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.

Of probabilistic wp and CSP
Carroll Morgan.

Its pCSP “equivalent”

\[
\text{initially } n := 0 \dfrac{1}{2} \oplus 1
\]

\[
\begin{align*}
\text{hic } &\triangleq n \neq 0 \rightarrow n := 0 \\
\text{haec } &\triangleq n = 0 \rightarrow n := -1 \dfrac{1}{3} \oplus 0 \\
\text{hoc } &\triangleq n < 0 \rightarrow n := \pm 1
\end{align*}
\]

Process \( \mathcal{H} \)

\[
\begin{align*}
\mathcal{H} &\triangleq \mathcal{H}_0 \dfrac{1}{2} \oplus (\mathcal{H}_1 \sqcap \mathcal{H}_2) \\
\mathcal{H}_0 &\triangleq \text{hic } \rightarrow \mathcal{H}_0 \\
\mathcal{H}_1 &\triangleq \text{haec } \rightarrow (\mathcal{H}_1 \dfrac{1}{3} \oplus \mathcal{H}_0) \\
\mathcal{H}_2 &\triangleq \text{hic } \rightarrow \mathcal{H}_0 \sqcap \text{hoc } \rightarrow (\mathcal{H}_1 \sqcap \mathcal{H}_0)
\end{align*}
\]

He; Josephs; Woodcock...and M.J. Butler. A CSP Approach to Action Systems.

A probabilistic Action System

System \( \mathcal{H} \)

\[
\begin{align*}
\text{var } n &\in \mathbb{Z} \\
\text{initially } n &:= 0 \dfrac{1}{2} \oplus 1
\end{align*}
\]

\[
\begin{align*}
\text{hic } &\triangleq n \neq 0 \rightarrow n := 0 \\
\text{haec } &\triangleq n = 0 \rightarrow n := -1 \dfrac{1}{3} \oplus 0 \\
\text{hoc } &\triangleq n < 0 \rightarrow n := \pm 1
\end{align*}
\]

Its traces

\[
\{ (\text{hic}), (\text{haec}), (\text{hoc}), \\
(\text{hic}, \text{haec}), \\
(\text{hic}, \text{haec}, \text{haec}), \\
(\text{hic}, \text{haec}, \text{hoc}), \\
(\text{hic}, \text{haec}, \text{haec}, \text{haec}), \\
\vdots \}
\]

R.-J.R. Back and R. Kurki-Suonio.
Decentralisation of process nets with centralised control.
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 \( \sum_{s \in S} \Delta.s \leq 1 \).
- The set of all such distributions is \( \mathbb{F}S \).
- A point distribution at \( s \) is written \( \bar{s} \).
- Non-demonic probabilistic programs have type \( S \rightarrow \mathbb{F}S \).
- Probabilistic/demonic programs have type \( S \rightarrow \mathbb{F}S \).
- The meaning of a program \( \text{prog} \) is written \( [\text{prog}] \).

Jilong He, Annabelle McIver and Karen Seidel.
Probabilistic models for the guarded command language.

Simple examples

- identity — \([\text{skip}] \) \( n = \{ \emptyset \} \)
  The “do-nothing” program \( \text{skip} \) 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 — \([n := n + 1] \) \( n = \{ n + 1 \} \)
  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) pro-
  babilistic choice.
- probabilistic choice — \([n := n + 1] 1/3 : n = n + 2] \) \( n \)
  where \( \Delta.(n + 1) = 1/3 \)
  \( \Delta.(n + 2) = 2/3 \)
  \( \Delta.n' = 0 \) for other values \( n' \)
  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.

Naked guarded commands

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

  \( \Delta' \in [\text{gd } \rightarrow \text{prog}].s \; \iff \; s \in [\text{gd} ] \land \Delta' \in [\text{prog}].s \)

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

Miracles of Morris, Nelson, Morgan, late 80's; used e.g. in Event-B and elsewhere.
Traces of a pAS

System A: Initially n := 0
bic := n ≥ 0 → n := +1
hoc := n ≤ 0 → n := −1

System D: Initially n := ±1
bic := n ≥ 0 → n := +1
hoc := n ≤ 0 → n := −1

System P: Initially n := −1/2ω + 1
bic := n ≥ 0 → n := +1
hoc := n ≤ 0 → 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...

Sequential composition

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

\[ f^\ast \Delta.s' = \left( \sum_{s:S} \Delta.s * f.s.s' \right) \]

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

\[ r \subseteq f \triangleq \left( \forall s:S \cdot r.s \ni f.s \right) \]

Sequential composition

- \( f^\ast \Delta.s' \triangleq \left( \sum_{s:S} \Delta.s * f.s.s' \right) \) \( r^+ \) 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:

\[ r^\ast \Delta \triangleq \{ f:S \rightarrow S | r^+ \subseteq f \cdot f^\ast \Delta \} \]

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

\[ [prog_1; prog_2].s \triangleq \{ \Delta : [prog_1].s \cdot [prog_2]^\ast \Delta \} \]

Traces of a pAS

\[ \text{Exp}.[prog].B.s \triangleq \left( \sum_{s':S} \Delta'.s' * B.s' \right) \]

given that \([prog].s = \{ \Delta' \} \).

For a non-demonic program prog, the \text{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

\[ \text{hic} \triangleq n \geq 0 \rightarrow n := +1 \]
\[ \text{hoc} \triangleq n \leq 0 \rightarrow n := -1 \]

We indicate characteristic functions with square brackets and, in that context, an action name stands for its guard; thus
\[ [\text{hic}] \cdot n = \begin{cases} 1 & \text{if } n \geq 0 \\ 0 & \text{otherwise} \end{cases} \]

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.

The expected value of a characteristic function is the probability assigned to its underlying set. Thus
\[ (\text{Exp.}[\text{ini}],[\text{hic}]) \cdot n \]

is the probability that event hic will be enabled initially.

Traces of a pAS

The probability that hic is initially enabled is
\[ \text{Exp.}[\text{ini}],[\text{hic}].n \]

initially \( n = -1 \%
\[ \text{hic} \triangleq n \geq 0 \rightarrow n := +1 \]
\[ \text{hoc} \triangleq n \leq 0 \rightarrow n := -1 \]

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
\[ [\text{hic}] \cdot \mathds{1} = \text{Exp.}[\text{hic}],[\text{true}] \cdot \mathds{1} \]

for the characteristic function of hic’s guard.

Traces of a pAS

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
\[ [\text{hic}] \cdot n = \text{Exp.}[\text{hic}],[\text{true}] \cdot n \]

for the characteristic function of hic’s guard.
The Exp duality

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

\[ \text{Exp.}[\text{prog}].B.s . \]

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

\[ \text{Exp.}[\text{prog}].B.s . \]

The two forms are dual, satisfying

\[ \text{Exp.}[\text{prog}].B.s = 1 - \text{Exp.}[\text{prog}] .(1 - B).s . \]

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

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

\[ \text{Exp.}[\text{prog}].B.s . \]

\[ \text{Exp.}[\text{prog}] \text{ and } \text{Exp.}[\text{prog}] \text{ are the same if prog is non-demonic.} \]

We have the distribution law

\[ \text{Exp.}[\text{prog}_1; \text{prog}_2].B.s = \text{Exp.}[\text{prog}_1] (\text{Exp.}[\text{prog}_2].B).s . \]

The Exp duality

1. Introduction to pGCL

\[ \text{wp.abort}.postE := 0 \]
\[ \text{wp.skip}.postE := postE \]
\[ \text{wp.(x := expr)}.postE := postE \ (x \leftarrow \text{expr}) \]
\[ \text{wp.(prog; prog').postE} := \text{wp.prog}.(\text{wp.prog'.postE}) \]
\[ \text{wp.(prog \& prog').postE} := \text{wp.prog}.postE \ \text{min} \ \text{wp.prog'.postE} \]
\[ \text{wp.(prog \&\& prog').postE} := p \cdot \text{wp.prog}.postE + \overline{p} \cdot \text{wp.prog'.postE} . \]

Recall that \( \overline{p} \) is the complement of \( p \).

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

1. Introduction to μGCL

\[
\begin{align*}
\text{wp.\text{abort}, post}_E & := 0 \\
\text{wp.\text{skip}, post}_E & := post_E \\
\text{wp.} (x:= \text{expr}) \cdot post_E & := post_E \{ x \mapsto \text{expr} \} \\
\text{wp.} (p \cdot \text{prog} \cdot \text{prog'}) \cdot post_E & := \text{wp.} (\text{wp.} \cdot \text{prog} \cdot \text{prog'}, post_E) \\
\text{wp.} (\text{prog} \cup \text{prog'}) \cdot post_E & := p \cdot \text{wp.} \cdot \text{prog, post}_E + \overline{p} \cdot \text{wp.} \cdot \text{prog'}, post_E.
\end{align*}
\]

Recall that \( \overline{p} \) is the complement of \( p \).

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

---

Probabilistic failures and divergences

- The failures of a process can be defined similarly: we have that
  \[
  \begin{align*}
  \text{Exp}_p [\text{init}; \text{es}] \cdot \neg \text{E}.s 
  \end{align*}
  \]
  is the maximum probability that the process can participate in the trace \( \text{es} \) and then refuse any event in \( \text{E} \), where \( \neg \text{E} \) is the characteristic function of the set in which no event of \( \text{E} \) is enabled.

- The divergences are defined using
  \[
  \begin{align*}
  \text{Exp}_p [\text{init}; \text{es}].[\text{false}].s 
  \end{align*}
  \]
  provided the relational model is (easily) extended to include a bottom “divergent” state \( \perp \).

- And the algebra of \( \text{Exp}_p [\cdot] \) allows us to prove probabilistic “healthiness laws” about the way these observations interact…

---

Standard healthiness

\begin{align*}
\text{C0} & \quad (\emptyset, \emptyset) \in F \\
\text{C1} & \quad (\text{es} \cdot \text{es}', \text{E}) \in F \Rightarrow (\text{es}, \emptyset) \in F \\
\text{C2} & \quad (\text{es}, \text{E}) \in F \land \text{E}' \subseteq \text{E} \Rightarrow (\text{es}, \text{E}') \in F \\
\text{C3} & \quad (\text{es}, \text{E}) \in F \Rightarrow (\text{es} \cdot \langle e \rangle, \emptyset) \in F \\
\text{C4} & \quad \text{es} \in D \Rightarrow \text{es} \cdot \text{es}' \in D \\
\text{C5} & \quad \text{es} \in D \Rightarrow (\text{es}, \text{E}) \in F
\end{align*}

These are the conditions that hold for the failures \( F \) and divergences \( D \) of a standard CSP process.
Probabilistic healthiness

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

$$p\text{Fail}.(\text{es}, E) \equiv \text{Exp.}[\text{ini}; \text{es}].[\neg E]$$

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

$$p\text{Tr}_.\text{es} \equiv p\text{Fail}.(\text{es}, \emptyset)$$

- Divergence is the only way to "establish" $false$.

$$p\text{Div}_.\text{es} \equiv \text{Exp.}[\text{ini}; \text{es}].[false]$$

The algebra of $\text{Exp}$ (reprise)

- Sub-conjunctivity (from probabilistic wp):

$$\text{Exp.}[\text{prog}].B \& \text{Exp.}[\text{prog}].B' \leq \text{Exp.}[\text{prog}].(B \& B')$$

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

$$\text{Exp.}[\text{prog}].B \parallel \text{Exp.}[\text{prog}].B' \geq \text{Exp.}[\text{prog}].(B \parallel B')$$

- Probabilistic conjunction

$$x \& y \equiv (x + y - 1) \max 0$$

- and its dual, probabilistic disjunction.

$$x \parallel y \equiv (x + y) \min 1$$
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.

Conclusions re compositionality

- This process treats hic and hoc without bias, however much it might favour haec over the two together:
  \[ \text{(hic } \rightarrow \text{ STOP } \frac{1}{2} \oplus \text{ hoc } \rightarrow \text{ STOP}) \]
  \[ \square \text{ haec } \rightarrow \text{ STOP} \]

- But this process might for example execute hic half the time and hoc not at all:
  \[ \text{(hic } \rightarrow \text{ STOP } \frac{1}{2}\oplus \text{ haec } \rightarrow \text{ STOP}) \]
  \[ \text{(hoc } \rightarrow \text{ STOP } \frac{1}{2}\oplus \text{ haec } \rightarrow \text{ STOP}) \]

Yet their pFail observations are identical.


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.