Refinement-oriented noninterference:

Comparison of a qualitative and a quantitative model

Annabelle McIver — Macquarie University
Larissa Meinicke — Macquarie University
Carroll Morgan — University of New South Wales

Summary

Context — Theoretical aspects of specification and development of (secure) computing systems... within a theory of system specification.

Summary

Context — Theoretical aspects of specification and development of (secure) computing systems... within a theory of system specification.

- Aims: Source-level program-algebraic reasoning over probability, demonic choice and hidden state.

- Semantic space(s), including a refinement order.
- Small programming language, and its denotations.
- Comparative security.
- Example.

- Prospects.

Aims and assumptions

We consider (sequential) programs whose variables have an extra attribute, that they can be either *visible* or *hidden*.

Our original motivation for this (1995) was to hide already-resolved probabilistic choices, *inside* program modules, from about-to-be-made demonic choices *outside* those modules, in order to allow data-refinement (a.k.a. simulation) between them.

This applies also to already-made choices in a concurrent process that should be hidden from about-to-be-made choices in another process, or e.g. to an adversarial concurrency scheduler that should not be able to exploit probabilistic processes' local state.

Aims and assumptions

Demonic/probabilistic choice plus hiding is hard, though progress has been made by many people.

In our own work we have treated the three pairs of features separately, in order to see how they interact in a simpler setting: we have done probabilistic choice with demonic choice (1995), hiding with demonic choice (2006) and hiding with probabilistic choice (2010).

1 2 3

We noticed that qualitative hiding with demonic choice (2006) could equally well model noninterference security with refinement/abstraction. That is why we later tried the same approach with quantitative hiding (2010).

Aims and assumptions

In this talk I will present both of those noninterference security models –qualitative and quantitative– side by side in order to highlight their similarities.

Qualitative slides will be marked with a $\sqcap$, and quantitative slides will be marked with a $\bigoplus$.

We noticed that qualitative hiding with demonic choice (2006) could equally well model noninterference security with refinement/abstraction. Later we tried the same approach with quantitative hiding (2010).

Aims and assumptions

In this talk I will present both of those noninterference security models –qualitative and quantitative– side by side in order to highlight their similarities.

Qualitative slides will be marked with a $\sqcap$, and quantitative slides will be marked with a $\bigoplus$.

We noticed that qualitative hiding with demonic choice (2006) could equally well model noninterference security with refinement/abstraction. Later we tried the same approach with quantitative hiding (2010).

Aims and assumptions

An attacker observes a program in action, seeing:

- values of visible variables, even if subsequently overwritten; and — perfect recall
- program flow, i.e. resolution of conditionals, loop guards and other "external" choices. — implicit flow

He knows the source code.

From all that, he attempts to deduce the value of hidden variables' final values. Either he succeeds, or does not $\sqcap$; or he succeeds with some probability $\oplus$.

Aims and assumptions

An attacker observes a program in action, seeing:

- values of visible variables, even if subsequently overwritten; and — perfect recall
- program flow, i.e. resolution of conditionals, loop guards and other "external" choices. — implicit flow

These assumptions are motivated by algebraic "experiments" based on how program refinement should behave: but they are not discussed in this talk.

The Shadow Knows: Refinement of ignorance in sequential programs. Carroll Morgan. *Proc. Maths. Prog. Construction.* 2006.

Philosophy and strategy

Abstraction from details.

A relation “at least as secure as” that allows compositional reasoning.

A specification (ideal system) that specifies explicitly all tolerable imperfections.

Philosophy and strategy

Abstraction from details.

A relation "at least as secure as" that allows compositional reasoning: (reactive) *simulatability*.

A specification (ideal system) that specifies explicitly all tolerable imperfections.

A general composition theorem for secure reactive systems. Backes, Pfitzmann and Waidner. *Proc. TCC* 2004.

A model for asynchronous reactive systems... Pfitzmann and Waidner. *Proc. 20th IEEE Symp. on Security and Privacy*, 2001.

Philosophy and strategy

Abstraction from details.

A relation "at least as secure as" that allows compositional reasoning.

A specification (ideal system) that specifies explicitly all tolerable imperfections.

Source-level reasoning based on denotations.
Program logic.
Connection between refinement relation and testing.
Information-theoretic (not complexity-theoretic).

Philosophy and strategy

Thus we don't judge a program, on its own, to be secure (or insecure) $\sqcap$, or to be partly secure $\oplus$. Rather we ask whether one program is at least as secure as another. In both cases the definition of "secure" is ultimately subjective. For example we could say that:

- $\sqcap$ If hidden variable's value can never be deduced in a specification *S*, then must not be deducible in any implementation *I* of it.
- $\oplus$ The chance of guessing a hidden variable's value in *I* must not exceed the chance of guessing it in *S*.

$\mathtt{v}$ — the visible variables $\mathtt{h}$ — the hidden variables

Small programming language

qualitative		quantitative
$\mathtt{v}{:=}\, E$	assign to visible	$\mathtt{v}{:=}\, E$
$\mathtt{h}{:=}\, E$	assign to hidden	$\mathtt{h}{:=}\, E$
$\mathtt{v}{:=}\, E \sqcap E'$	choose visible	$\mathtt{v}{:=}\, E \;{}_p\!\oplus\, E'$
$\mathtt{v}{:\in}\, \{E, E'\}$	(internal)	$\mathtt{v}{:\in}$ *later...*
$\mathtt{h}{:=}\, E \sqcap E'$	choose hidden	$\mathtt{h}{:=}\, E \;{}_p\!\oplus\, E'$
$\mathtt{h}{:\in}\, \{E, E'\}$	(internal)	$\mathtt{h}{:\in}$ *later...*

compound statements: conditional,
(external) demonic choice, OR
(external) probabilistic choice,
loop, local variables...

with hiding, not yet both probabilistic and demonic

Small programming language

qualitative		quantitative
$\mathtt{v}{:=}\,E$	assign to visible	$\mathtt{v}{:=}\,E$
$\mathtt{h}{:=}\,E$	assign to hidden	$\mathtt{h}{:=}\,E$
$\mathtt{v}{:=}\,E \sqcap E'$	choose visible	$\mathtt{v}{:=}\,E \;{}_p\!\oplus\, E'$
$\mathtt{v}{:\in}\,\{E, E'\}$	(internal)	$\mathtt{v}{:\in}\;\; later\ldots$
$\mathtt{h}{:=}\,E \sqcap E'$	choose hidden	$\mathtt{h}{:=}\,E \;{}_p\!\oplus\, E'$
$\mathtt{h}{:\in}\,\{E, E'\}$	(internal)	$\mathtt{h}{:\in}\;\; later\ldots$

compound statements: conditional,
(external) demonic choice, OR
(external) probabilistic choice,
loop, local variables...

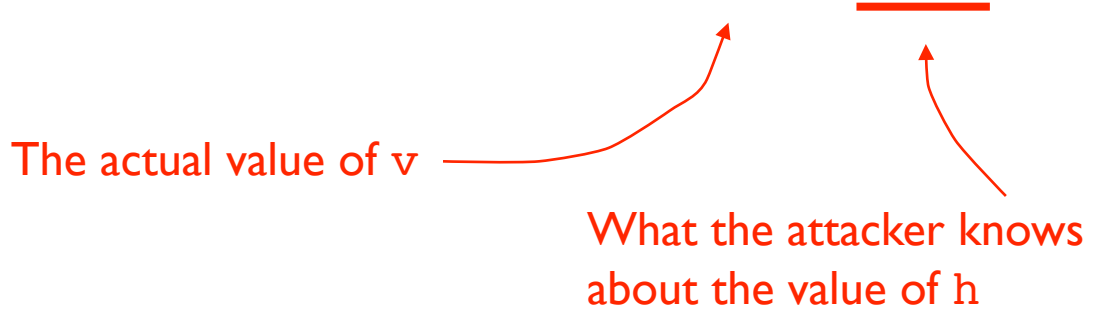
In this talk we don't consider loops or divergence.

SEMANTIC SPACE

Basic building-block is the *split-state*

Distinguish visible (low-security) variables of type $\mathcal{V}$
from hidden (high-security) variables of type $\mathcal{H}$

Classical split-state	$\mathcal{V} \times \mathcal{H}$
Qualitative split-state	$\mathcal{V} \times \underline{\mathbb{P}\mathcal{H}}$ Shadow
Quantitative split-state	$\mathcal{V} \times \underline{\mathbb{D}\mathcal{H}}$ Inner



What does a Qualitative split-state tell us?

The program variables are `v` and `h` in all three cases.

Qualitative split-state (v, H) tells us $\mathcal{V} \times \mathbb{P}\mathcal{H}$

- that `v`'s value is v, and
- that `h`'s value is in the "Shadow" set H.

The semantics constructs set H in $\mathbb{P}\mathcal{H}$ according to the attacker's "most intrusive" observations, about which more later.

What does a Quantitative split-state tell us?

The program variables are $\mathtt{v}$ and $\mathtt{h}$ in all three cases.

Quantitative split-state (v, δ) tells us $\mathcal{V} \times \mathbb{D}\mathcal{H}$

- that $\mathtt{v}$'s value is v, and
- that $\mathtt{h}$'s value has "Inner" distribution δ over $\mathcal{H}$.

The semantics constructs distribution $\delta : \mathbb{D}\mathcal{H}$ according to the attacker's observations.

Interpret these atomic programs using split-states

$\sqcap$ qualitative		$\oplus$ quantitative
$\mathtt{v}{:=}\,E$	assign to visible	$\mathtt{v}{:=}\,E$
$\mathtt{h}{:=}\,E$	assign to hidden	$\mathtt{h}{:=}\,E$
$\mathtt{v}{:=}\,E \sqcap E'$	choose visible	$\mathtt{v}{:=}\,E \;{}_{p}\!\oplus\, E'$
$\mathtt{v}{:\in}\,\{E, E'\}$	(internal)	$\mathtt{v}{:\in}\;\; later\ldots$
$\mathtt{h}{:=}\,E \sqcap E'$	choose hidden	$\mathtt{h}{:=}\,E \;{}_{p}\!\oplus\, E'$
$\mathtt{h}{:\in}\,\{E, E'\}$	(internal)	$\mathtt{h}{:\in}\;\; later\ldots$

Qualitative examples

From initial split-state (v, H) we execute the program shown, to give the final split-state at right.

	program	*split-state*
assign to visible	$\mathtt{v}{:=}\,0$	$(0, H)$
assign to hidden	$\mathtt{h}{:=}\,1$	$(v, \{1\})$
choose hidden	$\mathtt{h}{:\in}\,\{2,3,4\}$	$(v, \{2,3,4\})$
choose visible?	$\mathtt{v}{:\in}\,\{5,6,7\}$	*???*

Quantitative examples

From initial split-state (v, δ) we execute the program shown, to give the final split-state at right.

	program	*split-state*
assign to visible	$\mathtt{v} := 0$	$(0, \delta)$
assign to hidden	$\mathtt{h} := 1$	$(v, \{\!\{1\}\!\})$
choose hidden	$\mathtt{h} := 2 \oplus 3 \oplus 4$	$(v, \{\!\{2, 3, 4\}\!\})$
choose visible?	$\mathtt{v} := 5 \oplus 6 \oplus 7$	*???*

point distribution

uniform choices

uniform distribution

Qualitative examples continued

From initial split-state (v, H) we execute the two programs shown, to give the final split-state at right.

singleton set

	program	*split-state*
choose hidden	$\mathtt{h} :\in \{2,3,4\};$	$\{(v, \{2,3,4\})\}$
followed by		
assign to visible	$\mathtt{v} := \mathtt{h} \bmod 2$	$\{(0,\{2,4\}),(1,\{3\})\}$

two possible outcomes

Programs are thus of type $\mathcal{V} \times \mathbb{P}\mathcal{H} \to \mathbb{P}(\mathcal{V} \times \mathbb{P}\mathcal{H})$.

Kleisli...

Quantitative examples continued

From initial split-state (v, δ) we execute the two programs shown, to give the final split-state at right.

point distribution

	program	*split-state*
choose hidden	$\mathtt{h}{:=}\,2{\oplus}3{\oplus}4;$	$\{\!\{(v, \{\!\{2,3,4\}\!\})\}\!\}$
followed by		
assign to visible	$\mathtt{v}{:=}\,\mathtt{h}\ \mathbf{mod}\ 2$	$\{\!\{(0, \{\!\{2,4\}\!\})^{@\frac{2}{3}}, (1, \{\!\{3\}\!\})^{@\frac{1}{3}}\}\!\}$

discrete distribution over two pairs: the first pair has probability 2/3, the second has probability 1/3

the inner distributions are uniform

Programs are thus of type $\mathcal{V}\times\mathbb{D}\mathcal{H}\to\mathbb{D}(\mathcal{V}\times\mathbb{D}\mathcal{H})$.

Qualitative examples completed

From initial split-state (v, H) we execute the program shown, to give the final set of split-states at right.

	program	*split-state*
assign to visible	$\mathtt{v}{:=}\,0$	$\{(0, H)\}$
assign to hidden	$\mathtt{h}{:=}\,1$	$\{(v, \{1\})\}$
choose hidden	$\mathtt{h}{:\in}\,\{2,3,4\}$	$\{(v, \{2,3,4\})\}$
choose visible	$\mathtt{v}{:\in}\,\{5,6,7\}$	$\{(5, H), (6, H), (7, H)\}$

Programs are thus of type $\mathcal{V}\times\mathbb{P}\mathcal{H}\to\mathbb{P}(\mathcal{V}\times\mathbb{P}\mathcal{H})$.

Quantitative examples completed

From initial split-state (v, δ) we execute the program shown, to give the final distribution of split-states at right.

	program	*split-state*
assign to visible	$\mathtt{v}{:=}\,0$	$\{\!\{(0,\delta)\}\!\}$
assign to hidden	$\mathtt{h}{:=}\,1$	$\{\!\{(v,\{\!\{1\}\!\})\}\!\}$
choose hidden	$\mathtt{h}{:=}\,2 \oplus 3 \oplus 4$	$\{\!\{(v,\{\!\{2,3,4\}\!\})\}\!\}$
choose visible	$\mathtt{v}{:=}\,5 \oplus 6 \oplus 7$	$\{\!\{(5,\delta),(6,\delta),(7,\delta)\}\!\}$

An (outer) distribution over (inner) distribution is called a hyperdistribution. $\mathbb{D}(\mathcal{V}\times\mathbb{D}\mathcal{H})$

External vs internal demonic choice

$$\mathtt{h} :\in \{2,3,4\}; \mathtt{v} := \mathtt{h} \bmod 2$$

$$= \quad \mathtt{v} := 0; \mathtt{h} := 2 \sqcap 4$$

$$\sqcap \quad \mathtt{v} := 1; \mathtt{h} := 3$$

external demonic choice, visible to attacker

internal demonic choice, hidden from attacker

Both produce $\{(0, \{2,4\}), (1, \{3\})\}$.

External vs internal probabilistic choice

$$\mathtt{h} :\in \{\!\{2,3,4\}\!\}; \mathtt{v} := \mathtt{h} \bmod 2$$

$$= \quad \mathtt{v} := 0; \mathtt{h} := 2 \oplus 4$$

$$\quad {}_{2/3}\!\oplus \quad \mathtt{v} := 1; \mathtt{h} := 3$$

external probabilistic choice, visible to attacker

internal probabilistic choice, hidden from attacker

Both produce $\{\!\{(0,\{2,4\})^{@\frac{2}{3}}, (1,\{3\})^{@\frac{1}{3}}\}\!\}$.

PROGRAM DENOTATIONS

Don’t worry: just *one* of them, as an example.

Semantic definitions — qualitative sample

Assign to visible — qualitative

$$[\![\mathtt{v}{:=}\,E]\!](v,H) \quad = \quad \{(E^{v}_{\mathtt{v}}, \{h{:}\,H \mid E^{v}_{\mathtt{v}} = E^{v,h}_{\mathtt{v},\mathtt{h}}\}) \mid \mathtt{h}{:}\,H\}$$

where $E^{v}_{\mathtt{v}}$ denotes replacement of $\mathtt{v}$ by v throughout E

For each possible value of $\mathtt{h}$ in the incoming shadow H,

Construct a pair containing the evaluation of E for that value of $\mathtt{h}$ (and the incoming v),

And associate with it a new, possibly smaller H including only those h's that would have given the same E.

Semantic definitions — qualitative sample

Assign to visible — qualitative

$$[\![\mathtt{v}{:=}\,E]\!](v,H) \quad = \quad \{(E^{v}_{\mathtt{v}}, \{h{:}\,H \mid E^{v}_{\mathtt{v}} = E^{v,h}_{\mathtt{v},\mathtt{h}}\}) \mid \mathtt{h}{:}\,H\}$$

where $E^{v}_{\mathtt{v}}$ denotes replacement of $\mathtt{v}$ by v throughout E

For each possible value of $\mathtt{h}$ in the incoming shadow H,

Construct a pair containing the evaluation of E for that value of $\mathtt{h}$ (and the incoming v),

And associate with it a new, possibly smaller H including only those h's that would have given the same E.

Semantic definitions — qualitative sample

Assign to visible — qualitative

$$[\![\mathtt{v}{:=}\,E]\!](v,H) \quad = \quad \{(E^{v}_{\mathtt{v}}, \{h{:}\,H \mid E^{v}_{\mathtt{v}} {=} E^{v,h}_{\mathtt{v},\mathtt{h}}\}) \mid \mathtt{h}{:}\,H\}$$

where $E^{v}_{\mathtt{v}}$ denotes replacement of $\mathtt{v}$ by v throughout E

For each possible value of $\mathtt{h}$ in the incoming shadow H,

Construct a pair containing the evaluation of E for that value of $\mathtt{h}$ (and the incoming v),

And associate with it a new, possibly smaller H including only those h's that would have given the same E.

Semantic definitions — qualitative sample

Assign to visible — qualitative

$$[\![\mathtt{v}{:=}\,E]\!](v,H) \quad = \quad \{(E^{v}_{\mathtt{v}}, \{h{:}\,H \mid E^{v}_{\mathtt{v}} {=} E^{v,h}_{\mathtt{v},\mathtt{h}}\}) \mid \mathtt{h}{:}\,H\}$$

where $E^{v}_{\mathtt{v}}$ denotes replacement of $\mathtt{v}$ by v throughout E

For each possible value of $\mathtt{h}$ in the incoming shadow H,

Construct a pair containing the evaluation of E for that value of $\mathtt{h}$ (and the incoming v),

And associate with it a new, possibly smaller H including only those h's that would have given the same E.

Semantic definitions — qualitative sample

Assign to visible — qualitative

$$[\![\mathtt{v}{:=}\,E]\!](v,H) \quad = \quad \{(E^{v}_{\mathtt{v}}, \{h{:}\,H \mid E^{v}_{\mathtt{v}} {=} E^{v,h}_{\mathtt{v},\mathtt{h}}\}) \mid \mathtt{h}{:}\,H\}$$

where $E^{v}_{\mathtt{v}}$ denotes replacement of $\mathtt{v}$ by v throughout E

For each possible value of $\mathtt{h}$ in the incoming shadow H,

Construct a pair containing the evaluation of E for that value of $\mathtt{h}$ (and the incoming v),

And associate with it a new, possibly smaller H including only those h's that would have given the same E: it's a sort of conditioning.

Semantic definitions — qualitative sample

Assign to visible — qualitative

$$[\![\mathtt{v}{:=}\,E]\!](v,H) \quad = \quad \{(E^v_{\mathtt{v}}, \{h{:}\,H \mid E^v_{\mathtt{v}} = E^{v,h}_{\mathtt{v},\mathtt{h}}\}) \mid \mathtt{h}{:}\,H\}$$

where $E^v_{\mathtt{v}}$ denotes replacement of $\mathtt{v}$ by v throughout E

$$\begin{array}{ll} & [\![\mathtt{v}{:=}\,\mathtt{h}\ \mathbf{mod}\ 2]\!](v,\{2,3,4\}) \\ = & \{((\mathtt{h}\ \mathbf{mod}\ 2)^v_{\mathtt{v}}, \{h{:}\,\{2,3,4\} \mid (\mathtt{h}\ \mathbf{mod}\ 2)^v_{\mathtt{v}} = (\mathtt{h}\ \mathbf{mod}\ 2)^{v,h}_{\mathtt{v},\mathtt{h}}\}) \mid \\ & \quad \mathtt{h}{:}\,\{2,3,4\}\} \\ = & \{(\mathtt{h}\ \mathbf{mod}\ 2, \{h{:}\,\{2,3,4\} \mid \mathtt{h}\ \mathbf{mod}\ 2 = h\ \mathbf{mod}\ 2\}) \mid \\ & \quad \mathtt{h}{:}\,\{2,3,4\}\} \\ = & \{(0,\{2,4\}),(1,\{3\}),(0,\{2,4\})\} \\ \\ = & \{(0,\{2,4\}),(1,\{3\})\} \end{array}$$

Semantic definitions — quantitative sample

Assign to visible — quantitative

$$[\![\mathtt{v}{:=}\,E]\!](v, H)^{\delta} \quad = \quad \{(E^{v}_{\mathtt{v}}, \{h{:}\,H \mid E^{v}_{\mathtt{v}} = E^{v,h}_{\mathtt{v},\mathtt{h}}\}) \mid \mathtt{h}{:}\,H\}$$

where $E^{v}_{\mathtt{v}}$ denotes replacement of $\mathtt{v}$ by v throughout E

For each possible value of `h` in the support of the incoming inner δ,

Construct a pair containing the evaluation of E for that value of `h` (and the incoming v), and associate with it the probability in δ of the `h` that gave rise to it,

And associate with it a new δ conditioned on the fact that the same value of `v` is produced.

Semantic definitions — quantitative sample

Assign to visible — quantitative

$$[\![\mathtt{v}{:=}\,E]\!](v,H)^{\delta} \quad = \quad \{(E^{v}_{\mathtt{v}}, \{h{:}\,H \mid E^{v}_{\mathtt{v}} = E^{v,h}_{\mathtt{v},\mathtt{h}}\}) \mid \mathtt{h}{:}\,H\}$$

where $E^{v}_{\mathtt{v}}$ denotes replacement of $\mathtt{v}$ by v throughout E

First define function $f_v(h) = E^{v,h}_{\mathtt{v},\mathtt{h}}$ of type $\mathcal{H}{\to}\mathcal{V}$.

Then define $\delta_{v,v'}$ for $v, v'{:}\,\mathcal{V}$ as the conditional distribution given by $\Pr(h \mid f_v(h) = v')$ where Pr refers to δ.

Finally define pairing function $g_v(h) = (f_v(h), \delta_{v,f_v(h)})$ of type $\mathcal{H} \to \mathcal{V}{\times}\mathbb{D}\mathcal{H}$ by combining the two.

The output hyperdistribution is the *push-forward* given by $(g_v)_*(\delta)$ of g_v over δ.

Semantic definitions — quantitative sample

Given two sets X, Y and a function $f\colon X \to Y$ between them, the *push-forward* f_* of f acts between the distributions $\mathbb{D}X, \mathbb{D}Y$ over those sets, thus giving $f_*\colon \mathbb{D}X \to \mathbb{D}Y$.

In the discrete case, for $\delta\colon \mathbb{D}X$ we can define

$$f_*(\delta)(y) \quad := \quad \sum_{\substack{x\colon X \\ f(x)=y}} \quad \delta(x) \ .$$

Then the output hyperdistribution is the push-forward given by $(g_v)_*(\delta)$ of g_v over δ.

Semantic definitions — quantitative sample

Assign to visible — quantitative

$$[\![\mathtt{v}{:=}\,E]\!](v,H)^{\delta} \quad = \quad \{(E^{v}_{\mathtt{v}}, \{h{:}\,H \mid E^{v}_{\mathtt{v}} = E^{v,h}_{\mathtt{v},\mathtt{h}}\}) \mid \mathtt{h}{:}\,H\}$$

where $E^{v}_{\mathtt{v}}$ denotes replacement of $\mathtt{v}$ by v throughout E

First define function $f_v(h) = E^{v,h}_{\mathtt{v},\mathtt{h}}$ of type $\mathcal{H}{\to}\mathcal{V}$.

Then define $\delta_{v,v'}$ for $v, v'{:}\,\mathcal{V}$ as the conditional distribution given by $\Pr(h \mid f_v(h) = v')$ where Pr refers to δ.

Finally define pairing function $g_v(h) = (f_v(h), \delta_{v,f_v(h)})$ of type $\mathcal{H} \to \mathcal{V}{\times}\mathbb{D}\mathcal{H}$ by combining the two.

The output hyperdistribution is the *push-forward* given by $(g_v)_*(\delta)$ of g_v over δ.

Semantic definitions — quantitative sample

Assign to visible — quantitative

$$[\![\mathtt{v}{:=}\,E]\!](v,\delta) \quad = \quad \{\!\{(E^{v}_{\mathtt{v}}, \{\!\{h{:}\,\delta \mid E^{v}_{\mathtt{v}} {=} E^{v,h}_{\mathtt{v},\mathtt{h}}\}\!\}) \mid \mathtt{h}{:}\,\delta\}\!\}$$

where $E^{v}_{\mathtt{v}}$ denotes replacement of $\mathtt{v}$ by v throughout E

An appropriate notation for *Distribution Comprehensions* makes this a conditional-distribution operator...

...so that the qualitative- and quantitative definitions are very similar in appearance...

...but we won't describe that notation in detail in this talk.

struct distribution $\delta_{v,v'}$ for $v, v' : \mathcal{V}$ to be ... distribution given by $\Pr(h \mid f_v(h) = v')$ where Pr refers to δ. Define pairing function $g_v(h) = (f_v(h), \delta_{v,f_v(h)})$ of type $\mathcal{H} \to \mathcal{V} \times \mathbb{D}\mathcal{H}$ by combining the two.

Then the output hyperdistribution is the push-forward given by $(g_v)_*(\delta)$ of g_v over δ.

Semantic definitions — quantitative sample

Assign to visible — quantitative

$$[\![\mathtt{v}{:=}\,E]\!](v,\delta) \quad = \quad \{\!\{(E^{v}_{\mathtt{v}}, \{\!\{h{:}\,\delta \mid E^{v}_{\mathtt{v}} {=} E^{v,h}_{\mathtt{v},\mathtt{h}}\}\!\}) \mid \mathtt{h}{:}\,\delta\}\!\}$$

where $E^{v}_{\mathtt{v}}$ denotes replacement of $\mathtt{v}$ by v throughout E

An appropriate notation for *Distribution Comprehensions* makes this a conditional-distribution operator...

...so that the qualitative- and quantitative definitions are very similar in appearance...

...but we won't describe that notation in detail in this talk.

Define function $f_v(h) \ldots E^{v,h}_{\mathtt{v},\mathtt{h}}$ of type $\mathcal{H} \to \mathcal{V}$. Construct distribution $\delta_{v,v'}$ for $v, v' : \mathcal{V}$ to be the conditional distribution given by $\Pr(h \mid f_v(h) = v')$ where Pr refers to δ. Define pairing function $g_v(h) = (f_v(h), \delta_{v,f_v(h)})$ of type $\mathcal{H} \to \mathcal{V} \times \mathbb{D}\mathcal{H}$ by combining the two.

Then the output hyperdistribution is the push-forward given by $(g_v)_*(\delta)$ of g_v over δ.

Semantic definitions — quantitative sample

Assign to visible — quantitative

$$[\![\mathtt{v}{:=}\,E]\!](v,\delta) \quad = \quad \{\!\{(E^{v}_{\mathtt{v}}, \{\!\{h{:}\,\delta \mid E^{v}_{\mathtt{v}}{=}E^{v,h}_{\mathtt{v},\mathtt{h}}\}\!\}) \mid \mathtt{h}{:}\,\delta\}\!\}$$

where $E^{v}_{\mathtt{v}}$ denotes replacement of $\mathtt{v}$ by v throughout E

An appropriate notation for *Distribution Comprehensions* makes this a conditional-distribution operator...

...so that the qualitative- and quantitative definitions are very similar in appearance...

...but we won't describe that notation in detail in this talk.

$$\{\!\{(E^{v}_{\mathtt{v}}, \{\!\{h{:}\,H \mid E^{v}_{\mathtt{v}}{=}E^{v,h}_{\mathtt{v},\mathtt{h}}\}\!\}) \mid \mathtt{h}{:}\,H\}\!\}$$

Semantic definitions — quantitative sample

Assign to visible — quantitative

$$[\![\mathtt{v}{:=}\,E]\!](v,\delta) \quad = \quad \{\!\{(E^v_{\mathtt{v}}, \{\!\{h{:}\,\delta \mid E^v_{\mathtt{v}}{=}E^{v,h}_{\mathtt{v},\mathtt{h}}\}\!\}) \mid \mathtt{h}{:}\,\delta\}\!\}$$

where $E^v_{\mathtt{v}}$ denotes replacement of $\mathtt{v}$ by v throughout E

An appropriate notation for *Distribution Comprehensions* makes this a conditional-distribution operator...

...so that the qualitative- and quantitative definitions are very similar in appearance...

$$[\![\mathtt{v}{:=}\,\mathtt{h}\ \mathbf{mod}\ 2]\!](v, \{\!\{2,3,4\}\!\})$$

$$= \quad \vdots$$

...and then this calculation follows.

$$= \quad \{\!\{(0, \{\!\{2,4\}\!\})^{@\frac{2}{3}}, (1, \{\!\{3\}\!\})^{@\frac{1}{3}}\}\!\}$$

COMPARATIVE SECURITY

Security breach — qualitative

Definition 1. *Elementary-Testing Order for Noninterference* We say that $S \preceq I$, that S and I are in the elementary-testing order ($\preceq$) for *qualitative* noninterference, just when from some initial state

functional testing If implementation I can produce $\mathtt{v}{=}v$ and $\mathtt{h}{=}h$ for some v, h, then so can its specification S.

security testing If implementation I can allow an attacker to determine the value of $\mathtt{h}$ by observation of $\mathtt{v}$, the control flow and the source code, then so can its specification S.

Compositional noninterference from first principles. Carroll Morgan. *Formal Aspects of Computing*, 2011. DOI: 10.1007/s00165-010-0167-y

Compositional closure — qualitative

The order given in Definition 1 is *not compositional*, because it is not preserved by contexts. (More exactly, the denotations of contexts are not monotonic functions with respect to that order.)

So we define the qualitative *refinement order* to be the (unique) weakest strengthening that is compositional wrt to the programming language (and its meanings) we have defined, the so-called *compositional closure*.

Compositional noninterference from first principles. Carroll Morgan. *Formal Aspects of Computing*, 2011. DOI: 10.1007/s00165-010-0167-y

Compositional closure — qualitative

Recall that programs (denotations) are of type $(\mathcal{V}\times\mathbb{P}\mathcal{H})\to\mathbb{P}(\mathcal{V}\times\mathbb{P}\mathcal{H})$. Reasoning pointwise...

...for $S, I : \mathbb{P}(\mathcal{V}\times\mathbb{P}\mathcal{H})$ say that $S \sqsubseteq I$, that S *is refined by* I just when every (v, H) in I equals $(v, H_0 \cup H_1 \cup \cdots)$ for some collection of (v, H_i)'s in S.

In other words, a specification is refined by "unioning together" split-states that have the same v component. The canonical example is

$$\mathtt{h}{:=}\,0 \sqcap \mathtt{h}{:=}\,1 \quad \sqsubseteq \quad \mathtt{h}{:=}\,0 \sqcap 1 \ .$$

Alternative, monadic formulation

For simplicity we concentrate on $\mathcal{H}$ only, so that our output space is $\mathbb{P}(\mathbb{P}\mathcal{H})$, that is $\mathbb{P}^2\mathcal{H}$ i.e. (non-empty) sets of (non-empty) shadows. For $S, I : \mathbb{P}^2\mathcal{H}$ say that $S \sqsubseteq I$ just when there exists $X : \mathbb{P}^3\mathcal{H}$ such that

Multiply for the monad

$$S \supseteq \cup X \quad \text{and} \quad (\mathbb{P}\cup)X = I \ ,$$

The powerset functor.

where $\cup X$ is the union of all sets in X and $(\mathbb{P}\cup)X$ is the set formed by applying $(\cup)$ to all elements of X.

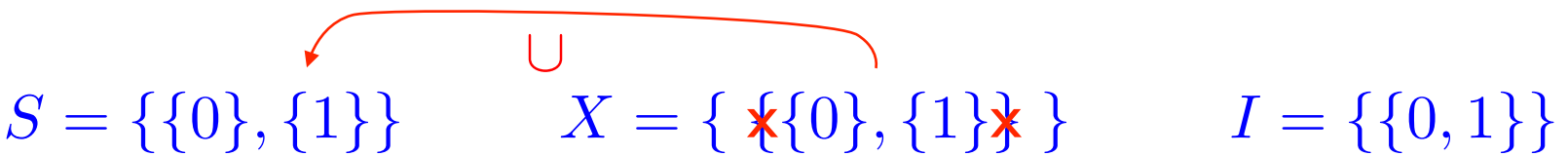
$S = \{\{0\}, \{1\}\}$ $\quad$ $X = \{\ \{\{0\}, \{1\}\}\ \}$ $\quad$ $I = \{\{0, 1\}\}$

Alternative, monadic formulation

For simplicity we concentrate on $\mathcal{H}$ only, so that our output space is $\mathbb{P}(\mathbb{P}\mathcal{H})$, that is $\mathbb{P}^2\mathcal{H}$ i.e. (non-empty) sets of (non-empty) shadows. For $S, I : \mathbb{P}^2\mathcal{H}$ say that $S \sqsubseteq I$ just when there exists $X : \mathbb{P}^3\mathcal{H}$ such that

$$S \supseteq \cup X \quad \text{and} \quad (\mathbb{P}\cup)X = I \ ,$$

where $\cup X$ is the union of all sets in X and $(\mathbb{P}\cup)X$ is the set formed by applying $(\cup)$ to all elements of X.

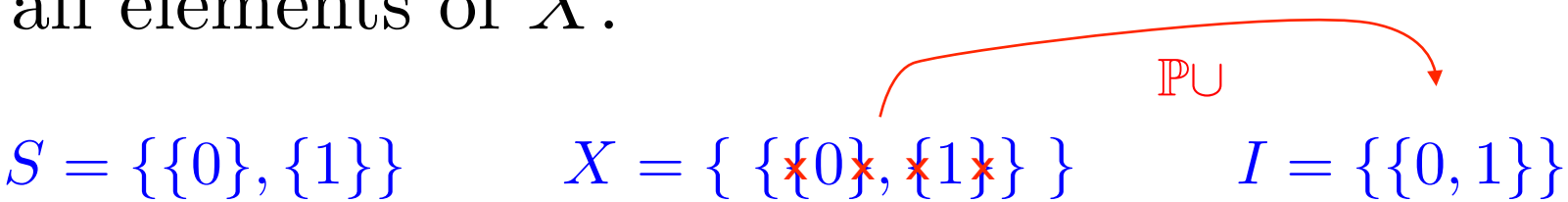


Alternative, monadic formulation

For simplicity we concentrate on $\mathcal{H}$ only, so that our output space is $\mathbb{P}(\mathbb{P}\mathcal{H})$, that is $\mathbb{P}^2\mathcal{H}$ i.e. (non-empty) sets of (non-empty) shadows. For $S, I : \mathbb{P}^2\mathcal{H}$ say that $S \sqsubseteq I$ just when there exists $X : \mathbb{P}^3\mathcal{H}$ such that

$$S \supseteq \cup X \quad \text{and} \quad (\mathbb{P}\cup)X = I \ ,$$

where $\cup X$ is the union of all sets in X and $(\mathbb{P}\cup)X$ is the set formed by applying $(\cup)$ to all elements of X.



Security breach — quantitative

Definition 1. *Elementary-Testing Order for Noninterference* We say that $S \preceq I$, that S and I are in the elementary-testing order ($\preceq$) for *quantitative* noninterference, just when from some initial state

functional testing For any v, h the specification and implementation produce that pair with equal probabilities.

security testing For any observed v (and possibly other observations based on perfect recall and implicit flow), the *Bayes Vulnerability* of $\mathtt{h}$ in the specification is never increased in the implementation.

Compositional closure for Bayes Risk in probabilistic noninterference.
McIver, Meinicke, Morgan. Proc *ICALP* 2010.

Bayes Risk/Vulnerability

The *Bayes Vulnerability* of $\mathtt{h}$ given that $\mathtt{v}{=}v$ (and possibly other observations) is the weighted average, across all those values v (and observations), of the conditional probability of the most likely value h of $\mathtt{h}$, the *maximum a posteriori probability* (MAP) of $\mathtt{h}$.

Adversaries and information leaks. G. Smith. *TGC* 2007.

Monadic definition of quantitative refinement

For simplicity we concentrate on $\mathcal{H}$ only, so that our output space is $\mathbb{D}^2\mathcal{H}$ i.e. hyper distributions, distributions of (inner) distributions. For $S, I : \mathbb{D}^2\mathcal{H}$ say that $S \sqsubseteq I$ just when there exists $X : \mathbb{D}^3\mathcal{H}$ such that

$$S = \mathsf{avg}(X) \quad \text{and} \quad (\mathbb{D}\,\mathsf{avg})(X) = I \ ,$$

where $\mathsf{avg}(X)$ is the average of all elements in X and $(\mathbb{D}\,\mathsf{avg})X$ is the push-forward of avg.

$$S = \{\!\{\{\!\{0\}\!\}, \{\!\{1\}\!\}\}\!\} \quad X = \{\!\{\ \{\!\{\{\!\{0\}\!\}, \{\!\{1\}\!\}\}\!\}\ \}\!\} \quad I = \{\!\{\{\!\{0, 1\}\!\}\}\!\}$$

Monadic definition of quantitative refinement

For simplicity we concentrate on $\mathcal{H}$ only, so that our output space is $\mathbb{D}^2\mathcal{H}$ i.e. hyper distributions, distributions of (inner) distributions. For $S, I : \mathbb{D}^2\mathcal{H}$ say that $S \sqsubseteq I$ just when there exists $X : \mathbb{D}^3\mathcal{H}$ such that

$$S = \mathsf{avg}(X) \quad \text{and} \quad (\mathbb{D}\,\mathsf{avg})(X) = I \ ,$$

where $\mathsf{avg}(X)$ is the average of all elements in X and $(\mathbb{D}\,\mathsf{avg})X$ is the push-forward of avg.

$$S = \{\!\{\{\!\{0\}\!\}, \{\!\{1\}\!\}\}\!\} \quad X = \{\!\{\ \{\!\{\{\!\{0\}\!\}, \{\!\{1\}\!\}\}\!\}\ \}\!\} \quad I = \{\!\{\{\!\{0, 1\}\!\}\}\!\}$$

Monadic definition of quantitative refinement

For simplicity we concentrate on $\mathcal{H}$ only, so that our output space is $\mathbb{D}^2\mathcal{H}$ i.e. hyper distributions, distributions of (inner) distributions. For $S, I : \mathbb{D}^2\mathcal{H}$ say that $S \sqsubseteq I$ just when there exists $X : \mathbb{D}^3\mathcal{H}$ such that

$$S \supseteq \cup\!:(X) \quad \text{and} \quad (\ \mathbb{P}\cup\ )(X) = I\ ,$$

where $\mathsf{avg}(X)$ is the average of all elements in X and $(\mathbb{D}\,\mathsf{avg})X$ is the push-forward of avg.

$$S = \{\!\{\{\!\{0\}\!\}, \{\!\{1\}\!\}\}\!\} \quad X = \{\!\{\ \{\!\{\{\!\{0\}\!\}, \{\!\{1\}\!\}\}\!\}\ \}\!\} \quad I = \{\!\{\{\!\{0, 1\}\!\}\}\!\}$$

Monadic definition of quantitative refinement

For simplicity we concentrate on $\mathcal{H}$ only, so that our output space is $\mathbb{D}^2\mathcal{H}$ i.e. hyper distributions, distributions of (inner) distributions. For $S, I : \mathbb{D}^2\mathcal{H}$ say that $S \sqsubseteq I$ just when there exists $X : \mathbb{D}^3\mathcal{H}$ such that

Multiply for the monad

$$S = \mathsf{avg}(X) \quad \text{and} \quad (\mathbb{D}\,\mathsf{avg})(X) = I \ ,$$

The Kantorovich functor

where $\mathsf{avg}(X)$ is the average of all elements in X and $(\mathbb{D}\,\mathsf{avg})X$ is the push-forward of avg.

$$S = \{\!\{\{\!\{0\}\!\}, \{\!\{1\}\!\}\}\!\} \quad X = \{\!|\ \{\!\{\{\!\{0\}\!\}, \{\!\{1\}\!\}\}\!\}\ |\!\} \quad I = \{\!\{\{\!\{0, 1\}\!\}\}\!\}$$

This formulation works well for distributions of infinite support, and (ultimately) proper measures. For Kantorovich, see van Breugel, *The Metric Monad for Probabilistic Nondeterminism*. 2005.

Monadic definition of quantitative refinement

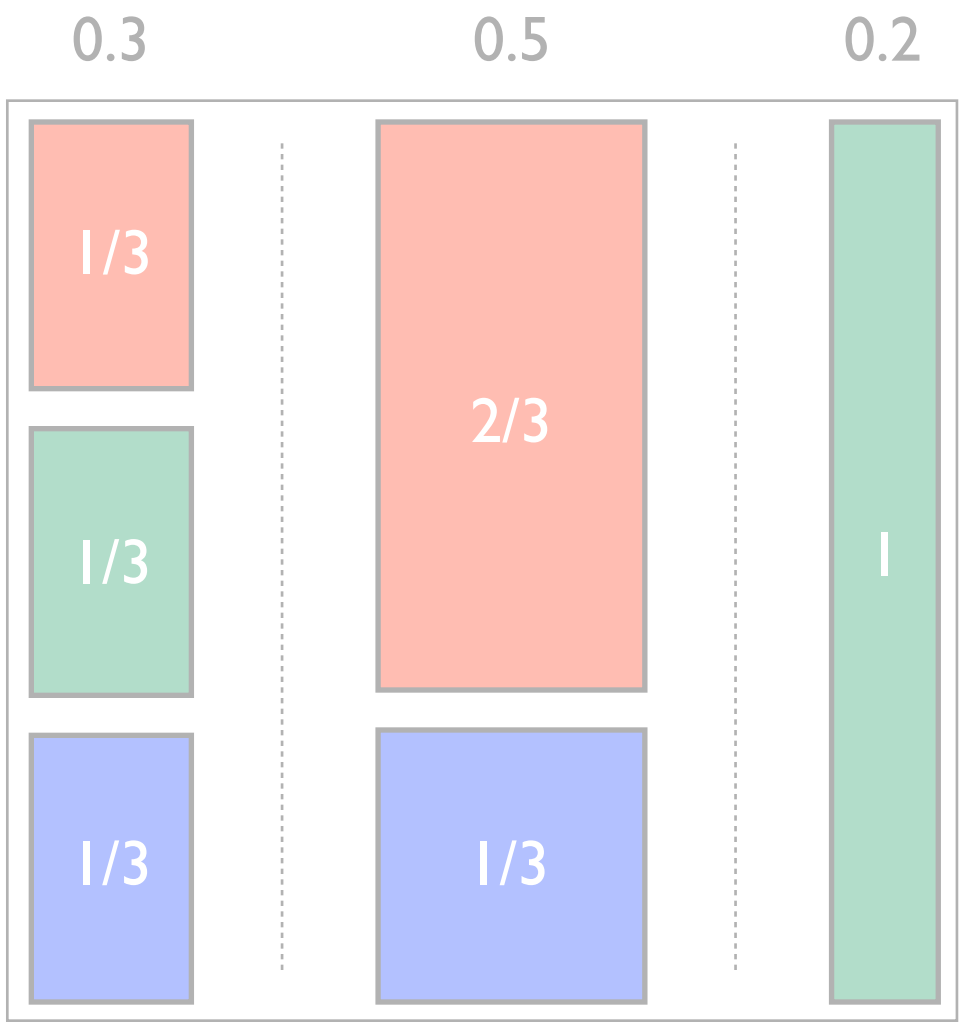
For simplicity we concentrate on $\mathcal{H}$ only, so that our output space is $\mathbb{D}^2\mathcal{H}$ i.e. hyper distributions, distributions of (inner) distributions. For $S, I : \mathbb{D}^2\mathcal{H}$ say that $S \sqsubseteq I$ just when there exists $X : \mathbb{D}^3\mathcal{H}$ such that

$$S = \mathsf{avg}(X) \quad \text{and} \quad (\mathbb{D}\,\mathsf{avg})(X) = I \ ,$$

where $\mathsf{avg}(X)$ is the average of all elements in X and $(\mathbb{D}\,\mathsf{avg})X$ is the push-forward of avg.

$$S = \{\!\{\{\!\{0\}\!\}, \{\!\{1\}\!\}\}\!\} \quad X = \{\!\{\ \{\!\{\{\!\{0\}\!\}, \{\!\{1\}\!\}\}\!\}\ \}\!\} \quad I = \{\!\{\{\!\{0, 1\}\!\}\}\!\}$$

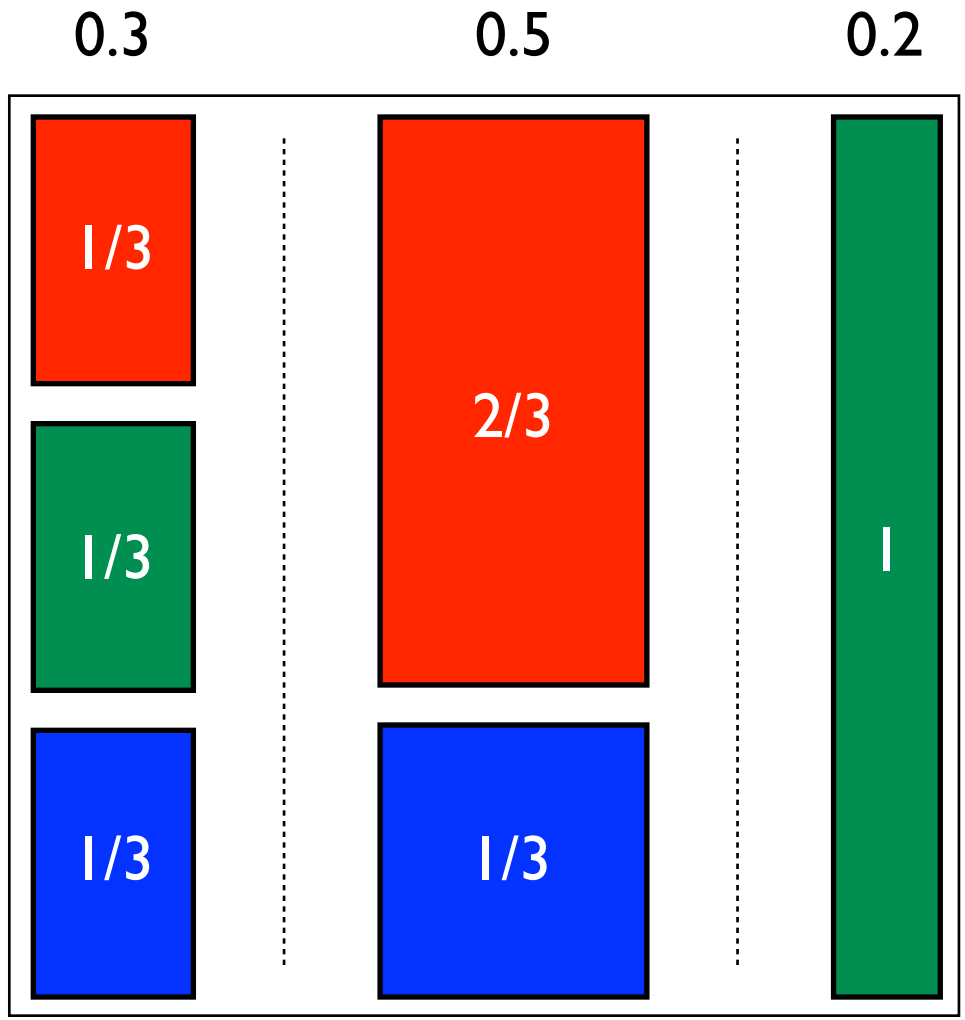
But in the discrete case, it's pretty straightforward: a "weighted sum" than (as earlier) a union.



Here's an example.

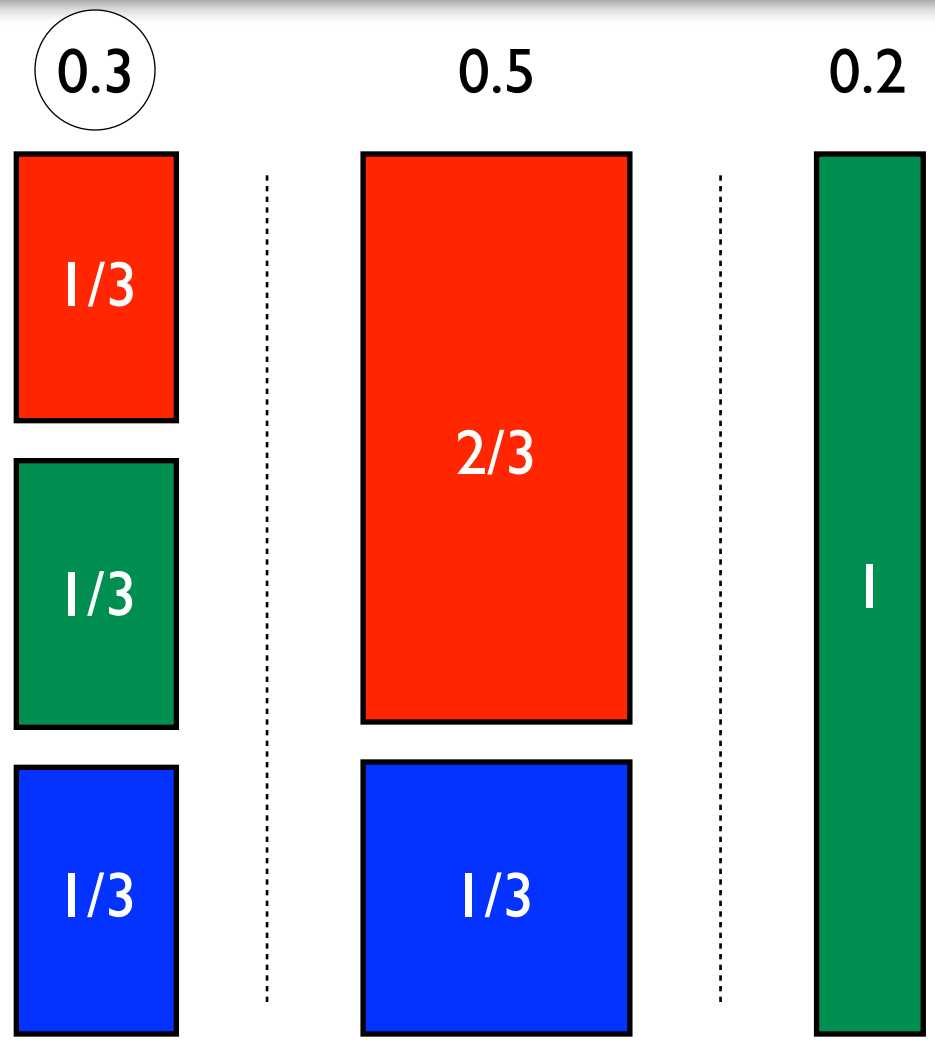
With probability 0.3 we know that $\mathtt{h}$ is distributed uniformly over R,G,B; with probability 0.5 we know it cannot be G, and is twice as likely to be R as B; with probability 0.2 we are certain it is G.

Specification,
over a split-state
of 3 elements
R,G,B,
given as an output
hyperdistribution
having three
inners.



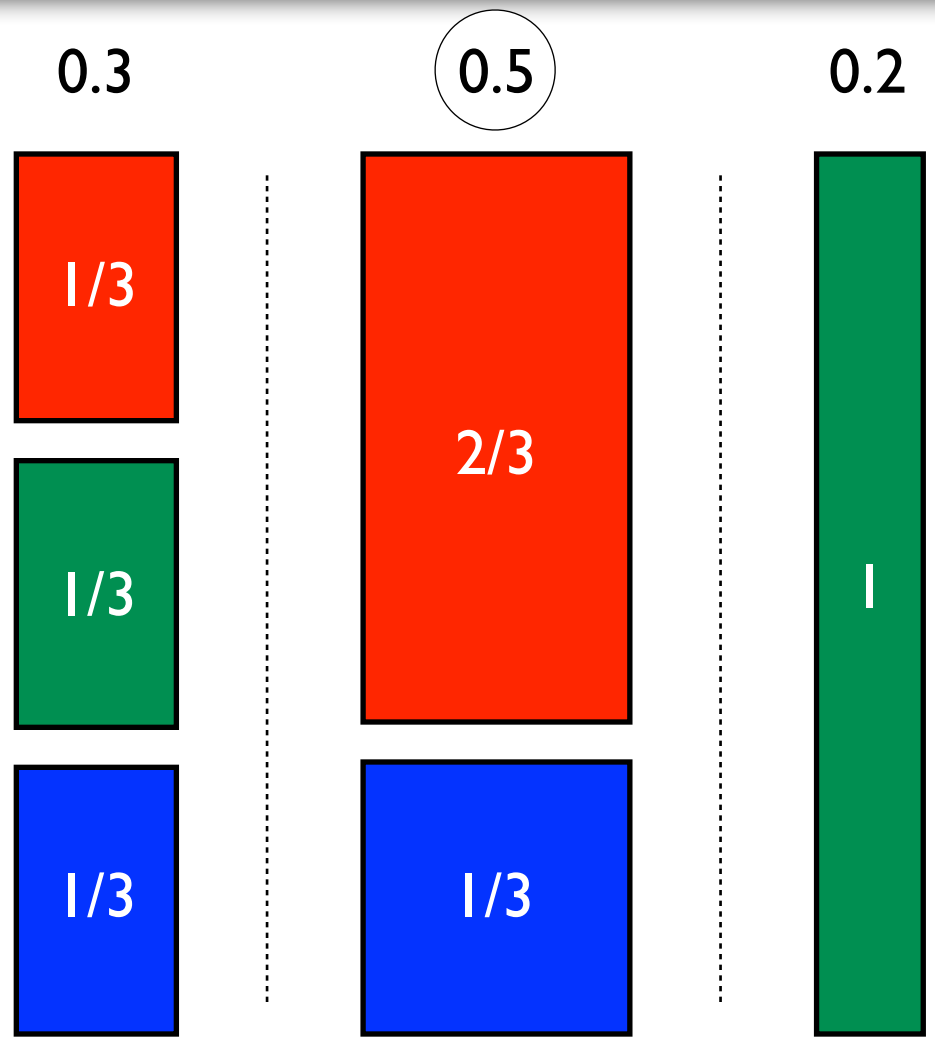
With probability 0.3 we know that $\mathtt{h}$ is distributed uniformly over R,G,B; with probability 0.5 we know it cannot be G, and is twice as likely to be R as B; with probability 0.2 we are certain it is G.

Specification,
over a split-state
of 3 elements
R,G,B,
given as an output
hyperdistribution
having three
inners.



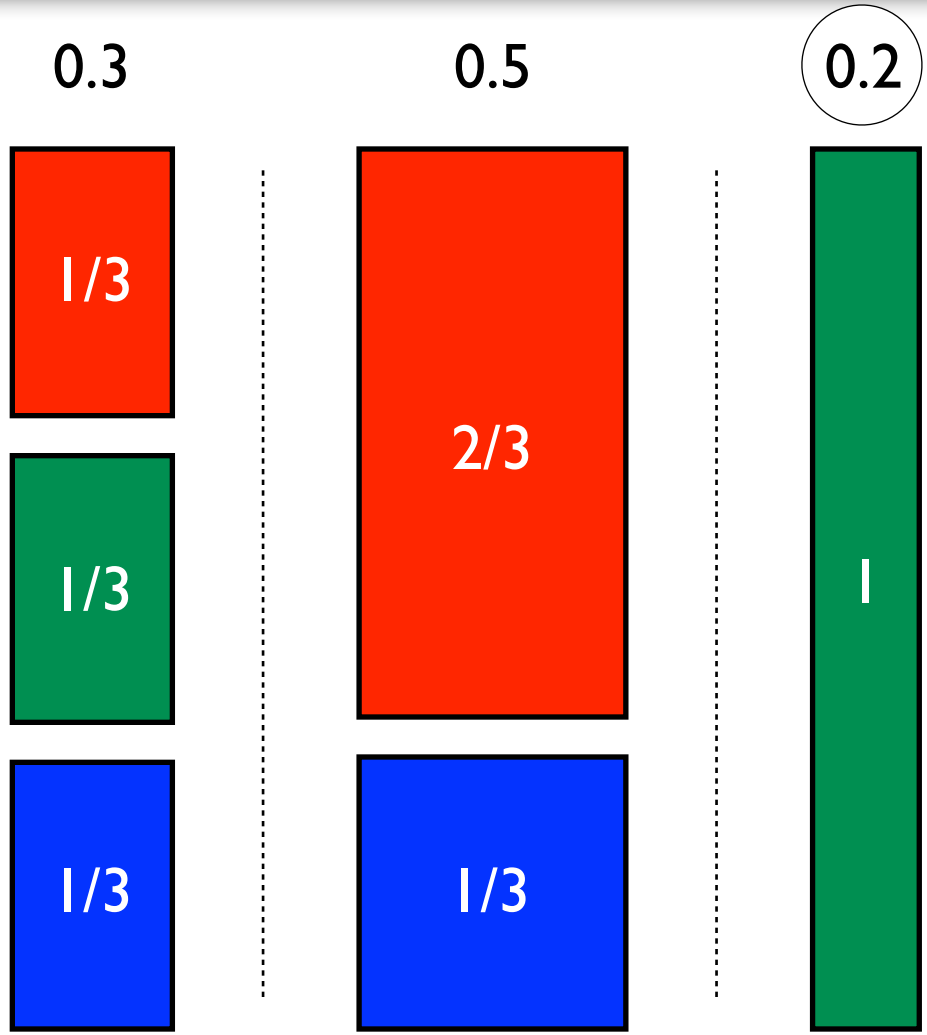
With probability 0.3 we know that $\mathtt{h}$ is distributed uniformly over R,G,B; with probability 0.5 we know it cannot be G, and is twice as likely to be R as B; with probability 0.2 we are certain it is G.

Specification,
over a split-state
of 3 elements
R,G,B,
given as an output
hyperdistribution
having three
inners.



With probability 0.3 we know that $\mathtt{h}$ is distributed uniformly over R,G,B; with probability 0.5 we know it cannot be G, and is twice as likely to be R as B; with probability 0.2 we are certain it is G.

Specification,
over a split-state
of 3 elements
R,G,B,
given as an output
hyperdistribution
having three
inners.



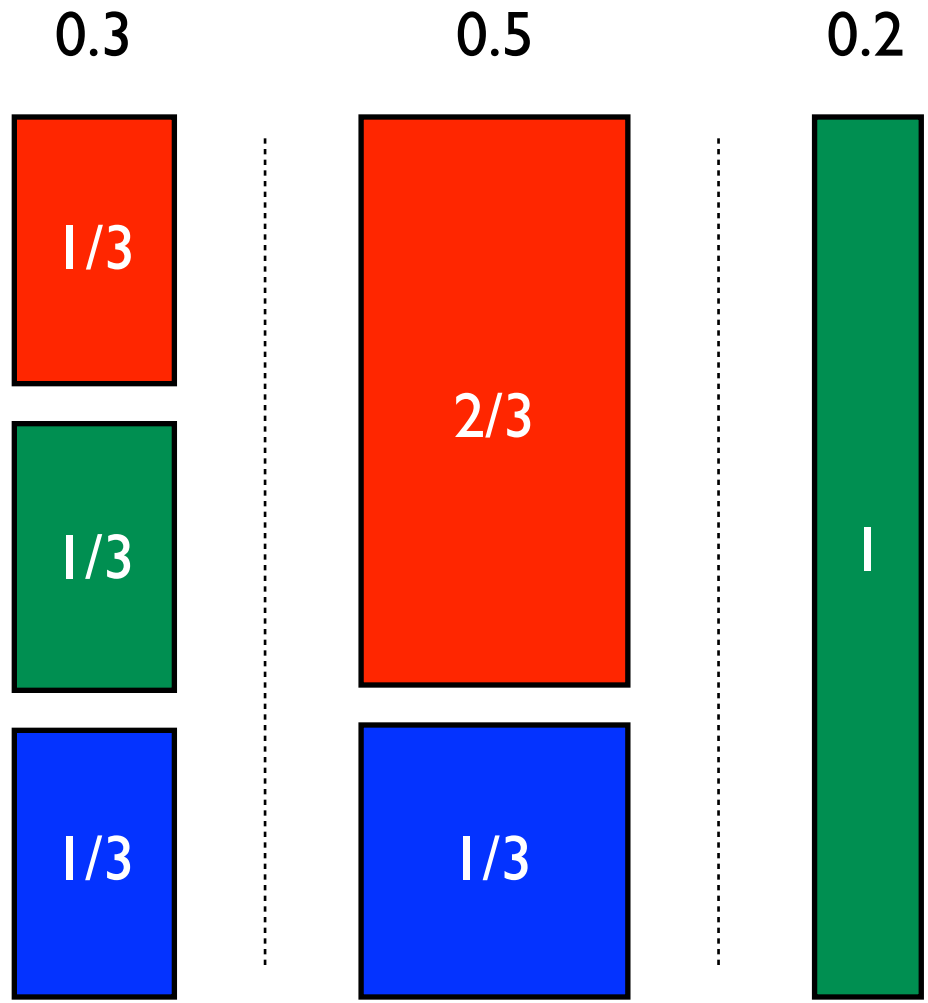
With probability 0.3 we know that h is distributed uniformly over R,G,B; with probability 0.5 we know it cannot be G, and is twice as likely to be R as B; with probability 0.2 we are certain it is G.

Specification

BV = 0.3/3 + 2(0.5)/3 + 0.2 = 0.633...

We illustrate quantitative refinement via a procedure of cut and paste.

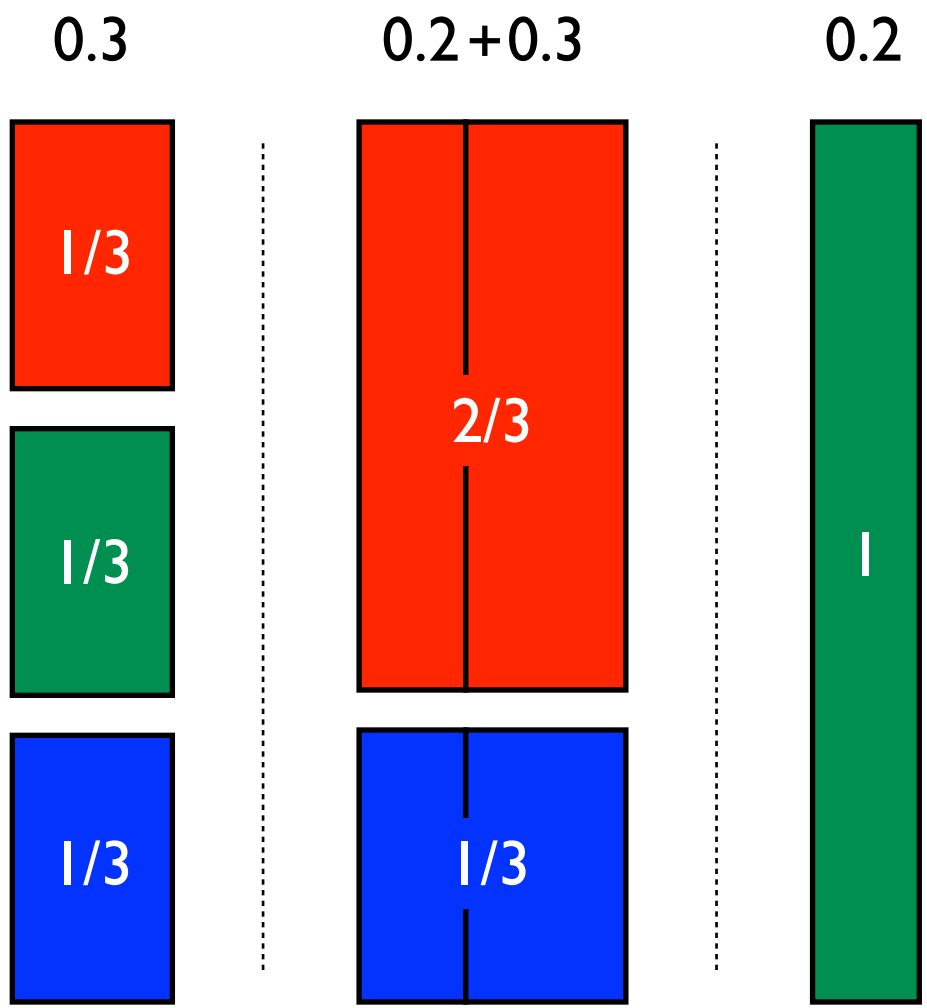
The BV should decrease, and that decrease should "flow through" to any context whatsoever.



Cut...

One or more inners can be cut into replicas of itself, provided the overall weight of each doesn't change.

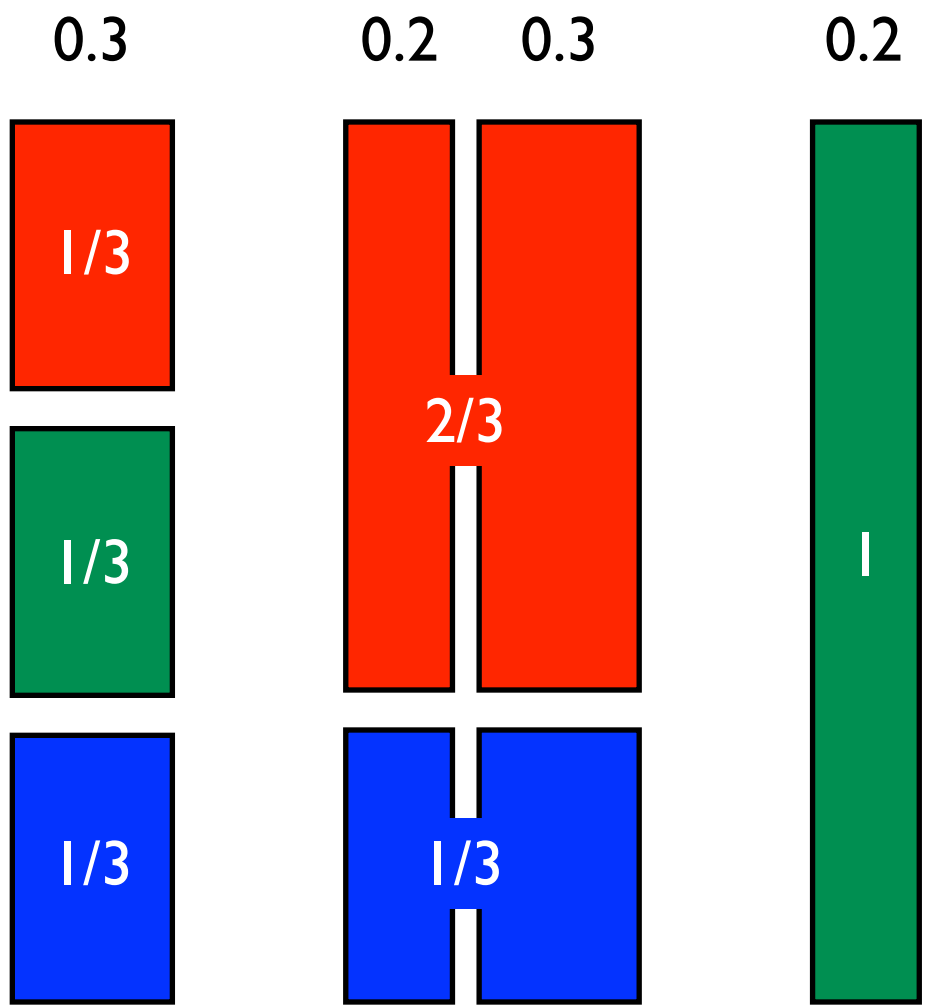
Here we have split the middle inner (of weight 0.5) into two replicas of weights 0.2 and 0.3.



Separate...

Now we have *four* inners (whereas we started with three); but two of them are the same.

The cutting makes no difference: in particular, it does not change the BV.

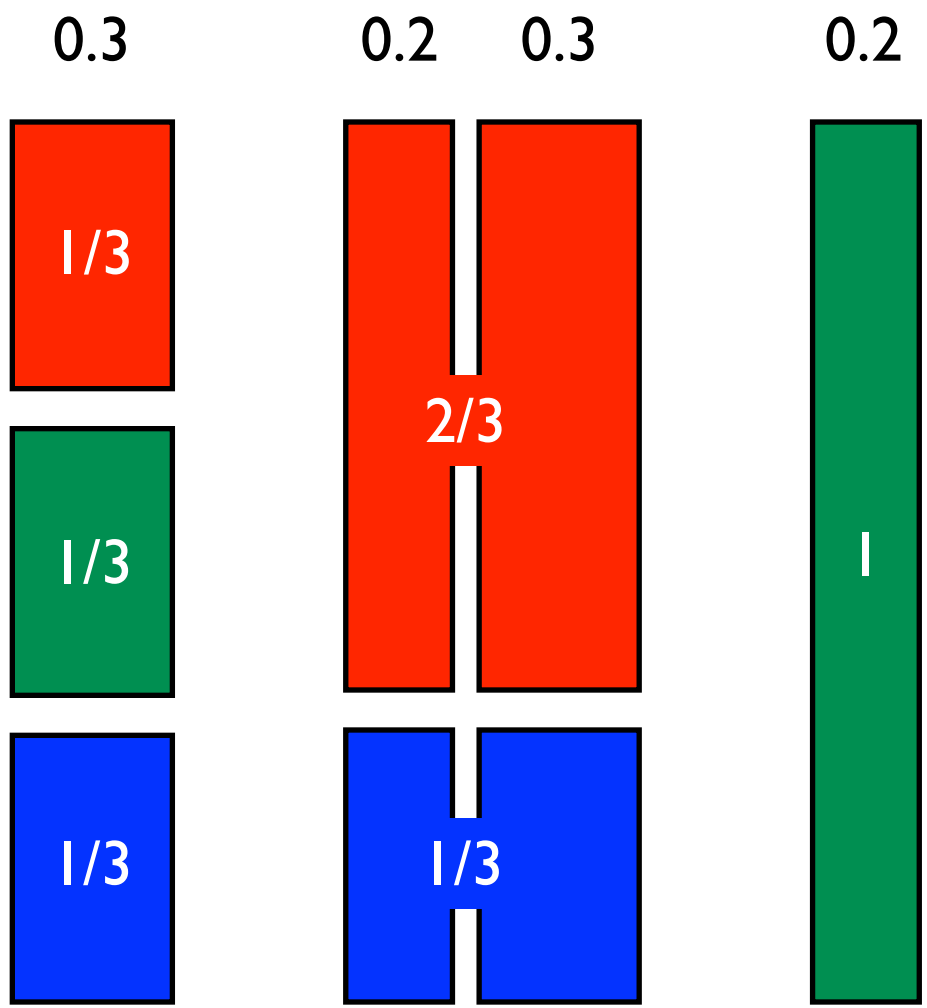


Separate...

Now we have *four* inners (whereas we started with three); but two of them are the same.

The cutting makes no difference: in particular, it does not change the BV.

But the next step does.

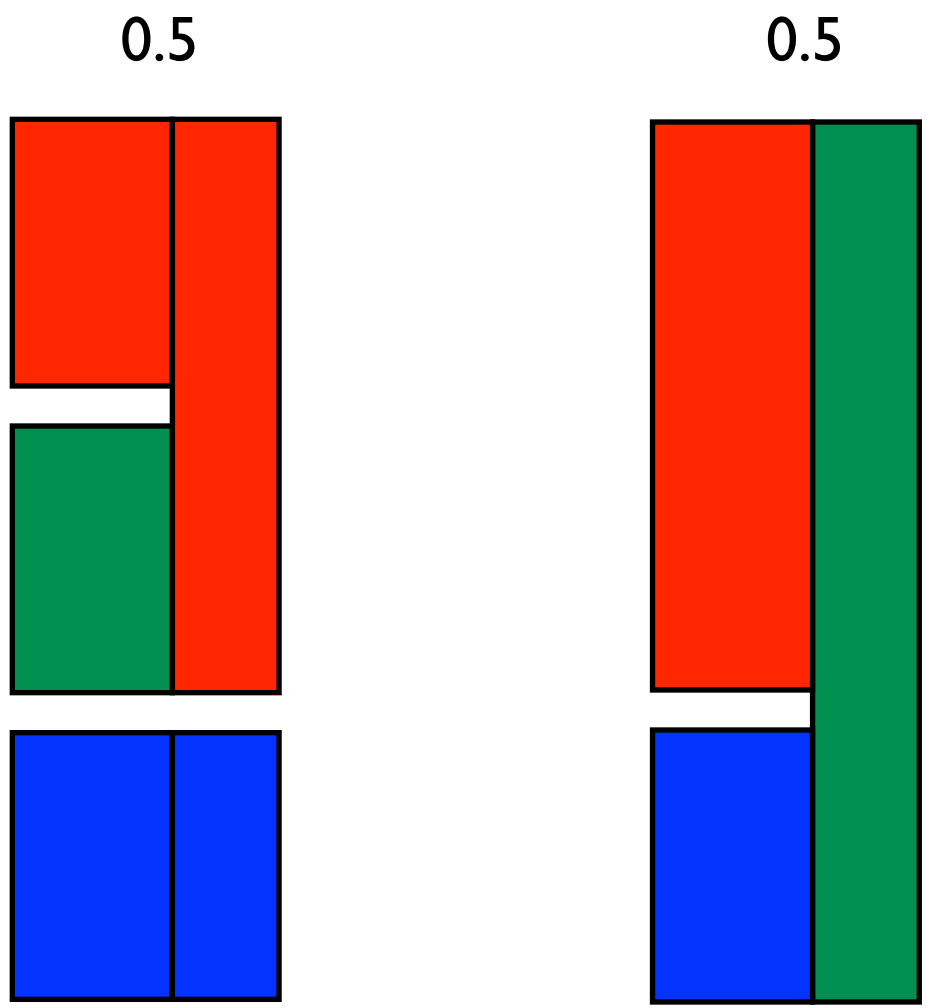


Merge...

After this cut-and-separate, the so-called *splitting* step (which can be the identity), we can then merge inners arbitrarily.

They are added together proportionally, based on their weights.

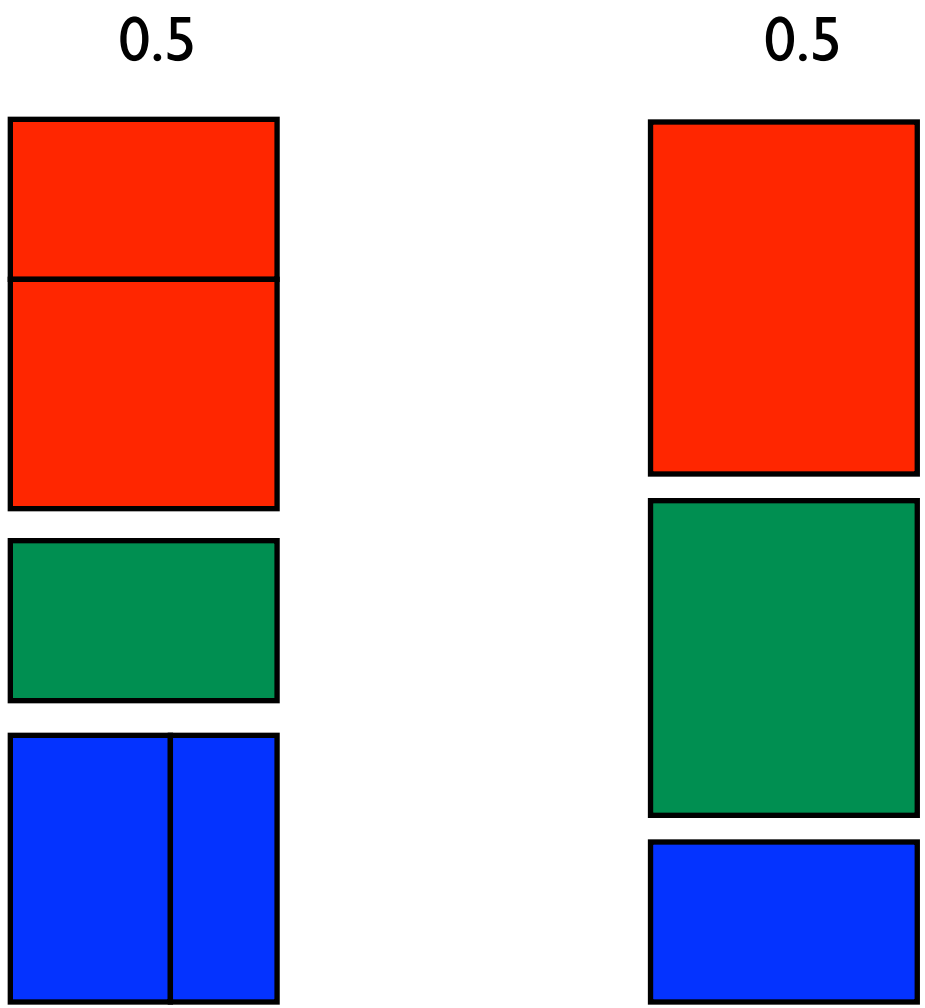
As a result, the BV probably decreases.



Normalise...

The result is (usually) a smaller number of inners.

In this case we started with three, via splitting made four, and then via merging reduced that to two.

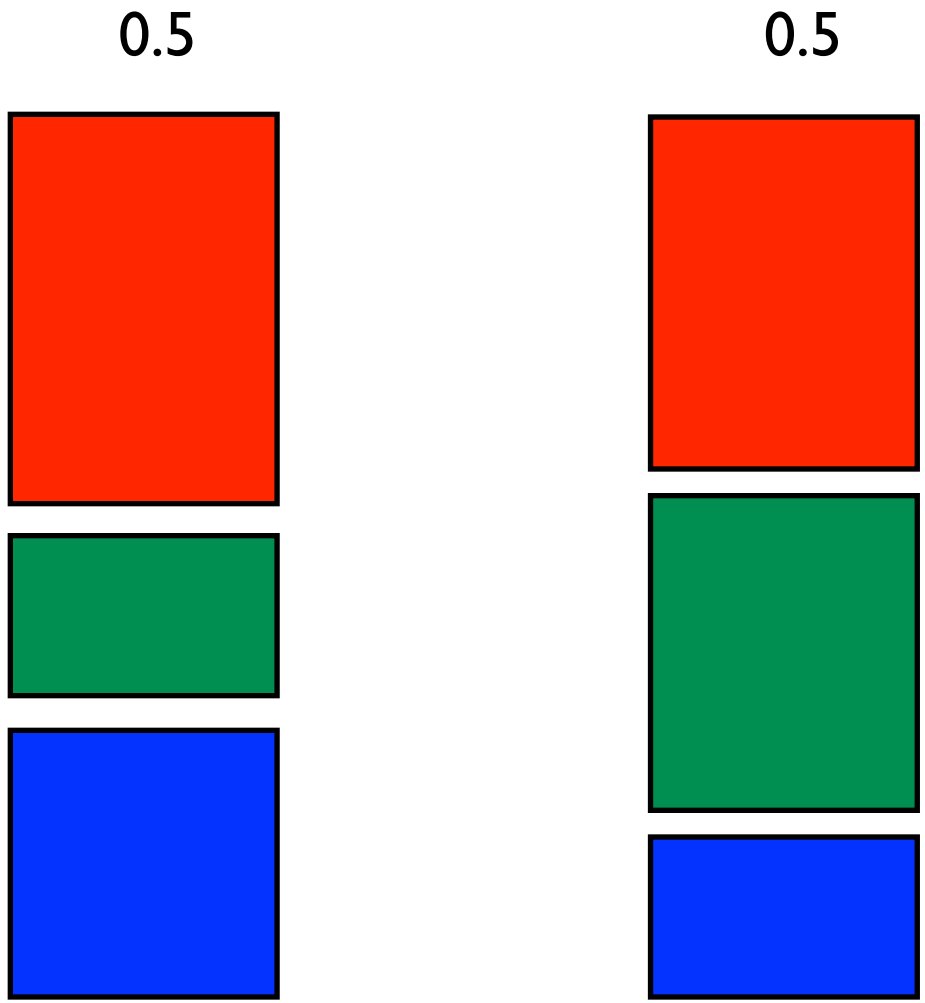


and Paste.

And we are done.

Notice that the total accumulated weight of each separate state R,G,B has not changed: consider the area occupied by each colour.

This is the functional equality that refinement demands (in the absence of demonic choice).

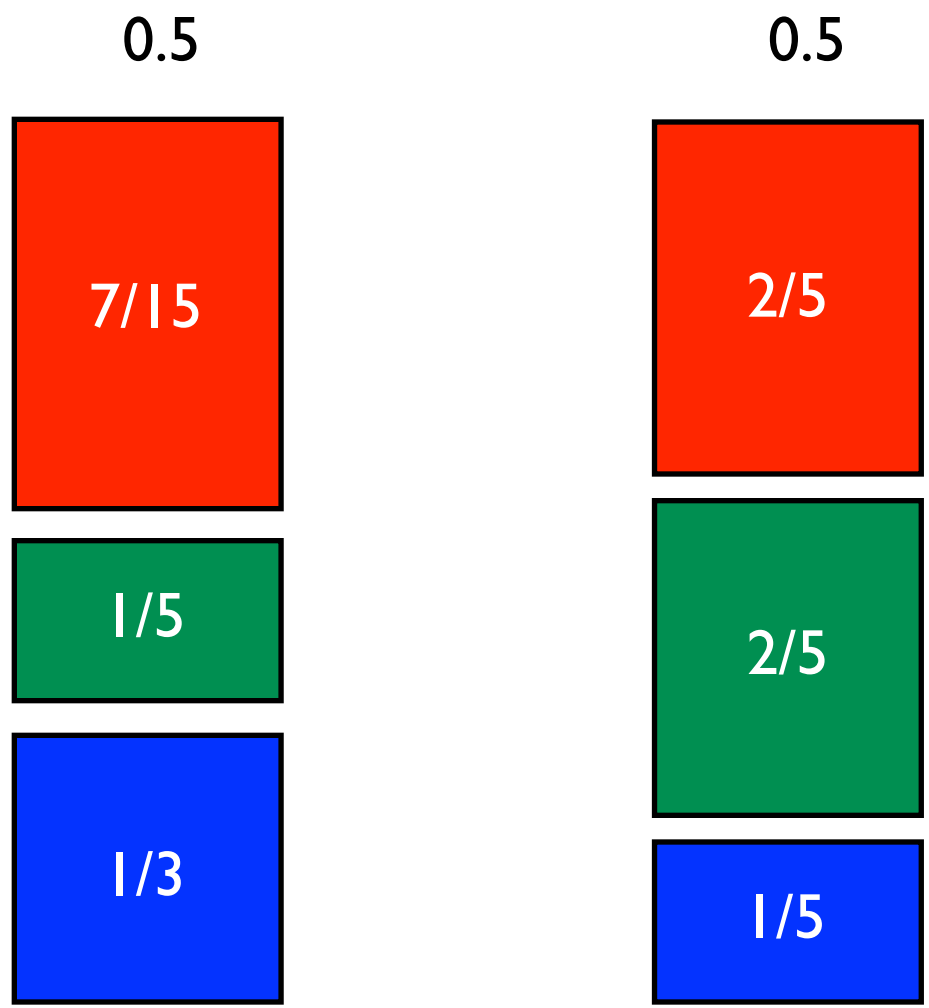


Half the time we know R,G,B are distributed in proportions 7,3,5; the other half of the time we know they are distributed in proportions 2,2,1.

Implementation

BV = 7(0.5)/15 + 2(0.5)/5 = 0.433... < 0.633...

This implementation hyperdistribution is (strictly) more secure than the specification hyper we started with, and that relation will be preserved in all programming contexts.



Compositional closure — quantitative

A specification is refined by "averaging together" split-states that have the same v component, according to their probabilities in the hyperdistribution.

The canonical example of this is

$$\mathtt{h}{:=}\,0 \oplus \mathtt{h}{:=}\,1 \quad \sqsubseteq \quad \mathtt{h}{:=}\,0 \oplus 1 \ .$$

$S = \{\!\{\{\!\{0\}\!\}, \{\!\{1\}\!\}\}\!\}$ $\quad X = \{\!|\ \{\!\{\{\!\{0\}\!\}, \{\!\{1\}\!\}\}\!\}\ |\!\}$ $\quad I = \{\!\{\{\!\{0,1\}\!\}\}\!\}$

Compositional closure — quantitative

A specification is refined by "averaging together" split-states that have the same v component, according to their probabilities in the hyperdistribution.

The canonical example of this is

$$\mathtt{h}{:=}\,0 \oplus \mathtt{h}{:=}\,1 \quad \sqsubseteq \quad \mathtt{h}{:=}\,0 \oplus 1 \ .$$

$1/2{\times}\{\!\{0\}\!\} + 1/2{\times}\{\!\{1\}\!\} \quad = \quad \{\!\{0,1\}\!\}$

$\mathbb{D}\mathsf{avg}$

$S = \{\!\{\{\!\{0\}\!\}, \{\!\{1\}\!\}\}\!\}$ $\quad X = \{\!\{ \{\!\{\{\!\{0\}\!\}, \{\!\{1\}\!\}\}\!\} \}\!\}$ $\quad I = \{\!\{\{\!\{0,1\}\!\}\}\!\}$

APPLICATIONS

Hierarchically structured protocols

The *Three Judges* protocol

Boolean/{0,1} hidden variables represent individual verdicts: guilty, or innocent. The aim is to reveal the majority verdict without revealing the individual verdicts.

vis v;
hid a, b, c;

$$\mathtt{v} := (\mathtt{a}+\mathtt{b}+\mathtt{c}) \geq 2$$

This is not (simply) the generalised Cryptographers: rather it is *Secret Voting* (Yao).

Hierarchically structured protocols

vis v;
hid a, b, c;

$$\mathtt{v} := (\mathtt{a}+\mathtt{b}+\mathtt{c}) \geq 2$$

This specification is very non-local: a sum must be constructed, preserving secrecy, of three variables held in three different places.

Our aim is to increase the locality while preserving both the functional and the security properties.

Hierarchically structured protocols

vis v;
hid a, b, c;

v:= (b∨c **if** a **else** b∧c)

This looks like an Oblivious Transfer; but its two arguments are still non-local. Thus we must go further...

vis v;
hid a, b, c;

v:= (a+b+c) ≥ 2

Hierarchically structured protocols

vis $\mathtt{v}$;
hid $\mathtt{a}, \mathtt{b}, \mathtt{c}$; **hid** $\mathtt{b}_0, \mathtt{b}_1$; **hid** $\mathtt{c}_0, \mathtt{c}_1$;

$\mathtt{v} := (\mathtt{b} \vee \mathtt{c}$ **if** $\mathtt{a}$ **else** $\mathtt{b} \wedge \mathtt{c})$

Hierarchically structured protocols

$$\textbf{vis } \mathsf{v};$$
$$\textbf{hid } \mathsf{a}, \mathsf{b}, \mathsf{c};\ \textbf{hid } \mathsf{b}_0, \mathsf{b}_1;\ \textbf{hid } \mathsf{c}_0, \mathsf{c}_1$$

$$(\mathsf{b}_0 \nabla \mathsf{c}_0) := \mathsf{b} \wedge \mathsf{c};$$
$$(\mathsf{b}_1 \nabla \mathsf{c}_1) := \mathsf{b} \vee \mathsf{c};$$

Choose variables "at random" on left-hand side to make their exclusive-or equal to the right-hand side.

$$\mathsf{v} := (\mathsf{b}_1 \nabla \mathsf{c}_1 \ \textbf{if } \mathsf{a} \ \textbf{else } \mathsf{b}_0 \nabla \mathsf{c}_0)$$

$$\textbf{vis } \mathsf{v};$$
$$\textbf{hid } \mathsf{a}, \mathsf{b}, \mathsf{c};\ \textbf{hid } \mathsf{b}_0, \mathsf{b}_1;\ \textbf{hid } \mathsf{c}_0, \mathsf{c}_1;$$
$$\mathsf{v} := (\mathsf{b} \vee \mathsf{c} \ \textbf{if } \mathsf{a} \ \textbf{else } \mathsf{b} \wedge \mathsf{c})$$

Hierarchically structured protocols

$$\textbf{vis } \mathtt{v};$$
$$\textbf{hid } \mathtt{a}, \mathtt{b}, \mathtt{c};\ \textbf{hid } \mathtt{b}_0, \mathtt{b}_1;\ \textbf{hid } \mathtt{c}_0, \mathtt{c}_1$$

$$(\mathtt{b}_0 \nabla \mathtt{c}_0) := \mathtt{b} \wedge \mathtt{c};$$
$$(\mathtt{b}_1 \nabla \mathtt{c}_1) := \mathtt{b} \vee \mathtt{c};$$

Choose variables "at random" on left-hand side to make their exclusive-or equal to the right-hand side.

$$\mathtt{v} := (\mathtt{b}_1 \nabla \mathtt{c}_1 \ \textbf{if } \mathtt{a} \ \textbf{else } \mathtt{b}_0 \nabla \mathtt{c}_0)$$

The "at random" choice is resolved either demonically or uniformly, depending on whether it is a qualitative or quantitative system.

Hierarchically structured protocols

$$\textbf{vis } \mathtt{v};$$
$$\textbf{hid } \mathtt{a}, \mathtt{b}, \mathtt{c};\ \textbf{hid } \mathtt{b}_0, \mathtt{b}_1;\ \textbf{hid } \mathtt{c}_0, \mathtt{c}_1$$

$$(\mathtt{b}_0 \nabla \mathtt{c}_0) := \mathtt{b} \wedge \mathtt{c};$$
$$(\mathtt{b}_1 \nabla \mathtt{c}_1) := \mathtt{b} \vee \mathtt{c};$$

Secure multi-party computations.

$$\mathtt{v} := (\mathtt{b}_1 \nabla \mathtt{c}_1 \ \textbf{if } \mathtt{a} \ \textbf{else } \mathtt{b}_0 \nabla \mathtt{c}_0)$$

Protocols for secure computations. A. C-C. Yao. *FOCS* 1982.
How to play any mental game. Goldreich, Micali, Wigderson. *STOC* 87.

Hierarchically structured protocols

$$\textbf{vis } \mathsf{v};$$
$$\textbf{hid } \mathsf{a}, \mathsf{b}, \mathsf{c};\ \textbf{hid } \mathsf{a}_b, \mathsf{a}_c;\ \textbf{hid } \mathsf{b}_0, \mathsf{b}_1;\ \textbf{hid } \mathsf{c}_0, \mathsf{c}_1$$

$$(\mathsf{b}_0 \nabla \mathsf{c}_0) := \mathsf{b} \wedge \mathsf{c};$$
$$(\mathsf{b}_1 \nabla \mathsf{c}_1) := \mathsf{b} \vee \mathsf{c};$$

$$\mathsf{a}_b := (\mathsf{b}_1 \textbf{ if } \mathsf{a} \textbf{ else } \mathsf{b}_0);$$
$$\mathsf{a}_c := (\mathsf{c}_1 \textbf{ if } \mathsf{a} \textbf{ else } \mathsf{c}_0);$$
$$\mathsf{v} := \mathsf{a}_b \nabla \mathsf{a}_c$$

$$\textbf{vis } \mathsf{v};$$
$$\textbf{hid } \mathsf{a}, \mathsf{b}, \mathsf{c};\ \textbf{hid } \mathsf{b}_0, \mathsf{b}_1;\ \textbf{hid } \mathsf{c}_0, \mathsf{c}_1$$

$$(\mathsf{b}_0 \nabla \mathsf{c}_0) := \mathsf{b} \wedge \mathsf{c};$$
$$(\mathsf{b}_1 \nabla \mathsf{c}_1) := \mathsf{b} \vee \mathsf{c};$$

$$\mathsf{v} := (\mathsf{b}_1 \nabla \mathsf{c}_1 \textbf{ if } \mathsf{a} \textbf{ else } \mathsf{b}_0 \nabla \mathsf{c}_0)$$

Hierarchically structured protocols

$$\textbf{vis } \mathtt{v};$$
$$\textbf{hid } \mathtt{a}, \mathtt{b}, \mathtt{c};\ \textbf{hid } \mathtt{a}_b, \mathtt{a}_c;\ \textbf{hid } \mathtt{b}_0, \mathtt{b}_1;\ \textbf{hid } \mathtt{c}_0, \mathtt{c}_1$$

$$(\mathtt{b}_0 \nabla \mathtt{c}_0) := \mathtt{b} \wedge \mathtt{c};$$
$$(\mathtt{b}_1 \nabla \mathtt{c}_1) := \mathtt{b} \vee \mathtt{c};$$

$$\mathtt{a}_b := (\mathtt{b}_1 \textbf{ if } \mathtt{a} \textbf{ else } \mathtt{b}_0);$$
$$\mathtt{a}_c := (\mathtt{c}_1 \textbf{ if } \mathtt{a} \textbf{ else } \mathtt{c}_0);$$
$$\mathtt{v} := \mathtt{a}_b \nabla \mathtt{a}_c$$

1-2 Oblivious Transfers

Rabin; Even, Goldreich, Lempel; Rivest.

Hierarchically structured protocols

$\mathbf{vis}\ \mathtt{v};$

$\mathbf{hid}\ \mathtt{a}, \mathtt{b}, \mathtt{c};\ \mathbf{hid}\ \mathtt{a}_b, \mathtt{a}_c;\ \mathbf{hid}\ \mathtt{b}_0, \mathtt{b}_1;\ \mathbf{hid}\ \mathtt{c}_0, \mathtt{c}_1$

$(\mathtt{b}_0 \nabla \mathtt{c}_0) := \mathtt{b} \wedge \mathtt{c};$

$(\mathtt{b}_1 \nabla \mathtt{c}_1) := \mathtt{b} \vee \mathtt{c};$

$\mathtt{a}_b := (\mathtt{b}_1\ \mathbf{if}\ \mathtt{a}\ \mathbf{else}\ \mathtt{b}_0);$

$\mathtt{a}_c := (\mathtt{c}_1\ \mathbf{if}\ \mathtt{a}\ \mathbf{else}\ \mathtt{c}_0);$

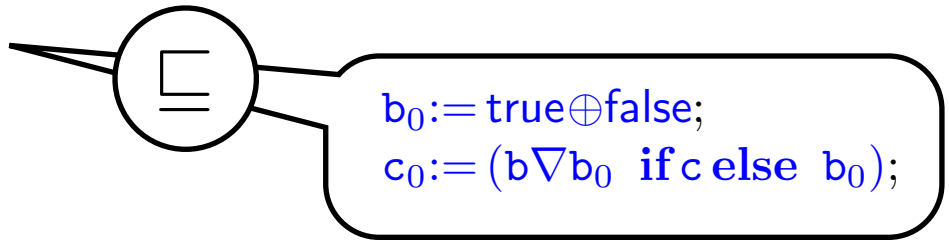
$\mathtt{v} := \mathtt{a}_b \nabla \mathtt{a}_c$

Hierarchically structured protocols

$\mathbf{vis}\ \mathtt{v};$

$\mathbf{hid}\ \mathtt{a}, \mathtt{b}, \mathtt{c};\ \mathbf{hid}\ \mathtt{a}_b, \mathtt{a}_c;\ \mathbf{hid}\ \mathtt{b}_0, \mathtt{b}_1;\ \mathbf{hid}\ \mathtt{c}_0, \mathtt{c}_1$

$(\mathtt{b}_0 \nabla \mathtt{c}_0) := \mathtt{b} \wedge \mathtt{c};$



Compositionality

$(\mathtt{b}_1 \nabla \mathtt{c}_1) := \mathtt{b} \vee \mathtt{c};$

$\mathtt{a}_b := (\mathtt{b}_1\ \mathbf{if}\ \mathtt{a}\ \mathbf{else}\ \mathtt{b}_0);$

$\mathtt{a}_c := (\mathtt{c}_1\ \mathbf{if}\ \mathtt{a}\ \mathbf{else}\ \mathtt{c}_0);$

$\mathtt{v} := \mathtt{a}_b \nabla \mathtt{a}_c$

Hierarchically structured protocols

$\mathbf{vis}\ \mathsf{v};$
$\mathbf{hid}\ \mathsf{a}, \mathsf{b}, \mathsf{c};\ \mathbf{hid}\ \mathsf{a}_b, \mathsf{a}_c;\ \mathbf{hid}\ \mathsf{b}_0, \mathsf{b}_1;\ \mathbf{hid}\ \mathsf{c}_0, \mathsf{c}_1$

$\mathsf{b}_0 := \mathsf{true} \oplus \mathsf{false};$
$\mathsf{c}_0 := (\mathsf{b} \nabla \mathsf{b}_0\ \ \mathbf{if}\ \mathsf{c}\ \mathbf{else}\ \ \mathsf{b}_0);$

$(\mathsf{b}_1 \nabla \mathsf{c}_1) := \mathsf{b} \vee \mathsf{c};$

$\mathsf{a}_b := \ (\mathsf{b}_1\ \mathbf{if}\ \ \mathsf{a}\ \ \mathbf{else}\ \mathsf{b}_0);$
$\mathsf{a}_c := \ (\mathsf{c}_1\ \mathbf{if}\ \ \mathsf{a}\ \ \mathbf{else}\ \mathsf{c}_0);$
$\mathsf{v} := \ \mathsf{a}_b \nabla \mathsf{a}_c$

$\mathbf{vis}\ \mathsf{v};$
$\mathbf{hid}\ \mathsf{a}, \mathsf{b}, \mathsf{c};\ \mathbf{hid}\ \mathsf{a}_b, \mathsf{a}_c;\ \mathbf{hid}\ \mathsf{b}_0, \mathsf{b}_1;\ \mathbf{hid}\ \mathsf{c}_0, \mathsf{c}_1$

$(\mathsf{b}_0 \nabla \mathsf{c}_0) := \mathsf{b} \wedge \mathsf{c};$

Hierarchically structured protocols

$\mathbf{vis}\ \mathsf{v};$

$\mathbf{hid}\ \mathsf{a}, \mathsf{b}, \mathsf{c};\ \mathbf{hid}\ \mathsf{a}_b, \mathsf{a}_c;\ \mathbf{hid}\ \mathsf{b}_0, \mathsf{b}_1;\ \mathbf{hid}\ \mathsf{c}_0, \mathsf{c}_1$

$\mathsf{b}_0 := \mathsf{true} \oplus \mathsf{false};$

$\mathsf{c}_0 := (\mathsf{b} \nabla \mathsf{b}_0\ \ \mathbf{if}\ \mathsf{c}\ \mathbf{else}\ \ \mathsf{b}_0);$

Oblivious Transfer, again

$(\mathsf{b}_1 \nabla \mathsf{c}_1) := \mathsf{b} \vee \mathsf{c};$

$\mathsf{a}_b := \ (\mathsf{b}_1\ \mathbf{if}\ \ \mathsf{a}\ \ \mathbf{else}\ \mathsf{b}_0);$

$\mathsf{a}_c := \ (\mathsf{c}_1\ \mathbf{if}\ \ \mathsf{a}\ \ \mathbf{else}\ \mathsf{c}_0);$

$\mathsf{v} := \ \mathsf{a}_b \nabla \mathsf{a}_c$

Hierarchically structured protocols

$\mathbf{vis}\ \mathsf{v};$
$\mathbf{hid}\ \mathsf{a}, \mathsf{b}, \mathsf{c};\ \mathbf{hid}\ \mathsf{a}_b, \mathsf{a}_c;\ \mathbf{hid}\ \mathsf{b}_0, \mathsf{b}_1;\ \mathbf{hid}\ \mathsf{c}_0, \mathsf{c}_1$

$\mathsf{b}_0 := \mathsf{true} \oplus \mathsf{false};$
$\mathsf{c}_0 := (\mathsf{b} \nabla \mathsf{b}_0\ \ \mathbf{if}\ \mathsf{c}\ \mathbf{else}\ \ \mathsf{b}_0);$ Oblivious Transfer

$(\mathsf{b}_1 \nabla \mathsf{c}_1) := \mathsf{b} \vee \mathsf{c};$ Secure multi-party computation

$\mathsf{a}_b := \ (\mathsf{b}_1\ \mathbf{if}\ \ \mathsf{a}\ \ \mathbf{else}\ \mathsf{b}_0);$ Oblivious Transfer
$\mathsf{a}_c := \ (\mathsf{c}_1\ \mathbf{if}\ \ \mathsf{a}\ \ \mathbf{else}\ \mathsf{c}_0);$ Oblivious Transfer
$\mathsf{v} := \ \mathsf{a}_b \nabla \mathsf{a}_c$

Prospects

For realistic protocols/programs, we need to combine all three features: hiding, probability and demonic choice. This extends the semantic space with (at least) a further powerset layer. In preparation.

Also we must treat loops and nontermination: this makes the outer distributions (at least) countably infinite, even over a countable state space. Under review.

Completion of the refinement order within this space seems to require proper measures. Done.

Automation? Event-B (Thai Son Hoang, Zurich) As we speak.