# Of Probabilistic *wp* and *CSP* —and Compositionality ⋆

Carroll Morgan⋆⋆

Dept. Comp. Sci. & Eng., University of New South Wales,
NSW 2052 Australia
carrollm@cse.unsw.edu.au

**Abstract.** We connect *probabilistic Action Systems* and *probabilistic CSP*, inducing healthiness conditions for the probabilistic traces, failures and divergences of the latter.

A probabilistic sequential semantics for *pGCL* [31] is "inserted underneath" an existing but non-probabilistic link between action systems and *CSP*. Thus the link, which earlier yielded the classic *CSP* healthiness conditions [34], is induced to produce probabilistic versions of them "for free".

Although probabilistic concurrency has enjoyed the attentions of a very large number of researchers over many years—including ourselves [37]—we nevertheless hope to gain new insights by combining the two approaches *CSP* and *pGCL*. In the meantime, however, we probably raise more questions than we answer: in particular, the issue of compositionality—for the moment—remains as delicate as ever.

## 1 Introduction

A typical *state-based* approach to concurrency is the *Action System* formalism of Back and Kurki-Suonio [3], in which the effects of transitions are described in some simple programming language such as Dijkstra's *Guarded-Command Language GCL* [10]; the transitions' enabling conditions are given by the commands' guards, which are predicates over the variables of some state space. By labelling the transitions we determine a labelled transition system.

That is, a state space is shared between a number of *actions*, each of which is enabled or not depending on the current state. The execution of an (enabled) action changes the state, which consequentially changes the set of enabled actions available for the very next step. *UNITY* [7] and *Event-B* [1] have essentially the same structure (although the former makes assumptions of scheduling fairness).

In contrast, a typical *event-based* formalisation of concurrency is the *Communicating Sequential Processes* approach due to C.A.R. Hoare [17]. There the

⋆ We retrace the path of an earlier work [34].

⋆⋆ The author is supported by a Fellowship from the Australian Research Council under its Discovery Grant *DP0345457*.

A.E. Abdallah, C.B. Jones, and J.W. Sanders (Eds.): CSP 2004, LNCS 3525, pp. 220–241, 2005.

actions are called *events*, have no internal structure, and affect no explicit state. The behaviour of a process is understood in terms of the sequences of events in which it might engage, called *traces* and, for finer distinctions, in terms of its *failures* (modelling deadlock) and its *divergences* (for livelock and other chaotic misbehaviour).

Linking state- and event-based approaches is attractive because there are so many real systems whose behaviour is partly controlled by state changes and partly by sequencing, and for that reason a great number of researchers have brought them together before.[1] For example, the contents of a buffer is probably best described by a state, *i.e.* the value(s) it contains; but the exchange of request- and confirm messages necessary to set up the communication channel, which the buffer serves, could well be best described by explicit sequencing.

In our earlier work [34] we linked *standard*, that is non-probabilistic, *GCL*-style action systems and *CSP* by giving three simple formulae for the traces, failures, and divergences of any action system; our approach differed from *e.g.* He's [12] and Josephs' [22] in its use of predicate transformers [10] rather than relations; we felt that the benefit of the predicate transformers was firstly a simpler formulation that included divergence naturally and automatically, and secondly the access to source-level reasoning afforded by predicate-transformer based (*i.e. wp*-style) programming logic. That *wp*-approach has led to further research [45, 4, 6].

In our work here we replace standard predicate transformers by the *probabilistic* predicate transformers [24, 36] that have been developed and extended since our earlier visit to this topic [37]: we (re-)construct and then explore a link between "probabilistic action systems" and what will be, in effect, part of a synthesised "probabilistic *CSP*".[2]

We present probabilistic Action Systems first—they will be action systems written in the probabilistic version *pGCL* of Dijkstra's *guarded-command language* [35, 31, 10]; the "*p*" in the name indicates that we have extended *GCL* with an explicit operator "$_p\oplus$" for the probabilistic choice between two commands.

Then we recall the details of *CSP* very briefly.

Finally, we use *pGCL*'s probabilistic relational model [23, 13] to make a link between probabilistic Action Systems and probabilistic *CSP*, resulting in a synthesis of probabilistic traces, failures and divergences. In the appendix we go on to show how the probabilistic program logic [36, 35, 31], accompanying that model, facilitates algebraic reasoning.

We conclude by discussing *compositionality*, which we regard as the key issue in any exercise of this kind.

[1] They have used both *CSP* and other styles of concurrency.

[2] It differs significantly from the probabilistic *CSP* we constructed via the probabilistic powerdomains of Jones and Plotkin [37, 19, 20, 32], and from other probabilistic *CSP*'s as well [25, 44, 32].

System $\mathcal{H}$:

$$\textbf{initially}\ \ \mathsf{n} := 0 \ {}_{1/2}\!\oplus\ \pm 1 \qquad \begin{array}{rcl} \mathsf{hic} & \widehat{=} & \mathsf{n} \neq 0 \ \rightarrow\ \mathsf{n} := 0 \\ \mathsf{haec} & \widehat{=} & \mathsf{n} = 0 \ \rightarrow\ \mathsf{n} := -1 \ {}_{1/3}\!\oplus 0 \\ \mathsf{hoc} & \widehat{=} & \mathsf{n} < 0 \ \rightarrow\ \mathsf{n} := \pm 1 \end{array}$$

The program variable is just $\mathsf{n}$ of type integer (*i.e.* $\mathbb{Z}$).

The initial state is chosen by flipping an unbiased coin: if it comes up heads, then $\mathsf{n}$ is set to 0; if it is tails, then a demonic choice is made between setting $\mathsf{n}$ to $+1$ or to $-1$.

When $\mathsf{n}$ is $+1$, only hic is enabled; if it is 0, only haec is enabled; if it is $-1$, both hic and hoc are enabled and an external choice is offered between them.

Action hic is neither probabilistic nor demonic; the haec action is purely probabilistic, without demonic choice; and hoc is purely demonic, without probabilistic choice.

Note we are using the given names (*e.g.* hic) both to refer to *actions*, which can have internal structure (*e.g.* can be demonic or probabilistic, and can include a guard), and to refer to *events* which are simply the *labels* of actions and have no structure in themselves.

**Fig. 1.** A probabilistic action system $\mathcal{H}$

## 2 Probabilistic Action Systems: *pAS*

A *probabilistic action system*—or *pAS*—is a set of labelled actions and an initialisation; an *action* is a guard and a command; a *guard* is a predicate; and a *command* is a program fragment in the probabilistic extension *pGCL* of Dijkstra's language of guarded commands [35, 31]. An *initialisation* is a command with no guard. We assume all of the above are given in the context of a collection of program variables over which the meanings of the commands and predicates are defined. Figure 1 is an example of a probabilistic action system in which the actions have been labelled hic, haec and hoc.

Execution of a probabilistic action system proceeds as follows:

1. First, the initialisation is executed; then
2. Repeatedly an enabled action is selected then executed.

An action is *enabled* if its guard is true; it is *executed* by carrying out its command as determined by the semantics of *pGCL*; at the same time, its associated event is deemed to have occurred.

If the repetition in Step 2 fails—because no action is enabled—then the system is *deadlocked*.

One of the possible behaviours of the probabilistic action system in Fig. 1 is to execute hic,haec,hoc repeatedly and forever. But—thankfully—there are many other possibilities: which one actually occurs depends on the outcomes of probabilistic and demonic choices made as the *pAS* evolves.

$$
\begin{array}{lcl}
\mathcal{H} & \widehat{=} & \mathcal{H}_0 \;{}_{1/2}\!\oplus (\mathcal{H}_{-1} \sqcap \mathcal{H}_1) \\
\\
\mathcal{H}_1 & \widehat{=} & \mathsf{hic} \to \mathcal{H}_0 \\
\mathcal{H}_0 & \widehat{=} & \mathsf{haec} \to (\mathcal{H}_{-1} \;{}_{1/3}\!\oplus \mathcal{H}_0) \\
\mathcal{H}_{-1} & \widehat{=} & \mathsf{hic} \to \mathcal{H}_0 \;\; \Box \;\; \mathsf{hoc} \to (\mathcal{H}_{-1} \sqcap \mathcal{H}_1)
\end{array}
$$

Process $\mathcal{H}$ is the entire system of Fig. 1 initially. Process $\mathcal{H}_1$ is then the system as it would behave when $\mathsf{n}$ is $+1$; and the processes $\mathcal{H}_0, \mathcal{H}_{-1}$ correspond to values $0, -1$ respectively.

**Fig. 2.** A "plausible" *pCSP*-style encoding of the system $\mathcal{H}$ from Fig. 1

## 3 Probabilistic Communicating Sequential Processes: *pCSP*

*Probabilistic CSP* is standard *CSP* extended with a ${}_p\!\oplus$ operator between processes: there are many versions, distinguished usually by the way in which internal choice, probabilistic choice and external choice interact, and by whether other features (*e.g.* priorities [25]) are included. When we say "*pCSP*" we mean "as defined here".[3]

Written in *pCSP*, the *pAS* in Fig. 1 would probably be as in Fig. 2: a set of mutually recursive process equations in which we have "coded up" the *pAS* by inventing one process term for each possible state.

The semantics of *CSP*–in its simplest form—includes a set of traces in which processes can engage, where a *trace* is a finite sequence of events and an *event* (as for a *pAS*) is the name of some action; we will see, however, that for *pCSP* we must also consider the *probability* that those traces can occur. Some of the possible traces for the *pAS* of Fig. 1, or equivalently (but informally) the *pCSP* process of Fig. 2, are set out in Fig. 3.

After the next section we will give a formula for the traces of a *pAS*, and for their associated probabilities. Subsequent sections introduce probabilistic failures and divergences.

## 4 Relational Semantics of *pGCL*

The *pGCL* we use for probabilistic action systems has both a relational semantics [23, 13] and a transformer semantics [24, 36]; they are consistent with each other [36, 31] in the same way that conventional relational semantics is consistent with Dijkstra's original predicate-transformer semantics [14, 10]. In this section we concentrate on relational semantics, because it is more intuitive (than transformer semantics, at least at first); we develop the associated probabilistic predicate transformers in the appendix.

[3] We do not mean "*the*" probabilistic *CSP*, since there are many.

$$
\begin{array}{l}
\{\ \langle\rangle, \\
\quad \langle \mathsf{hic}\rangle,\ \langle \mathsf{haec}\rangle,\ \langle \mathsf{hoc}\rangle, \\
\quad \langle \mathsf{hic},\ \ \mathsf{haec}\rangle, \\
\quad \langle \mathsf{haec},\ \mathsf{hic}\rangle, \\
\quad \langle \mathsf{haec},\ \mathsf{haec}\rangle, \\
\quad \langle \mathsf{haec},\ \mathsf{hoc}\rangle, \\
\quad \langle \mathsf{hoc},\ \ \mathsf{hic}\rangle, \\
\quad \langle \mathsf{hoc},\ \ \mathsf{hoc}\rangle, \\
\quad \langle \mathsf{hic},\ \ \mathsf{haec},\ \ \mathsf{hic}\rangle, \\
\quad \langle \mathsf{hic},\ \ \mathsf{haec},\ \ \mathsf{haec}\rangle, \\
\quad \langle \mathsf{hic},\ \ \mathsf{haec},\ \ \mathsf{hoc}\rangle, \\
\qquad\quad \vdots \\
\}
\end{array}
$$

If the left alternative $\mathsf{n} := 0$ is taken in the *pAS* initialisation of $\mathcal{H}$–with probability 1/2—then only haec is offered initially in the associated *pCSP* process. If the right alternative $\mathsf{n} := \pm 1$ is taken—also probability 1/2—then the choice between setting n to $-1$ or to $+1$ is made demonically. If n is set to $-1$, then an external choice between hic and hoc is offered initially. In this case we thus have probabilistic, then internal (demonic), then finally external (angelic) choice in succession. If n is set to $+1$, then only hic is offered.

After hic, only haec can be offered.

After haec, with probability 1/3 an external choice hic/hoc is offered; with probability 2/3, only haec is offered (again).

After hoc, the choice between the offers hic and hic/hoc is demonic.

**Fig. 3.** A partial *pCSP* view of $\mathcal{H}$ from Figs. 1 and 2: its set of traces

---

Let the state space be $S$; we assume it is countable. A *sub-distribution* over $S$ is a function $\Delta$ from $S$ into the unit interval $[0, 1]$ such that $(\sum_{s:S} \Delta.s) \leq 1$, that is such that the total probability over all states $s$ of the individual probabilities $\Delta.s$ is no more than one.[4] A sub-distribution that sums to one may be called a *distribution* (*i.e.* dropping the "sub-").

The set of all sub-distributions over $S$ is written $\overline{S}$, and as a special case we write $\overline{s}$ for the element of $\overline{S}$ that is one at $s$ and zero elsewhere, *i.e.* is the *point distribution* on $s$. We say that a distribution is *standard* if it is $\overline{s}$ for some $s$.

*Non-demonic*—but possibly probabilistic—programs are functions from $S$ to $\overline{S}$, so that program $f$ takes initial state $s$ to the final sub-distribution $f.s$.

[4] In general for function $f$ and argument $x$ we write $f.x$ for the application of $f$ to $x$, and the operator associates to the left: thus $f.g.x$ is $(f(g))(x)$.

Equivalently, the probability that $f$ takes $s$ to $s'$ is just $f.s.s'$. If some program $f$ is such that $f.s.s'$ is either zero or one for all $s, s'$, then we say that $f$ itself is *standard*; clearly such $f$'s are the representatives of traditional deterministic programs.

*Demonic* probabilistic programs are functions from $S$ to *subsets* rather than to simple elements of $\overline{S}$, that is they are of type $S \to \mathbb{P}\overline{S}$, so that program $r$ can take initial state $s$ to final sub-distribution $\Delta'$ just when $\Delta' \in r.s$. (In this multi-valued case we are writing "$r$"—instead of "$f$"—as a mnemonic for "relation".) Thus for example the possible probabilities that program $r$ can take initial $s$ to some final $s'$ ranges (demonically) between the minimum and the maximum of $\Delta'.s'$ over all $\Delta'$ in $r.s$.

### 4.1 Examples of Simple Programs

Let the state space again be $\mathbb{Z}$, and for *pGCL* program *prog* (*i.e.*, given syntactically), let $[\![prog]\!]$ be its relational interpretation as described above. We use $\mathsf{n}$ for the program variable and $n$ for the whole state.

We begin with atomic programs, and then introduce simple compounds.

**identity** — $[\![\mathbf{skip}]\!].n \;=\; \{\overline{n}\}$

The "do-nothing" program **skip** takes any state to itself. Because of our demonic/probabilistic type for programs, however, the result is not just $n$ again, nor even the set $\{n\}$, but rather is the singleton set containing just the point distribution on $n$.

**assignment** — $[\![\mathsf{n}{:=}\ \mathsf{n}+1]\!].n \;=\; \{\overline{n+1}\}$

Non-demonic and non-probabilistic assignments deliver singleton sets of point distributions: singleton sets because there is no demonic choice; point distributions because there is no (non-trivial) probabilistic choice.

**probabilistic choice** —
$$[\![\mathsf{n}{:=}\ \mathsf{n}+1 \;{}_{1/3}\!\oplus\; \mathsf{n}{:=}\ \mathsf{n}+2]\!].n \;=\; \{\Delta'\}$$
where
$$\Delta'.(n+1) \;=\; 1/3 \qquad \Delta'.(n+2) \;=\; 2/3 \qquad \Delta'.n' \;=\; 0 \quad \text{for other values } n'$$

Non-demonic but probabilistic assignments deliver singleton sets of non-trivial sub-distributions: again the sets are singleton because there is no demonic choice; but the single element of the set is a proper sub-distribution.

**demonic choice** —
$$[\![\mathsf{n}{:=}\ \mathsf{n}+1 \;\sqcap\; \mathsf{n}{:=}\ \mathsf{n}+2]\!].n \;=\; \{\overline{n+1}, \overline{n+2}\}$$

A purely demonic (and non-probabilistic) binary choice delivers the sub-distributions contributed by each of its operands.

**demonic probabilistic choice** —
$$[\![\mathsf{n}:= \mathsf{n}+1 \;{}_{1/3}\oplus_{1/3}\; \mathsf{n}:= \mathsf{n}+2]\!].n$$
$$= \{\Delta'_{1/3}, \Delta'_{2/3}\}$$
$$\text{where } \Delta'_p.(n+1) = p$$
$$\Delta'_p.(n+2) = 1-p$$
$$\Delta'_p.n' = 0 \quad \text{for other values } n'$$

The notation ${}_p\oplus_q$, for $p+q \leq 1$, abbreviates the demonic choice between the two probabilistic choices ${}_p\oplus$ and ${}_{1-q}\oplus$: it executes the left branch with probability *at least* $p$, the right with probability *at least* $q$ and—in any case—it is certain to execute one or the other.

### 4.2 "Naked" Guarded Commands and Miracles

The *pGCL* commands in probabilistic action systems are equipped with a guard that controls whether or not they are enabled in the current state. We build that in to the relational semantics of *pGCL* by "erasing" the parts of transitions that the guard does not enable: if a state does not make the guard true, then its result set is empty from that state. That is, for predicate $gd$ we define

$$\Delta' \in [\![gd \to prog]\!].s \quad \widehat{=} \quad s \in [\![gd]\!] \;\wedge\; \Delta' \in [\![prog]\!].s \;,$$

where by $[\![gd]\!]$ we mean the subset of $S$ denoted by the guard $gd$.

This is of course the "normal" way of dealing with *miracles* when considered relationally: because a miraculous command has no final states at all, every final state it produces satisfies $\mathsf{false}$–and therefore we imagine that its execution cannot even be started [41, 33, 40, 18].

The enabling/disabling property of a guard is very convenient when moving between action systems and *CSP* [34, 45, 22] —whether probabilistic or not— since it automatically excludes the traces which the action system cannot produce.

### 4.3 Sequential Composition in *pGCL*

As an action system executes, it carries out one (guarded) command after another; the overall effect is the sequential composition of all the (finitely many) commands concerned. Given two commands $prog_1$ and $prog_2$, we therefore want to construct the relational semantics of their composition.

We begin with non-demonic programs $f$ (*i.e.* with their meanings). If we are given some sub-distribution $\Delta$ of *initial* states from which $f$ will repeatedly be run, the overall effect can be obtained by averaging $f$'s output sub-distributions for each initial state over the known "incoming" sub-distribution $\Delta$ for them: thus we define

$$f^*.\Delta.s' \quad \widehat{=} \quad \left(\sum_{s:S} \Delta.s * f.s.s'\right) \;, \tag{1}$$

where we distinguish the $f$-over-sub-distributions from the original $f$ by writing $f^*$ for the former. Note that the original can be recovered, since $f.s = f^*.\overline{s}$.

Now to determine the effect of a possibly demonic $r$ on an initial sub-distribution $\Delta$, we construct the collection of its non-demonic "refinements" $f$ and then refer to (1) above: that is, we say that "$r$ is refined by $f$" just when $f$ satisfies $(\forall s: S \cdot r.s \ni f.s)$, and we write it $r \sqsubseteq f$. Then we define

$$r^*.\Delta \quad \widehat{=} \quad \{f: S \to \overline{S} \mid r^+ \sqsubseteq f \cdot f^*.\Delta\} \ , \quad [5]$$

where again we use $(\cdot)^*$ to indicate "lifting" a function to act over sub-distributions rather than individual states, and where the relation $r^+$ is the "down closure" of $r$ obtained by adding the everywhere-zero sub-distribution to $r.s$ whenever $r.s$ is empty.

Finally, we describe sequential composition simply by applying the "lifted" semantics of the second component to every final sub-distribution the first component could produce: that is, for initial state $s$ we define

$$[\![prog_1;\ prog_2]\!].s \quad \widehat{=} \quad \{\Delta: [\![prog_1]\!].s \cdot [\![prog_2]\!]^*.\Delta\} \ . \quad [6]$$

## 5 Traces of a *pAS*

We now use the sequential composition of Sec. 4.3 to determine the traces of a *pAS*. Let its initialisation be command *ini* and let its events be $\mathsf{e}_1, \mathsf{e}_2, \cdots$. The *alphabet* of the action system *pAS* is the set of all its events (whether or not they actually can be executed).

We write subsets of the state space in three different ways, as convenient: as *sets of states* directly (whether enumerated or given as a comprehension); as *predicates* over program variables, denoting sets whose variables' values satisfy the predicate; and as *sets of events*, in which case we will mean the set of states corresponding to the disjunction of the events' guards. Note that the empty set $\emptyset$, whether of states or of events, corresponds to the predicate $\mathsf{false}$.

When we write events or sequences of events between semantic brackets $[\![\cdot]\!]$, we mean the relational semantics of the corresponding actions, with their guards, sequentially composed if appropriate.

In standard *CSP*, a trace is a finite sequence of events drawn from the alphabet; and the *traces* model of a process is the set of all the traces it could carry out [17]. Because a particular trace is included in the trace-semantics of a processes if it *can* occur, our probabilistic view will be that we are interested in the *maximum probability* of that occurrence. For example, the sets of traces for the two standard processes

[5] This set comprehension is read "vary bound variable $f$ over its type $S \to \overline{S}$; select those values satisfying the condition $r^+ \sqsubseteq f$; form set elements from them according to the expression $f^*.\Delta$."

[6] In this comprehension the omitted condition defaults to *true*; refer Footnote 5 immediately above.

$$
\begin{array}{lll}
\text{System } \mathcal{A}\text{:} & \textbf{initially}\ \ \mathsf{n}:= 0 & \mathsf{hic} \ \widehat{=}\ \ \mathsf{n} \geq 0 \ \rightarrow\ \mathsf{n}:= +1 \\
 & & \mathsf{hoc} \ \widehat{=}\ \ \mathsf{n} \leq 0 \ \rightarrow\ \mathsf{n}:= -1 \\
\text{System } \mathcal{D}\text{:} & \textbf{initially}\ \ \mathsf{n}:= \pm 1 & \mathsf{hic} \ \widehat{=}\ \ \mathsf{n} \geq 0 \ \rightarrow\ \mathsf{n}:= +1 \\
 & & \mathsf{hoc} \ \widehat{=}\ \ \mathsf{n} \leq 0 \ \rightarrow\ \mathsf{n}:= -1 \\
\text{System } \mathcal{P}\text{:} & \textbf{initially}\ \ \mathsf{n}:= -1 \,{}_{1/2}\!\oplus +1 & \mathsf{hic} \ \widehat{=}\ \ \mathsf{n} \geq 0 \ \rightarrow\ \mathsf{n}:= +1 \\
 & & \mathsf{hoc} \ \widehat{=}\ \ \mathsf{n} \leq 0 \ \rightarrow\ \mathsf{n}:= -1
\end{array}
$$

All three systems exhibit the same potential traces, *i.e.* any trace comprising either all hic's or all hoc's; but in System $\mathcal{P}$ the associated probabilities can be included by giving a set of trace-probability pairs

$$
\begin{array}{l}
\{\ (\langle\rangle, 1), \quad (\langle \mathsf{hic}\rangle, 1/2),\ (\langle \mathsf{hic}, \mathsf{hic}\rangle, 1/2), \quad (\langle \mathsf{hic}, \mathsf{hic}, \mathsf{hic}\rangle, 1/2), \cdots \\
\qquad\quad (\langle \mathsf{hoc}\rangle, 1/2), (\langle \mathsf{hoc}, \mathsf{hoc}\rangle, 1/2), (\langle \mathsf{hoc}, \mathsf{hoc}, \mathsf{hoc}\rangle, 1/2), \cdots\ \}\ ,
\end{array}
$$

where the second element of each pair is the (maximum) probability with which the first element can occur.

In Systems $\mathcal{A}$ and $\mathcal{D}$ the "trace-probability" would be just one when a trace can occur, and zero when it cannot. The standard trace semantics in those cases is obtained by removing the probability-zero pairs, and then "projecting away" the probability-one information from those that remain.

**Fig. 4.** Three action systems: angelic, demonic, probabilistic

---

$$
\begin{array}{lll}
 & \mathsf{hic} \rightarrow \mathit{STOP} \ \ \Box\ \ \mathsf{hoc} \rightarrow \mathit{STOP} & \text{—external choice} \\
\text{and} & \mathsf{hic} \rightarrow \mathit{STOP} \ \ \sqcap\ \ \mathsf{hoc} \rightarrow \mathit{STOP} & \text{—internal choice}
\end{array}
$$

are the same, being just $\{\langle\rangle, \langle \mathsf{hic}\rangle, \langle \mathsf{hoc}\rangle\}$ in each case and not taking account of the fact that the second process—with its "demon" $\sqcap$ representing the internal choice—cannot be forced to produce either of the non-empty traces separately.

Accordingly, in our probabilistic view, we will associate *probability one* with all three traces, for both processes, with the same caveat about ignoring the demon (for now).

Action systems for two similar processes $\mathcal{A}$ (for angelic) and $\mathcal{D}$ (for demonic) are given in Fig. 4, together with a third system $\mathcal{P}$ which chooses probabilistically between the two events. As we are about to see, it exhibits proper probabilities.

We begin by considering System $\mathcal{P}$. By (informal) inspection, the probability that hic will occur is just the probability that the initialisation *ini* establishes the guard $\mathsf{n} \geq 0$ of that event. From our relational semantics of Sec. 4 we know that the initialisation produces the single distribution

$$
\Delta_{ini} \quad \widehat{=} \quad \{-1 \mapsto 1/2, \quad +1 \mapsto 1/2\} \tag{2}
$$

which assigns probability 1/2 to the subset $\{0, 1, \cdots\}$ of $\mathbb{Z}$ in which hic is enabled.[7] We could also write that as the singleton set of guards "{hic}".

That probability is in fact the *expected value* of the characteristic function of the set concerned, taken over the distribution (2) above that the initialisation produces. However that set is written, whether explicitly or as a predicate or as a set of events (*i.e.* as the disjunction of their guards, in the last case), we use the notation $[\cdot]$ to form the associated characteristic function: thus we would write $[0, 1, \cdots]$ or $[\mathsf{n} \geq 0]$ or [hic] here, meaning in each case the function over the integers that takes non-negative arguments to one and negative arguments to zero.

In general, for the expected value over a sub-distribution $\Delta$ of some random variable $B$ (itself a function from the state space into the reals), we write

$$\int_{\Delta} B \quad \widehat{=} \quad \left( \sum_{s:S} \Delta.s * B.s \right) , \tag{3}$$

so that the probability 1/2 we calculated above is just $\int_{\Delta_{ini}} [\mathsf{hic}]$ .

We now form a combined notation for all the above operations, that is of determining the relational semantics of a non-demonic command, applying it to an initial state, and then taking the expected value of some function: we define

$$\mathrm{Exp}.[\![prog]\!].B.s \quad \widehat{=} \quad \int_{\Delta'} B \qquad \text{given that } [\![prog]\!].s = \{\Delta'\} .$$

As a result, we know that when $B$ is some standard $[Q]$ for predicate $Q$, the expression $\mathrm{Exp}.[\![prog]\!].[Q].s$ is the probability that the non-demonic *prog* will reach $Q$ from $s$.[8]

If we now look at the *action* associated with hic, we see that its relational semantics is given by

$$[\![\mathsf{n} \geq 0 \to \mathsf{n} := +1]\!] \quad = \quad \{\, n{:}\mathbb{Z} \mid n \geq 0 \cdot \{n \mapsto \{\, \overline{+1} \,\}\}\, \} ,$$

the partial function defined only on non-negative arguments which produces the singleton result set of sub-distributions $\{\, \overline{+1} \,\}$ for each one of them. If we ask "what is the *maximum* possible expected value of random variable [true] after executing hic?" from initial state $n$—for which we could invent the notation

$$\overline{\mathrm{Exp}}.[\![\mathsf{n} \geq 0 \to \mathsf{n} := +1]\!].[\mathsf{true}].n \quad {}^{9} \tag{4}$$

by incorporating the "maximum" as an overbar— we find it is just the random variable [hic] itself, since whenever $n$ does not satisfy hic's guard $n \geq 0$ we

[7] In the usual terminology of probability theory we would speak of the probabilistic *event* $\{0, 1, \cdots\}$ rather than subset; but we must avoid confusion with the "events" of *CSP*.

[8] We will deal with the demonic-*prog* case shortly.

[9] We are defining $\overline{\mathrm{Exp}}.[\![prog]\!].B.s \;\widehat{=}\; (\sqcup \Delta' {:}\, [\![prog]\!].s \cdot \int_{\Delta'} B)$ .

are taking the maximum over an *empty* set of non-negative reals, yielding zero. When $n$ does satisfy hic's guard, the expression (4) gives one, the probability assigned by the distribution $\overline{+1}$ to the whole state space (of which [true] is the characteristic function). That is, we find that

$$[\mathsf{hic}] \quad = \quad \overline{\mathrm{Exp}}.[\![\mathsf{hic}]\!].[\mathsf{true}] \ , \tag{5}$$

where we recall that "hic" between semantic brackets refers to the corresponding action, including its guard.

We can now put our two experiments with hic together: since the initialisation is unguarded, terminating and purely probabilistic, it produces a single distribution from every initial state and so is unaffected if we use $\overline{\mathrm{Exp}}.[\![\cdot]\!]$ rather than $\mathrm{Exp}.[\![\cdot]\!]$. Thus we have that the (maximum) probability of the occurrence of the trace $\langle \mathsf{hic} \rangle$ in System $\mathcal{P}$ can be written

$$\overline{\mathrm{Exp}}.[\![\mathit{ini}]\!].[\mathsf{hic}].n \ , \quad \text{that is} \quad \overline{\mathrm{Exp}}.[\![\mathit{ini}]\!].(\overline{\mathrm{Exp}}.[\![\mathsf{hic}]\!].[\mathsf{true}]).n \ , \tag{6}$$

where on the right we have appealed to (5). But, as we prove later in Fig. 7 of App. A, the "cascaded" use of expectations at (6) on the right can be simplified to just

$$\overline{\mathrm{Exp}}.[\![\mathit{ini};\ \mathsf{hic}]\!].[\mathsf{true}].n \tag{7}$$

because $\overline{\mathrm{Exp}}.[\![\cdot]\!]$ distributes over sequential composition, becoming functional composition.

From our operational intuition, we believe that the expression (7) will equal $1/2$ for any initial $n$, as will the further extended $\overline{\mathrm{Exp}}.[\![\mathit{ini};\ \mathsf{hic};\ \mathsf{hic}]\!].[\mathsf{true}].n$, and so on.

Now to give the "trace semantics" of a probabilistic action system we can use the above to map every potential trace (finite sequence of events) to the maximum probability of its occurrence.

Let the *pAS* be $\mathcal{S}$ over a state space $S$. As before, for a given finite trace say $\mathsf{es} = \langle \mathsf{e}_1, \mathsf{e}_2, \cdots, \mathsf{e}_n \rangle$ of events from $\mathcal{S}$ we mean the sequential composition $\mathsf{e}_1; \mathsf{e}_2; \cdots; \mathsf{e}_n$ of the events' corresponding actions whenever es appears within semantic brackets $[\![\cdot]\!]$. Also, we continue to use $[\cdot]$ to form characteristic functions, and we let *ini* be the initialisation of $\mathcal{S}$. Then for any predicate $Q$ over the state space $S$ we define

$$\mathcal{S}.\langle\!\langle \mathsf{es} \rangle\!\rangle.Q \quad \widehat{=} \quad (\sqcup s\colon S \cdot \overline{\mathrm{Exp}}.[\![\mathit{ini};\ \mathsf{es}]\!].[Q].s) \ , \tag{8}$$

where on the right the terms $S, \mathit{ini}, \mathsf{es}, Q$ are to be interpreted within the system $\mathcal{S}$ mentioned on the left. When $\mathcal{S}$ is clear from context, however, we omit it and write just $\langle\!\langle \mathsf{es} \rangle\!\rangle.Q$ on the left.

Thus Eqns. (6) and (7)—the maximum probability that trace $\langle \mathsf{hic} \rangle$ can occur in $\mathcal{P}$—would be written simply $\mathcal{P}.\langle\!\langle \mathsf{hic} \rangle\!\rangle.\mathsf{true}$. What the notation of (8) has done is simply to bundle up the choice of action system, the inclusion of the initialisation, and the maximising over all initial states.[10]

[10] Maximising over *initial* states is usually unnecessary: since the initialisation rarely depends on *its* initial state, the effect of the quantification is merely to replace some constant *function* (of the initial $s$) by the constant itself.

We can now give the probabilistic trace-semantics of $\mathcal{S}$: it is a function from finite sequences of the events of $\mathcal{S}$ into the real interval $[0,1]$, giving for each sequence $\mathsf{es}$ the maximum probability of its occurrence. We call the function $\mathrm{pTr}_{\mathcal{S}}$, and define

$$\mathrm{pTr}_{\mathcal{S}}.\mathsf{es} \quad \widehat{=} \quad \mathcal{S}.\langle\!\langle \mathsf{es} \rangle\!\rangle.\mathsf{true} \ . \tag{9}$$

Again, we omit the $\mathcal{S}$ when it is obvious, so that $\mathrm{pTr}.\mathsf{es} = \langle\!\langle \mathsf{es} \rangle\!\rangle.\mathsf{true}$.

## 6 Failures

The traces of *CSP* are sufficient only for describing deterministic behaviour: when describing (internal) nondeterminism as well, *CSP* makes more detailed observations. A *failure* is a pair comprising a trace and a refusal; a *refusal* is a set of events in which the system can "refuse" to engage.

Let $\mathsf{es}$ be a trace and $\mathsf{E}$ a refusal. The behaviour $(\mathsf{es}, \mathsf{E})$ is observed whenever the process first engages in all the events in $\mathsf{es}$ and then refuses to extend the trace with any event in $\mathsf{E}$.

Systems $\mathcal{A}$, $\mathcal{D}$ and $\mathcal{P}$ from Fig. 4 have the same standard traces, as we have already seen; and the first two agree even for probabilistic traces, mapping each possible trace to probability one and all others to zero. But $\mathcal{A}$ and $\mathcal{D}$ are distinguished by their failures, since for example $(\langle\rangle, \{\mathsf{hic}\})$ is a failure of $\mathcal{D}$ but not of $\mathcal{A}$. Operationally we see this by noting that after initialisation of $\mathcal{A}$ the event $\mathsf{hic}$ cannot fail to be enabled; but if the initialisation of $\mathcal{D}$ sets $\mathsf{n}$ to $-1$, then $\mathsf{hic}$ will be disabled, and so can be refused.

In the previous section we considered expressions $\overline{\mathrm{Exp}}.[\![prog]\!].[\mathsf{true}].s$, for trace semantics; but we know more generally that for standard $[Q]$ (*i.e.* for predicate $Q$ not necessarily $\mathsf{true}$), the expression $\overline{\mathrm{Exp}}.[\![prog]\!].[Q].s$ is the *maximum* probability that *prog* will reach $Q$ from $s$; and again that maximum is zero whenever the guard of *prog* is false, since in that case *prog* cannot reach anything.

Now the "failure semantics" of an action system should give for each potential failure $(\mathsf{es}, \mathsf{E})$ the maximum probability that it will be observed. Since this is the maximum probability that the system can engage in $\mathsf{es}$ and reach a state *not* enabling any event in $\mathsf{E}$, we define

$$\mathrm{pFail}.(\mathsf{es}, \mathsf{E}) \quad \widehat{=} \quad \langle\!\langle \mathsf{es} \rangle\!\rangle.(\neg \mathsf{E}) \ , \tag{10}$$

where $\neg\mathsf{E}$ is the complement of $\mathsf{E}$, that is the subset of $S$ in which no event of $\mathsf{E}$ is enabled. Thus, as in standard *CSP*, we have

$$\mathrm{pTr}.\mathsf{es} \quad = \quad \mathrm{pFail}.(\mathsf{es}, \emptyset) \ .$$

## 7 Divergences

A *divergence* of a *CSP* process is a trace after which the process behaves chaotically. In a *pAS* that behaviour is deemed to result from a potentially "aborting" command, one which we will model by adding a special element $\bot$ to our

state space, to represent non-termination. Sub-distributions are now taken over $S_\bot = S \cup \{\bot\}$, with the value they assign to $\bot$ itself being the probability that the command fails to terminate normally.

Sequential composition is handled (in the usual way) by insisting that every command preserves "having failed to terminate"; that is, in extending our relational semantics we insist that for all programs *prog* we have $[\![prog]\!].\bot = \{\overline{\bot}\}$. And we add to our earlier list of relational semantics examples (Sec. 4.1) the item

**abort** — $[\![\mathbf{abort}]\!].n \;=\; \{\overline{\bot}\}$

The diverging program **abort** takes every state to the special "bottom" state $\bot$.

We need not extend our random variables, however, which remain functions of $S$ alone; instead, we adjust the definition of $\overline{\text{Exp}}.[\![\cdot]\!]$ (from (4) and its Footnote 9), which becomes

$$\overline{\text{Exp}}.[\![prog]\!].B.s \quad \widehat{=} \quad (\sqcup \Delta' : [\![prog]\!].s \cdot \Delta'.\bot + \int_{\Delta'} B) \;, \tag{11}$$

where the $\int$ notation continues to denote a summation over proper (*i.e.* non-$\bot$) values of $S$ only, as at (3). This reflects our interest in the traces and failures a process *might* do (as opposed to "can be forced to do"): the maximum probability of *any* behaviour, after divergence, is one; and that is why we introduce the extra additive term $\Delta'.\bot$, which assigns a value of one to a command's reaching $\bot$. (Recall that $B$ itself is not defined for $\bot$.)

With this new apparatus, we now define

$$\text{pDiv}.\mathsf{es} \quad \widehat{=} \quad \langle\!\langle \mathsf{es} \rangle\!\rangle.\mathsf{false} \;,$$

giving for any sequence of events $\mathsf{es}$ the maximum probability that executing the corresponding actions can achieve the predicate $\mathsf{false}$–because the only way an action can "achieve" $\mathsf{false}$ is to diverge, and that is precisely the behaviour we are trying to quantify.

In Fig. 5 we give several examples of potentially diverging probabilistic action systems, all with alphabet $\{\mathsf{hic}, \mathsf{hoc}\}$. System $\mathcal{X}_1$ aborts immediately, and is equivalent to the *CSP* process *CHAOS*; for example (writing the $\mathcal{X}_1$ explicitly) we have $\text{pDiv}_{\mathcal{X}_1}.\mathsf{es} = 1$ for all traces $\mathsf{es}$, including the empty trace.[11]

System $\mathcal{X}_2$ literally (but informally) translated into *CSP* appears to be the process that can execute (and indeed can be forced to execute) any number of $\mathsf{hic}$'s; but as soon as it does a $\mathsf{hoc}$, it diverges. As in System $\mathcal{X}_1$, all traces have probability one; but we have $\text{pDiv}_{\mathcal{X}_2}.\langle\rangle = 0$ whereas we have seen that $\text{pDiv}_{\mathcal{X}_1}.\langle\rangle = 1$. The shortest nonzero-probability divergence for $\mathcal{X}_2$ is $\langle \mathsf{hoc} \rangle$; it and all its extensions have (maximum) probability one of divergence.

[11] Divergence has implications for the failures of a system as well, as we see in Sec. 7 below: any trace or failure extending a divergence has probability at least as great as the divergence.

| | | | |
|---|---|---|---|
| System $\mathcal{X}_1$ | **initially abort** | hic $\widehat{=}$ false $\rightarrow$ **skip** [12] | |
| | | hoc $\widehat{=}$ false $\rightarrow$ **skip** | |
| System $\mathcal{X}_2$ | **initially** $\mathsf{n} := 0$ | hic $\widehat{=}$ $\mathsf{n} \geq 0 \rightarrow$ **skip** | |
| | | hoc $\widehat{=}$ $\mathsf{n} \leq 0 \rightarrow$ **abort** | |
| System $\mathcal{X}_3$ | **initially** $\mathsf{n} := \pm 1$ | hic $\widehat{=}$ $\mathsf{n} \geq 0 \rightarrow$ **skip** | |
| | | hoc $\widehat{=}$ $\mathsf{n} \leq 0 \rightarrow$ **abort** | |
| System $\mathcal{X}_4$ | **initially** $\mathsf{n} := -1 \,{}_{1/2}\!\oplus +1$ | hic $\widehat{=}$ $\mathsf{n} \geq 0 \rightarrow$ **skip** | |
| | | hoc $\widehat{=}$ $\mathsf{n} \leq 0 \rightarrow$ **abort** | |

**Fig. 5.** Action systems that can diverge

System $\mathcal{X}_3$ contains demonic choice in its initialisation, and so the process decides internally whether to begin with hic or with hoc. If the former, it must continue with hic's forever (and cannot diverge); if the latter, it can execute hoc and then diverge, continuing after that with hic's, hoc's or deadlock *ad lib*.

System $\mathcal{X}_4$ is like $\mathcal{X}_3$ except that the initial choice —still not accessible externally—is at least predictable to the extent that it is made with the probability shown; after that, it behaves like $\mathcal{X}_3$. We give the complete traces, failures and divergences of $\mathcal{X}_4$ in App. B.

## 8 Healthiness Conditions for Probabilistic Action Systems

The failures and divergences of standard *CSP* satisfy the conditions listed in Fig. 6. We discuss the probabilistic version for each one in turn; they all have straightforward proofs in the program logic of *pGCL*, and as an example of that the proof of **pC3** below is given in App. A.1. Throughout, by "probability" we mean "maximum probability".

**pC0** — $$\text{pFail.}(\langle\rangle, \emptyset) \;=\; 1$$
It is always possible for a system to start, since its initialisation is unguarded.

**pC1** — $$\text{pFail.}(\mathsf{es} +\!\!+ \mathsf{es}', \mathsf{E}) \;\leq\; \text{pFail.}(\mathsf{es}, \emptyset)$$
The probability of continuing a trace is no more than the probability of achieving the trace itself.

[12] Events with guard false are in the alphabet of the system but can never be explicitly enabled.

$$
\begin{array}{ll}
\mathbf{C0} & (\langle\rangle, \emptyset) \in F \\
\mathbf{C1} & (\mathsf{es} +\!\!+ \mathsf{es}', \mathsf{E}) \in F \;\Rightarrow\; (\mathsf{es}, \emptyset) \in F \quad {}^{13} \\
\mathbf{C2} & (\mathsf{es}, \mathsf{E}) \in F \;\wedge\; \mathsf{E}' \subseteq \mathsf{E} \;\Rightarrow\; (\mathsf{es}, \mathsf{E}') \in F \\
\mathbf{C3} & (\mathsf{es}, \mathsf{E}) \in F \;\Rightarrow\; (\mathsf{es} +\!\!+ \langle \mathsf{e} \rangle, \emptyset) \in F \;\vee\; (\mathsf{es}, \mathsf{E} \cup \{\mathsf{e}\}) \in F \\
\mathbf{C4} & \mathsf{es} \in D \;\Rightarrow\; \mathsf{es} +\!\!+ \mathsf{es}' \in D \\
\mathbf{C5} & \mathsf{es} \in D \;\Rightarrow\; (\mathsf{es}, \mathsf{E}) \in F
\end{array}
$$

For any set of failures $F$ and divergences $D$ of a standard *CSP* process, the above conditions hold for any event $\mathsf{e}$, traces $\mathsf{es}, \mathsf{es}'$ and sets of events $\mathsf{E}, \mathsf{E}'$ over the alphabet of the process.

**Fig. 6.** Healthiness conditions for standard *CSP* over a finite alphabet

---

**pC2** — $\mathrm{pFail}.(\mathsf{es}, \mathsf{E}) \;\geq\; \mathrm{pFail}.(\mathsf{es}, \mathsf{E} \cup \mathsf{E}')$

The probability of refusing a set of events is no less than the probability of refusing a superset of it.

**pC3** — $\mathrm{pFail}.(\mathsf{es}, \mathsf{E}) \;\leq\; \mathrm{pFail}.(\mathsf{es} +\!\!+ \langle \mathsf{e} \rangle, \emptyset) + \mathrm{pFail}.(\mathsf{es}, \mathsf{E} \cup \{\mathsf{e}\})$

If an event cannot be refused, then it must be accepted.

**pC4** — $\mathrm{pDiv}.\mathsf{es} \;\leq\; \mathrm{pDiv}.(\mathsf{es} +\!\!+ \mathsf{es}')$

Any event is possible after divergence.

**pC5** — $\mathrm{pDiv}.\mathsf{es} \;\leq\; \mathrm{pFail}.(\mathsf{es}, \mathsf{E})$

Any refusal is possible after divergence.

Recall that **pC3** is proved in App. A.1.

## 9 What Now?

In fact almost everything still remains to be done.

- The *refinement* order for *pCSP*–when one process can be said to be implemented by another—is suggested by the refinement order for *pGCL* that we describe briefly in App. A, provided care is taken with the guards of the generating *pAS*. This has been shown already by a number of authors for the standard case [22, 11, 12, 45, 4], and it should be checked for the probabilistic case.

[13] We use $+\!\!+$ for concatenation of traces.

- Because there is a *pGCL* construction [36, 31] taking transformer semantics (as in App. A) back to relational semantics (as in Sec. 4), we should expect that there is a canonical mapping from *pCSP* back to a *pAS* in what we would consider a "normal form", using the technique earlier employed in the standard case [22, 11] where the normal-form state space is the set of *CSP*-style refusals over the alphabet of the process. This induces an equivalence relation on the *pAS*'s directly, and it should be verified that it is intuitively reasonable.
- The combining operations between *pAS*, especially their parallel composition but also prefixing, internal and external choice, probabilistic choice, hiding... are suggested by the corresponding operations defined for standard action systems [4, 5]: it must be checked that they respect the normal-form equivalence. But parallel composition raises interesting problems, since it must in turn be based on the parallel composition *of commands* (*e.g.* [2]), which operation requires great care when those commands include both probabilistic and demonic choice. (In fact parallel composition of initialisation commands is necessary for external choice also.)
- Most important of all—and subsuming much of the above—is that once the *pAS* operations have been defined, there should be *pCSP* operations corresponding to them that are expressed only in terms of our semantic observations pFail and pDiv. This would be *compositionality*.

### 9.1 Compositionality

Unfortunately, it has been known for some time that compositionality is not possible in terms of observations like pFail and pDiv alone [26]; indeed, we know that "probability-of-attaining-a-postcondition" -style semantics is not compositional even for *sequential* demonic/probabilistic programs [29].

The *expectation-transformer* semantics of *pGCL* however uses a generalised form of postcondition in which states are associated with non-negative reals (the states' "value") rather than simply with a Boolean (whether the state is "acceptable" or not); and *pGCL* semantics is compositional for sequential programs, even when demonic- and probabilistic choice appear together [36, 35, 31].

The corresponding extension which that suggests for *pCSP* is that a refusal should be a function from event to $\mathbb{R}^{\geq}$ (the "cost" of refusing the event?) rather than simply a function from event to Boolean (whether it can or cannot be refused). A failure pFail.$(\mathsf{es}, \mathsf{E})$ would then be the (maximum possible) expected value of the real-valued function $\mathsf{E}$ after observations of the trace $\mathsf{es}$.

Unfortunately (again), it has already been shown that this does not offer an easy road to compositionality [21, 15]: and so there probably will be even further extensions required, for example a form of "may/must" testing but with respect to testing *trees* (rather than the simpler "broom-like" shapes offered by failures [42]), *together with* delivering quantitive rewards rather than only "yes" or "no" [9, 21].

Compositionality of course is the key to a successful abstraction. We take our favourite example—and it is probabilistic—from genetics.

Knowing parents' eye colour, on its own, cannot be used to predict distribution of eye colour among their children: some brown-eyed parents are virtually certain to produce brown-eyed children, *i.e.* with probability one; other brown-eyed parents may produce blue-eyed children with a predictable probability of one in four. Since the distribution of children's eye colour cannot be predicted from their parents' eye colour alone, the eye-colour abstraction (of a person) is not compositional for the binary operation "having children" between people: it is too severe. This is in effect where we find ourselves with pFail and pDiv.

At the other extreme, we have the full genetic profile of both parents; though still an abstraction, since it ignores phenotype, it may be sufficient *in principle* to predict the distribution of genotype in their children: as such it would be compositional. But it is far too costly a method if eye colour is all that interests us. This is where we might be if we worked with *pAS* directly, or (equivalently) probabilistic labelled transition systems or even probabilistic nondeterministic automata.

Thus "eye colour" alone is economical but not reliable; and "full genetic profile" is reliable but not economical. The right level of abstraction—the crucial breakthrough of Mendel—came from understanding the role of dominant and recessive characteristics (*alleles*), and led to a method of analysis which is both accurate and cheap to perform. Although very difficult to find, once discovered the abstraction "eye colour *together with* its dominant/recessive characteristic" turned out to be economical, reliable and easy to understand. Most importantly, it is compositional.

This is what we seek: the "alleles" for probabilistic, nondeterministic concurrent systems.

## 10 Conclusion

We are aware that many *CSP*-researchers—not to mention the even more numerous membership of the *CCS*-based community—have "thought long and hard" about how to introduce probability and nondeterminism together into a concurrent setting.

Clearly that has not stopped us from trying again, even using a very simple approach. In the ten years since our earlier encounter with "*pCSP*" [37], we have learned a great deal about the subtleties of probabilistic *vs.* demonic choice from having worked extensively on probabilistic semantics [36, 31]—both for sequential programs (and abstraction/nondeterminism), and for two-player games with probabilistic, demonic and angelic choice treated together [27, 28].

Treating concurrency in the "behavioural" style seems to be an inescapable point of view for anyone who has ever seriously been exposed to "the *CSP* effect" [43]. Its astonishing conceptual power and beauty—that it can express such subtle concepts with such simple means—is undiminished, even twenty-five years later. Nothing less elegant can ever suffice.

## Acknowledgements

We thank the *Australian Research Council* for their support under their *Discovery Grants* Programme, and Christine Paulin-Mohring and the Laboratoire de Recherche en Informatique (*LRI*) at Orsay for their hospitality during the period March–June 2004.

We are grateful also to the organisers of the *25 Years of* CSP *Meeting* at which this work was first presented, and to the referee who advised us on the preparation of the final version of the article.

## References

1. J.-R. Abrial. Extending *B* without changing it (for developing distributed systems). In H. Habrias, editor, *First Conference on the B Method*, pages 169–190. Laboratoire LIANA, L'Institut Universitaire de Technologie (IUT) de Nantes, November 1996.
2. R.-J.R. Back and M.J. Butler. Fusion and simultaneous execution in the refinement calculus. *Acta Informatica*, 35(11):921–949, 1998.
3. R.-J.R. Back and R. Kurki-Suonio. Decentralisation of process nets with centralised control. In *2nd ACM SIGACT-SIGOPS Symp. Principles of Distributed Computing*, pages 131–142, 1983.
4. M.J. Butler. A CSP approach to action systems. Technical report, Oxford University, 1992. (DPhil Thesis).
5. M.J. Butler. *csp2B*: A practical approach to combining CSP and B. *Formal Aspects of Computing*, pages 182–196, 2000.
6. M.J. Butler and C.C. Morgan. Action systems, unbounded nondeterminism and infinite traces. *Formal Aspects of Computing*, 7(1):37–53, 1995.
7. K.M. Chandy and J. Misra. *Parallel Program Design: A Foundation*. Addison-Wesley, Reading, Mass., 1988.
8. J.W. Davies, A.W. Roscoe, and J.C.P. Woodcock, editors. *Millennial Perspectives in Computer Science*. Cornerstones of Computing. Palgrave, 2000.
9. M. de Nicola and M. Hennessy. Testing equivalence for processes. *Theoretical Computer Science*, 34, 1984.
10. E.W. Dijkstra. *A Discipline of Programming*. Prentice Hall International, Englewood Cliffs, N.J., 1976.
11. Jifeng He. Process refinement. In J. McDermid, editor, *The Theory and Practice of Refinement*. Butterworths, 1989.
12. Jifeng He. Process simulation and refinement. *Formal Aspects of Computing*, 1(3):229–241, 1989.
13. Jifeng He, K. Seidel, and A.K. McIver. Probabilistic models for the guarded command language. *Science of Computer Programming*, 28:171–192, 1997. Available at [30–key HSM95].
14. Wim H. Hesselink. *Programs, Recursion and Unbounded Choice*. Number 27 in Cambridge Tracts in Theoretical Computer Science. Cambridge University Press, Cambridge, U.K., 1992.
15. Chris Ho-Stuart. Private communication. 1996.
16. C.A.R. Hoare. An axiomatic basis for computer programming. *Communications of the ACM*, 12(10):576–580, 583, October 1969.

17. C.A.R. Hoare. *Communicating Sequential Processes.* Prentice-Hall International, 1985.
18. C.A.R. Hoare and Jifeng He. *Unifying Theories of Programming.* Prentice-Hall, 1998.
19. C. Jones. Probabilistic nondeterminism. Monograph ECS-LFCS-90-105, Edinburgh University, 1990. (Ph.D. Thesis).
20. C. Jones and G. Plotkin. A probabilistic powerdomain of evaluations. In *Proceedings of the IEEE 4th Annual Symposium on Logic in Computer Science*, pages 186–195, Los Alamitos, Calif., 1989. Computer Society Press.
21. B. Jonsson, C. Ho-Stuart, and W. Yi. Testing and refinement for nondeterministic and probabilistic processes. In Langmaack, de Roever, and Vytopil, editors, *Formal Techniques in Real-Time and Fault-Tolerant Systems*, volume 863 of *LNCS*, pages 418–430. Springer Verlag, 1994.
22. M.B. Josephs. A state-based approach to communicating processes. *Distributed Computing*, 3(1):9–18, December 1988.
23. D. Kozen. Semantics of probabilistic programs. *Jnl. Comp. Sys. Sciences*, 22:328–350, 1981.
24. D. Kozen. A probabilistic PDL. *Jnl. Comp. Sys. Sciences*, 30(2):162–178, 1985.
25. G. Lowe. Probabilities and priorities in timed CSP. Technical Monograph PRG-111, Oxford University Computing Laboratory, 1993. (DPhil Thesis).
26. G. Lowe. Representing nondeterministic and probabilistic behaviour in reactive processes.
`web.comlab.ox.ac.uk/oucl/work/gavin.lowe/Papers/prob.html`, 1993.
27. A.K McIver and C.C. Morgan. Games, probability and the quantitative $\mu$-calculus qMu. In *Proc. LPAR*, volume 2514 of *LNAI*, pages 292–310. Springer-Verlag, 2002. Revised and expanded at [28].
28. A.K. McIver and C.C. Morgan. Results on the quantitative $\mu$-calculus $qM\mu$. *ACM Trans. Comp. Logic*, provisionally accepted, 2004.
29. A.K. McIver, C.C. Morgan, and J.W. Sanders. Probably Hoare? Hoare probably! In Davies et al. [8], pages 271–282.
30. A.K. McIver, C.C. Morgan, J.W. Sanders, and K. Seidel. Probabilistic Systems Group: Collected reports.
`web.comlab.ox.ac.uk/oucl/research/areas/probs/bibliography.html`.
31. Annabelle McIver and Carroll Morgan. *Abstraction, Refinement and Proof for Probabilistic Systems.* Technical Monographs in Computer Science. Springer Verlag, 2004.
32. M. Mislove. Nondeterminism and probabilistic choice: Obeying the laws.
`math.tulane.edu/~mwm/`.
33. C.C. Morgan. The specification statement. *ACM Transactions on Programming Languages and Systems*, 10(3):403–419, July 1988. Reprinted in [39].
34. C.C. Morgan. Of wp and CSP. In W.H.G. Feijen, A.J.M. van Gasteren, D. Gries, and J. Misra, editors, *Beauty is Our Business.* Springer Verlag, 1990.
35. C.C. Morgan and A.K. McIver. *pGCL*: Formal reasoning for random algorithms. *South African Computer Journal*, 22, March 1999. Available at [30–key pGCL].
36. C.C. Morgan, A.K. McIver, and K. Seidel. Probabilistic predicate transformers. *ACM Transactions on Programming Languages and Systems*, 18(3):325–353, May 1996. `doi.acm.org/10.1145/229542.229547`.
37. C.C. Morgan, A.K. McIver, K. Seidel, and J.W. Sanders. Refinement-oriented probability for CSP. *Formal Aspects of Computing*, 8(6):617–647, 1996.

38. C.C. Morgan and A.K. McIver. Cost analysis of games using program logic. Proc. 8th Asia-Pacific Software Engineering Conference (APSEC 2001), December 2001. Abstract only: full text available at [30–key MDP01].
39. C.C. Morgan and T.N. Vickers, editors. *On the Refinement Calculus*. FACIT Series in Computer Science. Springer Verlag, Berlin, 1994.
40. J.M. Morris. A theoretical basis for stepwise refinement and the programming calculus. *Science of Computer Programming*, 9(3):287–306, December 1987.
41. G. Nelson. A generalization of Dijkstra's calculus. *ACM Transactions on Programming Languages and Systems*, 11(4):517–561, October 1989.
42. Amir Pnueli. Linear and branching structures in the semantics and logics of reactive systems. In *Proc. 12th Colloq. on Automata, Languages and Programming*, pages 15–32. Springer Verlag, 1985.
43. A.W. Roscoe, G.M. Reed, and R. Forster. The successes and failures of behavioural models. In Davies et al. [8].
44. K. Seidel. Probabilistic communicating processes. Technical Monograph PRG-102, Oxford University, 1992. (DPhil Thesis).
45. J.C.P. Woodcock and C.C. Morgan. Refinement of state-based concurrent systems. In *Proc. VDM-90*, volume 428 of *LNCS*, 1990.

## A Probabilistic Program Logic for *pAS*

The sequential probabilistic/demonic relational semantics for *pAS* that we gave in Sec. 4 is based on the work of Jifeng He and his colleagues [13]—however we have simplified the presentation here by omitting the closure conditions they defined for their relations and, for reasons we explained in Sec. 5, we have taken an angelic (maximum) rather than their demonic (minimum) view.

Noting that our explicitly given random variables are all standard (*i.e.* they are characteristic functions of some enabling predicate of an action), we restrict *all* the random variables we use to the real interval $[0,1]$ throughout. This is possible due to the probabilistic *feasibility* [31–Def. 1.6.2] that is the quantitative version of Dijkstra's *Law of the Excluded Miracle* [10–Property 1 p. 18]. In our case, it says that the random variables we generate via $\overline{\text{Exp}}.[\![\cdot]\!]$ are pointwise dominated by the characteristic functions we started with.

Then we define the dual

$$\underline{\text{Exp}}.[\![prog]\!].B.s \quad = \quad 1 - \overline{\text{Exp}}.[\![prog]\!].(1-B).s \ , \tag{12}$$

made possible by the fact that, by the remarks above, we can assume $B \leq 1$. The 1-bounded demonic behaviour defined by $\underline{\text{Exp}}.[\![\cdot]\!]$, including miracle-producing guards, is isomorphic to our *Lamington model* [38], obtained by extending our original demonic/probabilistic but miracle-free model [36] with a miraculous command **magic** satisfying $\underline{\text{Exp}}.[\![\textbf{magic}]\!].B.s = 1$ for all $B$ and $s$.

From the Lamington semantics for *pGCL* we can induce a sequential $\overline{\text{Exp}}.[\![\cdot]\!]$-style semantics for the commands of our *pAS*, as in Fig. 7, and we note crucially that it includes the sequential-composition property appealed to at (6) in Sec. 5 above. That is, it need not be proved from the relational semantics directly—duality has given it to us for free.

$$
\begin{array}{rcl}
\overline{\mathrm{Exp}}.[\![\mathbf{abort}]\!].B & \widehat{=} & 1 \\
\overline{\mathrm{Exp}}.[\![\mathbf{skip}]\!].B & \widehat{=} & B \\
\overline{\mathrm{Exp}}.[\![\mathsf{n} := \mathit{expr}]\!].B & \widehat{=} & B^{\mathit{expr}}_{\mathsf{n}} \quad {}^{14} \\
\overline{\mathrm{Exp}}.[\![G \to \mathit{prog}]\!].B & \widehat{=} & [G] * \overline{\mathrm{Exp}}.[\![\mathit{prog}]\!].B \\
\overline{\mathrm{Exp}}.[\![\mathit{prog};\ \mathit{prog}']\!].B & \widehat{=} & \overline{\mathrm{Exp}}.[\![\mathit{prog}]\!].(\overline{\mathrm{Exp}}.[\![\mathit{prog}']\!].B) \\
\overline{\mathrm{Exp}}.[\![\mathit{prog} \sqcap \mathit{prog}']\!].B & \widehat{=} & \overline{\mathrm{Exp}}.[\![\mathit{prog}]\!].B \quad \mathbf{max} \quad \overline{\mathrm{Exp}}.[\![\mathit{prog}']\!].B \\
\overline{\mathrm{Exp}}.[\![\mathit{prog}\ {}_p\!\oplus\ \mathit{prog}']\!].B & \widehat{=} & p * \overline{\mathrm{Exp}}.[\![\mathit{prog}]\!].B \ + \ (1-p) * \overline{\mathrm{Exp}}.[\![\mathit{prog}']\!].B\ .
\end{array}
$$

**Fig. 7.** Structurally inductive definition of $\overline{\mathrm{Exp}}.[\![\cdot]\!]$ for *pAS*

### A.1 Super-Disjunctivity for $\overline{\mathrm{Exp}}.[\![\cdot]\!]$

A second spinoff of duality relates to the algebra of $\overline{\mathrm{Exp}}.[\![\cdot]\!]$.

The Lamington transformers, with their **magic**, do not satisfy the sublinearity property of our original demonic/probabilistic transformers—for example, **magic** itself is clearly not scaling. Nevertheless they do satisfy *sub-conjunctivity*, that is that for all programs *prog* and $[0,1]$-valued random variables $B, B'$ we have

$$
\underline{\mathrm{Exp}}.[\![\mathit{prog}]\!].B \quad \& \quad \underline{\mathrm{Exp}}.[\![\mathit{prog}]\!].B' \quad \leq \quad \underline{\mathrm{Exp}}.[\![\mathit{prog}]\!].(B \,\&\, B')\ , \tag{13}
$$

where for $0 \leq x, y \leq 1$ we define $x \,\&\, y \mathrel{\widehat{=}} (x + y - 1)\,\mathbf{max}\,0$. From the duality (12) we then have immediately that

$$
\overline{\mathrm{Exp}}.[\![\mathit{prog}]\!].B \quad [\!] \quad \overline{\mathrm{Exp}}.[\![\mathit{prog}]\!].B' \quad \geq \quad \overline{\mathrm{Exp}}.[\![\mathit{prog}]\!].(B \,[\!]\, B')\ , \tag{14}
$$

where the duality has induced a definition $x \,[\!]\, y \mathrel{\widehat{=}} (x+y)\,\mathbf{min}\,1$ of a "probabilistic disjunction". We call this *super-disjunctivity*.

This important inequality—which is fully general, applying even when *prog* is both probabilistic and angelic[15]—can be used for example to prove the healthiness condition **pC3** for probabilistic action systems that we gave in Sec. 8. Thus we have for trace $\mathsf{es}$, event $\mathsf{e}$ and set of events $\mathsf{E}$ the calculation

$$
\begin{array}{lll}
 & \mathrm{pFail}.(\mathsf{es} +\!\!+ \langle \mathsf{e} \rangle, \emptyset) \ + \ \mathrm{pFail}.(\mathsf{es}, \mathsf{E} \cup \{\mathsf{e}\}) & \\
\geq & \mathrm{pFail}.(\mathsf{es} +\!\!+ \langle \mathsf{e} \rangle, \emptyset) \ [\!] \ \mathrm{pFail}.(\mathsf{es}, \mathsf{E} \cup \{\mathsf{e}\}) & \text{arithmetic} \\
= & \langle\!\langle \mathsf{es} +\!\!+ \langle \mathsf{e} \rangle \rangle\!\rangle.(\neg\emptyset) \ [\!] \ \langle\!\langle \mathsf{es} \rangle\!\rangle.(\neg(\mathsf{E} \cup \{\mathsf{e}\})) & \text{definition pFail at (10)} \\
= & \langle\!\langle \mathsf{es} \rangle\!\rangle.\{\mathsf{e}\} \ [\!] \ \langle\!\langle \mathsf{es} \rangle\!\rangle.(\neg(\mathsf{E} \cup \{\mathsf{e}\})) & \text{sequential composition} \\
\geq & \langle\!\langle \mathsf{es} \rangle\!\rangle.(\ \{\mathsf{e}\} \ [\!] \ (\neg\mathsf{E} \cup \{\mathsf{e}\})\ ) & \text{definition } \langle\!\langle \mathsf{es} \rangle\!\rangle. \text{ at (8); Property (14)} \\
= & \langle\!\langle \mathsf{es} \rangle\!\rangle.(\neg(\mathsf{E} - \{\mathsf{e}\})) & \text{set algebra} \\
= & \mathrm{pFail}.(\mathsf{es},\ \mathsf{E} - \{\mathsf{e}\}) & \text{definition pFail} \\
\geq & \mathrm{pFail}.(\mathsf{es},\ \mathsf{E})\ . & \text{Condition } \mathbf{pC2}
\end{array}
$$

[14] By $B^{\mathit{expr}}_{\mathsf{n}}$ we mean syntactic replacement of $\mathsf{n}$ by *expr* in $B$, respecting bound variables.

[15] Neither sub-conjunctivity nor super-disjunctivity applies however if the probabilistic programs are *both* demonic and angelic.

## B Complete Traces *etc.* for System $\mathcal{X}_4$ of Fig. 5

| Trace | Associated maximum probability | |
|---|---|---|
| $\langle\rangle$ | 1 | Empty trace always gives 1. |
| $\langle \mathsf{hic} \rangle$ | 1/2 | Initialisation sets $\mathsf{n}$ to $+1$. |
| $\langle \mathsf{hoc} \rangle$ | 1/2 | Initialisation sets $\mathsf{n}$ to $-1$. |
| $\langle \mathsf{hic}, \mathsf{hic} \rangle$ | 1/2 | Variable $\mathsf{n}$ remains $+1 \ldots$ |
| $\langle \mathsf{hic}, \mathsf{hoc} \rangle$ | 0 | $\ldots$ so that $\mathsf{hoc}$ is never enabled; |
| $\langle \mathsf{hoc}, \mathsf{hic} \rangle$ | 1/2 | but divergence after $\mathsf{hoc} \ldots$ |
| $\langle \mathsf{hoc}, \mathsf{hoc} \rangle$ | 1/2 | $\ldots$ allows anything. |

| | |
|---|---|
| Any non-empty trace comprising only $\mathsf{hic}$'s: | Probability 1/2. |
| Any trace beginning $\mathsf{hic}$ but containing a $\mathsf{hoc}$: | Probability 0. |
| Any trace beginning $\mathsf{hoc}$: | Probability 1/2. |

**Fig. 8.** Complete traces for System $\mathcal{X}_4$ of Fig. 5

| Failure | Associated maximum probability | |
|---|---|---|
| $(\langle\rangle, \{\})$ | 1 | Empty offer is always refused. |
| $(\langle\rangle, \{\mathsf{hoc}\})$ | 1/2 | Initialisation sets $\mathsf{n}$ to $+1$. |
| $(\langle\rangle, \{\mathsf{hic}\})$ | 1/2 | Initialisation sets $\mathsf{n}$ to $-1$. |
| $(\langle\rangle, \{\mathsf{hic}, \mathsf{hoc}\})$ | 0 | Initialisation does not diverge or deadlock. |
| $(\langle \mathsf{hic} \rangle, \{\})$ | 1/2 | Empty offer refused... *if* we get this far. |
| $(\langle \mathsf{hic} \rangle, \{\mathsf{hic}\})$ | 0 | Event $\mathsf{hic}$ must follow $\mathsf{hic} \ldots$ |
| $(\langle \mathsf{hic} \rangle, \{\mathsf{hoc}\})$ | 1/2 | $\ldots$ but $\mathsf{hoc}$ cannot. |
| $(\langle \mathsf{hic} \rangle, \{\mathsf{hic}, \mathsf{hoc}\})$ | 0 | Action *hic* does not diverge or deadlock. |
| $(\langle \mathsf{hoc} \rangle, \mathsf{E})$ | 1/2 | Anything can be refused after divergence, including the entire alphabet. |

| | |
|---|---|
| Any failure whose non-empty trace comprises only $\mathsf{hic}$'s: | *As for trace* $\langle \mathsf{hic} \rangle$. |
| Any failure whose trace begins $\mathsf{hic}$ but contains a $\mathsf{hoc}$: | Probability 0. |
| Any failure whose trace begins $\mathsf{hoc}$, no matter what refusal: | Probability 1/2. |

**Fig. 9.** Complete failures for System $\mathcal{X}_4$ of Fig. 5

| Divergence | Associated maximum probability |
|---|---|
| Any trace beginning $\mathsf{hoc}$ | Probability 1/2. |
| Any other trace | Probability 0. |

**Fig. 10.** Complete divergences for System $\mathcal{X}_4$ of Fig. 5