

# Of probabilistic $wp$ and $CSP$

Carroll Morgan

University of New South Wales

- *Action Systems* (probabilistic)
- Informal translation to  $CSP$  (probabilistic)
- Probabilistic *relational model* (à la Jifeng He)
- Probabilistic traces via *expected values*
- Expected-value algebra and *duality*
- Probabilistic *healthiness conditions*
- *Compositionality?*

Carroll Morgan.

Of  $wp$  and  $CSP$ .

In *Beauty is our Business*, Feijen *et al.*, Springer, 1990.

# A probabilistic Action System

System  $\mathcal{H}$

**var**  $n: \mathbb{Z}$

**initially**  $n := 0 \quad \frac{1}{2} \oplus \pm 1$

**hic**  $\hat{=}$   $n \neq 0 \rightarrow n := 0$

**haec**  $\hat{=}$   $n = 0 \rightarrow n := -1 \quad \frac{1}{3} \oplus 0$

**hoc**  $\hat{=}$   $n < 0 \rightarrow n := \pm 1$

probabilistic choice

demonic choice

R.-J.R. Back and R. Kurki-Suonio.

Decentralisation of process nets with centralised control.

*2nd ACM SIGACT-SIGOPS Symp. PODC*, 131-142, 1983.

# Its *pCSP* “equivalent”

initially  $n := 0 \quad \text{1/2} \oplus \pm 1$

hic  $\hat{=}$   $n \neq 0 \rightarrow n := 0$

haec  $\hat{=}$   $n = 0 \rightarrow n := -1 \quad \text{1/3} \oplus 0$

hoc  $\hat{=}$   $n < 0 \rightarrow n := \pm 1$

## Process $\mathcal{H}$

$$\mathcal{H} \quad \hat{=} \quad \mathcal{H}_0 \quad \text{1/2} \oplus (\mathcal{H}_{-1} \sqcap \mathcal{H}_1)$$

$$\mathcal{H}_1 \quad \hat{=} \quad \text{hic} \rightarrow \mathcal{H}_0$$

$$\mathcal{H}_0 \quad \hat{=} \quad \text{haec} \rightarrow (\mathcal{H}_{-1} \quad \text{1/3} \oplus \mathcal{H}_0)$$

$$\mathcal{H}_{-1} \quad \hat{=} \quad \text{hic} \rightarrow \mathcal{H}_0 \quad \square \quad \text{hoc} \rightarrow (\mathcal{H}_{-1} \sqcap \mathcal{H}_1)$$

# Its traces

$$\{ \langle \rangle, \\ \langle \text{hic} \rangle, \langle \text{haec} \rangle, \langle \text{hoc} \rangle, \\ \langle \text{hic}, \text{haec} \rangle, \\ \langle \text{haec}, \text{hic} \rangle, \\ \langle \text{haec}, \text{haec} \rangle, \\ \langle \text{haec}, \text{hoc} \rangle, \\ \langle \text{hoc}, \text{hic} \rangle, \\ \langle \text{hoc}, \text{hoc} \rangle, \\ \langle \text{hic}, \text{haec}, \text{hic} \rangle, \\ \langle \text{hic}, \text{haec}, \text{haec} \rangle, \\ \langle \text{hic}, \text{haec}, \text{hoc} \rangle, \\ \vdots \\ \}$$

initially  $n := 0 \quad \frac{1}{2} \oplus \pm 1$

$$\begin{array}{l} \text{hic} \hat{=} n \neq 0 \rightarrow n := 0 \\ \text{haec} \hat{=} n = 0 \rightarrow n := -1 \frac{1}{3} \oplus 0 \\ \text{hoc} \hat{=} n < 0 \rightarrow n := \pm 1 \end{array}$$

# The relational model for a $pAS$

- A (*sub-*) *distribution*  $\Delta$  over a state space  $S$  is a function from  $S$  to  $[0, 1]$  such that  $(\sum_{s:S} \Delta.s) \leq 1$ .
- The *set of all such distributions* is  $\bar{S}$ .
- A *point distribution* at  $s$  is written  $\bar{s}$ .
- *Non-demonic* probabilistic programs have type  $S \rightarrow \bar{S}$ .
- *Probabilistic/demonic* programs have type  $S \rightarrow \mathbb{P}\bar{S}$ .
- The *meaning* of a program  $prog$  is written  $\llbracket prog \rrbracket$ .

Jifeng He, Annabelle McIver and Karen Seidel.

Probabilistic models for the guarded command language.

*Science of Computer Programming* 28:171-192, 1997.

# Simple examples

- **identity** —  $\llbracket \text{skip} \rrbracket . n = \{\bar{n}\}$

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** —  $\llbracket n := n + 1 \rrbracket . n = \{\overline{n + 1}\}$

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** —  $\llbracket n := n + 1 \quad \frac{1}{3} \oplus \quad n := n + 2 \rrbracket . n$   
 $= \{\Delta'\}$

$$\begin{aligned} \text{where } \Delta'.(n + 1) &= 1/3 \\ \Delta'.(n + 2) &= 2/3 \\ \Delta'.n' &= 0 \quad \text{for other values } n' \end{aligned}$$

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

# Simple examples

○ **demonic choice** — 
$$\llbracket n := n + 1 \sqcap n := n + 2 \rrbracket . n$$
  

$$= \{\overline{n + 1}, \overline{n + 2}\}$$

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

○ **demonic probabilistic choice** — 
$$\llbracket n := n + 1 \text{ }_{1/3} \oplus_{1/3} \text{ } n := n + 2 \rrbracket . n$$
  

$$= \{\Delta'_{1/3}, \Delta'_{2/3}\}$$
  
 where 
$$\begin{aligned} \Delta'_p.(n + 1) &= p \\ \Delta'_p.(n + 2) &= 1 - p \\ \Delta'_p.n' &= 0 \quad \text{for other values } n' \end{aligned}$$

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.

# Naked guarded commands

- A “naked” guarded command (*i.e.* not “clothed” by an **if...fi**) is executed only if its guard is true; otherwise it “cannot start”.

$$\Delta' \in \llbracket gd \rightarrow prog \rrbracket.s \quad \hat{=} \quad \underline{s \in \llbracket gd \rrbracket} \wedge \Delta' \in \llbracket prog \rrbracket.s$$

- If the guard is false in the current state (here  $s$ ), then the result set of distributions is empty.

# Sequential composition

- First, “lift” functions  $f$  so that they act over whole incoming initial distributions:

$$f^*.\Delta.s' \hat{=} \left( \sum_{s:S} \Delta.s * f.s.s' \right)$$

- Then identify the “deterministic refinements” of a general demonic/probabilistic command:

$$r \sqsubseteq f \hat{=} (\forall s: S \cdot r.s \ni f.s)$$

# Sequential composition

$$\bullet f^*. \Delta . s' \hat{=} \left( \sum_{s:S} \Delta . s * f . s . s' \right)$$

$$\bullet r \sqsubseteq f \hat{=} (\forall s:S \cdot r.s \ni f.s)$$

$r^+$  is a form of “down-closure” of  $r$ , adding the everywhere-zero sub-distribution where  $r$  is empty.

- Then lift the general command by lifting its deterministic components:

$$r^*. \Delta \hat{=} \{ f: S \rightarrow \bar{S} \mid r^+ \sqsubseteq f \cdot f^*. \Delta \}$$

- And, finally, form the sequential composition by applying the second command “lifted” to all outcomes of the first:

$$[[prog_1; prog_2]].s \hat{=} \{ \Delta: [[prog_1]].s \cdot [[prog_2]]^*. \Delta \}$$

# Traces of a *pAS*

System $\mathcal{A}$ :	<b>initially</b> $n := 0$	hic $\hat{=}$ $n \geq 0 \rightarrow n := +1$
		hoc $\hat{=}$ $n \leq 0 \rightarrow n := -1$
System $\mathcal{D}$ :	<b>initially</b> $n := \pm 1$	hic $\hat{=}$ $n \geq 0 \rightarrow n := +1$
		hoc $\hat{=}$ $n \leq 0 \rightarrow n := -1$
System $\mathcal{P}$ :	<b>initially</b> $n := -1 \text{ }_{1/2} \oplus +1$	hic $\hat{=}$ $n \geq 0 \rightarrow n := +1$
		hoc $\hat{=}$ $n \leq 0 \rightarrow n := -1$

All three systems have the same potential traces — but the first is “angelic” (due to external choice), and the second is “demonic” (internal choice).

The third is *probabilistic*, and we look at it more closely...

## Traces of a *pAS*

$$\text{Exp.} \llbracket prog \rrbracket . B . s \quad \hat{=} \quad \left( \sum_{s' : S} \Delta' . s' * B . s' \right) ,$$

given that  $\llbracket prog \rrbracket . s = \{ \Delta' \}$  .

For a *non-demonic* program *prog*, the Exp function gives

the *expected value* of the function *B* over the final distribution produced by *prog* from initial state *s*.

# Traces of a $pAS$

$$\begin{aligned} \text{hic} &\hat{=} n \geq 0 \rightarrow n := +1 \\ \text{hoc} &\hat{=} n \leq 0 \rightarrow n := -1 \end{aligned}$$

We indicate *characteristic functions* with square brackets and, in that context, an action name stands for its guard; thus

$$[\text{hic}] \quad [\text{hic}].n = \begin{array}{l} 1 \text{ if } n \geq 0 \\ 0 \text{ otherwise} \end{array}$$

is the function of the state space  $S$ , that is the integers  $\mathbb{Z}$  here, that is one for non-negative arguments and zero elsewhere.

# Traces of a $pAS$

$$\text{initially } n := -1 \quad \frac{1}{2} \oplus +1 \quad \begin{array}{l} \text{hic} \hat{=} n \geq 0 \rightarrow n := +1 \\ \text{hoc} \hat{=} n \leq 0 \rightarrow n := -1 \end{array}$$

The expected value of a characteristic function is the probability assigned to its underlying set. Thus

$$(\text{Exp.}[[ini]].[hic]).n$$

is the probability that event `hic` will be enabled initially.

# Traces of a *pAS*

The probability that **hic** is initially enabled is

$$\text{Exp}.\llbracket ini \rrbracket.\llbracket hic \rrbracket.n$$

$$\text{initially } n := -1 \oplus_{1/2} +1 \quad \begin{array}{l} \text{hic} \hat{=} n \geq 0 \rightarrow n := +1 \\ \text{hoc} \hat{=} n \leq 0 \rightarrow n := -1 \end{array}$$

But **[hic]** itself is the expected value of the constant, everywhere-one function **[true]** over the distributions produced by the *action* **hic**, provided we take that value to be *zero* when the guard is false since — in that case — there is no final distribution. Thus we have

$$\llbracket hic \rrbracket.n = \text{Exp}.\llbracket hic \rrbracket.\llbracket true \rrbracket.n$$

for the characteristic function of **hic**'s guard.

If it's enabled, its probability of establishing **true** is one; if it's not enabled, its probability of establishing **true** — or indeed anything at all — is zero.

# Traces of a *pAS*

The probability that `hic` is initially enabled is

$$\text{Exp}.\llbracket ini \rrbracket.\llbracket hic \rrbracket.n$$

$$\text{initially } n := -1 \oplus_{1/2} +1 \quad \begin{array}{l} \text{hic} \hat{=} n \geq 0 \rightarrow n := +1 \\ \text{hoc} \hat{=} n \leq 0 \rightarrow n := -1 \end{array}$$

But `[hic]` itself is the expected value of the constant, everywhere-one function `[true]` over the distributions produced by the *action* `hic`, provided we take that value to be *zero* when the guard is false since — in that case — there is no final distribution. Thus we have

$$\llbracket hic \rrbracket.n = \text{Exp}.\llbracket hic \rrbracket.\llbracket true \rrbracket.n$$

for the characteristic function of `hic`'s guard.

If it's enabled, its probability of establishing `true` is one; if it's not enabled, its probability of establishing `true` — or indeed anything at all — is zero.

# Traces of a $pAS$

The probability that **hic** is initially enabled is

$$\text{Exp}.\llbracket ini \rrbracket.\llbracket hic \rrbracket.n$$

$$\text{initially } n := -1 \oplus_{1/2} +1 \quad \begin{array}{l} \text{hic} \hat{=} n \geq 0 \rightarrow n := +1 \\ \text{hoc} \hat{=} n \leq 0 \rightarrow n := -1 \end{array}$$

“The probability of the occurrence of the trace  $\langle \text{hic} \rangle$  from initial state  $n$ ”

$$= \text{Exp}.\llbracket ini \rrbracket.\llbracket hic \rrbracket.n$$

$$= \text{Exp}.\llbracket ini \rrbracket.(\text{Exp}.\llbracket hic \rrbracket.\llbracket true \rrbracket).n$$

$$\text{☂} = \text{Exp}.\llbracket ini; hic \rrbracket.\llbracket true \rrbracket.n .$$


$$\llbracket hic \rrbracket = \text{Exp}.\llbracket hic \rrbracket.\llbracket true \rrbracket$$

# The algebra of the *Exp* functions

- Because of internal choice (demonic nondeterminism), there may be several final distributions produced by a program *prog*.
- Since standard *CSP* includes a trace if it *could* occur, we take the *maximum* probability over all possible distributions, writing

$$\overline{\text{Exp}}.\llbracket prog \rrbracket.B.s .$$

- $\text{Exp}.\llbracket prog \rrbracket$  and  $\overline{\text{Exp}}.\llbracket prog \rrbracket$  are the same if *prog* is non-demonic.

 We have the distribution law

$$\overline{\text{Exp}}.\llbracket prog_1; prog_2 \rrbracket.B.s = \overline{\text{Exp}}.\llbracket prog_1 \rrbracket.(\overline{\text{Exp}}.\llbracket prog_2 \rrbracket.B).s .$$

# The *Exp* duality

- Since standard *CSP* includes a trace if it *could* occur, we take the *maximum* probability over all possible distributions, writing

$$\overline{\text{Exp}}.\llbracket prog \rrbracket.B.s .$$

- But if we took the “normal”, demonic view for sequential programs we would take the *minimum* expected value, writing instead

$$\underline{\text{Exp}}.\llbracket prog \rrbracket.B.s .$$

- The two forms are dual, satisfying

$$\overline{\text{Exp}}.\llbracket prog \rrbracket.B.s = 1 - \underline{\text{Exp}}.\llbracket prog \rrbracket.(1 - B).s .$$

- ☂ And the latter is exactly the *greatest pre-expectation* for sequential, probabilistic, demonic programs.

# The *Exp* duality

26 1. Introduction to  $pGCL$

$$\begin{aligned}
 wp.\mathbf{abort}.postE &:= 0 \\
 wp.\mathbf{skip}.postE &:= postE \\
 wp.(x := expr).postE &:= postE \langle x \mapsto expr \rangle \\
 \img alt="umbrella icon" data-bbox="95 405 135 455"/> wp.(prog; prog').postE &:= wp.prog.(wp.prog'.postE) \\
 wp.(prog \sqcap prog').postE &:= wp.prog.postE \min wp.prog'.postE \\
 wp.(prog_p \oplus prog').postE &:= p * wp.prog.postE + \bar{p} * wp.prog'.postE .
 \end{aligned}$$

Recall that  $\bar{p}$  is the complement of  $p$ .

The expression on the right gives the greatest pre-expectation of  $postE$  with respect to each  $pGCL$  construct, where  $postE$  is an expression of type  $\mathbb{E}S$  over the variables in state space  $S$ . (For historical reasons we continue to write  $wp$  instead of  $gp$ .)

# The *Exp* duality

26 1. Introduction to  $pGCL$

$$\begin{aligned}
 wp.\mathbf{abort}.postE &:= 0 \\
 wp.\mathbf{skip}.postE &:= postE \\
 wp.(x := expr).postE &:= postE \langle x \mapsto expr \rangle \\
 \img alt="umbrella icon" data-bbox="95 405 135 455"/> wp.(prog; prog').postE &:= wp.prog.(wp.prog'.postE) \\
 wp.(prog \sqcap prog').postE &:= wp.prog.postE \min wp.prog'.postE \\
 wp.(prog_p \oplus prog').postE &:= p * wp.prog.postE + \bar{p} * wp.prog'.postE .
 \end{aligned}$$

Recall that  $\bar{p}$  is the complement of  $p$ .

The expression on the right gives the greatest pre-expectation of  $postE$  with respect to each  $pGCL$  construct, where  $postE$  is an expression of type  $\mathbb{E}S$  over the variables in state space  $S$ . (For historical reasons we continue to write  $wp$  instead of  $gp$ .)

C.C. Morgan, A.K. McIver and K. Seidel.

Probabilistic predicate transformers.

*ACM TOPLAS* 18(3):325-353, 1996.

# The *Exp* duality

$$\overline{\text{Exp.}} \llbracket \mathbf{abort} \rrbracket . B \quad \hat{=} \quad 1$$

$$\overline{\text{Exp.}} \llbracket \mathbf{skip} \rrbracket . B \quad \hat{=} \quad B$$

$$\overline{\text{Exp.}} \llbracket n := \text{expr} \rrbracket . B \quad \hat{=} \quad B_n^{\text{expr}}$$

$$\overline{\text{Exp.}} \llbracket G \rightarrow \text{prog} \rrbracket . B \quad \hat{=} \quad [G] * \overline{\text{Exp.}} \llbracket \text{prog} \rrbracket . B$$

$$\overline{\text{Exp.}} \llbracket \text{prog}; \text{prog}' \rrbracket . B \quad \hat{=} \quad \overline{\text{Exp.}} \llbracket \text{prog} \rrbracket . (\overline{\text{Exp.}} \llbracket \text{prog}' \rrbracket . B)$$

$$\overline{\text{Exp.}} \llbracket \text{prog} \sqcap \text{prog}' \rrbracket . B \quad \hat{=} \quad \overline{\text{Exp.}} \llbracket \text{prog} \rrbracket . B$$

$$\max \quad \overline{\text{Exp.}} \llbracket \text{prog}' \rrbracket . B$$

$$\overline{\text{Exp.}} \llbracket \text{prog}_p \oplus \text{prog}' \rrbracket . B \quad \hat{=} \quad p * \overline{\text{Exp.}} \llbracket \text{prog} \rrbracket . B$$

$$+ \quad (1 - p) * \overline{\text{Exp.}} \llbracket \text{prog}' \rrbracket . B$$

26 1. Introduction to *pGCL*

```

wp.abort.postE := 0
wp.skip.postE := postE
wp.(x := expr).postE := postE (x ↦ expr)
wp.(prog; prog').postE := wp.prog.(wp.prog'.postE)
wp.(prog □ prog').postE := wp.prog.postE min wp.prog'.postE
wp.(prog_p ⊕ prog').postE := p * wp.prog.postE + ¬p * wp.prog'.postE.

```

Recall that  $\bar{p}$  is the complement of  $p$ .

The expression on the right gives the *greatest pre-expectation* of *postE* with respect to each *pGCL* construct, where *postE* is an expression of type  $\mathbb{E}S$  over the variables in state space  $S$ . (For historical reasons we continue to write *wp* instead of *gp*.)

# Probabilistic *failures* and *divergences*

- The *failures* of a process can be defined similarly: we have that

$$\overline{\text{Exp.}}.[ini; es].[\neg E].s$$

is the maximum probability that the process can participate in the trace  $es$  and then refuse any event in  $E$ , where  $[\neg E]$  is the characteristic function of the set in which no event of  $E$  is enabled.

- The *divergences* are defined using

$$\overline{\text{Exp.}}.[ini; es].[\text{false}].s ,$$

provided the relational model is (easily) extended to include a bottom “divergent” state  $\perp$ .

- And the *algebra* of  $\overline{\text{Exp.}}.[\cdot]$  allows us to prove probabilistic “healthiness laws” about the way these observations interact...

# Standard healthiness

$$\mathbf{C0} \quad (\langle \rangle, \emptyset) \in F$$

$$\mathbf{C1} \quad (es \ ++ \ es', E) \in F \Rightarrow (es, \emptyset) \in F$$

$$\mathbf{C2} \quad (es, E) \in F \wedge E' \subseteq E \Rightarrow (es, E') \in F$$

$$\mathbf{C3} \quad (es, E) \in F \Rightarrow \begin{array}{l} (es \ ++ \ \langle e \rangle, \emptyset) \in F \\ \vee \\ (es, E \cup \{e\}) \in F \end{array}$$

$$\mathbf{C4} \quad es \in D \Rightarrow es \ ++ \ es' \in D$$

$$\mathbf{C5} \quad es \in D \Rightarrow (es, E) \in F$$

These are the conditions that hold for the failures  $F$  and divergences  $D$  of a standard *CSP* process.

# Probabilistic healthiness

- The probabilistic failure  $\text{pFail.}(es, E)$  is the maximum probability that the action system can perform its initialisation, then execute  $es$ , then enter a state where no actions in  $E$  are enabled.

$$\text{pFail.}(es, E) \hat{=} \overline{\text{Exp.}} \llbracket ini; es \rrbracket. [\neg E]$$

- The traces are derived from the failures in the usual way.

$$\text{pTr.}es \hat{=} \text{pFail.}(es, \emptyset)$$

- Divergence is the only way to “establish” false.

$$\text{pDiv.}es \hat{=} \overline{\text{Exp.}} \llbracket ini; es \rrbracket. [\text{false}]$$

# Probabilistic healthiness

To compare these with the standard conditions **C0** to **C5**, read “ $\leq$ ” as implication.

$$\mathbf{pC0} \text{ --- } \text{pFail}.\langle \rangle, \emptyset = 1$$

It is always possible for a system to start, since its initialisation is unguarded.

$$\mathbf{pC1} \text{ --- } \text{pFail}.\text{es} \text{ ++ } \text{es}', \mathbf{E} \leq \text{pFail}.\text{es}, \emptyset$$

The probability of continuing a trace is no more than the probability of achieving the trace itself.

$$\mathbf{pC2} \text{ --- } \text{pFail}.\text{es}, \mathbf{E} \geq \text{pFail}.\text{es}, \mathbf{E} \cup \mathbf{E}'$$

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

$$\mathbf{pC3} \text{ --- } \text{pFail}.\text{es}, \mathbf{E} \leq \text{pFail}.\text{es} \text{ ++ } \langle \mathbf{e} \rangle, \emptyset + \text{pFail}.\text{es}, \mathbf{E} \cup \{ \mathbf{e} \}$$

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

$$\mathbf{pC4} \text{ --- } \text{pDiv}.\text{es} \leq \text{pDiv}.\text{es} \text{ ++ } \text{es}'$$

Any event is possible after divergence.

$$\mathbf{pC5} \text{ --- } \text{pDiv}.\text{es} \leq \text{pFail}.\text{es}, \mathbf{E}$$

Any refusal is possible after divergence.

# Probabilistic healthiness

$$\begin{aligned}
 & \geq \frac{\text{pFail}.\langle es \ ++ \ \langle e \rangle, \emptyset \rangle + \text{pFail}.\langle es, E \cup \{e\} \rangle}{\text{pFail}.\langle es \ ++ \ \langle e \rangle, \emptyset \rangle \sqcup \text{pFail}.\langle es, E \cup \{e\} \rangle} && \text{arithmetic} \\
 & = \frac{\overline{\text{Exp}}.\langle ini; es; e \rangle.\langle \text{true} \rangle \sqcup \overline{\text{Exp}}.\langle ini; es \rangle.\langle \neg(E \cup \{e\}) \rangle}{\overline{\text{Exp}}.\langle ini; es \rangle.\langle e \rangle \sqcup \overline{\text{Exp}}.\langle ini; es \rangle.\langle \neg(E \cup \{e\}) \rangle} && \begin{array}{l} \text{definition pFail} \\ \text{sequential composition} \end{array} \\
 & \geq \frac{\overline{\text{Exp}}.\langle ini; es \rangle.\langle [e] \sqcup [\neg(E \cup \{e\})] \rangle}{\overline{\text{Exp}}.\langle ini; es \rangle.\langle \neg E \rangle} && \begin{array}{l} \text{super-disjunctivity} \\ \text{set algebra; monotonicity} \end{array} \\
 & = \text{pFail}.\langle es, E \rangle && \text{definition pFail}
 \end{aligned}$$

# The algebra of *Exp* (reprise)

- Sub-conjunctivity (from probabilistic *wp*):

$$\underline{\text{Exp}}.\llbracket prog \rrbracket.B \quad \& \quad \underline{\text{Exp}}.\llbracket prog \rrbracket.B' \quad \leq \quad \underline{\text{Exp}}.\llbracket prog \rrbracket.(B \& B')$$

- Super-disjunctivity (from the above, via duality):

$$\overline{\text{Exp}}.\llbracket prog \rrbracket.B \quad \sqcup \quad \overline{\text{Exp}}.\llbracket prog \rrbracket.B' \quad \geq \quad \overline{\text{Exp}}.\llbracket prog \rrbracket.(B \sqcup B')$$

- Probabilistic conjunction

$$x \& y \quad \hat{=} \quad (x + y - 1) \mathbf{max} \ 0$$

- and its dual, probabilistic disjunction.

$$x \sqcup y \quad \hat{=} \quad (x + y) \mathbf{min} \ 1$$

# Conclusions *re* compositionality

- The probabilistic traces, failures and divergences are *observations* of a probabilistic process, but seem to be too weak to be compositional.
- This is as expected, since the same occurs in sequential programming, but...
- We won't know exactly where the difficulties are until the action-system operations themselves have been defined.
- Of those, parallel composition and hiding seem to be particularly tricky, because of the interaction between probabilistic- and internal choice.

Gavin Lowe.

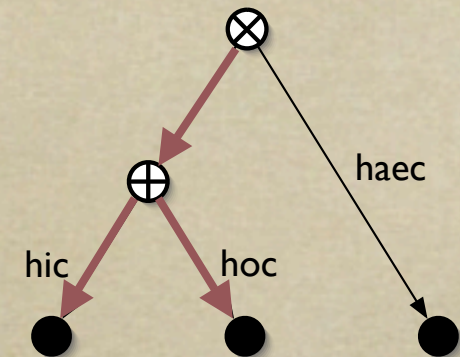
Representing nondeterministic and probabilistic behaviour in reactive processes.

[web.comlab.ox.ac.uk/oucl/work/gavin.lowe/papers/prob.html](http://web.comlab.ox.ac.uk/oucl/work/gavin.lowe/papers/prob.html), 1993.

# Conclusions *re* compositionality

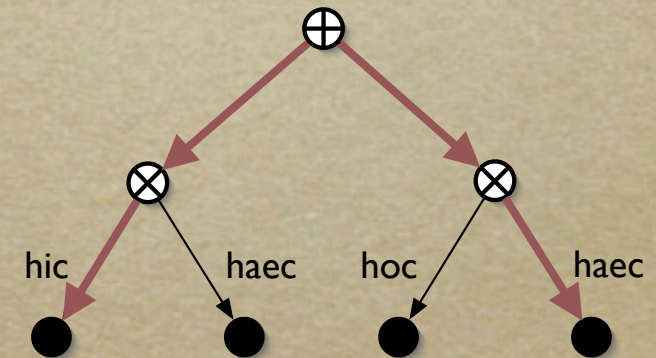
- This process treats **hic** and **hoc** without bias, however much it might favour **haec** over the two together:

$$\begin{aligned} & (\text{hic} \rightarrow \text{STOP} \quad \text{hoc} \rightarrow \text{STOP}) \\ \sqcap & \quad \text{haec} \rightarrow \text{STOP} \end{aligned}$$



- But this process might for example execute **hic** half the time and **hoc** not at all:

$$\begin{aligned} & (\text{hic} \rightarrow \text{STOP} \quad \sqcap \quad \text{haec} \rightarrow \text{STOP}) \\ \text{1/2} \oplus & \quad (\text{hoc} \rightarrow \text{STOP} \quad \sqcap \quad \text{haec} \rightarrow \text{STOP}) \end{aligned}$$



- ☹ Yet their pFail observations are identical.

A.K. McIver, C.C. Morgan and J.W. Sanders. Probably Hoare? Hoare Probably!

In *Millennial Perspectives in Computer Science*. Davies, Roscoe, Woodcock (eds.), pp 271-282.

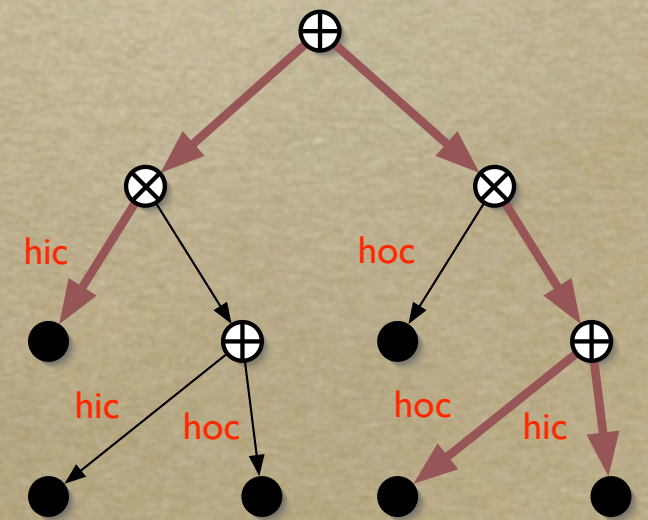
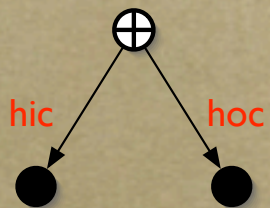
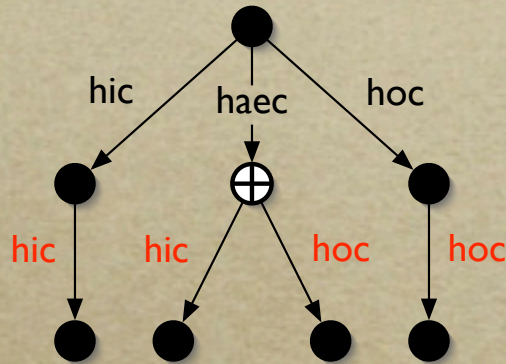
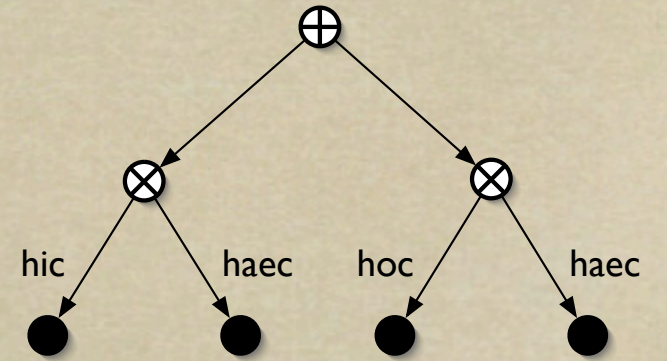
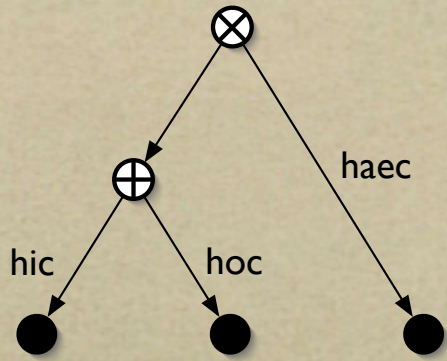
Palgrave, 2000

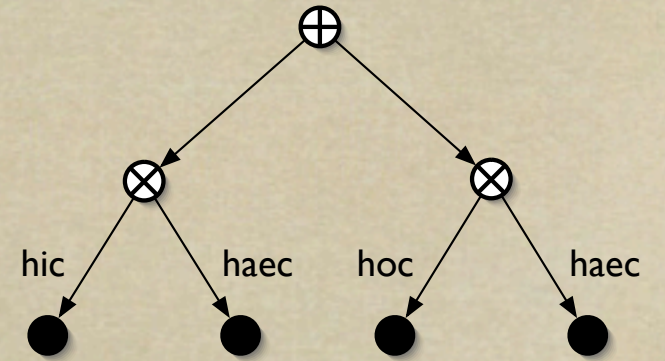
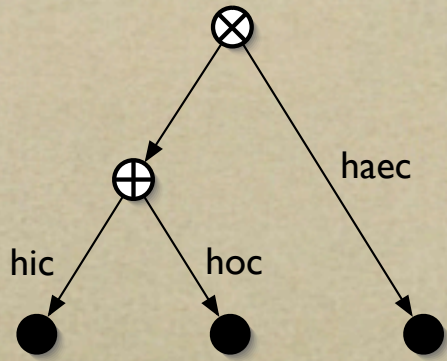
# Conclusions *re* compositionality

- One could extend the `pFail` and `pDiv` operations to act on full *real-valued* expectations rather than only sets of events' guards.

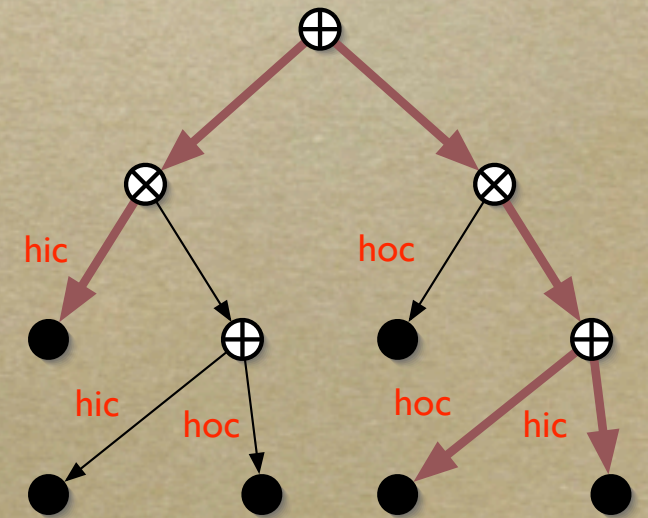
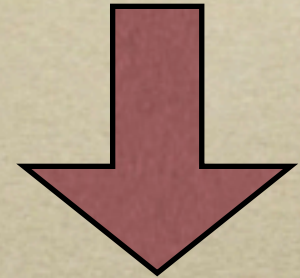
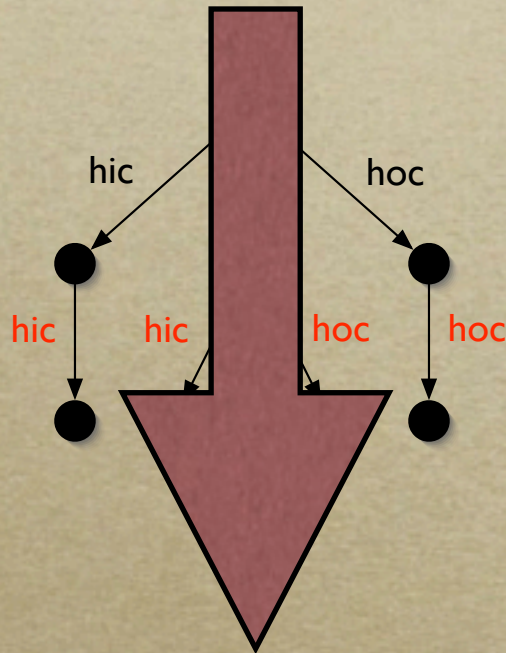
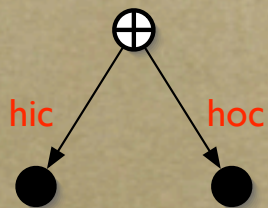
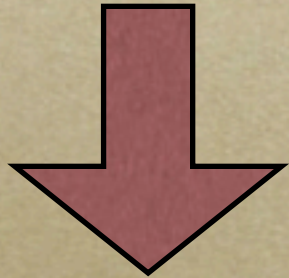
- 😊 This recovers compositionality for sequential programming but, for concurrency, ...

- ❓ It might be necessary to extend further, to allow more elaborate “tree-like” *and quantitative* testing observations, especially for hiding.

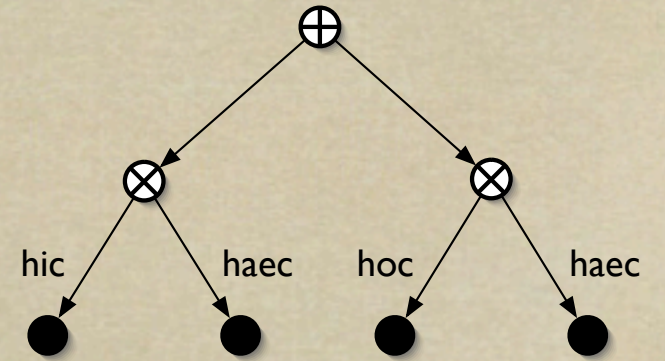
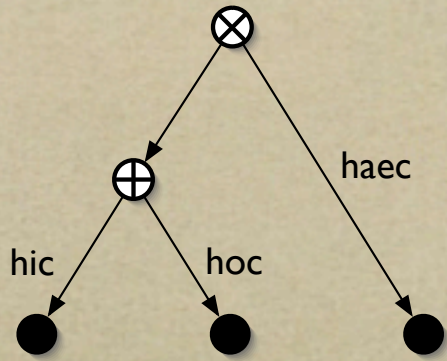




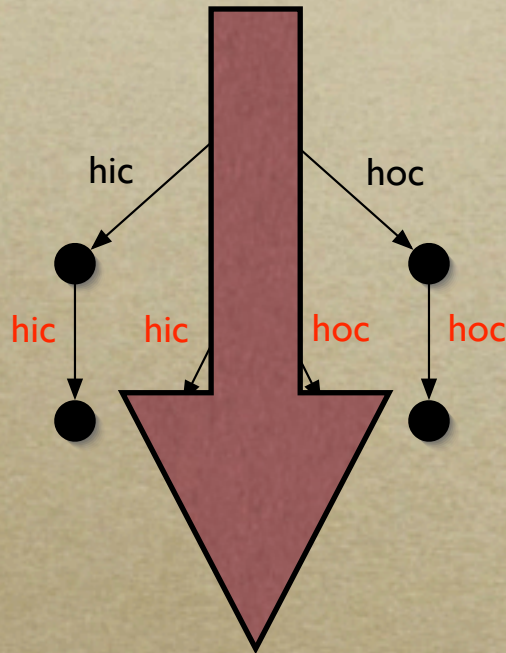
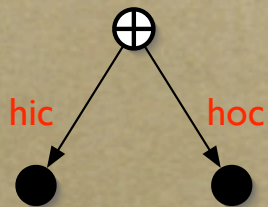
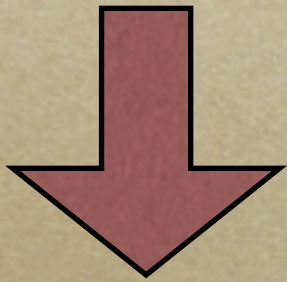
*Parallel composition...*



*...then hide black events.*



*Parallel composition...*



*...then hide black events.*

