & \text{if } B(s) = \text{true} \\ & & Q(s) & \text{if } B(s) = \text{false}\ .\end{array}$$

$\square$

Finally, general external choice is defined directly as a function of its CSP_s-process argument:

Definition 2.4 Let β be some non-empty set $\{b_1, \cdots, b_N\}$ of events, a subset of α, let b be a variable, and let P be a CSP_s process. Then

$$\begin{array}{rcll}(\mathbin{[\!]} b{:}\,\beta \cdot P) & := & & P[b{:=}\,b_1] \\ & & [\!] & \vdots \\ & & [\!] & P[b{:=}\,b_N]\ .\end{array}$$

$\square$

It might not be clear at first how Def. 2.4 differs from the original definition 'by syntactic equivalence'. In fact at the (standard) level of CSP and CSP_s there is no difference: it is only in the probabilistic spaces PCSP and PCSP_s that the two approaches are no longer the same.

The original syntactic method used that equivalence not only in the standard space, but in the probabilistic space as well, asserting the equality

$$
\begin{array}{rcll}
([\!] b{:}\, \beta \cdot A) & =? & & A[b{:}= b_1] \\
& & [\!] & \vdots \\
& & [\!] & A[b{:}= b_N]
\end{array}
\tag{9}
$$

even for probabilistic process A: the result is a duplication on the right of any probabilistic choices that A contains. In PCSP_s however, general external choice is applied to the probabilistic A by using the embedding of the *standard* function defined in Def. 2.4; thus we have instead of (9) the equality

$$
([\!] b{:}\, \beta \cdot A) \quad = \quad \overline{([\!] b{:}\, \beta \cdot \diamond)}(A)\ ,
$$

where the overbar denotes embedding, and the symbol $\diamond$ indicates the argument position of the expression taken as a unary function.

We now explore the effect of our definitions.

2.4 Consequences of state

The first — and intended — consequence of using PCSP_s rather than PCSP is that $([\!] b{:}\, \beta \cdot \diamond)$ is now a standard function, in $\mathrm{CSP}_s \to \mathrm{CSP}_s$, and so its embedding $\overline{([\!] b{:}\, \beta \cdot \diamond)}$ distributes ${}_p\!\oplus$. Thus in particular we have

$$
\begin{array}{ll}
& (\mathit{left}?a{:}\, \{0,1\} \to (\mathit{right}!a \to PB \;{}_{0.5}\!\oplus\; \mathit{right}!\overline{a} \to PB)) \\[1em]
= & \qquad\quad (\mathit{left}?a{:}\, \{0,1\} \to \mathit{right}!a \to PB) \\
& {}_{0.5}\!\oplus \quad (\mathit{left}?a{:}\, \{0,1\} \to \mathit{right}!\overline{a} \to PB)\ ,
\end{array}
$$

which is just what we wanted, but is inconsistent with the equality of (6) and (8) — thus the latter must now fail, which we might call the second consequence.

The third consequence concerns the conditional. In both (standard) CSP and CSP_s, we have the equality

$$P \;\underline{\text{if}}\; B \;\underline{\text{else}}\; P \quad = \quad P\ ,$$

for any (standard) process P and Boolean-valued B. The same is not necessarily true in PCSP_s however, since the left-hand side contains two occurrences of the process P: that is, we cannot conclude

$$A \;\underline{\text{if}}\; B \;\underline{\text{else}}\; A \quad =? \quad A \tag{10}$$

for arbitrary probabilistic A. And indeed (10) is false in general, as this example shows: we have the embedded equality

$$\begin{array}{ll} & \mathit{left}.0 \to A[a\!:=0] \;[\!]\; \mathit{left}.1 \to B[a\!:=1] \\ = & (\mathit{left}?a\!:\{0,1\} \to A \;\underline{\text{if}}\; (a=0) \;\underline{\text{else}}\; B) \end{array}$$

because it is a standard law, and there is no duplication on either side. Taking B to be A (after embedding) and using idempotence of ($\underline{\text{if}}$ B $\underline{\text{else}}$) illegitimately gives

$$\begin{array}{lll} & \mathit{left}.0 \to A[a\!:=0] \;[\!]\; \mathit{left}.1 \to A[a\!:=1] & \\ = & (\mathit{left}?a\!:\{0,1\} \to A \;\underline{\text{if}}\; (a=0) \;\underline{\text{else}}\; A) & \text{law above} \\ =? & (\mathit{left}?a\!:\{0,1\} \to A) & \text{not allowed} \end{array}$$

which, as discussed earlier, is an equality we no longer have.

We do however have the equality (10) when Boolean condition is constant (that is, a constant function of the state). In that case, taking $B(s) := \text{true}$ without loss of generality, we have the standard equality

$$P \;\underline{\text{if}}\; \text{true} \;\underline{\text{else}}\; Q \quad = \quad P\ ,$$

which can indeed be embedded. The result then follows by taking both arguments P, Q to be the probabilistic process A.

Note that if one is defining an indexed family of processes, the indices are not state variables: thus a Boolean expression depending only on indices is a constant function for our purposes here.

2.5 Discussion

The modification that takes us from CSP to CSP$_s$ has no effect at the standard level: the two structures may validly be viewed as alternative semantics for the same phenomena, under the restrictions we impose (for example, that the alphabet is finite).

The probabilistic spaces, PCSP and PCSP$_s$ respectively, are however sensitive to the structure of the standard space from which they are constructed. In PCSP, we do not have an embedded general external choice operator, because CSP does not have one to embed; and in CSP it cannot be defined as a semantic operator because we have no semantic notion of substitution.

In CSP$_s$, the presence of a state allows substitution to be defined, and hence a general external choice operator as well. That can be embedded, and takes us to PCSP$_s$, in which general external choice distributes probabilistic choice ${}_p\oplus$, as all embedded operators do: that was exactly what we wanted.

In fact it is convenient to take the same approach to general *internal* choice:

Definition 2.5 Let β be some non-empty set $\{b_1, \cdots, b_N\}$ of events, a subset of α, let b be a variable, and let P be a CSP$_s$ process. Then

$$(\sqcap b{:}\, \beta \cdot P) \quad := \quad P[b{:}= b_1] \sqcap \cdots \sqcap P[b{:}= b_N] \ .$$

$\square$

That definition allows us, at the PCSP$_s$ level, to use laws like

$$([\!]b{:}\, \beta \cdot b \to A)\ \underline{\text{hide}}\ \beta \quad = \quad (\sqcap b{:}\, \beta \cdot A\ \underline{\text{hide}}\ \beta)\ ,$$

which we could not have done if the right-hand side were instead given as a syntactic equivalence. An example of the use of Def. 2.5 occurs at the end of Sec. 4.

3 <u>where</u>-clauses

3.1 More general duplication

The difficulties illustrated by the *PB* example of Sec. 2.1 can occur in a more general context for which the state-based solution of Sec. 2 is insufficient. Consider for example the standard CSP equality

$$a \to P \parallel b \to Q \quad = \quad a \to (P \parallel b \to Q) \quad [\!] \quad b \to (a \to P \parallel Q)\ , \tag{11}$$

in which we assume that neither of the events a or b belongs to the synchronising alphabet.

Equation (11) does not embed into PCSP because of the duplication of P, and of Q, on the right-hand side. Yet without it we have essentially no way of distributing prefix ($\to$) through parallel composition ($\parallel$).

Our approach is to provide a means of indicating that (for example) the two P's on the right-hand side of (11) refer to 'the same' process, and not separately to each of two (identical) copies. Operationally, just as for our earlier work with state, that will be understood as a 'single copy' of P that may however be invoked from several places.

3.2 Definition of <u>where</u>

Continuing with the example (11), but using now A, B to indicate that we are dealing with probabilistic processes, we aim for an equality

$$\begin{array}{rl} & a \to A \parallel b \to B \\ = & a \to (P \parallel b \to Q) \\ [\!] & b \to (a \to P \parallel Q) \quad \underline{\text{where}}\ P, Q := A, B\ . \end{array} \tag{12}$$

The P, Q on the right are now bound variables indicating the places at which the 'real' processes A, B are invoked: there are two such places for each of A, B, but we intend that there be 'only one copy' of A and of B.

Naturally such a <u>where</u> clause cannot in general be removed by simple syntactic substitution. To give a proper definition we must recall some of the details of the construction of PCSP from CSP. We begin by considering just a single bound variable (rather than two, as above).

A PCSP expression with free occurrences of a variable A can be considered to be a function, of A, from PCSP to itself. (Take other free variables, if present, to be 'arbitrary but fixed' for this discussion.) Applying that function to a process in the usual way gives the value of the original expression when A has the value of the argument: if f

were the function corresponding to the 'A-abstracted' expression, then by application we mean the usual $f(A)$.

For <u>where</u> clauses we define a different notion of application (or, equivalently, of abstraction). Recall that an element of PCSP is a continuous evaluation over CSP, thus a function from Scott-open subsets of $(\mathrm{CSP}, \sqsubseteq)$ to the closed interval $[0, 1]$ of real numbers. We now give the definition of '<u>where</u>-application', with an explanation of the notation immediately following.

Definition 3.1 For (as usual, continuous) function $f \colon \mathrm{PCSP} \to \mathrm{PCSP}$, probabilistic process $A \colon \mathrm{PCSP}$ (a continuous evaluation), and arbitrary Scott-open subset $\mathcal{P}$ of $(\mathrm{CSP}, \sqsubseteq)$, we define

$$(f * A)(\mathcal{P}) \quad := \quad \int_A f(\overline{P})(\mathcal{P}) \; \mathrm{d}P \; .$$

$\square$

(If the left-hand side above were written with <u>where</u>, it would appear

$$(f(P) \; \underline{\text{where}} \; P := A)(\mathcal{P}) \; ,$$

a <u>where</u>-clause applied to the Scott-open set $\mathcal{P}$.)

The body $f(\overline{P})(\mathcal{P})\;\mathrm{d}P$ of the integral is a function taking a standard CSP process (P) to probability in $[0, 1]$ — it is the probability that f, applied to the embedding $\overline{P}$ of P, gives a result that lies in $\mathcal{P}$.

The subscript A of the integral indicates that it is taken over the continuous evaluation A, and is written there by analogy with limits of definite integration (as in $\int_a^b$) in analysis.[1]

[1] Usually integrals over measures are written

$$\int_{x \in X} f(x) \; \mathrm{d}\mu \; ,$$

where μ is the measure concerned, and the subscripted $x \in X$ indicates both the carrier for the measure and the name of the bound variable. We prefer to write

$$\int_\mu f(x) \; \mathrm{d}x \quad \text{or equivalently in this simple case} \quad \int_\mu f \; ,$$

preserving the convention that in $\mathrm{d}x$, say, the variable x is a binding occurrence (unlike μ in the conventional notation $\mathrm{d}\mu$), and that the characteristics of the space over which the integral is taken are indicated next to the integral symbol itself (as the interval $[a, b]$, for example, in Riemann integration).

The integral of Def. 3.1 is well-defined [1, Sec. 3.9] provided the function $(\lambda P \cdot f(\overline{P})(\mathcal{P}))$ is upper-continuous, that is that the inverse image through it of any upper-open interval $(r, \infty]$ is a Scott-open set in $(\mathrm{CSP}, \sqsubseteq)$; that function is indeed upper-continuous provided f itself is continuous, which we have assumed.

Roughly speaking, the effect of $(f*)$ applied to A is to split A into its standard components, apply f to each (in the normal way), and then to re-combine the results in their original probabilistic proportions. We see in the following sections how that gives us exactly the properties we need.

3.3 Well-definedness of <u>where</u>

Def. 3.1 shows that $f*A$ is a function from Scott-open subsets of CSP to the interval $[0, 1]$; to show further that it is a continuous evaluation, we must check that it satisfies the conditions given for them [2, Def. 3.2],[1, p. 50]. We omit proofs for the simpler properties; and in each case f is a continuous function in $\mathrm{PCSP} \to \mathrm{PCSP}$, A is an element of PCSP, and $\mathcal{P}, \mathcal{P}'$ are Scott-open subsets of CSP.

Lemma 3.2 *(Co-) Strictness*

$$(f * A)(\emptyset) = 0 \quad \text{and} \quad (f * A)(\mathrm{CSP}) = 1 \ .$$

□

Lemma 3.3 *Monotonicity*

$$\mathcal{P} \subseteq \mathcal{P}' \quad \text{implies} \quad (f * A)(\mathcal{P}) \leq (f * A)(\mathcal{P}') \ .$$

□

Lemma 3.4 *Modularity*

$$\begin{array}{rl} & (f * A)(\mathcal{P} \cup \mathcal{P}') + (f * A)(\mathcal{P} \cap \mathcal{P}') \\ = & (f * A)(\mathcal{P}) + (f * A)(\mathcal{P}') \ . \end{array}$$

□

Lemma 3.5 *Continuity* Let $\boldsymbol{\mathcal{P}}$ be a $\subseteq$-directed set of Scott-open subsets of $(CSP, \sqsubseteq)$. Then

$$(f * A)(\sqcup \boldsymbol{\mathcal{P}}) \quad = \quad (\sqcup \mathcal{P} : \boldsymbol{\mathcal{P}} \cdot (f * A)(\mathcal{P}))\ .$$

Proof:

$$\begin{array}{lll}
 & (f * A)(\bigsqcup \boldsymbol{\mathcal{P}}) & \\
= & \int_A f(\overline{P})(\bigsqcup \boldsymbol{\mathcal{P}})\ \mathrm{d}P & \text{Def. 3.1} \\
= & \int_A (\sqcup \mathcal{P} : \boldsymbol{\mathcal{P}} \cdot f(\overline{P})(\mathcal{P}))\ \mathrm{d}P & \text{continuity of } f(\overline{P}) \\
= & (\sqcup \mathcal{P} : \boldsymbol{\mathcal{P}} \cdot \int_A f(\overline{P})(\mathcal{P})\ \mathrm{d}P) & \text{see below} \\
= & (\sqcup \mathcal{P} : \boldsymbol{\mathcal{P}} \cdot (f * A)(\mathcal{P}))\ . & \text{Def. 3.1}
\end{array}$$

The distribution of $\sqcup$ through $\int_A$ is shown to be valid in [1, Thm. 3.13 directed monotone convergence]. □

The effect of this technical section has just been to show that $f * A$ is indeed an element of PCSP, given appropriate f and A. We now turn to its more interesting properties.

3.4 Continuity of <u>where</u>

If we are to be able to (continue to) write recursive expressions for processes, we need to know that <u>where</u> is continuous in both its arguments (and hence, if necessary, in the arguments jointly). We look first at the right argument.

Lemma 3.6 The function $F(P)$ <u>where</u> $P := A$ is continuous, considered as a function of A: PCSP.

Proof: The continuity is given directly by [1, Eqn. 4.4]. □

For the left argument we have the following.

Lemma 3.7 The function $F(P)$ <u>where</u> $P := A$ is continuous, considered as a function of F: PCSP $\to$ PCSP.

Proof: Using Def. 3.1, we show (more generally) that

$$\int_A \sqcup \mathcal{F} = (\sqcup F : \mathcal{F} \cdot \int_A F)\ ,$$

where $\mathcal{F}$ is a directed set of elements each of type CSP $\to [0, 1]$. As in Lem. 3.5, we appeal directly to [1, Thm. 3.31 directed monotone convergence]. □

3.5 where introduction and elimination

In this section we show how where clauses are introduced and eliminated during calculation. The conditions are simpler than they appear, and may be summarised

> where-application is equivalent to ordinary application if either:
>
> 1. the bound variable occurs in at most one position in the where body (so that there is no replication); or
> 2. the argument is standard (so that it contains no probabilistic choice to be affected by replication).

To prove the above, we need to define linearity for process-to-process functions.

Definition 3.8 Function $f\colon \mathrm{PCSP} \to \mathrm{PCSP}$ is said to be *linear* if it distributes probabilistic choice: for all processes A, B in PCSP and probabilities p,

$$f(A\,{}_p\!\oplus B) \quad = \quad f(A)\,{}_p\!\oplus f(B)\ .$$

$\square$

Recall that all embedded functions (functions $\overline{f}$ for f in $\mathrm{CSP} \to \mathrm{CSP}$) are linear by construction [2, Thm. 8.3], so that the lemma to follow applies as a special case to expressions built from (embedded) standard operators only, provided the argument occurs in only one position syntactically. Thus, for example, $A \sqcap B$ is linear in A (and in B), but $A \sqcap A$ is not (A appears twice), since

$$
\begin{array}{lll}
 & (A\,{}_p\!\oplus A') \sqcap (A\,{}_p\!\oplus A') & \\
 & & \\
= & \begin{array}{ll} A \sqcap A & @\, p^2 \\ A \sqcap A' & @\, 2p\overline{p} \\ A' \sqcap A' & @\, \overline{p}^2 \end{array} & \text{distribute both } {}_p\!\oplus\text{'s} \\
 & & \\
\neq & (A \sqcap A)\,{}_p\!\oplus (A' \sqcap A')\ . & \text{in general, unless } p \text{ is 0 or 1}
\end{array}
$$

We now have our first lemma for where-introduction or -elimination.

Lemma 3.9 If $F(A)$ is a PCSP-expression which is linear in A, then

$$F(A) \quad = \quad F(P) \; \underline{\text{where}} \; P := A \; .$$

Proof:

Letting f denote $(\lambda A \cdot F(A))$, we must show that linearity of f implies equality of $f * A$ and $f(A)$. Now in [1, p. 77] it is stated (transliterated) that if $(\text{CSP}, \sqsubseteq)$ is continuous (as an *ipo* — in fact it is algebraic), then any linear function f is "super-linear", and satisfies

$$f(A) \quad = \quad (\lambda \mathcal{P} \cdot \int_A f(\overline{P})(\mathcal{P}) \; \mathrm{d}P) \; ,$$

which is precisely what we require. □

Note that linearity is more general than "occurs at most once", since for example $F(A)_p \oplus G(A)$ is linear in A if both F and G are:[2] informally one might say that A may occur at most once in each probabilistic alternative.

The other lemma applies when the argument is standard.

Lemma 3.10 If P is an embedded standard process, then

$$F(P) \quad = \quad F(Q) \; \underline{\text{where}} \; Q := P \; ,$$

for all PCSP expressions $F(Q)$ (whether linear in Q or not).

Proof: It is an elementary property of integration over evaluations that

$$\int_{\overline{P}} f \quad = \quad f(P) \; ;$$

instantiating f to $(\lambda Q \cdot F(\overline{Q})(\mathcal{P}))$ gives

$$\int_{\overline{P}} F(\overline{Q})(\mathcal{P}) \; \mathrm{d}Q \quad = \quad F(\overline{P})(\mathcal{P})$$

as required. □

Lem. 3.10 assures us, among other things, that for entirely standard expressions <u>where</u> application is the same as ordinary functional application.

The final lemma for this section concerns distribution of one <u>where</u> clause over another.

[2]...because $(B \;_q\oplus\; C) \;_p\oplus\; (D \;_q\oplus\; E) = (B \;_p\oplus\; D) \;_q\oplus\; (C \;_p\oplus\; E)$ for any B, C, D, E.

Lemma 3.11 Provided PCSP-expression A contains no variable Q, and B contains no P, we have

$$
\begin{aligned}
& (F(P,Q)\ \underline{\text{where}}\ P := A)\ \underline{\text{where}}\ Q := B \\
= \quad & (F(P,Q)\ \underline{\text{where}}\ Q := B)\ \underline{\text{where}}\ P := A \\
= \quad & F(P,Q)\ \underline{\text{where}}\ P,Q := A,B \ .
\end{aligned}
$$

Proof: Working through the first of the three expressions, applied to a given Scott-open set $\mathcal{P}$, gives

$$\int_B (\int_A F(\overline{P},\overline{Q})(\mathcal{P})\ \mathrm{d}P)\ \mathrm{d}Q\ .$$

Reversing the order of integration (second expression), or integrating directly over the product space $\overline{\mathrm{CSP}^2}$ (third expression), is then justified by Fubini's theorem for evaluations [1, Thm. 3.17]; the result then follows. □

3.6 Use of standard laws

Our principal use for <u>where</u> is to expand the collection of algebraic laws available to us at the probabilistic level, mainly by gaining access to those containing duplication. The following lemma shows how that is achieved.

Lemma 3.12 Let $F(P), G(P)$ be two PCSP expressions each possibly containing a variable P (possibly duplicated), and let A be an arbitrary PCSP process. Then if $F(P)$ and $G(P)$ are equal for all embedded standard values of P, we have also

$$F(P)\ \underline{\text{where}}\ P := A \quad = \quad G(P)\ \underline{\text{where}}\ P := A\ .$$

Proof: Directly from Def. 3.1 we have that the left-hand side, when applied to an arbitrary Scott-open $\mathcal{P}$, is

$$\int_A F(\overline{P})(\mathcal{P})\ \mathrm{d}P\ ,$$

in which F is applied only to embedded standard arguments. The result follows immediately. □

Lem. 3.12 tells us that in the body of a $\underline{\text{where}}$ $P := \cdots$ we can regard P as standard, and thus ignore any duplication of it.

We now look at a few examples. Recall first that in PCSP generally we do not have idempotence of non-deterministic choice,

$$A \quad =? \quad A \sqcap A \ ,$$

for reasons discussed at length in [2]. But we do indeed have

$$\begin{array}{lll} & A & \\ = & P \ \underline{\text{where}}\ P := A & \text{Lem. 3.9} \\ = & P \sqcap P \ \underline{\text{where}}\ P := A \ . & \text{Lem. 3.12} \end{array}$$

The conclusion, that A equals $P \sqcap P$ $\underline{\text{where}}$ $P := A$, suggests the equality holds provided 'there is only one A': indeed in that case a non-deterministic choice between a thing and itself makes no difference (but a choice between two separate copies of a thing could well).

For another example we return to (12): we can now see it is justified because the left-hand side can be written (Lem. 3.9, Lem. 3.11)

$$a \to P \parallel b \to Q \quad \underline{\text{where}}\ P, Q := A, B \ ,$$

and the standard equality (11) is then applied within the $\underline{\text{where}}$ body.

3.7 Distribution of probability

Another useful property of $\underline{\text{where}}$ is that it distributes probabilistic choice ${}_p\oplus$ in both arguments. We have

Lemma 3.13 For expressions $F(P), G(P)$ and PCSP process A,

$$\begin{array}{lllll} F(P)\,{}_p\oplus\, G(P) & \underline{\text{where}}\ P := A & = & & F(P) \quad \underline{\text{where}}\ P := A \\ & & & {}_p\oplus & G(P) \quad \underline{\text{where}}\ P := A \ . \end{array}$$

Proof: Applied to Scott-open $\mathcal{P}$, the left-hand side is

$$\begin{array}{lll} & \int_A (F(\overline{P})\,{}_p\oplus\, G(\overline{P}))(\mathcal{P}) \ \mathrm{d}P & \\ = & \int_A p \times F(\overline{P})(\mathcal{P}) + \overline{p} \times G(\overline{P})(\mathcal{P}) \ \mathrm{d}P \ . & \text{definition of } {}_p\oplus \end{array}$$

Distribution of addition through $\int_A$ is shown at [1, Lem. 3.4], and distribution of multiplication is obvious. □

For the other distributive property we have

Lemma 3.14 For expression $F(P)$ and PCSP processes A, B,

$$\begin{array}{rcl} F(P)\ \underline{\text{where}}\ P := A\ {}_p\!\oplus\ B & = & F(P)\ \ \underline{\text{where}}\ P := A \\ & & {}_p\!\oplus\ \ F(P)\ \ \underline{\text{where}}\ P := B\ . \end{array}$$

Proof: The result is immediate from the general equality

$$\int_{A_p\oplus B} f \quad = \quad \int_A f\ {}_p\!\oplus \int_B f\ ,$$

where on the right we give ${}_p\!\oplus$ between reals the obvious meaning. □

3.8 Distribution of standard operators

Our final two lemmas concern the interaction of <u>where</u> with standard operators. Given for example

$$F(P)\ \underline{\text{where}}\ P := a \to A\ ,$$

we might want to use within $F(P)$ the fact that P is a process of the form $a \to \cdots$. That is achieved by distributing the standard function $(a \to)$ into the left-hand side, giving

$$F(a \to Q)\ \underline{\text{where}}\ Q := A\ .$$

The general case is justified by the following lemma.

Lemma 3.15 If $F(P)$ is a process expression, g a standard operator in $CSP \to CSP$ and A a PCSP process, then

$$F(P)\ \underline{\text{where}}\ P := \overline{g}(A) \quad = \quad F(\overline{g}(Q))\ \underline{\text{where}}\ Q := A\ .$$

Proof: Using Def. 3.1, our equality would follow from

$$\int_{\overline{g}(A)} F(\overline{P})(\mathcal{P})\ \mathrm{d}P \quad = \quad \int_A F(\overline{g}(\overline{Q}))(\mathcal{P})\ \mathrm{d}Q\ ,$$

or more generally, by taking $f := (\lambda P \cdot F(\overline{P})(\mathcal{P}))$ and noting that $\overline{g}(\overline{Q}) = \overline{g(Q)}$, from

$$\int_{\overline{g}(A)} f \quad = \quad \int_A f \circ g\ .$$

That equation is given at (4.6) in [1, p. 74]. □

The other distribution property concerns application of a linear function to the where clause as a whole.

Lemma 3.16 If $F(P)$ is a process expression, g a linear (and continuous) function in PCSP $\rightarrow$ PCSP and A a PCSP process, then

$$g(F(P) \ \underline{\text{where}}\ P := A) \quad = \quad g(F(P)) \ \underline{\text{where}}\ P := A\ .$$

Proof: Consider first the case where A is of the simple form $R\,{}_p\!\oplus S$, for standard embedded R and S. Then we have

$$\begin{array}{lll}
 & g(F(P) \ \underline{\text{where}}\ P := R\,{}_p\!\oplus S) & \\
= & \quad g(F(P) \ \underline{\text{where}}\ P := R) & \text{Lem. 3.14; } g \text{ linear} \\
 & {}_p\!\oplus \quad g(F(P) \ \underline{\text{where}}\ P := S) & \\
= & g(F(R))\,{}_p\!\oplus g(F(S)) & R, S \text{ standard} \\
= & \quad g(F(P)) \ \underline{\text{where}}\ P := R & R, S \text{ standard} \\
 & {}_p\!\oplus \quad g(F(P)) \ \underline{\text{where}}\ P := S & \\
= & g(F(P)) \ \underline{\text{where}}\ P := R\,{}_p\!\oplus S\ . & \text{Lem. 3.14}
\end{array}$$

The reasoning above clearly extends to the case in which A is a larger but still finite combination of standard processes.

By [1, Thm. 5.2], every A is the limit of finite probabilistic combinations as above: let $A = \sqcup\mathcal{A}$, where $\mathcal{A}$ is a directed set of such combinations. Then

$$\begin{array}{lll}
 & g(F(P) \ \underline{\text{where}}\ P := A) & \\
= & g(F(P) \ \underline{\text{where}}\ P := \sqcup\mathcal{A}) & \\
= & (\sqcup A' \colon \mathcal{A} \cdot g(F(P) \ \underline{\text{where}}\ P := A')) & \text{Lem. 3.6; } g \text{ continuous} \\
= & (\sqcup A' \colon \mathcal{A} \cdot g(F(P)) \ \underline{\text{where}}\ P := A') & A' \text{ is of form above} \\
= & g(F(P)) \ \underline{\text{where}}\ P := \sqcup\mathcal{A} & \text{Lem. 3.6} \\
= & g(F(P)) \ \underline{\text{where}}\ P := A\ . &
\end{array}$$

□

With the lemmas of the last three sections, we are now ready to treat a more substantial example.

4 Example: buffer chaining

At (6) we saw a probabilistic one-place buffer that corrupted its contents with probability 0.5. More generally, we define

$$PB_p \quad := \quad (\mathit{left}?a\colon \{0,1\} \rightarrow (\mathit{right}!a \rightarrow PB_p \,{}_p\!\oplus\, \mathit{right}!\overline{a} \rightarrow PB_p)) \ ,$$

and investigate the effect of putting two such buffers in sequence

$$PB_p \gg PB_q \ .$$

We begin by expanding both PB_p and PB_q, to give an expression in which there are two occurrences of probabilistic choice: on the left (of $\gg$) is ${}_p\!\oplus$, and on the right is ${}_q\!\oplus$. Assuming that we use PCSP_s, those choices distribute through the inputs ($\mathit{left}?x \rightarrow \diamond$); they distribute through $\gg$ in any case, giving four alternatives, with probabilities pq, $p\overline{q}$, $\overline{p}q$ and $\overline{p}\,\overline{q}$. Taking $\overline{p}q$ as representative, we have (omitting $:\{0,1\}$ to suppress clutter)

$$(\mathit{left}?a \rightarrow \mathit{right}!\overline{a} \rightarrow PB_p) \gg (\mathit{left}?a \rightarrow \mathit{right}!a \rightarrow PB_q)$$

$$\begin{array}{ll}
= & \hfill \text{standard law for chaining} \\
 & (\mathit{left}?a \rightarrow (PB_p \gg \mathit{right}!\overline{a} \rightarrow PB_q[a\colon= \overline{a}])) \\
= & \hfill PB_q \text{ unaffected by substitution} \\
 & (\mathit{left}?a \rightarrow (PB_p \gg \mathit{right}!\overline{a} \rightarrow PB_q)) \ .
\end{array}$$

The other three alternatives work out similarly, and we find that the pq and $\overline{p}\,\overline{q}$ alternatives are equal, as are the $p\overline{q}$ and the $\overline{p}q$. Bringing in those alternatives, and collapsing the identical pairs, gives

$$\begin{array}{ll}
 & (\mathit{left}?a \rightarrow (PB_p \gg \mathit{right}!a \rightarrow PB_q)) \\
{}_r\!\oplus & (\mathit{left}?a \rightarrow (PB_p \gg \mathit{right}!\overline{a} \rightarrow PB_q)) \ ,
\end{array}$$

where we define $r := pq + \overline{p}\,\overline{q}$. Taking it one step further we get $(\mathit{left}?a \rightarrow B)$, with

$$\begin{array}{rl}
B := & C[b\colon= a] \,{}_r\!\oplus\, C[b\colon= \overline{a}] \\
C := & PB_p \gg \mathit{right}!b \rightarrow PB_q \ .
\end{array}$$

Now we must continue with C; we substitute the text of PB_p and PB_q and expand, again selecting a typical summand:

$$\begin{array}{ll}
 & (\mathit{left}?a \rightarrow \mathit{right}!\overline{a} \rightarrow PB_p) \\
\gg & \mathit{right}!b \rightarrow (\mathit{left}?a \rightarrow \mathit{right}!a \rightarrow PB_q) \ .
\end{array}$$

$$
\begin{array}{lcl}
A & := & (\mathit{left}?a \rightarrow B) \\
B & := & C[b := a] \,{}_r\!\oplus\, C[b := \overline{a}] \\
C & := & \left.\begin{array}{l} \quad (\mathit{left}?a \rightarrow \mathit{right}!b \rightarrow P) \\ [\!] \quad \mathit{right}!b \rightarrow (\mathit{left}?a \rightarrow P) \end{array}\right\} \ \underline{\text{where}}\ P := B
\end{array}
$$

Figure 1: Process A is a probabilistic two-place buffer.

But this time we must fix PB_p and PB_q with a <u>where</u> clause, because the standard law will result in their duplication. We continue

$$
\begin{array}{lll}
= & \quad (\mathit{left}?a \rightarrow \mathit{right}!b \rightarrow (P \gg \mathit{right}!\overline{a} \rightarrow Q)) & \text{standard law for } \gg \\
 & [\!] \quad \mathit{right}!b \rightarrow (\mathit{left}?a \rightarrow (P \gg \mathit{right}!\overline{a} \rightarrow Q)) & \\
 & \underline{\text{where}}\ P, Q := PB_p, PB_q & \\
 & & \\
= & \quad (\mathit{left}?a \rightarrow \mathit{right}!b \rightarrow R) & \text{Lem. 3.12; Lem. 3.10} \\
 & [\!] \quad \mathit{right}!b \rightarrow (\mathit{left}?a \rightarrow R) & \\
 & \underline{\text{where}}\ R := (P \gg \mathit{right}!\overline{a} \rightarrow Q) & \\
 & \underline{\text{where}}\ P, Q := PB_p, PB_q & \\
 & & \\
= & \quad (\mathit{left}?a \rightarrow \mathit{right}!b \rightarrow R) & \text{Lem. 3.11; Lem. 3.9 twice} \\
 & [\!] \quad \mathit{right}!b \rightarrow (\mathit{left}?a \rightarrow R) & \\
 & \underline{\text{where}}\ R := (PB_p \gg \mathit{right}!\overline{a} \rightarrow PB_q) & \\
 & & \\
= & \quad (\mathit{left}?a \rightarrow \mathit{right}!b \rightarrow R) & \text{definition of } C \\
 & [\!] \quad \mathit{right}!b \rightarrow (\mathit{left}?a \rightarrow R) & \\
 & \underline{\text{where}}\ R := C[b := \overline{a}] &
\end{array}
$$

As before, we bring in the other three alternatives and group in pairs; then Lem. 3.14 allows us to move the ${}_r\!\oplus$ within the <u>where</u> clause, giving finally

$$
\begin{array}{l}
\quad (\mathit{left}?a \rightarrow \mathit{right}!b \rightarrow R) \\
[\!] \quad \mathit{right}!b \rightarrow (\mathit{left}?a \rightarrow R) \\
\underline{\text{where}}\ R := B\ .
\end{array}
$$

with the result overall that $PB_p \gg PB_q$ is equivalent to the process A shown in Fig. 1. Note that it is a two-place buffer.

In fact the equivalence of $PB_p \gg PB_q$ with A of Fig. 1 shows that

$$PB_p \gg PB_q \quad = \quad PB_{p'} \gg PB_{q'}$$

whenever $p \otimes q = p' \otimes q'$, where we define

$$p \otimes q \quad := \quad pq + \overline{p}\,\overline{q} \ .$$

The operator $\otimes$ is commutative and associative, and has identity 1 and zero 1/2.[3] For longer chains we can reach similar conclusions, for

$$\begin{array}{lll}
 & PB_p \gg (PB_q \gg PB_r) & \\
= & PB_p \gg (PB_{q\otimes r} \gg PB_1) & \\
= & (PB_p \gg PB_{q\otimes r}) \gg PB_1 & \gg \text{ associative} \\
= & PB_{p\otimes q\otimes r} \gg PB_1 \gg PB_1 & \\
= & PB^3_{p\otimes q\otimes r} \ , &
\end{array}$$

where we define the n-place probabilistic buffer

$$PB^n_p \quad := \quad PB_p \gg \overbrace{PB_1 \gg \cdots \gg PB_1}^{n-1} \ .$$

That gives the more general identity

$$PB^m_p \gg PB^n_q \quad = \quad PB^{m+n}_{p\otimes q} \ .$$

Exercise: Define an nondeterministic buffer to be one that *might* corrupt a message, with probability $1 - p$:

$$NB_p \quad := \quad (\mathit{left}?a{:}\{0,1\} \rightarrow \quad \begin{array}{l} \mathit{right}!a \rightarrow NB_p \\ {}_p\!\oplus \quad \left(\begin{array}{ll} & \mathit{right}!0 \rightarrow P \\ \sqcap & \mathit{right}!1 \rightarrow P \end{array} \right\} \ \underline{\text{where}} \ P := NB_p)) \end{array}$$

Why is the <u>where</u> necessary? By investigating $NB_p \gg NB_q$ discover equalities similar to the above. *Hint*: First eliminate the <u>where</u>: recall Sec. 2.5.

5 Conclusion

[3] Consider $(1 + x)/2 \otimes (1 + y)/2 = (1 + xy)/2$.

1. where-clauses may be introduced or eliminated as for simple substitution, provided either:
   (a) the bound variable occurs only once in each probabilistic branch of the where body (Lem. 3.9); or
   (b) the expression substituted by the where is (or may be considered to be) standard (Lem. 3.10).
2. where substitutions may be grouped provided the usual rules for (avoiding) variable capture are observed (Lem. 3.11).
3. Within a where body the bound variable may be considered to denote a standard (rather than, more generally, a probabilistic) process (Lem. 3.12).
4. Probabilistic choice distributes through where in both its arguments (Lem. 3.13, Lem. 3.14).
5. Standard functions distribute through the right argument of where: the function is applied to all instances of the bound variable in the where body (Lem. 3.15).
6. Linear functions distribute through the left argument of where (Lem. 3.16).

Figure 2: Informal summary of where algebra

The introduction of state, forming PCSP$_s$, was without much cost: the only loss was the equality of (6) and (7), which one could argue we didn't want anyway.

The where-clauses, while not terribly complicated, could not be said to be cost-free however: in Fig. 2 we summarise the introduction and elimination rules for them. With some care, as the examples show, where-clauses can be successfully deployed to access the standard laws required. Experience shows, however, that the larger the example the more of a nuisance they may become.

A Answer to exercise

Here we work through the exercise posed at the end of Sec. 4.

First, the where can be eliminated by noting that within its body the name P may be considered to denote a standard process (Lem. 3.12), thus we can use Def. 2.5 to rewrite it as

$$(\sqcap a\colon \{0,1\} \cdot \mathit{right}!a \to P)\ .$$

Then noting that P occurs only once, we can with Lem. 3.9 eliminate the where altogether. The result is an equivalent definition (omitting $:\{0,1\}$ to avoid clutter)

$$\begin{array}{lll} NB_p & := & (\mathit{left}?a \to \quad \mathit{right}!a \to NB_p \\ & & \qquad\qquad {}_p\!\oplus \quad (\sqcap a \cdot \mathit{right}!a \to NB_p))\ , \end{array}$$

and it is the one we shall use. Thus we consider two such buffers in sequence,

$$NB_p \gg NB_q\ .$$

As before, we expand both arguments; but this time we must have a closer look at each component resulting from the expansion. The first (probability pq) is

$$\begin{array}{lll} & (\mathit{left}?a \to \mathit{right}!a \to NB_p) \gg (\mathit{left}?a \to \mathit{right}!a \to NB_q) & \\ = & (\mathit{left}?a \to (NB_p \gg \mathit{right}!a \to NB_q))\ . & \text{by standard reasoning} \end{array}$$

For the second (probability $p\overline{q}$) we have

$$\begin{array}{lll} & (\mathit{left}?a \to \mathit{right}!a \to NB_p) \gg (\mathit{left}?a \to (\sqcap a \cdot \mathit{right}!a \to NB_q)) & \\ = & (\mathit{left}?a \to (\sqcap a \cdot NB_p \gg \mathit{right}!a \to NB_q))\ . & \text{by standard reasoning} \end{array}$$

The third and fourth components (probabilities $\overline{p}q$ and $\overline{p}\,\overline{q}$) give the same result as the second, producing overall for this first stage the process $(\mathit{left}?a \to B)$, where we define

$$\begin{array}{rl} B := & C[b\!:=a] \;_{pq}\!\oplus (\sqcap b \cdot C) \\ C := & NB_p \gg \mathit{right}!b \to NB_q \;. \end{array}$$

Now continuing with C, we have for the first summand (probability pq)

$$\begin{array}{rl} & (\mathit{left}?a \to \mathit{right}!a \to NB_p) \\ \gg & \mathit{right}!b \to (\mathit{left}?a \to \mathit{right}!a \to NB_q)\;; \end{array}$$

using a <u>where</u>-clause as before, we have

$$\begin{array}{lll} = & \quad (\mathit{left}?a \to \mathit{right}!b \to (P \gg \mathit{right}!a \to Q)) & \text{standard law for } \gg \\ & [\!]\;\; \mathit{right}!b \to (\mathit{left}?a \to (P \gg \mathit{right}!a \to Q)) & \\ & \underline{\text{where}}\; P, Q := NB_p, NB_q & \\ & & \\ = & \quad (\mathit{left}?a \to \mathit{right}!b \to R) & \text{as for } PB \text{ in Sec. 4} \\ & [\!]\;\; \mathit{right}!b \to (\mathit{left}?a \to R) & \\ & \underline{\text{where}}\; R := C[b\!:=a]\;. & \end{array}$$

As representative of the other three summands (total $\overline{pq}$), we consider the $\overline{p}\,\overline{q}$ case, giving

$$\begin{array}{rl} & (\mathit{left}?a \to (\sqcap a \cdot \mathit{right}!a \to NB_p)) \\ \gg & \mathit{right}!b \to (\mathit{left}?a \to (\sqcap a \cdot \mathit{right}!a \to NB_p)) \end{array}$$

$$\begin{array}{lll} = & \quad (\mathit{left}?a \to \mathit{right}!b \to (\sqcap a \cdot P \gg \mathit{right}!a \to Q)) & \\ & [\!]\;\; \mathit{right}!b \to (\mathit{left}?a \to (\sqcap a \cdot P \gg \mathit{right}!a \to Q)) & \\ & \underline{\text{where}}\; P, Q := NB_p, NB_q & \\ & & \\ = & \quad (\mathit{left}?a \to \mathit{right}!b \to R) & \text{as for } PB \\ & [\!]\;\; \mathit{right}!b \to (\mathit{left}?a \to R) & \\ & \underline{\text{where}}\; R := (\sqcap b \cdot C) & \end{array}$$

Finally, we group as before, distributing the probabilistic choice $_{pq}\!\oplus$ into the right-hand side of the <u>where</u>, showing that $NB_p \gg NB_q$ is equivalent to process A of Fig. 3.

The algebra of these buffers is clearly simpler than before, giving this time the general identity

$$NB_p^m \gg NB_q^n \quad = \quad NB_{pq}^{m+n}\;.$$

$$
\begin{array}{lcl}
A & := & (\mathit{left}?a \rightarrow B) \\
B & := & C[b{:=}\,a]\;{}_{pq}\!\oplus (\sqcap b \cdot C) \\
C & := & \left.\begin{array}{ll} & (\mathit{left}?a \rightarrow \mathit{right}!b \rightarrow P) \\ [] & \mathit{right}!b \rightarrow (\mathit{left}?a \rightarrow P) \end{array}\right\} \ \underline{\text{where}}\ P := B
\end{array}
$$

Figure 3: Process A is a nondeterministic two-place buffer.

References

[1] C. Jones. Probabilistic nondeterminism. Monograph ECS-LFCS-90-105, Edinburgh University, 1990. (PhD Thesis).

[2] Carroll Morgan, Annabelle McIver, Karen Seidel, and J. W. Sanders. Refinement-oriented probability for CSP. Technical Report PRG-TR-12-94, Programming Research Group, August 1994.