Models of Randomness

Part I: a sketchy survey

Fritz Obermeyer

Department of Mathematics
Carnegie-Mellon University

2008:04:08
Outline

Programs with randomness

Abstract probability theories

Finite sets
Probability measures
Pointless probability
Probability on dcpos
Probability on lattices
Abstract probability algebras

Summary and prospects
Motivational outline

Want to prove equivalence between randomized algorithms.
Motivational outline

Want to prove equivalence between randomized algorithms.

Need a programming language with random monad;
Motivational outline

Want to prove equivalence between randomized algorithms.

Need a programming language with random monad; and a domain-theoretic model of this language.
Motivational outline

Want to prove equivalence between randomized algorithms.

Need a programming language with random monad; and a domain-theoretic model of this language.

This talk surveys some attempts at models.
Want to prove equivalence between randomized algorithms.

Need a programming language with random monad; and a domain-theoretic model of this language.

This talk surveys some attempts at models.

Later Part II tries to find a random monad in a type-as-ambiguity framework (closures).
Bayesian networks

Consider entirely first-order programs, with no looping/recursion,
Bayesian networks

Consider entirely first-order programs, with no looping/recursion, e.g.,

```
sample x from unif(0, 1) in
sample y from unif(0, x) in
sample z from unif(x, 0) in
if y + z < 1/2 then 0 else 1
```

This is a Bayesian network (see picture).
All Bayesian networks are so expressible.
Bayesian networks

Consider entirely first-order programs, with no looping/recursion, e.g.,

```
sample x from unif(0, 1) in
sample y from unif(0, x) in
sample z from unif(x, 0) in
if y + z < 1/2 then 0 else 1
```

(notice we can forget x after sampling y,z)
Bayesian networks

Consider entirely first-order programs, with no looping/recursion, e.g.,

sample x from unif(0, 1) in
sample y from unif(0, x) in
sample z from unif(x, 0) in
if y + z < 1/2 then 0 else 1

(notice we can forget x after sampling y,z)
This is a Bayesian network (see picture).
Bayesian networks

Consider entirely first-order programs, with no looping/recursion, e.g.,

\[
\text{sample } x \text{ from } \text{unif}(0, 1) \text{ in } \\
\text{sample } y \text{ from } \text{unif}(0, x) \text{ in } \\
\text{sample } z \text{ from } \text{unif}(x, 0) \text{ in } \\
\text{if } y + z < 1/2 \text{ then } 0 \text{ else } 1
\]

(notice we can forget \( x \) after sampling \( y, z \))

This is a Bayesian network (see picture).
All Bayesian networks are so expressible.
Consider iterative loops,

Markov chains

This is a Markov chain (see picture).

All Markov chains are so expressible.
Markov chains

Consider iterative loops, e.g.

\[ x \leftarrow \text{sample } x_0 \text{ from normal}(0, 1) \in x_0. \]
\[ \text{while } \ldots : \]
\[ \text{sample } n \text{ from normal}(0, 1) \text{ in} \]
\[ \text{let } x' = 1 + x/2 \text{ in} \]
\[ x \leftarrow x' + n \]
Markov chains

Consider iterative loops, e.g.

\[
x \leftarrow \text{sample } x_0 \text{ from } \text{normal}(0, 1) \in x_0. \\
\text{while } \ldots : \\
\quad \text{sample } n \text{ from } \text{normal}(0, 1) \text{ in} \\
\quad \text{let } x' = 1 + x/2 \text{ in} \\
\quad x \leftarrow x' + n
\]

This is a Markov chain (see picture).
Consider iterative loops, e.g.

\[
x \leftarrow \text{sample } x_0 \text{ from normal}(0, 1) \in x_0.
\]

while \ldots :

\[
\begin{align*}
\text{sample } n \text{ from normal}(0, 1) \text{ in} \\
\text{let } x' = 1 + x/2 \text{ in} \\
x \leftarrow x' + n
\end{align*}
\]

This is a Markov chain (see picture). All Markov chains are so expressible.
Prototype for a probability theory

- set of states/points (maybe with structure)

We start with finite sets, generalize to probability measures, then we weaken the event language, add structure among points, and end with fully algebraic approaches.
Prototype for a probability theory

- set of states/points (maybe with structure)
- space of events/predicates
Prototype for a probability theory

- set of states/points (maybe with structure)
- space of events/predicates
- morphisms between state-event spaces

We start with finite sets, generalize to probability measures, then weaken the event language, add structure among points, and end with fully algebraic approaches.
Prototype for a probability theory

- set of states/points (maybe with structure)
- space of events/predicates
- morphisms between state-event spaces
- random monad \(\equiv\) probability distributions

We start with finite sets, generalize to probability measures, then weaken the event language, add structure among points, and end with fully algebraic approaches.
Prototype for a probability theory

- set of states/points (maybe with structure)
- space of events/predicates
- morphisms between state-event spaces
- random monad = probability distributions
- extra structure: products, exponentials, ...

We start with finite sets,
Prototype for a probability theory

- set of states/points (maybe with structure)
- space of events/predicates
- morphisms between state-event spaces
- random monad = probability distributions
- extra structure: products, exponentials, ...

We start with finite sets,
generalize to probability measures,
Prototype for a probability theory

- set of states/points (maybe with structure)
- space of events/predicates
- morphisms between state-event spaces
- random monad \(=\) probability distributions
- extra structure: products, exponentials, ...

We start with finite sets, generalize to probability measures, then weaken the event language,
Prototype for a probability theory

- set of states/points (maybe with structure)
- space of events/predicates
- morphisms between state-event spaces
- random monad = probability distributions
- extra structure: products, exponentials, ...

We start with finite sets, generalize to probability measures, then weaken the event language, add structure among points,
Prototype for a probability theory

- set of states/points (maybe with structure)
- space of events/predicates
- morphisms between state-event spaces
- random monad = probability distributions
- extra structure: products, exponentials, ...

We start with finite sets, generalize to probability measures, then weaken the event language, add structure among points, and end with fully algebraic approaches.
Finite sets: states, morphisms, extra structure

Start with a finite set $X$ of states.
Start with a finite set $X$ of states. No need for event space.
Finite sets: states, morphisms, extra structure

Start with a finite set $X$ of states. No need for event space.

Morphisms are just functions,
Start with a finite set $X$ of states. No need for event space.

Morphisms are just functions, product, exponential are standard.
Finite sets: states, morphisms, extra structure

Start with a finite set $X$ of states. No need for event space.

Morphisms are just functions, product, exponential are standard. NO recursive types,
Finite sets: states, morphisms, extra structure

Start with a finite set $X$ of states. No need for event space.

Morphisms are just functions, product, exponential are standard. NO recursive types, NO infinite types.
Finite sets: Random functor?

Like the powerset, Rand is functorial,
Finite sets: Random functor?

Like the powerset, Rand is functorial,
On objects: $\text{Rand}((X, \mathcal{F})) = (X', \mathcal{F}')$ where

- $X' = \text{probability measures over } (X, \mathcal{F})$
Finite sets: Random functor?

Like the powerset, Rand is functorial,
On objects: \( \text{Rand}((X, F)) = (X', F') \) where

- \( X' = \) probability measures over \((X, F)\)
- \( F' = \) generated by (right?)

\[
\{ \{ f \in X' \mid f^{-1}B \subseteq A \} \mid A \in F, \ B \in G \}
\]

But random functor doesn't land in finite sets.
Finite sets: Random functor?

Like the powerset, Rand is functorial,
On objects: $\text{Rand}((X, F)) = (X', F')$ where

- $X' = \text{probability measures over } (X, F)$
- $F' = \text{generated by (right?)}$

$$\{ \{ f \in X' \mid f^{-1}B \subseteq A \} \mid A \in F, \ B \in G \}$$

On arrows: for $h : (X, F) \to (Y, G)$,
$\text{Rand}(h) = h' : (X', F') \to (Y', G')$ where for $p : X'$ a probability measure on $(X, F), \ B \in G,$

$$\text{Rand}(h)(p)(B) = p(h^{-1}B)$$
Finite sets: Random functor?

Like the powerset, Rand is functorial,
On objects: \( \text{Rand}((X, F)) = (X', F') \) where

- \( X' = \) probability measures over \((X, F)\)
- \( F' = \) generated by (right?)

\[
\{ \{ f \in X' \mid f^{-1}B \subseteq A \} \mid A \in F, B \in G \}
\]

On arrows: for \( h : (X, F) \to (Y, G) \),
\( \text{Rand}(h) = h' : (X', F') \to (Y', G') \) where for \( p : X' \) a probability measure on \((X, F)\), \( B \in G \),

\[
\text{Rand}(h)(p)(B) = p(h^{-1}B)
\]

But random functor doesn’t land in finite sets.
Random monad (pieces)

The probability functor forms a monad with natural
always : \( \forall a. a \to \text{Rand} \ a \)
mix : \( \forall a. \text{Rand}(\text{Rand} \ a) \to \text{Rand} \ a \)
The probability functor forms a **monad** with natural

\[
\text{always} : \forall a. \ a \to \text{Rand} \ a
\]

\[
\text{mix} : \forall a. \ \text{Rand}(\text{Rand} \ a) \to \text{Rand} \ a
\]

equivalently a Kleisli triple with ’always’ and

\[
\text{sample} : \forall a, b. \ \text{Rand} \ a \to (a \to \text{Rand} \ b) \to \text{Rand} \ b
\]
Random monad (pieces)

The probability functor forms a monad with natural

\[
\text{always} : \forall a. \ a \to \text{Rand } a \\
\text{mix} : \forall a. \text{Rand(Rand } a) \to \text{Rand } a
\]
equivalently a Kleisli triple with 'always' and

\[
\text{sample} : \forall a, b. \text{Rand } a \to (a \to \text{Rand } b) \to \text{Rand } b
\]

In finite-sets, semantics is

\[
[\text{always } x](y) = \delta_{x,y} \\
[\text{mix } p](x) = \int [p](q) \ q(x) \ dq \\
[sample \ p \ f](y) = \sum_x [p](x) \ [f](y)(x)
\]

*oops: p infinite*
Random monad (properties)

Being a monad requires equations

sample x from p in always x
Random monad (properties)

Being a monad requires equations

\[
sample \ x \ from \ p \ in \ always \ x = p,
\]

▶ 'always' is mono
▶ monad plays nicely with products and sums
Random monad (properties)

Being a monad requires equations

sample \(x\) from \(p\) in always \(x = p\),
sample \(x\) from always \(y\) in \(f\ \times\)

Random monad (properties)

Being a monad requires equations

\[
\text{sample } x \text{ from } p \text{ in always } x = p, \\
\text{sample } x \text{ from } \text{always } y \text{ in } f x = f y,
\]
Random monad (properties)

Being a monad requires equations

\[ \text{sample } x \text{ from } p \text{ in always } x = p, \]
\[ \text{sample } x \text{ from always } y \text{ in } f x = f y, \]
\[ \text{sample } y \text{ from } (\text{sample } x \text{ from } p \text{ in } f x) \text{ in } g y \]
Random monad (properties)

Being a monad requires equations

sample \( x \) from \( p \) in always \( x = p \),
sample \( x \) from always \( y \) in \( f x = f y \),
sample \( y \) from (sample \( x \) from \( p \) in \( f x \)) in \( g y \)
\[
= \text{sample } x \text{ from } p \text{ in } \\
\quad \text{sample } y \text{ from } f x \text{ in } \\
\quad g y
\]
Random monad (properties)

Being a monad requires equations

\[
\begin{align*}
\text{sample } x \text{ from } p & \text{ in always } x = p, \\
\text{sample } x \text{ from always } y \text{ in } f & \text{ } x = f \ y, \\
\text{sample } y \text{ from } (\text{sample } x \text{ from } p \text{ in } f \ x) \text{ in } g \ y \\
& = \ \text{sample } x \text{ from } p \text{ in } \\
& \quad \text{sample } y \text{ from } f \ x \text{ in } \\
& \quad g \ y
\end{align*}
\]

Being a computational monad (a la Moggi) requires also:

- 'always' is mono
- monad plays nicely with products and sums
Probability measures: states, events, morphisms

Start with a state set $X$ (unstructured).
Probability measures: states, events, morphisms

Start with a state set $X$ (unstructured). The event space is a sigma-algebra $F$ of $X$, 
Probability measures: states, events, morphisms

Start with a state set $X$ (unstructured). The event space is a sigma-algebra $F$ of $X$, a structure $\langle X, \ F \subseteq P(\Omega), \ \neg, \ \bigvee^{\leq \omega} \rangle$. 
Probability measures: states, events, morphisms

Start with a state set $X$ (unstructured). The event space is a sigma-algebra $F$ of $X$, a structure $\langle X, F \subseteq \mathcal{P}(\Omega), \neg, \bigvee^{\leq \omega}\rangle$, where $\bot = \bigvee \emptyset$, $\top = \bigvee F$ are definable.
Start with a state set \( X \) (unstructured). The event space is a sigma-algebra \( F \) of \( X \), a structure \( \langle X, F \subseteq \mathcal{P}(\Omega), \neg, \bigvee^{\leq \omega} \rangle \), where \( \bot = \bigvee \emptyset \), \( \top = \bigvee F \) are definable.

A morphism is a sigma-algebra hom,
Start with a state set $X$ (unstructured).
The event space is a sigma-algebra $F$ of $X$, a structure
$$\langle X, F \subseteq \mathcal{P}(\Omega), \neg, \bigvee^{\leq \omega} \rangle,$$
where $\bot = \bigvee \emptyset$, $\top = \bigvee F$ are definable.

A morphism is a sigma-algebra hom, a measurable
function $f : X \rightarrow Y$, whose preimage induces a hom

$$f' : \langle F, \neg, \bigcup \rangle \leftarrow \langle G, \neg, \bigcup \rangle$$
Probability measures: states, events, morphisms

Start with a state set \( X \) (unstructured). The event space is a sigma-algebra \( F \) of \( X \), a structure \( \langle X, F \subseteq \mathcal{P}(\Omega), \neg, \bigvee^{\leq \omega} \rangle \), where \( \bot = \bigvee \emptyset \), \( \top = \bigvee F \) are definable.

A morphism is a sigma-algebra hom, a measurable function \( f: X \to Y \), whose preimage induces a hom \( f': \langle F, \neg, \bigcup \rangle \leftarrow \langle G, \neg, \bigcup \rangle \).
Measures: Random states

A random state is a probability measure, a hom

\[ \langle F, \emptyset, X, \bigcup^{\omega} \rangle \rightarrow \langle [0, 1], 0, 1, \sum^{\omega} \rangle \]
Measures: Random states

A random state is a probability measure, a hom

\[ \langle F, \emptyset, X, \bigcup^\omega \rangle \rightarrow \langle [0, 1], 0, 1, \sum^\omega \rangle \]

i.e., functions \( p \) satisfying

- \( p(\bot) = 0 \)
- \( p(\top) = 1 \)
- \( p(\bigcup_i A_i) = \sum_i p(A_i) \)

Question equivalent to additivity + continuity?
Measures: Random states

A random state is a probability measure, a hom

$$\langle F, \emptyset, X, \bigcup^\omega \rangle \longrightarrow \langle \mathbb{[0,1]}, 0, 1, \sum^\omega \rangle$$

i.e., functions $p$ satisfying

$$p(\bot) = 0$$
$$p(\top) = 1$$
$$p\left(\bigcup_i A_i\right) = \sum_i p(A_i)$$

and hence $p(\neg A) = 1 - p(A)$

**Question** equivalent to additivity + continuity?
Measures: extra structure

Product sigma-algebras are generated by rectangles
Measures: extra structure

Product sigma-algebras are generated by rectangles.
Exponentials have pointwise sigma-algebra (right?)
Measures: extra structure

Product sigma-algebras are generated by rectangles
Exponentials have pointwise sigma-algebra (right?)

NO untyped/unityped model of lambda-calculus
Probability valuations

Relax event logic from sigma-algebra to topology;

Definition
A probability valuation on $F$ is a monotone $p : F \to [0, 1]$ satisfying

1. $p(\bot) = 0$
2. $p(\top) = 1$
3. $p(A) + p(B) = p(A \cap B) + p(A \cup B)$

we also assume continuity (some authors don't).

(analogous to countable additivity?)

...extension theorems
e.g. Theorem (Jones) Every continuous valuation on a continuous dcpo extends uniquely to a measure.
Probability valuations

Relax event logic from sigma-algebra to topology;
abstract away points to frames/locales/CHAs.

Definition

A probability valuation on $F$ is a monotone $p : F \rightarrow [0, 1]$ satisfying

- $p(\bot) = 0$,
- $p(\top) = 1$,
- $p(A) + p(B) = p(A \cap B) + p(A \cup B)$

we also assume continuity (some authors don’t).

(analogous to countable additivity?)

...extension theorems

e.g. Theorem (Jones) Every continuous valuation on a continuous dcpo extends uniquely to a measure.
Probability valuations

Relax event logic from sigma-algebra to topology;
abstract away points to frames/locales/CHAs.

Definition

A **probability valuation** on $F$ is a monotone $p : F \rightarrow [0, 1]$ satisfying

$$p(\bot) = 0,$$
Probability valuations

Relax event logic from sigma-algebra to topology;
abstract away points to frames/locales/CHAs.

Definition

A probability valuation on $F$ is a monotone $p : F \rightarrow [0, 1]$ satisfying

$$p(\bot) = 0, \quad p(\top) = 1$$
Probability valuations

Relax event logic from sigma-algebra to topology;
abstract away points to frames/locales/CHAs.

Definition

A probability valuation on $F$ is a monotone $p: F \rightarrow [0, 1]$ satisfying

\[ p(\bot) = 0, \quad p(\top) = 1 \]
\[ p(A) + p(B) = p(A \sqcap B) + p(A \sqcup B) \]
Probability valuations

Relax event logic from sigma-algebra to topology;
abstract away points to frames/locales/CHAs.

Definition

A probability valuation on $F$ is a monotone $p : F \rightarrow [0, 1]$ satisfying

\[ p(\bot) = 0, \quad p(\top) = 1 \]
\[ p(A) + p(B) = p(A \sqcap B) + p(A \sqcup B) \]

we also assume continuity (some authors don’t).
(Analogous to countable additivity?)
Probability valuations

Relax event logic from sigma-algebra to topology;
abstract away points to frames/locales/CHAs.

Definition

A probability valuation on $F$ is a monotone $p : F \rightarrow [0, 1]$ satisfying

\[ p(\bot) = 0, \quad p(\top) = 1 \]
\[ p(A) + p(B) = p(A \sqcap B) + p(A \sqcup B) \]

we also assume continuity (some authors don’t).
(analogous to countable additivity?)

...extension theorems
Probability valuations

Relax event logic from sigma-algebra to topology;
abstract away points to frames/locales/CHAs.

Definition

A probability valuation on $F$ is a monotone $p : F \rightarrow [0, 1]$ satisfying

\[
p(\bot) = 0, \quad p(\top) = 1
\]
\[
p(A) + p(B) = p(A \cap B) + p(A \sqcup B)
\]

we also assume continuity (some authors don’t).
(-analogous to countable additivity?)

... extension theorems e.g.
Probability valuations

Relax event logic from sigma-algebra to topology;
abstract away points to frames/locales/CHAs.

Definition

A **probability valuation** on $F$ is a monotone $p : F \rightarrow [0, 1]$ satisfying

\[
p(\bot) = 0, \quad p(\top) = 1 \\
p(A) + p(B) = p(A \sqcap B) + p(A \sqcup B)
\]
we also assume continuity (some authors don’t).
(analogous to countable additivity?)

...extension theorems e.g.

**Theorem**

*(Jones)* Every continuous valuation on a continuous dcpo
Directed-complete partial orders.

Start with a dcpo of states.
Directed-complete partial orders.

Start with a dcpo of states.
Events are Scott-open sets (a frame).
Directed-complete partial orders.

Start with a dcpo of states. Events are Scott-open sets (a frame). Morphisms are Scott-continuous functions (preserving joins, inducing frame-homs).
Directed-complete partial orders.

Start with a dcpo of states.  
Events are Scott-open sets (a frame).  
Morphisms are Scott-continuous functions  
(preserving joins, inducing frame-homs).  

Directed joins of random things are random things
Directed-complete partial orders.

Start with a dcpo of states.
Events are Scott-open sets (a frame).
Morphisms are Scott-continuous functions
(preserving joins, inducing frame-homs).

Directed joins of random things are random things
so dcpo is closed under random monad.
Directed-complete partial orders.

Start with a dcpo of states.  
Events are Scott-open sets (a frame).  
Morphisms are Scott-continuous functions  
(preserving joins, inducing frame-homs).

Directed joins of random things are random things  
so dcpo is closed under random monad.

dcpo is also closed under products, function spaces,  
coinductive types, ...
Directed-complete partial orders.

Start with a dcpo of states.
Events are Scott-open sets (a frame).
Morphisms are Scott-continuous functions (preserving joins, inducing frame-homs).

Directed joins of random things are random things so dcpo is closed under random monad.

dcpo is also closed under products, function spaces, coinductive typess, ...
Continuous dcpos

...but dcpos don’t have enough structure to be “domains”.
Continuous dcpos

...but dcpos don’t have enough structure to be “domains”.

Continuous domains have more.
Continuous dcpos

...but dcpo don’t have enough structure to be “domains”.

Continuous domains have more.
CONT is closed under probability monad.
Continuous dcpos

...but dcpos don’t have enough structure to be “domains”.

Continuous domains have more.
CONT is closed under probability monad.
But NOT closed under function spaces.
Lattices I: states, randomness

Start with a lattice $X$
Start with a lattice $X$ (e.g. real line).
Lattices I: states, randomness

Start with a lattice $X$ (e.g. real line).
Let event space be the upper sets.
Lattices I: states, randomness

Start with a lattice $X$ (e.g. real line).
Let event space be the upper sets.
To each valuation $p$, define a cpdf

$$p'(x) = p(\text{upper } x)$$

Dually, each cpdf $d$ extends to a unique valuation

$$d'(\text{upper } x) = d(x)$$

Try again: (no event space)
Start with a lattice $X$ (e.g. real line). Let event space be the upper sets. To each valuation $p$, define a cpdf

$$p'(x) = p(\text{upper } x)$$

Dually, each cpdf $d$ extends to a unique valuation

$$d'(\text{upper } x) = d(x)$$

Try again: (no event space) Morphisms are lattice homs.
Start with a lattice \( X \) (e.g. real line). Let event space be the upper sets. To each valuation \( p \), define a cpdf

\[
p'(x) = p(\text{upper } x)
\]

Dually, each cpdf \( d \) extends to a unique valuation

\[
d'(\text{upper } x) = d(x)
\]

Try again: (no event space) Morphisms are lattice homs. Random states are cpdfs, but...
A space of random lattice elements need not itself be a lattice.
A space of random lattice elements need not itself be a lattice.
Hence $\text{Rand} \circ \text{Rand}$ need not exist.
NO random monad

A space of random lattice elements need not itself be a lattice.

Hence Rand ∘ Rand need not exist.

Example

the square lattice ⊥ ⊑ tr, fa ⊑ ⊤,
NO random monad

A space of random lattice elements need not itself be a lattice.

Hence $\text{Rand} \circ \text{Rand}$ need not exist.

Example

the square lattice $\bot \sqsubseteq \text{tr}, \text{fa} \sqsubseteq \top$,

$\bot + \text{tr} \mid \bot + \text{fa} \sqsubseteq \bot + \top$, $\text{tr} + \text{fa}$
NO random monad

A space of random lattice elements need not itself be a lattice.

Hence $\text{Rand} \circ \text{Rand}$ need not exist.

Example

the square lattice $\perp \sqsubseteq \text{tr}, \text{fa} \sqsubseteq \top$,

$\perp + \text{tr} \mid \perp + \text{fa} \sqsubseteq \perp + \top$, $\text{tr} + \text{fa}$

no unique minimal upper bound

The max of two cdfs may lead to negative densities.
Abstract probability algebras

Start with a dcpo with \( \bot \).
Abstract probability algebras

Start with a dcpo with \( \bot \).
Generate initial “R-algebra” with binary mixing \( x + y \)
Abstract probability algebras

Start with a dcpo with \( \perp \).
Generate initial “R-algebra” with binary mixing \( x + y \) subject to monotonicity and

\[
\begin{align*}
x + x &= x & \text{idempotence} \\
x + y &= y + x & \text{commutativity}
\end{align*}
\]
Abstract probability algebras

Start with a dcpo with $\bot$.
Generate initial “R-algebra” with binary mixing $x + y$
subject to monotonicity and

\[
\begin{align*}
    x + x &= x & \text{idempotence} \\
    x + y &= y + x & \text{commutativity} \\
    (\omega + x) + (y + z) &= (\omega + z) + (y + x) & \text{associativity}
\end{align*}
\]
Abstract probability algebras

Start with a dcpo with \( \bot \).
Generate initial “R-algebra” with binary mixing \( x + y \) subject to monotonicity and

\[
\begin{align*}
x + x &= x \quad \text{idempotence} \\
x + y &= y + x \quad \text{commutativity} \\
(\omega + x) + (y + z) &= (\omega + z) + (y + x) \quad \text{associativity}
\end{align*}
\]

- equivalent to arbitrary real mixing
- equivalent to valuations
Abstract probability algebras

Start with a dcpo with ⊥.
Generate initial “R-algebra” with binary mixing $x + y$
subject to monotonicity and

$$x + x = x \quad \text{idempotence}$$

$$x + y = y + x \quad \text{commutativity}$$

$$(\omega + x) + (y + z) = (\omega + z) + (y + x) \quad \text{associativity}$$

- equivalent to arbitrary real mixing
- equivalent to valuations

Compare with initial join-semilattice (“J-algebra”)

$$x \mid x = x$$

$$x \mid y = y \mid x$$

$$x \mid (y \mid x) = (x \mid y) \mid z$$
Problem the join/meet of two random things may not be a random thing.
Probability and lattices

**Problem** the join/meet of two random things may not be a random thing.

R-algebra models randomness.
Probability and lattices

**Problem** the join/meet of two random things may not be a random thing.

R-algebra models randomness.
J-algebra models parallelism.
Probability and lattices

Problem the join/meet of two random things may not be a random thing.

R-algebra models randomness.
J-algebra models parallelism.
JR-algebra with distributivity.

\[(x + y) \mid z = (x \mid z) + (y \mid z)\]
Probability and lattices

Problem the join/meet of two random things may not be a random thing.

R-algebra models randomness.
J-algebra models parallelism.
JR-algebra with distributivity.

\[(x + y) \mid z = (x \mid z) + (y \mid z)\]

models parallelism with randomness,
Probability and lattices

Problem  the join/meet of two random things may not be a random thing.

R-algebra models randomness.
J-algebra models parallelism.
JR-algebra with distributivity.

\[(x + y) \mid z = (x \mid z) + (y \mid z)\]

models parallelism with randomness, allows random normal form,
Probability and lattices

**Problem** the join/meet of two random things may not be a random thing.

R-algebra models randomness.
J-algebra models parallelism.
JR-algebra with distributivity.

\[(x + y) | z = (x | z) + (y | z)\]

models parallelism with randomness,
allows random normal form,
sampling semantics
Probability and lattices

Problem the join/meet of two random things may not be a random thing.

R-algebra models randomness.
J-algebra models parallelism.
JR-algebra with distributivity.

\[(x + y) \parallel z = (x \parallel z) + (y \parallel z)\]

models parallelism with randomness, allows random normal form, sampling semantics
JR-algebra models... nothing nice,
Probability and lattices

Problem  the join/meet of two random things may not be a random thing.

R-algebra models randomness.
J-algebra models parallelism.
JR-algebra with distributivity.

\[(x + y) | z = (x | z) + (y | z)\]

models parallelism with randomness,
allows random normal form,
sampling semantics
JR-algebra models... nothing nice,
NO random normal form
Probability and lattices

Problem  the join/meet of two random things may not be a random thing.

R-algebra models randomness.
J-algebra models parallelism.
JR-algebra with distributivity.

\[(x + y) \upharpoonright z = (x \upharpoonright z) + (y \upharpoonright z)\]

to models parallelism with randomness,
allows random normal form,
sampling semantics
JR-algebra models... nothing nice,
NO random normal form
Next time: can JR-algebras be made to work?
Summary and prospects

(again)
We started with finite sets,
Summary and prospects

(again)
We started with finite sets,
genralized to probability measures,
Summary and prospects

(again)
We started with finite sets, generalized to probability measures, then weakened the event language,
Summary and prospects

(again)
We started with finite sets, generalized to probability measures, then weakened the event language, added structure among points,
Summary and prospects

(again)
We started with finite sets, generalized to probability measures, then weakened the event language, added structure among points, and ended with fully algebraic approaches.