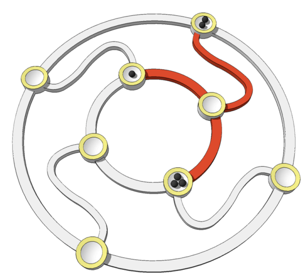
Nondeterminism and probability: A tour of probabilistic programming, from abstraction to the “refinement paradox”.



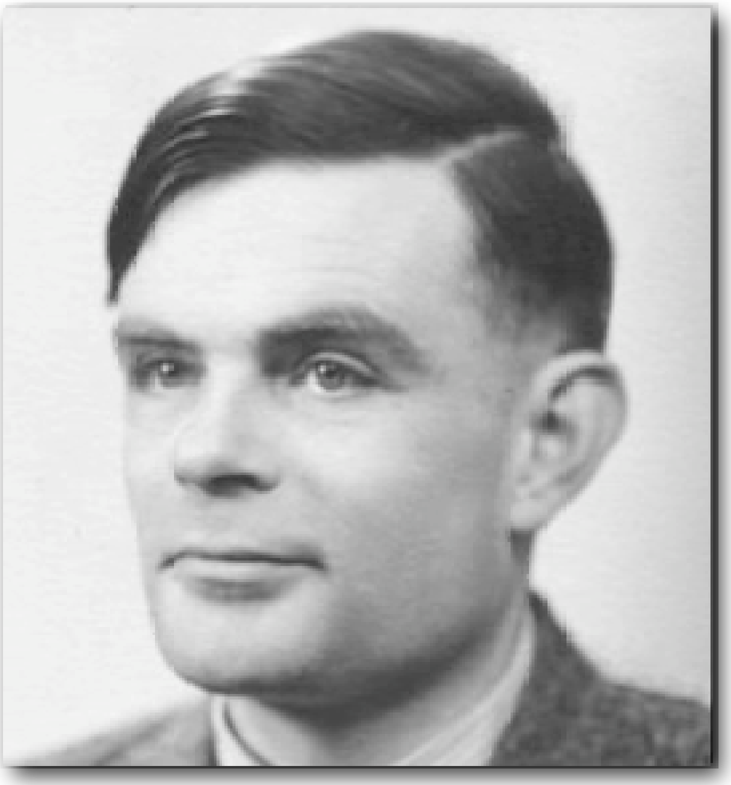
Annabelle McIver,
Macquarie University,
Sydney

Quick summary

- A short history of probabilistic programming;
- How to build the semantics you want in four easy stages;
- A cute application;
- The refinement paradox, and how probability can help shed light.

A brief History of Computing

A brief History of Computing



From working-out the theory...

to working, in theory.

What we want.

Computations
Probability
Nondeterminism
(Refinement;
abstraction)

→

Program algebra;
Behavioural model;
Program logic;
Proof techniques;
Compositionality;
Tools.

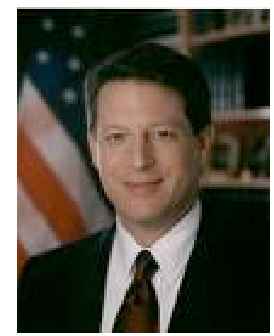
But what does this all mean?
How do these things interact?

In particular what does it mean for refinement of probability: intuitively you might think that probability should increase up the refinement order....

$$\text{Good} \ {}_{1/3}\!\oplus \ \text{Bad} \qquad \sqsubseteq \qquad \text{Good} \ {}_{1/2}\!\oplus \ \text{Bad}$$

In particular what does it mean for refinement of probability: intuitively you might think that probability should increase up the refinement order....

Good $_{1/3}\oplus$ Bad $\sqsubseteq$ Good $_{1/2}\oplus$ Bad

 $_{1/3}\oplus$  $\qquad$  $_{1/2}\oplus$ 

We're going to look for inspiration from a general mathematical construction called powerdomains.

…. but now if we reinstantiate “Good” and “Bad”, shouldn’t we also have this?

$$\text{Good} \;{}_{2/3}\!\oplus\; \text{Bad} \quad \sqsubseteq \quad \text{Good} \;{}_{1/2}\!\oplus\; \text{Bad}$$

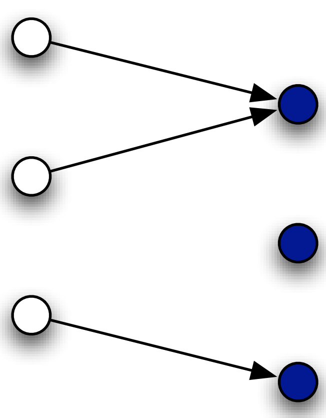
${}_{2/3}\!\oplus$

${}_{1/2}\!\oplus$

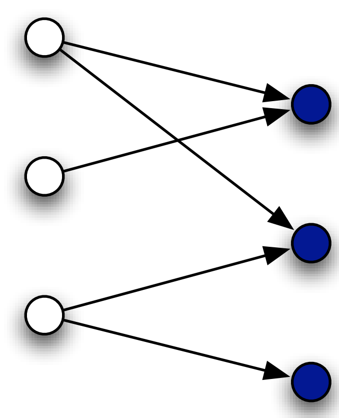
We’re going to look for inspiration from a general mathematical construction called powerdomains to guide the principles underlying refinement.

Powerdomains

(Originally) a general technique by which a semantic model can be augmented to include nondeterminism in such a way that the underlying computational structure of the original model is maintained.



Functions,
$S \longrightarrow S$

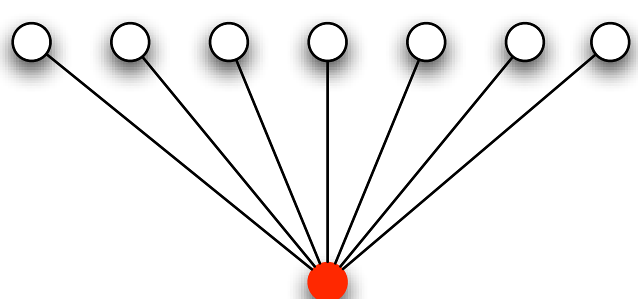


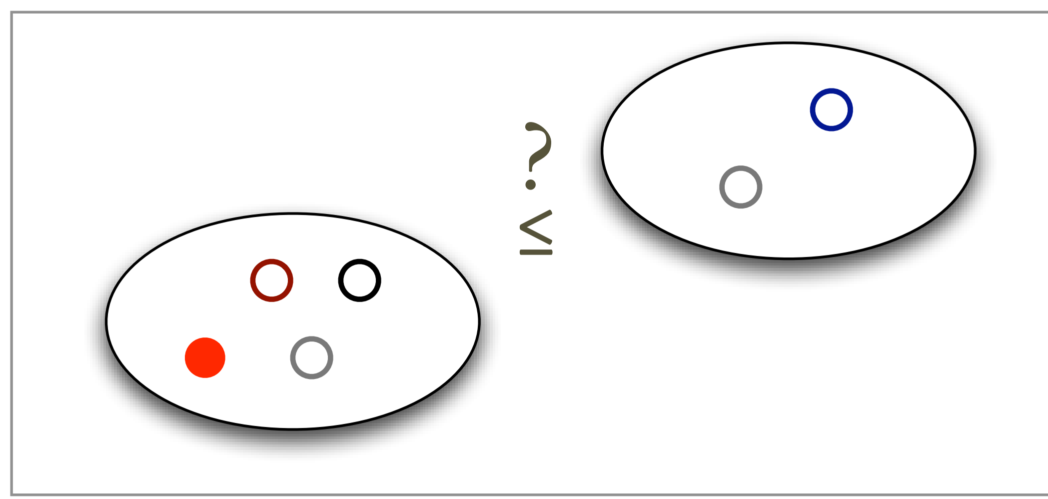
Relations,
$S \longrightarrow PS$

Powerdomains

When we want to distinguish nontermination from other behaviour, we introduce a special “bottom state”

How do we order the programs so that is worse than everything, and reducing the range of behaviours corresponds to “more refined”.

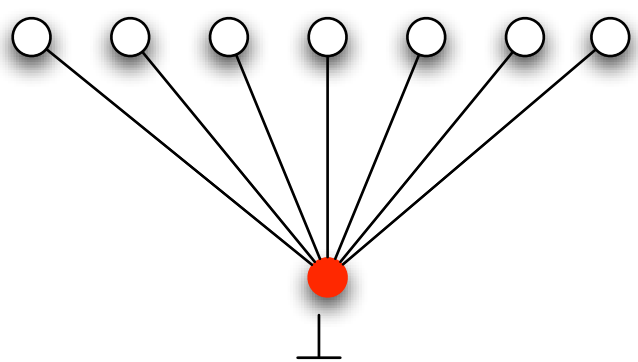


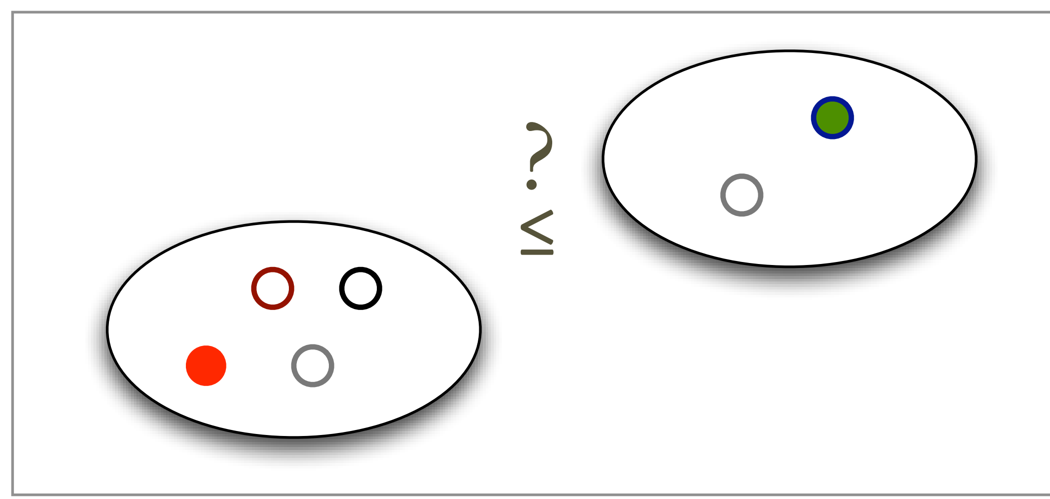


Powerdomains: the Smyth order

$$A \leq_S B \quad \textit{iff} \quad (\forall b \in B(\exists a \in A \cdot a \leq b))$$

$$A \leq_S B \quad \textit{iff} \quad (\perp \in A) \vee (B \subseteq A)$$



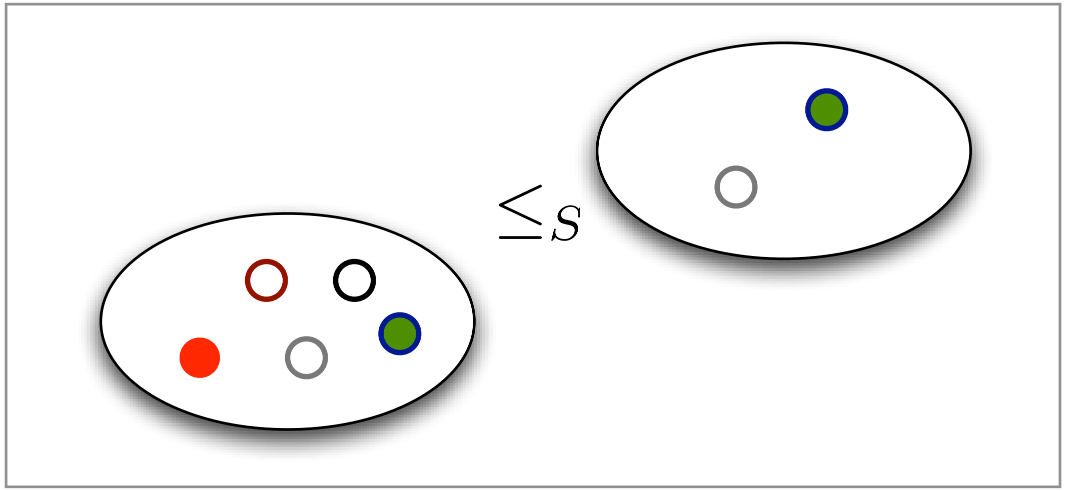


Powerdomains: the Smyth order

$$A \uparrow \quad \hat{=} \quad \{s \in S_\perp \mid (\exists a \in A \cdot a \leq s\}$$

$\leq_S$ becomes an order (rather than a pre-order) on up-closed sets.

On up-closed sets, refinement is simply reverse subset inclusion.



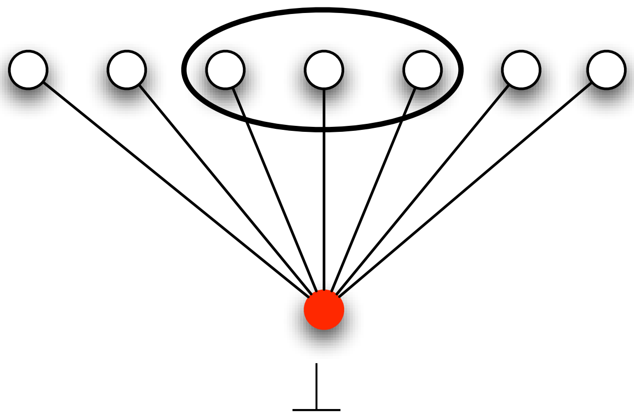
Probabilistic powerdomains

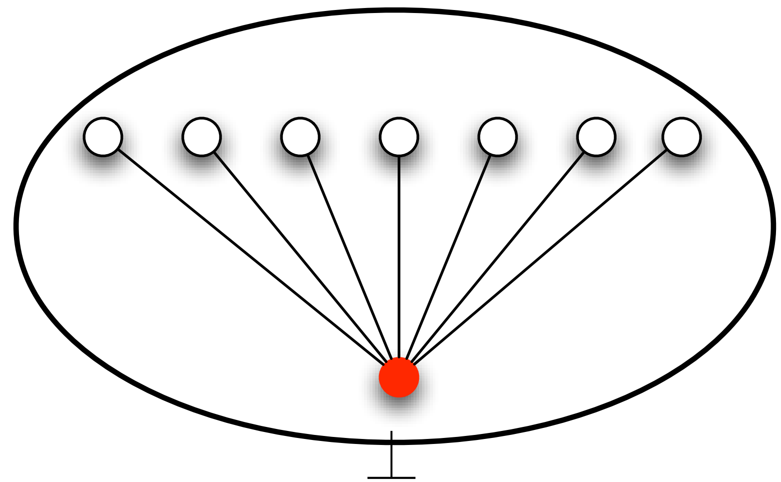
Given a structure (D, $\leq$), we can construct a powerdomain (Eval.D, $\leq$) where objects are evaluations over D, and the order is defined to make “appropriate distinctions”.

- Evaluations are real-valued functions which are defined over the open sets of a (fixed) topology; under certain conditions they can be extended to probability distributions.

- Computational domains can be reformulated in terms of the Scott Topology: a set is Scott open if it is “up-closed” and “inaccessible” (any limit of a chain inside the set can only happen if the chain intersects the set).

Probabilistic powerdomains: Evaluations





$Eval.S_\perp \mathrel{\hat{=}} \mathcal{O}S_\perp \to [0,1]$

Monotone; additive

$$d \le d' \quad \mathit{iff} \quad (\forall O \in \mathcal{O}S_\perp \cdot d.O \le d'.O)$$

$$d \le d' \quad \mathit{iff} \quad (\forall s \in S \cdot d.\{s\} \le d'.\{s\})$$

Probability can increase up the refinement order!

Probabilistic powerdomains: Semantics

Given a structure (D, ≤), we can construct a powerdomain (Eval.D, ≤) where objects are evaluations over D, and the order is defined to make "appropriate distinctions".

Programs $$S_\perp \rightarrow Eval.S_\perp$$

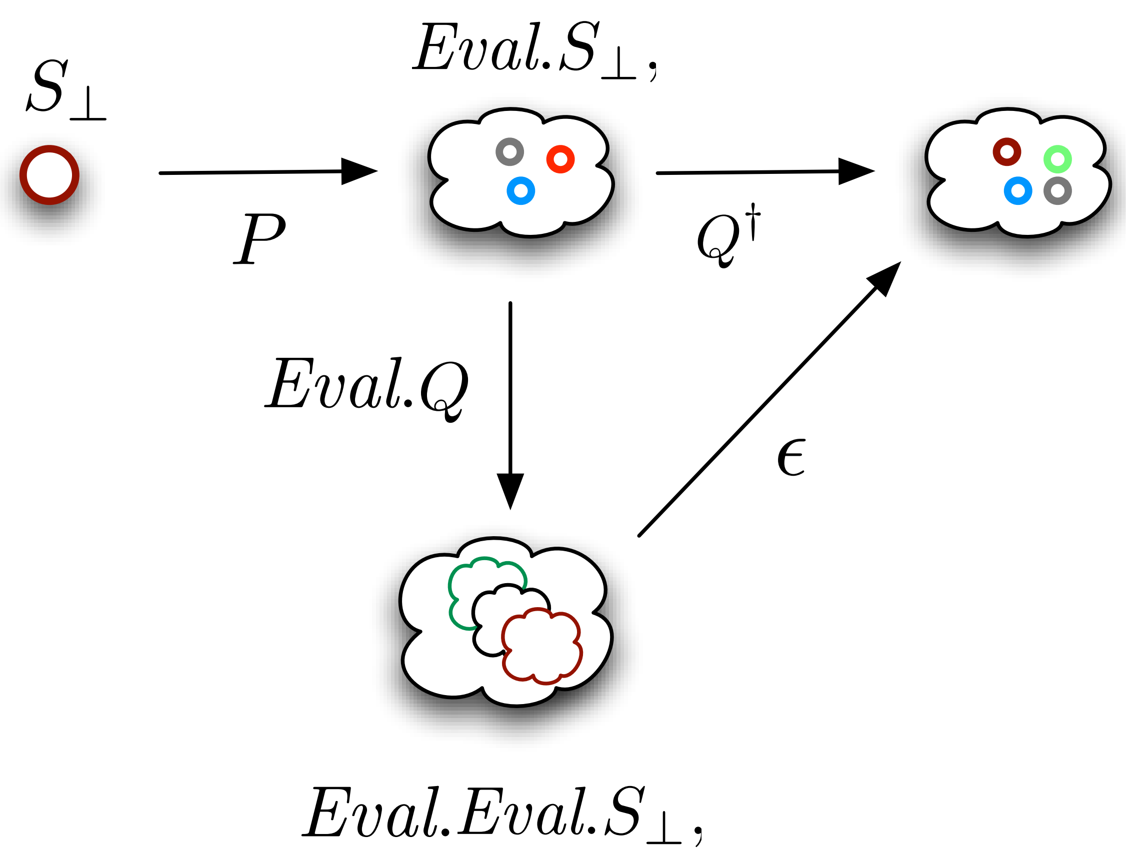
Probabilistic choice $$(P \;{}_p\!\oplus Q).s \quad \hat{=} \quad p \times P.s + (1-p) \times Q.s$$

Sequence $$P; Q \quad \hat{=} \quad P \circ Q^\dagger$$

$$Q^\dagger : Eval.S_\perp \rightarrow Eval.S_\perp$$
$$Q^\dagger.d \quad \hat{=} \quad \textstyle\sum_{s:S} (d.s) \times Q.s$$

Probabilistic powerdomains: Defining $Q^\dagger$

Sequence $\qquad P;Q \quad \hat{=} \quad P \circ Q^\dagger$



Now we have all the ingredients for instant probabilistic semantics.

"It's marvelous! You just add water."

First try:

You will need a flat domain, the Smyth Powerdomain, and the probabilistic powerdomain.

- First add nondeterminism to $(S_\perp \to S_\perp, \sqsubseteq)$
- Next add nondeterminism to get $(S_\perp \to \mathbb{P}S_\perp, \sqsubseteq_S)$
- Finally add probability to get $(\mathit{Eval.}(S_\perp \to \mathbb{P}S_\perp), \sqsubseteq_S)$

First try: What do we get?

- Probabilistic arithmetic $\quad (P\ {}_p\!\oplus Q)\ {}_q\!\oplus R \quad = \quad P\ {}_{pq}\!\oplus (Q\ {}_{\frac{(1-p)q}{1-pq}}\!\oplus R)$

- Universal probabilistic distributivity..... $\quad (P\ {}_p\!\oplus Q) \sqcap R \quad = \quad (P \sqcap R)\ {}_p\!\oplus (Q \sqcap R)$

.... which implies this ....

$$
\begin{array}{lll}
 & P \sqcap P & \\
= & (a\ {}_{1/2}\!\oplus b) \sqcap (a\ {}_{1/2}\!\oplus b) & P \mathrel{\hat{=}} a\ {}_{1/2}\!\oplus b \\
= & (a \sqcap (a\ {}_{1/2}\!\oplus b))\ {}_{1/2}\!\oplus (b \sqcap (a\ {}_{1/2}\!\oplus b)) & \text{Distribution} \\
= & (a \sqcap a)\ {}_{1/4}\!\oplus ((b \sqcap b)\ {}_{1/3}\!\oplus (a \sqcap b)) & \text{Distribution, arithmetic} \\
= & a\ {}_{1/4}\!\oplus (b\ {}_{1/3}\!\oplus (a \sqcap b)) & a \sqcap a = a \\
\neq & P & P \mathrel{\hat{=}} a\ {}_{1/2}\!\oplus b
\end{array}
$$

!!

Probability versus nondeterminism

$$(y := 0 \sqcap y := 1); (x := 0 \ {}_{1/2}\oplus \ x := 1)$$

What's the chance that the demon can guess the value of x?

Probability versus nondeterminism

$$(y := 0 \sqcap y := 1); (x := 0 \;{}_{1/2}\!\oplus x := 1) \qquad \text{Prob distributes over nondet}$$

$$= \qquad (y := 0 \sqcap y := 1); x := 0 \;{}_{1/2}\!\oplus\; (y := 0 \sqcap y := 1); x := 1$$

In this model, we can reproduce the demon's choice within each probabilistic branch....

.... effectively making the demon able to see into the future.

Whoops!

Next try:

You will need a flat domain, the Smyth Powerdomain, the probabilistic powerdomain, and compactness and convexity.

- First add probability to get $(Eval.S_\perp, \leq)$
- Next add nondeterminism to get $(S_\perp \rightarrow \mathbb{P}Eval.S_\perp, \sqsubseteq_P)$
- We need some extra closure conditions:
  (a) up-closed - for termination
  (b) Convex closed - $P\ {}_p\!\oplus P = P$
  (c) Compact - so that iteration can be approximated by "finite" computations.

As before, refinement is reverse subset inclusion

Relational-style semantics for a small sequential language

$$
\begin{array}{lll}
\textit{identity} & [\![\mathsf{skip}]\!].s & \hat{=} \quad \{\overline{s}\} \\
\textit{assignment} & [\![x := a]\!].s & \hat{=} \quad \{\overline{s[x \mapsto a]}\} \\[1em]
\textit{composition} & [\![P; P']\!].s & \hat{=} \quad \{\sum_{s' : S} d.s' \times f'.s' \mid d \in [\![P]\!].s; f' \sqsubseteq [\![P']\!]\} \\
 & \multicolumn{2}{l}{\text{where } f' \in S \to \overline{S}_{\perp} \text{ and in general } f' \sqsubseteq r' \text{ means } f'.s \in r'.s \text{ for all } s.} \\[0.5em]
\textit{choice} & [\![\mathsf{if}\ B\ \mathsf{then}\ P\ \mathsf{else}\ P']\!].s & \hat{=} \quad \text{if } B.s \text{ then } [\![P]\!].s \text{ else } [\![P']\!].s \\
\textit{probability} & [\![P\ {}_p\!\oplus P']\!].s & \hat{=} \quad \{d\ {}_p\!\oplus d' \mid d \in [\![P]\!].s; d' \in [\![P']\!].s\} \\[1em]
\textit{nondeterminism} & [\![P \sqcap P']\!].s & \hat{=} \quad \lceil\, [\![P]\!].s \cup [\![P']\!].s\, \rceil\ , \\
 & \multicolumn{2}{l}{\quad \text{where in general } \lceil D \rceil \text{ is the up-, convex- and Cauchy closure of } D.} \\[0.5em]
\textit{iteration} & \mathsf{do}\ \ G \to P\ \ \mathsf{od} & \hat{=} \quad (\mu X \cdot \mathsf{if}\ \ G\ \ \mathsf{then}\ [\![P]\!]; X\ \ \mathsf{else}\ \ [\![\mathsf{skip}]\!])\ .
\end{array}
$$

Some nice laws....

$$P \sqcap P = P$$

$$(P \sqcap Q) \,{}_p\!\oplus (P \sqcap R) \sqsubseteq_P P \sqcap (Q \,{}_p\!\oplus R)$$

$$P \sqcap P \sqsubseteq_P P \,{}_p\!\oplus P = P$$

$$P \,{}_p\!\oplus (Q \sqcap R) = (P \,{}_p\!\oplus Q) \sqcap (P \,{}_p\!\oplus R)$$

$$P; (Q \,{}_p\!\oplus R) \sqsubseteq_P P; Q \,{}_p\!\oplus P; R$$

$$(Q \,{}_p\!\oplus R); P = (Q; P \,{}_p\!\oplus R; P)$$

This nondeterminism can see what happened after a coin flip, but not before.

Probability versus nondeterminism

$$(y := 0 \ \sqcap \ y := 1); (x := 0 \ {}_{1/2}{\oplus} \ x := 1)$$

What's the chance that the demon can guess the value of x?

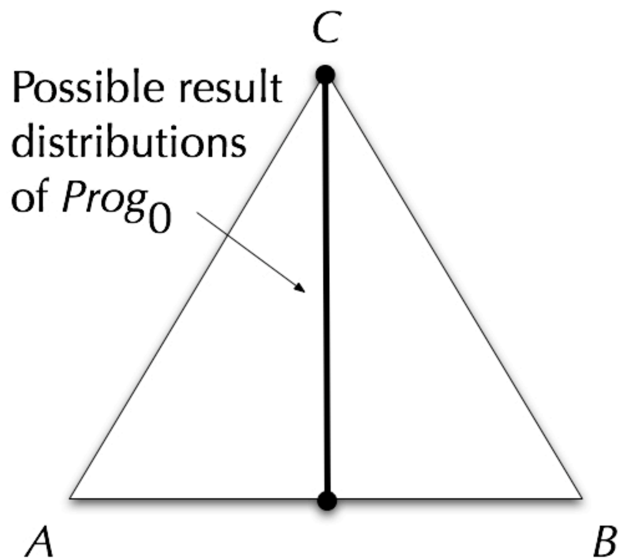
Probability versus nondeterminism

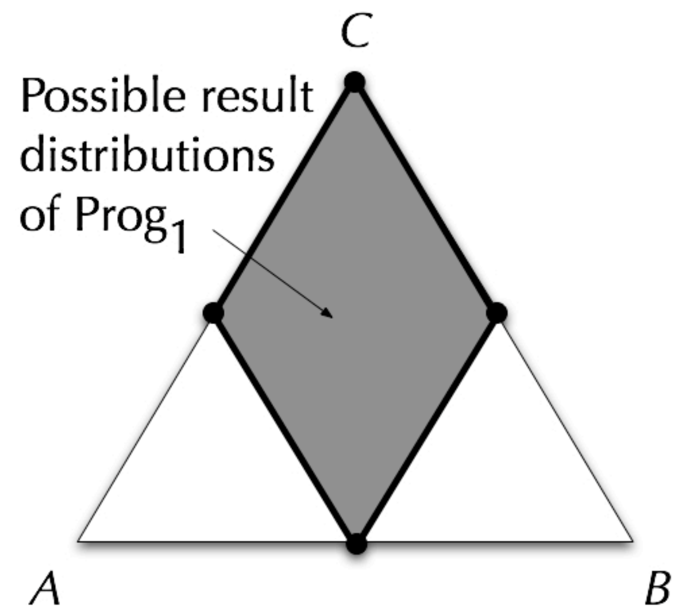
$$(y := 0 \sqcap y := 1); (x := 0 \,{}_{1/2}\!\oplus x := 1) \qquad \text{Nondet distributes over pro}$$

$$= \qquad (y := 0); (x := 0 \,{}_{1/2}\!\oplus x := 1) \;\sqcap\; (y := 1); (x := 0 \,{}_{1/2}\!\oplus x := 1)$$

What's the chance that the demon can guess the value of x?
Answer is 1/2.

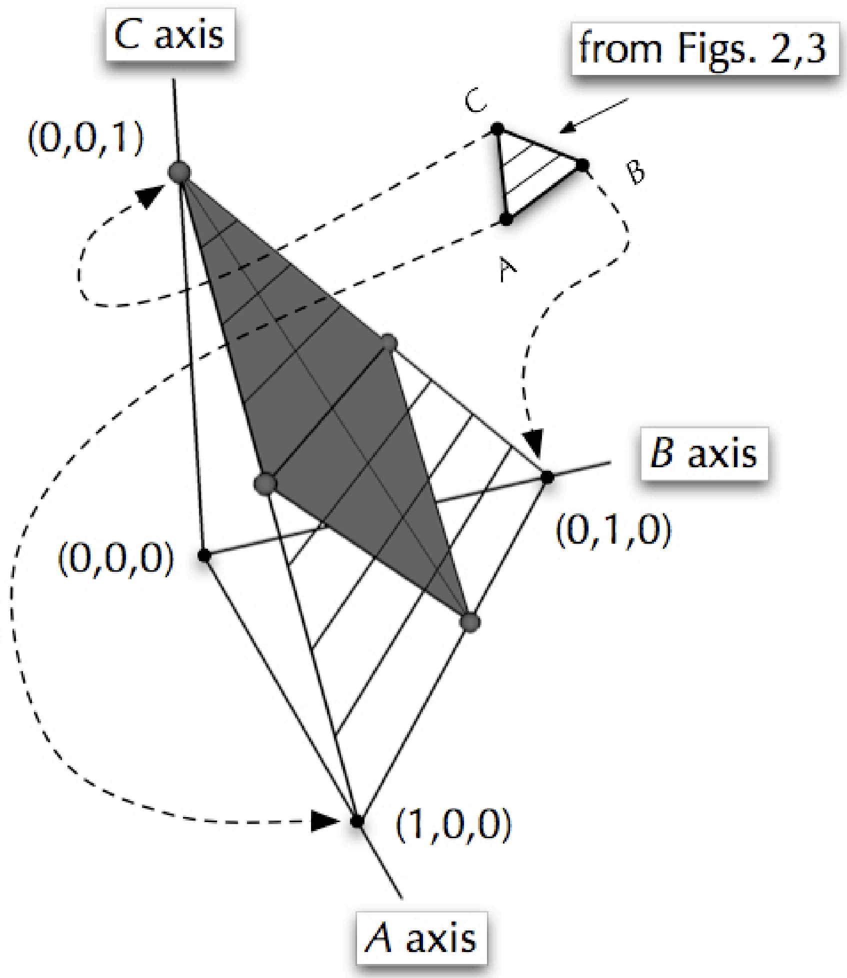
Geometrical interpretation.





$$
\begin{array}{lcl}
Prog_0 & \hat{=} & (s := A \;{}_{0.5}\!\oplus\, s := B) \;\sqcap\; s := C \\
Prog_1 & \hat{=} & (s := A \sqcap s := C) \;{}_{0.5}\!\oplus\; (s := B \sqcap s := C)
\end{array}
$$

Geometrical interpretation.



Plotted on the same diagram, we can see immediately the relationship between the two programs.

$$Prog_0 \quad \hat{=} \quad (s := A \ {}_{0.5}\!\oplus\ s := B) \ \sqcap \ s := C$$

$$Prog_1 \quad \hat{=} \quad (s := A \sqcap s := C) \ {}_{0.5}\!\oplus\ (s := B \sqcap s := C)$$

Logic and properties

Properties are now quantitative

$$\begin{aligned} \mathbb{E}S &\hat{=} S \rightarrow [0,1] \\ e \leq e' &= (\forall s : S \cdot e.s \leq e'.s) \end{aligned}$$

$$\mathsf{wp}.P.e.s \quad \hat{=} \quad (\sqcap \quad d \in P.s \cdot \int_d e)$$

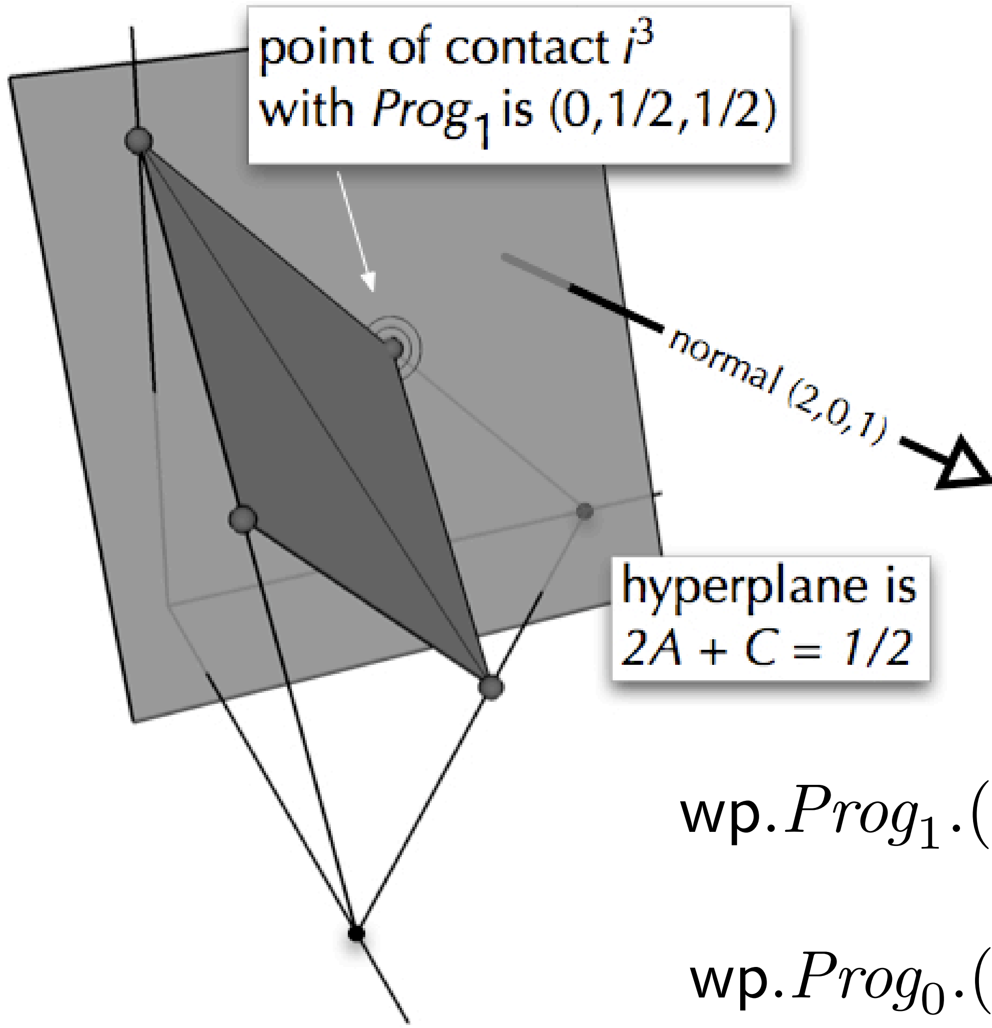
Greatest guaranteed expected value of e with respect to the results of P from initial state s.

$$d \in EvalS_{\perp}, \;\; e \in \mathbb{E}S, \qquad \int_d e \hat{=} \sum_{s:S} d.s \times e.s$$

Transformer semantics for a small sequential language

$$
\begin{array}{lll}
\mathit{identity} & \mathsf{wp.skip}.\mathit{expt} & \hat{=} \quad \mathit{expt} \\
\mathit{assignment} & \mathsf{wp}.(x := E).\mathit{expt} & \hat{=} \quad \mathit{expt}[x := E] \\
\mathit{composition} & \mathsf{wp}.(P; P').\mathit{expt} & \hat{=} \quad \mathsf{wp}.P.(\mathsf{wp}.P'.\mathit{expt}) \\
\mathit{choice} & \mathsf{wp}.(\mathsf{if}\ B\ \mathsf{then}\ P\ \mathsf{else}\ P'\ \mathsf{fi}).\mathit{expt} & \\
 & \qquad \hat{=}\ [B] \times \mathsf{wp}.P.\mathit{expt} + [\neg B] \times \mathsf{wp}.P'.\mathit{expt} & \\
\mathit{probability} & \mathsf{wp}.(P\ {}_p\!\oplus P').\mathit{expt} & \\
 & \qquad \hat{=}\ p \times \mathsf{wp}.P.\mathit{expt} + (1-p) \times \mathsf{wp}.P'.\mathit{expt} & \\
\mathit{nondeterminism} & \mathsf{wp}.(P \sqcap P').\mathit{expt} & \hat{=} \quad \mathsf{wp}.P.\mathit{expt}\ \mathbf{min}\ \mathsf{wp}.P'.\mathit{expt} \\
\mathit{iteration} & \mathsf{wp}.(\mathsf{do}\ B \rightarrow r\ \mathsf{od}).e\ \hat{=}\ (\mu X \bullet [B] \times \mathsf{wp}.r.X + [\neg B] \times e)\ . &
\end{array}
$$

Geometrical interpretation:



Expectations are "hyperplanes".

$$\mathsf{wp}.Prog_1.(2[s = A] + [s = C]) \quad = \quad 1/2$$

$$\mathsf{wp}.Prog_0.(2[s = A] + [s = C]) \quad = \quad 1$$

Logic and properties: the monotonic transformers

$$\mathbb{T}S \quad \hat{=} \quad \mathbb{E}S \leftarrow \mathbb{E}S$$

$$S_\perp \rightarrow \mathbb{P}Eval.S_\perp \quad \underset{rp}{\overset{\mathsf{wp}}{\rightleftarrows}} \quad \mathbb{T}S$$

$$\mathsf{wp} \circ rp \quad = \quad id$$

$$rp \circ \mathsf{wp} \quad = \quad id \text{ , if } \begin{cases} t.(e \,{}_p{\oplus}\, e') \geq t.e \,{}_p{\oplus}\, t.e' \\ t.(\mathbf{k}e) = \mathbf{k}t.e \\ t.(e - \mathbf{k}) \geq t.e - \mathbf{k} \end{cases}$$

"Sublinear"

Why so complicated: can't we just have a whole logic based on probabilities, rather than random variables?

It's a question of compositionality:

$$
\begin{array}{lcl}
Prog_0 & \hat{=} & (s := A \ {}_{0.5}\oplus \ s := B) \ \sqcap \ s := C \\
Prog_1 & \hat{=} & (s := A \sqcap s := C) \ {}_{0.5}\oplus \ (s := B \sqcap s := C)
\end{array}
$$

Allowed final value(s) of s	A	B	C	A, B	B, C	C, A
Maximim possible probability	1/2	1/2	1	1	1	1
Minimim possible probability	0	0	0	0	1/2	1/2

A quantitative logic based on probabilities *is not compositional.*

Consider the following "context":

$$Prog_0;\quad \mathsf{if}\ s{=}C\ \mathsf{then}\ (s:=\ A\ {}_{0.5}{\oplus}\ s:=\ B)\ \mathsf{fi}$$
$$Prog_1;\quad \mathsf{if}\ s{=}C\ \mathsf{then}\ (s:=\ A\ {}_{0.5}{\oplus}\ s:=\ B)\ \mathsf{fi}$$

What's the probability that the state is A finally?

As we have seen, the two programs can be distinguished in the transformer semantics (by a random variable encoded as an expectation).

$$
\begin{aligned}
\mathsf{wp}.Prog_1.(2[s = A] + [s = C]) &\quad=\quad 1/2 \\
\mathsf{wp}.Prog_0.(2[s = A] + [s = C]) &\quad=\quad 1
\end{aligned}
$$

The transformer semantics, based on full random variables, *is* compositional.

A nice proof rule, proved using the transformer semantics:

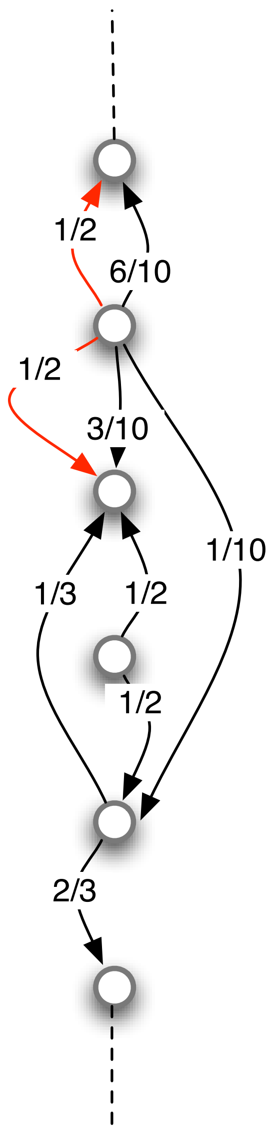
A loop: $\quad \mathsf{do} \;\; G \;\rightarrow\; \mathit{body} \;\; \mathsf{od}$

An invariant: $\quad [G] \times I \quad \leq \quad \mathsf{wp}.\mathit{body}.I$

Termination condition: $\quad T \quad \hat{=} \quad \mathsf{wp}.(\mathsf{do} \;\; G \;\rightarrow\; \mathit{body} \;\; \mathsf{od}).1$

A rule: $\quad I \leq T \quad \Rightarrow \quad I \leq \mathsf{wp}.(\mathsf{do} \;\; G \;\rightarrow\; \mathit{body} \;\; \mathsf{od}).I$

The "jumping bean" : specification.



$$[n = N] \quad \leq \quad \lceil \mathsf{wp}.\mathit{jump}.[n \neq N] \rceil$$

The bean must move...

$$[n = N] \quad \leq \quad \mathsf{wp}.\mathit{jump}.[N{-}K \leq n \leq N{+}K]$$

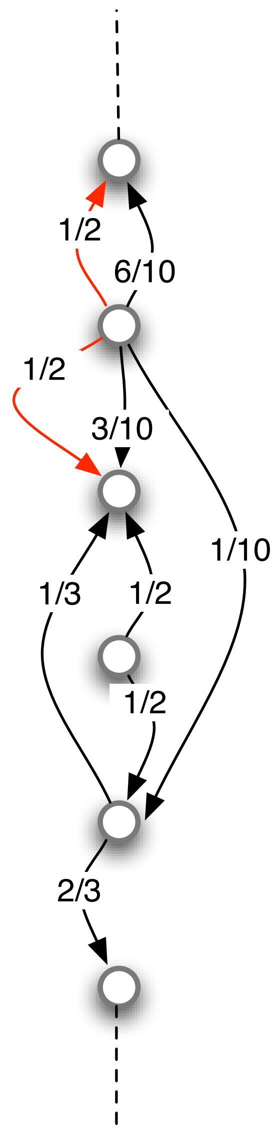
(K is a fixed constant.)

The bean can't move too much...

$$n \quad \leq \quad \mathsf{wp}.\mathit{jump}.n$$

The expected move is at least 0.

The “jumping bean”.



$$Bean \;\hat{=}\; \mathsf{wp}.(\mathsf{do}\;\; (n \leq N) \rightarrow \; jump \;\; \mathsf{od})$$

The bean continues to jump, until it exceeds N.

$$1 \;=\; \mathsf{wp}.Bean.[n > N]$$

The conditions on its behaviour guarantee that it will eventually exceed any bound.

Exercise: use the properties of the transformers to prove this. (Should be about 10 lines of proof.)

Probability versus nondeterminism:

$$
\begin{aligned}
& (x := 0 \sqcap x := 1); (y := 0 \;{}_{1/2}{\oplus}\; y := 1) \\
= \quad & (x := 0 \sqcap x := 1); (y := 0) \\
& {}_{1/2}{\oplus} \\
& (x := 0 \sqcap x := 1); y := 1)
\end{aligned}
$$

The demon can predict the future.

$$
\begin{aligned}
& (y := 0 \;{}_{1/2}{\oplus}\; y := 1); (x := 0 \sqcap x := 1) \\
= \quad & y := 0; (x := 0 \sqcap x := 1) \\
& {}_{1/2}{\oplus} \\
& y := 1; (x := 0 \sqcap x := 1)
\end{aligned}
$$

The demon can access the past.

Probability versus nondeterminism:

Smyth powerdomain, for nondeterminism; then the probabilistic powerdomain on top of that.

The demon can predict the future.

Probabilistic powerdomain to make $EvalS_{\perp}$; then the Smyth powerdomain to make, $S_{\perp} \rightarrow \mathbb{P}Eval.S_{\perp}$ with a special definition of ";"

The demon can access the past.

Suppose we wanted to prevent the demon from accessing the past, i.e.

$$
\begin{array}{ll}
 & (y := 0 \ {}_{1/2}\!\oplus\, y := 1); (x := 0 \ \sqcap \ x := 1) \\
= & (y := 0 \ {}_{1/2}\!\oplus\, y := 1); x := 0 \\
 & \sqcap \\
 & (y := 0 \ {}_{1/2}\!\oplus\, y := 1); x := 1
\end{array}
$$

How would we build a semantic domain justifying this algebraic property?

Suppose we wanted to prevent the demon from accessing the past, i.e.

Use the probabilistic powerdomain to build $Eval.S_\perp \rightarrow Eval.S_\perp$,
and then the Smyth powerdomain to build $Eval.S_\perp \rightarrow \mathbb{P}Eval.S_\perp$

How would we build a semantic domain justifying this algebraic property?

With this model, we lose some refinements:

$$\boxed{A \sqcap B \quad \not\sqsubseteq \quad \textsf{if} \ \ G \ \ \textsf{then} \ A \ \ \textsf{else} \ \ B}$$

In fact, if we did not ban this refinement and we still had the equality we started with, then we would also have to accept this refinement!

$$(x, y := \ 0, 0) \ {}_{1/2}\!\oplus (x, y := \ 1, 1) \quad \sqsubseteq \quad x, y := \ 0, 0$$

An algebraic "thought experiment"

$$
\begin{array}{lll}
 & (y := 0 \; {}_{1/2}{\oplus} \; y := 1); (x := 0 \sqcap x := 1); & \\
 & \mathsf{if}\ (x = y)\ \mathsf{then}\ (x, y := 1, 1)\ \mathsf{else}\ (x, y := 0, 0) & \\
\sqsubseteq & (y := 0 \; {}_{1/2}{\oplus} \; y := 1); (\mathsf{if}\ (y = 0)\ \mathsf{then}\ x := 0)\ \mathsf{else}\ x := 1; & \ddagger \\
 & \mathsf{if}\ (x = y)\ \mathsf{then}\ Z\ \mathsf{else}\ W & \\
= & (y := 0; x := 0 \; {}_{1/2}{\oplus} \; y := 1; x := 0) & \\
 & \mathsf{if}\ (x = y)\ \mathsf{then}\ Z\ \mathsf{else}\ W & \\
= & Z & \\
= & x, y := 0, 0 &
\end{array}
$$

$$
\begin{array}{ll}
 & (y := 0 \;{}_{1/2}\!\oplus\; y := 1); (x := 0 \;\sqcap\; x := 1); \\
 & \textsf{if}\;\; (x = y) \;\; \textsf{then} \;\; Z \;\; \textsf{else} \;\; W \\[1em]
= & ((y := 0 \;{}_{1/2}\!\oplus\; y := 1); x := 0) \;\sqcap\; (y := 0 \;{}_{1/2}\!\oplus\; y := 1); x := 1; \qquad \dagger \\
 & \textsf{if}\;\; (x = y) \;\; \textsf{then} \;\; Z \;\; \textsf{else} \;\; W \\[1em]
 & ((y := 0 \;{}_{1/2}\!\oplus\; y := 1); x := 0); \textsf{if}\;\; (x = y) \;\; \textsf{then} \;\; Z \;\; \textsf{else} \;\; W \\
 & \;\sqcap \\
 & ((y := 0 \;{}_{1/2}\!\oplus\; y := 1); x := 1); \textsf{if}\;\; (x = y) \;\; \textsf{then} \;\; Z \;\; \textsf{else} \;\; W \\[1em]
 & (Z \;{}_{1/2}\!\oplus\; W) \;\sqcap\; (Z \;{}_{1/2}\!\oplus\; W) \\[1em]
= & (x, y := 0, 0) \;{}_{1/2}\!\oplus\; (x, y := 1, 1)
\end{array}
$$

$$
(x, y := 0, 0) \;{}_{1/2}\!\oplus\; (x, y := 1, 1) \;\sqsubseteq\; x, y := 0, 0
$$

The "refinement paradox"

Properties of the logic/algebra in the context where "hidden state" is an issue are hard to get right, even when there are no probabilities.

It turns out to be a really hard problem to find a formalisation which behaves properly for refinement

h "High security" variables (are "private")

l "Low security" variables (are "public")

"Obviously" we want to make sure that going up the refinement order preserves our security properties.

The "refinement paradox"

$$h :\in \{0, \ldots, H\}$$

Choose some value for h, but keep it hidden.

$$h :\in \{0, \ldots, H\} \quad \sqsubseteq \quad h := 0$$

Refinement doesn't preserve the security of the choice?

Even refinement of low security variables is a problem.

$$l := 0 \sqcap l := 1 \quad \sqsubseteq \quad \mathsf{if}\ (h = 0)\ \mathsf{then}\ l := 0\ \mathsf{else}\ h := 1$$

And distribution through nondeterminism...

$$(P \sqcap Q); R \quad \sqsubseteq \quad P; R \sqcap Q; R$$

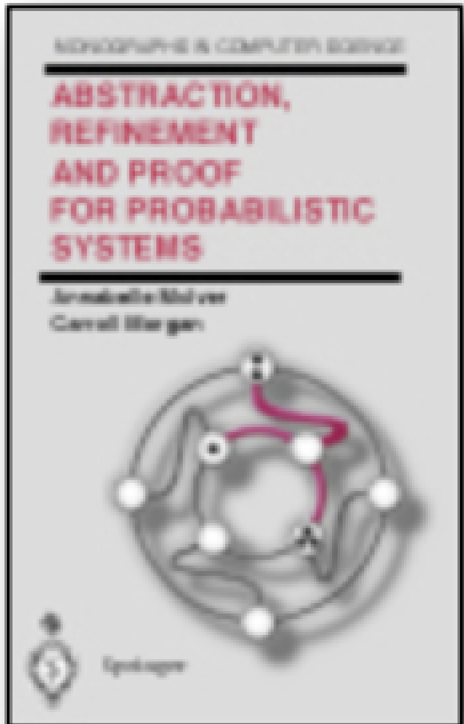
... implies

$$\begin{array}{ll} & (h := 0 \sqcap h := 1); (l := 0 \sqcap l := 1) \\ = & h := 0; (l := 0 \sqcap l := 1) \sqcap h := 1; (l := 0 \sqcap l := 1) \\ \sqsubseteq & h := 0; l := 0 \sqcap h := 1; l := 1 \end{array}$$

Resolving the paradox

Building a model with “hidden probabilities”, based on the ideas of the final example domain provides the right insights to build a model and logic for “refinement” and knowledge:

- “The Shadow Knows: Refinement of Ignorance in Sequential Programs”, CC Morgan, to appear, Science of Computer Programming



http://www.cse.unsw.edu.au/~carrollm/probs/bibliography.html