Laboratoire Méthodes Formelles

université PARIS-SACLAY

cnrs

école normale supérieure paris—saclay

with some parts in collaboration with
Xiaodong Jia (贾晓东),
Clément Theron

Jean Goubault-Larrecq

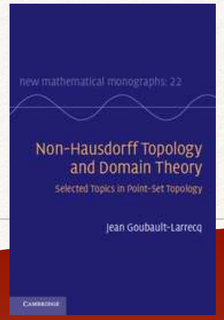
A journey through the semantics of higher-order probabilistic languages, domain theory, and topology

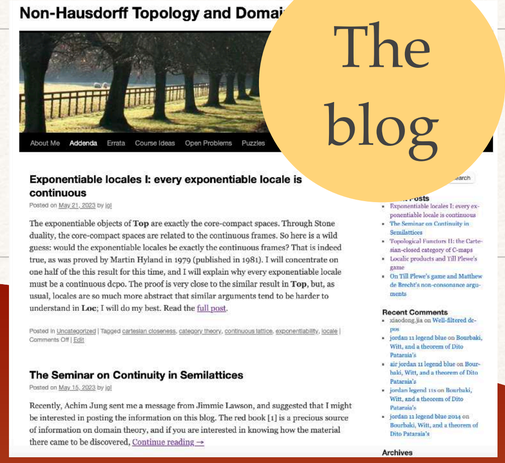
Etiolles 2023

Topology and Applications

The purpose of this talk

- I was asked the following in May 2023:
  « Would you accept to make a long presentation on a topic related to topology in computer science (like a tutorial, or an introduction « for dummies »)? »
  « est-ce que tu accepterais de faire une présentation longue sur un sujet lié à la topologie en informatique (type tutoriel, ou intro "pour les nuls ») ?»
- Let me first give you an overview of my main activities in topology at LMF





The blog

Directed alg topolo

Tru concurr

Non-Hausdorff topology

Noetherian spaces

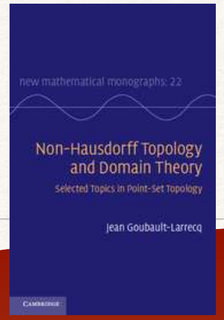
Quasi-metric spaces

Domain theory

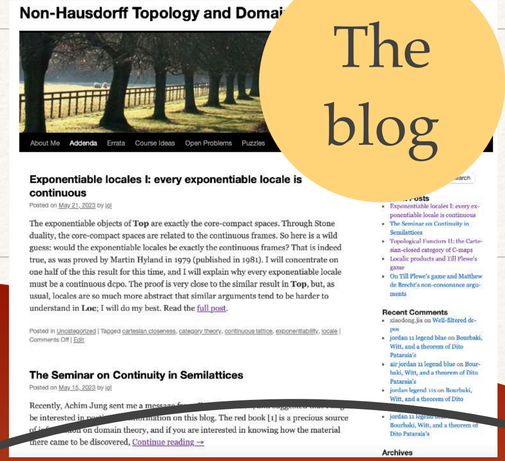
Topological measure theory

Verification f infinite-state systems

Semantics of programming languages



Non-Hausdorff topology



The blog

Directed alg
topolo

Tru
concurr

Noetherian spaces

Quasi-metric spaces

Verification
f infinite-state
systems

Domain theory

Topological
measure theory

Semantics of
programming
languages

Topology

What topology is, in this talk

- A **topology** on a set X is a collection of sets, then **open sets**, s.t.:
  — every (arbitrary) union of open sets is open
  — every finite intersection of open sets is open

What topology is, in this talk

- A **topology** on a set X is a collection of sets, then **open sets**, s.t.:
  — every (arbitrary) union of open sets is open
  — every finite intersection of open sets is open
- A **topological space** is a set X with a topology

What topology is, in this talk

- A **topology** on a set X is a collection of sets, then **open sets**, s.t.:
  — every (arbitrary) union of open sets is open
  — every finite intersection of open sets is open
- A **topological space** is a set X with a topology
- A map $f : X \to Y$ is **continuous** iff
  for every open subset V of Y, $f^{-1}(V)$ is open in X

Domain theory

- An approach to the semantics of programming languages initiated by Dana S. Scott

Theoretical Computer Science 121 (1993) 411–440
Elsevier

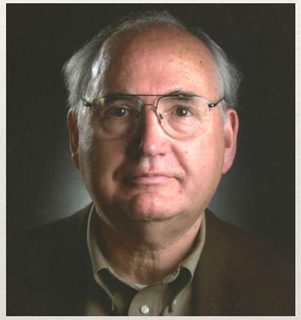
A type-theoretical alternative to ISWIM, CUCH, OWHY

Dana S. Scott
Carnegie-Mellon University, Pittsburgh, PA, USA, and RISC-Linz, Austria

Abstract

Scott, D.S., A type-theoretical alternative to ISWIM, CUCH, OWHY, Theoretical Computer Science 121 (1993) 411–440.

The paper (first written in 1969 and circulated privately) concerns the definition, axiomatization, and applications of the hereditarily monotone and continuous functionals generated from the integers and the Booleans (plus "undefined" elements). The system is formulated as a typed system of combinators (or as a typed λ-calculus) with a recursion operator (the least fixed-point operator), and its proof rules are contrasted to a certain extent with those of the untyped λ-calculus. For publication (1993), a new preface has been added, and many bibliographical references and comments in footnotes have been appended.

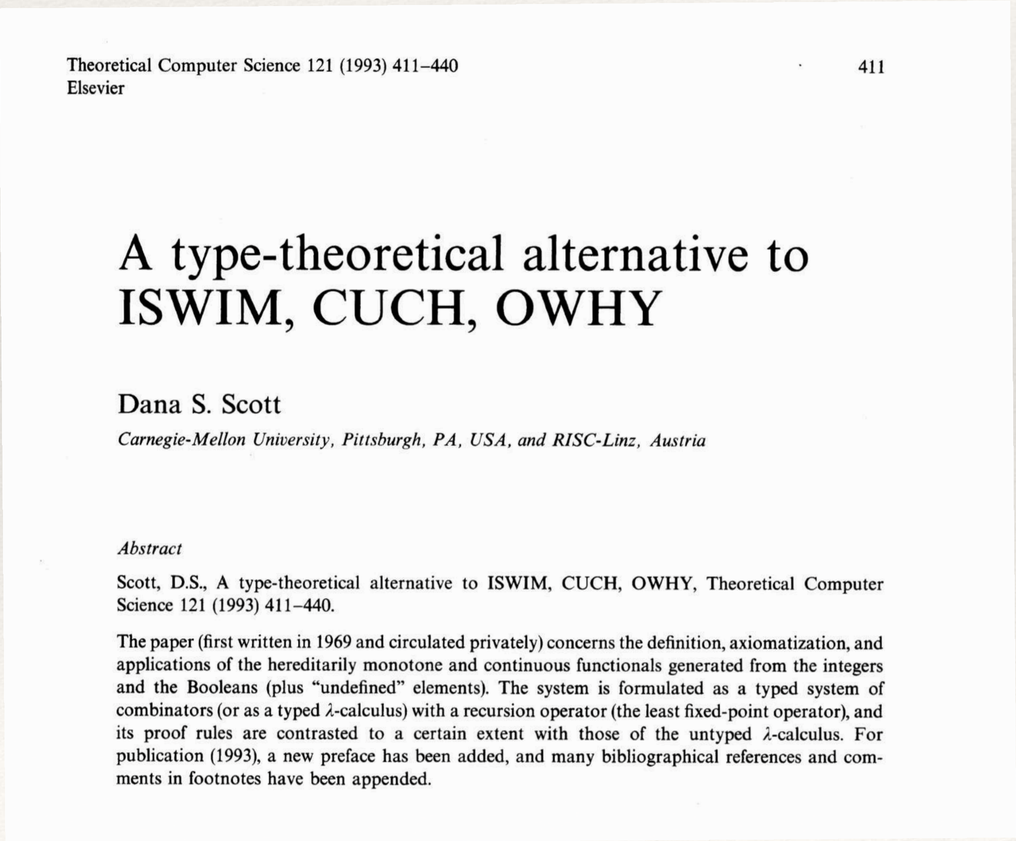


https://www.scs.cmu.edu/sites/default/files/styles/news_item_image/public/DanaScott_b.jpg?itok=T9voOevm

(1969, unpublished, but widely circulated - 1st published 1993)

Domain theory

- An approach to the semantics of programming languages initiated by Dana S. Scott



Theoretical Computer Science 121 (1993) 411–440
Elsevier

A type-theoretical alternative to ISWIM, CUCH, OWHY

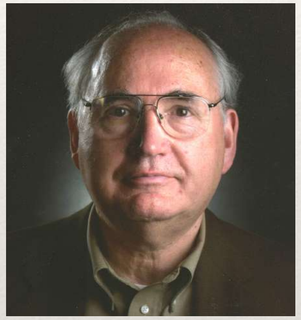
Dana S. Scott

Carnegie-Mellon University, Pittsburgh, PA, USA, and RISC-Linz, Austria

Abstract

Scott, D.S., A type-theoretical alternative to ISWIM, CUCH, OWHY, Theoretical Computer Science 121 (1993) 411–440.

The paper (first written in 1969 and circulated privately) concerns the definition, axiomatization, and applications of the hereditarily monotone and continuous functionals generated from the integers and the Booleans (plus "undefined" elements). The system is formulated as a typed system of combinators (or as a typed λ-calculus) with a recursion operator (the least fixed-point operator), and its proof rules are contrasted to a certain extent with those of the untyped λ-calculus. For publication (1993), a new preface has been added, and many bibliographical references and comments in footnotes have been appended.



https://www.scs.cmu.edu/sites/default/files/styles/news_item_image/public/DanaScott_b.jpg?itok=T9voOevm

(1969, unpublished, but widely circulated - 1st published 1993)

- based on **partial orders**,
- with some particular **topology**

Dcpos, partial values, and the order of information

- A **dcpo** (= directed-complete partial order)
  is a set of « partial values »
  — total values among them
  represent terminated computations

Dcpos, partial values, and the order of information

- A **dcpo** (= directed-complete partial order)
  is a set of « partial values »
  — total values among them
  represent terminated computations

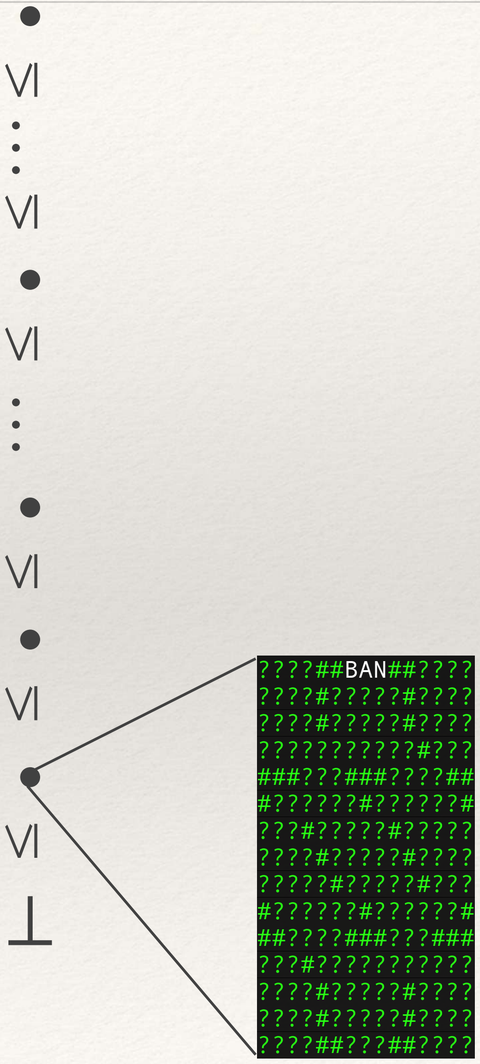
- **partially ordered**:
  $x \leq y$ means « y contains **more information**,
  is more precise than x »

Dcpos, partial values, and the order of information

- A **dcpo** (= directed-complete partial order) is a set of « partial values »
  — total values among them
  represent terminated computations
- **partially ordered**:
  $x \leq y$ means « y contains **more information**,
  is more precise than x »
- with a notion of **limit** (=**suprema**) of increasing chains of partial values
  giving the « value at infinity »

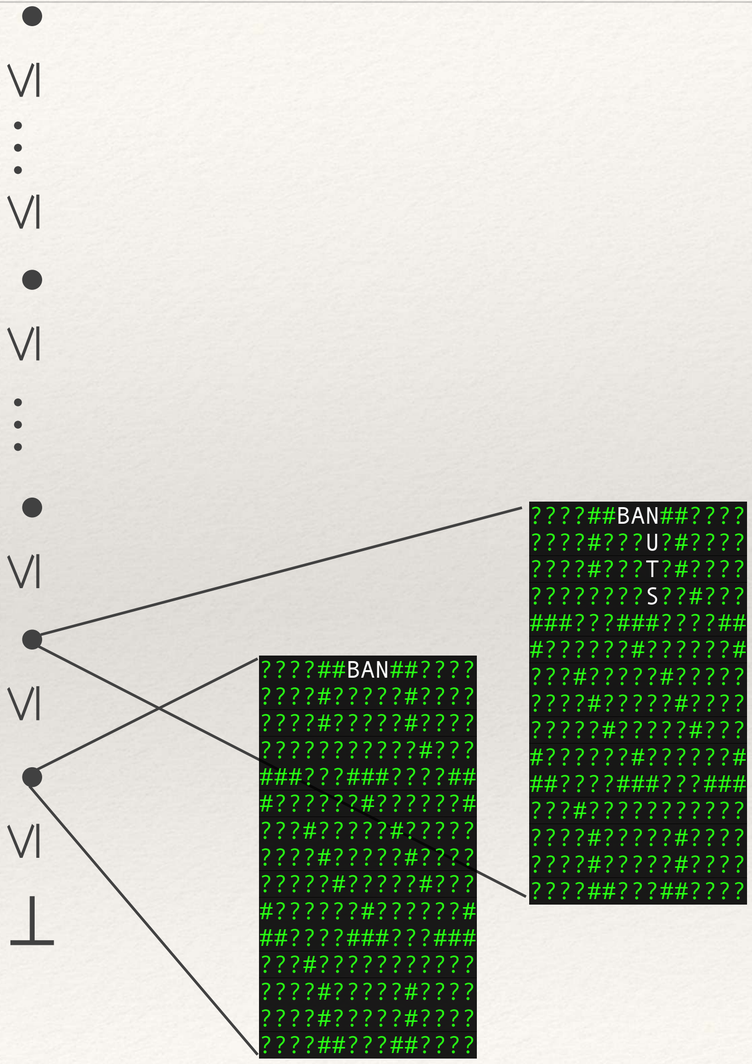
Dcpos, partial values, and the order of information

- A **dcpo** (= directed-complete partial order) is a set of « partial values »
  — total values among them
  represent terminated computations
- **partially ordered**:
  $x \leq y$ means « y contains **more information**,
  is more precise than x »
- with a notion of **limit** (=**suprema**) of increasing chains of partial values
  giving the « value at infinity »



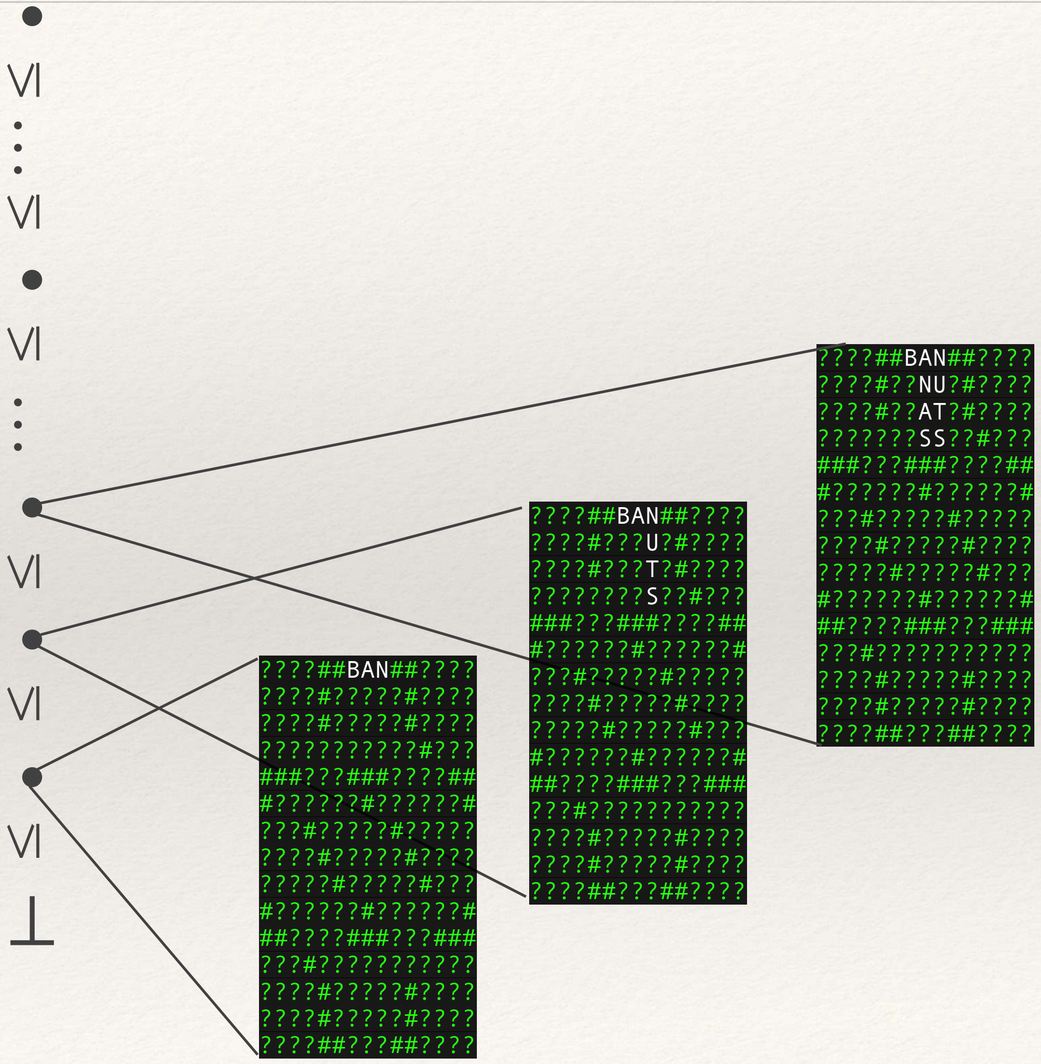
Dcpos, partial values, and the order of information

- A **dcpo** (= directed-complete partial order) is a set of « partial values »
  — total values among them
  represent terminated computations
- **partially ordered**:
  $x \leq y$ means « y contains **more information**,
  is more precise than x »
- with a notion of **limit** (=**suprema**) of increasing chains of partial values
  giving the « value at infinity »



Dcpos, partial values, and the order of information

- A **dcpo** (= directed-complete partial order) is a set of « partial values »
  — total values among them
  represent terminated computations
- **partially ordered**:
  $x \leq y$ means « y contains **more information**,
  is more precise than x »
- with a notion of **limit** (=**suprema**) of increasing chains of partial values
  giving the « value at infinity »



Dcpos, partial values, and the order of information

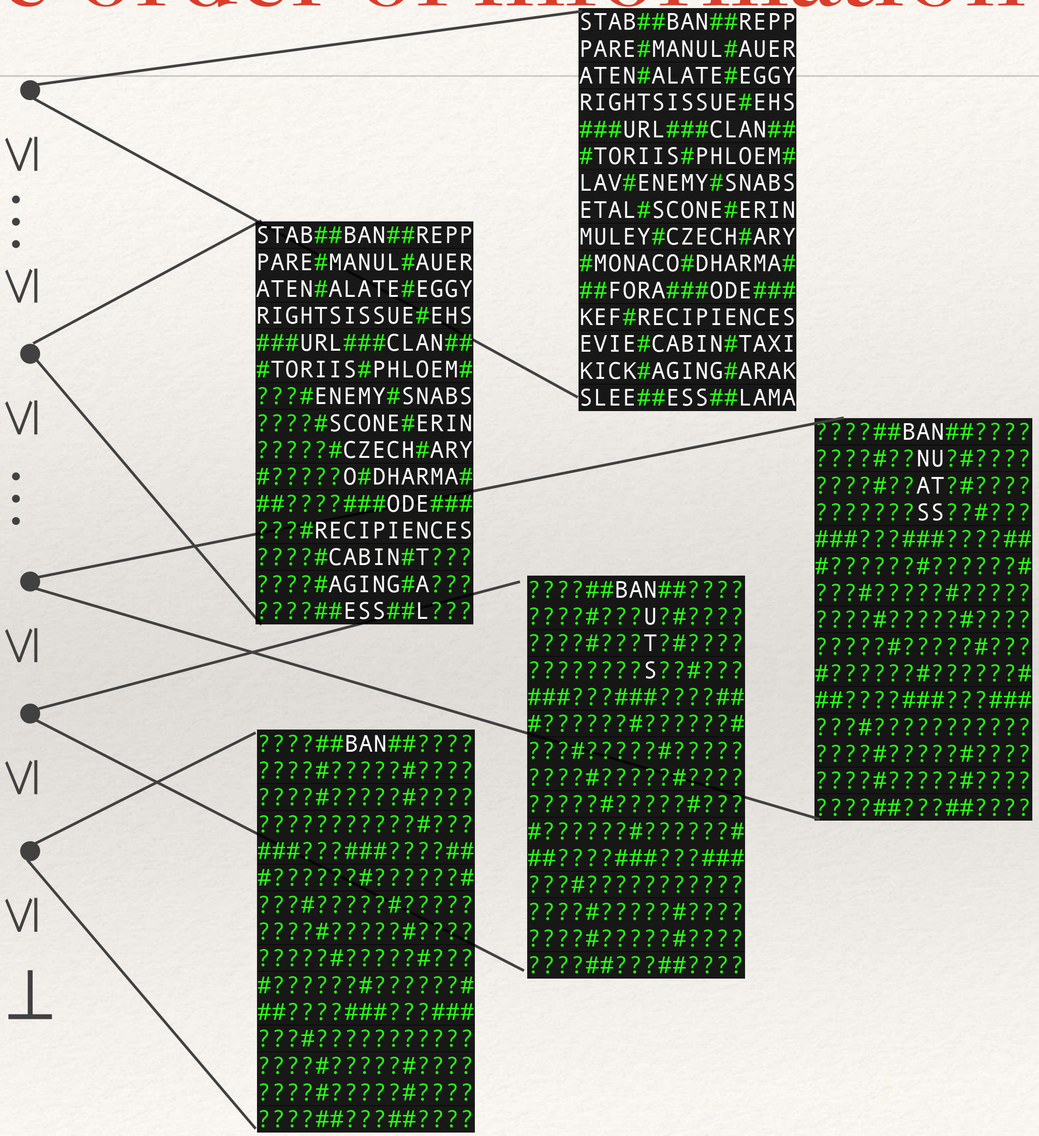
- A **dcpo** (= directed-complete partial order) is a set of « partial values »
  — total values among them represent terminated computations
- **partially ordered**:
  $x \leq y$ means « y contains **more information**, is more precise than x »
- with a notion of **limit** (=**suprema**) of increasing chains of partial values giving the « value at infinity »

Dcpos, partial values, and the order of information

- A **dcpo** (= directed-complete partial order) is a set of « partial values »
  — total values among them
  represent terminated computations

- **partially ordered**:
  $x \leq y$ means « y contains **more information**,
  is more precise than x »

- with a notion of **limit** (=**suprema**) of increasing chains of partial values
  giving the « value at infinity »

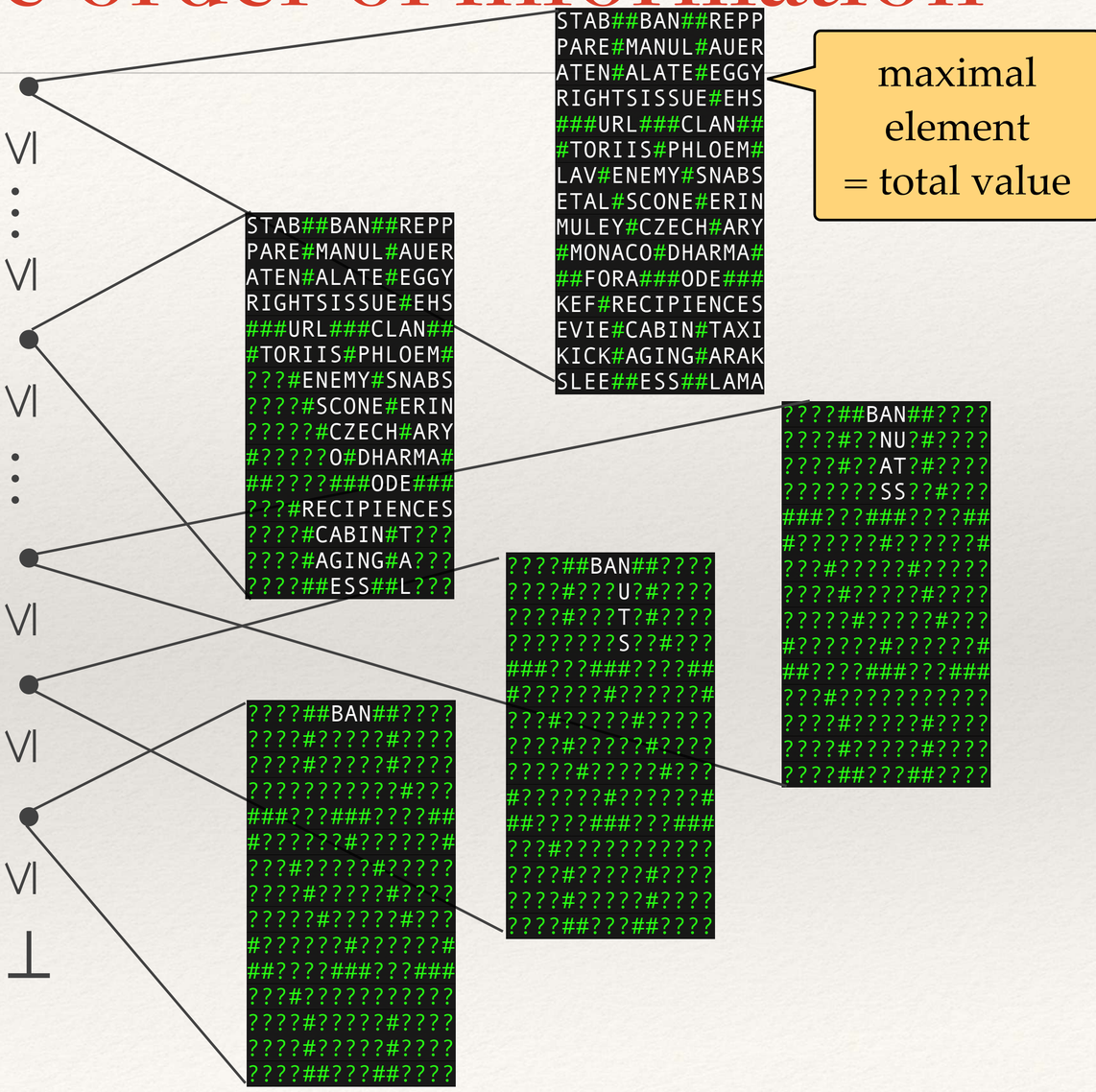


Dcpos, partial values, and the order of information

- A **dcpo** (= directed-complete partial order) is a set of « partial values »
  — total values among them
  represent terminated computations

- **partially ordered**:
  $x \leq y$ means « y contains **more information**,
  is more precise than x »

- with a notion of **limit** (=**suprema**) of increasing chains of partial values
  giving the « value at infinity »

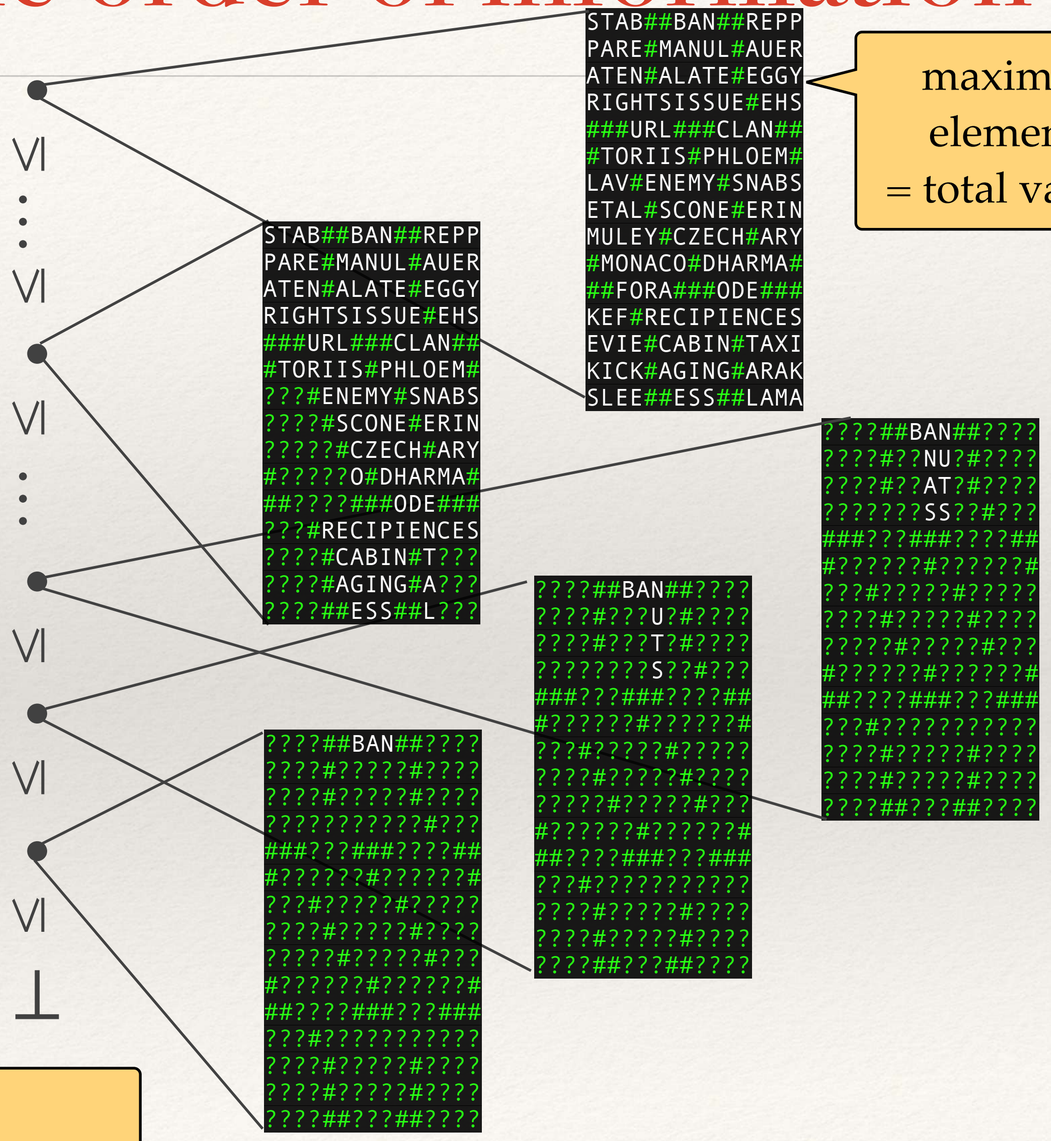


Dcpos, partial values, and the order of information

- A **dcpo** (= directed-complete partial order) is a set of « partial values »
  — total values among them
  represent terminated computations

- **partially ordered**:
  $x \leq y$ means « y contains **more information**,
  is more precise than x »

- with a notion of **limit** (=**suprema**) of increasing chains of partial values
  giving the « value at infinity »



well, directed families, really:
D is **directed** iff $\neq \emptyset$ and every pair of elements of D has an upper bound in D

Scott-continuous maps, least fixed points

- A map $f : X \to Y$ between dcpos is **Scott-continuous** iff:
  — monotonic: $x \leq x' \Longrightarrow f(x) \leq f(x')$
  — preserves directed suprema: $\sup_i^{\uparrow} f(x_i) = f(\sup_i^{\uparrow} x_i)$

Scott-continuous maps, least fixed points

- A map $f : X \to Y$ between dcpos is **Scott-continuous** iff:
  — monotonic: $x \leq x' \Longrightarrow f(x) \leq f(x')$
  — preserves directed suprema: $\sup_i^{\uparrow} f(x_i) = f(\sup_i^{\uparrow} x_i)$

- **Theorem 1.** On a pointed dcpo X (one with a least element $\perp$), every Scott-continuous map $f : X \to X$ has a **least fixed point**.

Will serve to interpret **recursion**

Scott-continuous maps, least fixed points

- A map $f : X \to Y$ between dcpos is **Scott-continuous** iff:
  — monotonic: $x \leq x' \Longrightarrow f(x) \leq f(x')$
  — preserves directed suprema: $\sup_i^{\uparrow} f(x_i) = f(\sup_i^{\uparrow} x_i)$

represents **non-termination**

Will serve to interpret **recursion**

- **Theorem 1.** On a pointed dcpo X (one with a least element $\perp$), every Scott-continuous map $f : X \to X$ has a **least fixed point**.

Scott-continuous maps, least fixed points

- A map $f : X \to Y$ between dcpos is **Scott-continuous** iff:
  — monotonic: $x \leq x' \Longrightarrow f(x) \leq f(x')$
  — preserves directed suprema: $\sup_i^{\uparrow} f(x_i) = f(\sup_i^{\uparrow} x_i)$

represents **non-termination**

Will serve to interpret **recursion**

- **Theorem 1.** On a pointed dcpo X (one with a least element $\perp$), every Scott-continuous map $f : X \to X$ has a **least fixed point**.

- *Proof.* Let $x \triangleq \sup_{n \in \mathbb{N}}^{\uparrow} f^n(\perp)$. Since f preserves directed suprema, $f(x) = x$. That x is least is left as an exercise. $\square$

Scott-continuous maps, least fixed points

- A map $f : X \to Y$ between dcpos is **Scott-continuous** iff:
  — monotonic: $x \leq x' \Longrightarrow f(x) \leq f(x')$
  — preserves directed suprema: $\sup_i^{\uparrow} f(x_i) = f(\sup_i^{\uparrow} x_i)$

represents **non-termination**

Will serve to interpret **recursion**

- **Theorem 1.** On a pointed dcpo X (one with a least element $\perp$), every Scott-continuous map $f : X \to X$ has a **least fixed point**.

- *Proof.* Let $x \triangleq \sup_{n \in \mathbb{N}}^{\uparrow} f^n(\perp)$. Since f preserves directed suprema, $f(x) = x$. That x is least is left as an exercise. $\square$

- **Theorem 2.** $[X \to Y] \triangleq \{f : X \to Y \text{ Scott-continuous}\}$ is a dcpo, where $f \leq g$ iff for every $x \in X$, $f(x) \leq g(x)$; it is pointed if Y is.

Will serve to interpret **higher-order** types

PCF

https://www.research.ed.ac.uk/files-asset/17539241/gdp.jpg?w=320&f=webp

- Consider the higher-order, functional programming language PCF [Plotkin 77]

$M, N, P, \ldots ::= x, y, z, \ldots$	variables
$\mid MN$	application
$\mid \lambda x_\sigma . M$	abstraction
$\mid \mathbf{rec}(M)$	recursion
$\mid 0 \mid 1 \mid 2 \mid \ldots$	natural numbers
$\mid \mathbf{s}(M)$	successor
$\mid \mathbf{p}(M)$	predecessor
$\mid \mathbf{if}\ M = 0\ \mathbf{then}\ N\ \mathbf{else}\ P$	conditional

PCF

https://www.research.ed.ac.uk/files-asset/17539241/gdp.jpg?w=320&f=webp

- Consider the higher-order, functional programming language PCF [Plotkin 77]

$M, N, P, \ldots ::= x, y, z, \ldots$	variables
$\mid MN$	application
$\mid \lambda x_\sigma . M$	abstraction
$\mid \mathbf{rec}(M)$	recursion
$\mid 0 \mid 1 \mid 2 \mid \ldots$	natural numbers
$\mid \mathbf{s}(M)$	successor
$\mid \mathbf{p}(M)$	predecessor
$\mid \mathbf{if}\ M = 0\ \mathbf{then}\ N\ \mathbf{else}\ P$	conditional

(simply-typed) lambda-calculus

PCF

https://www.research.ed.ac.uk/files-asset/17539241/gdp.jpg?w=320&f=webp

- Consider the higher-order, functional programming language PCF [Plotkin 77]

$M, N, P, \ldots ::= x, y, z, \ldots$	variables
$\mid MN$	application
$\mid \lambda x_\sigma . M$	abstraction
$\mid \mathbf{rec}(M)$	recursion
$\mid 0 \mid 1 \mid 2 \mid \ldots$	natural numbers
$\mid \mathbf{s}(M)$	successor
$\mid \mathbf{p}(M)$	predecessor
$\mid \mathbf{if}\ M = 0\ \mathbf{then}\ N\ \mathbf{else}\ P$	conditional

(simply-typed) lambda-calculus

YM, for purists

PCF

https://www.research.ed.ac.uk/files-asset/17539241/gdp.jpg?w=320&f=webp

- Consider the higher-order, functional programming language PCF [Plotkin 77]

$M, N, P, \ldots ::= x, y, z, \ldots$	variables
$\mid MN$	application
$\mid \lambda x_\sigma . M$	abstraction
$\mid \mathbf{rec}(M)$	recursion
$\mid 0 \mid 1 \mid 2 \mid \ldots$	natural numbers
$\mid \mathbf{s}(M)$	successor
$\mid \mathbf{p}(M)$	predecessor
$\mid \mathbf{if}\ M = 0\ \mathbf{then}\ N\ \mathbf{else}\ P$	conditional

(simply-typed) lambda-calculus

YM, for purists

elementary operations on type **nat**

Types

- PCF terms are **typed**: $\sigma, \tau, \ldots ::= \mathbf{nat} \mid \sigma \to \tau$

$$\frac{}{x_\sigma : \sigma} \qquad \frac{M : \tau}{\lambda x_\sigma . M : \tau}$$

$$\frac{M : \sigma \to \tau \quad N : \sigma}{MN : \tau}$$

$$\frac{M : \tau \to \tau}{\mathbf{rec}(M) : \tau}$$

$$\frac{}{0 : \mathbf{nat}} \quad \frac{}{1 : \mathbf{nat}} \quad \ldots$$

$$\frac{M : \mathbf{nat}}{\mathbf{s}(M) : \mathbf{nat}} \qquad \frac{M : \mathbf{nat}}{\mathbf{p}(M) : \mathbf{nat}}$$

$$\frac{M : \mathbf{nat} \quad N : \tau \quad P : \tau}{\mathbf{if}\ M = 0\ \mathbf{then}\ N\ \mathbf{else}\ P : \tau}$$

Types

- PCF terms are **typed**: $\sigma, \tau, \ldots ::= \mathbf{nat} \mid \sigma \to \tau$
- Semantics of types: $[\![\tau]\!]$ will be a pointed dcpo

$$\frac{}{x_\sigma : \sigma} \qquad \frac{M : \tau}{\lambda x_\sigma . M : \tau}$$

$$\frac{M : \sigma \to \tau \quad N : \sigma}{MN : \tau}$$

$$\frac{M : \tau \to \tau}{\mathbf{rec}(M) : \tau}$$

$$\frac{}{0 : \mathbf{nat}} \quad \frac{}{1 : \mathbf{nat}} \quad \ldots$$

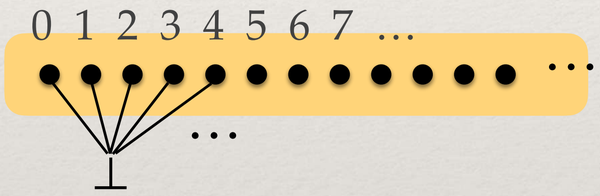
$$\frac{M : \mathbf{nat}}{\mathbf{s}(M) : \mathbf{nat}} \qquad \frac{M : \mathbf{nat}}{\mathbf{p}(M) : \mathbf{nat}}$$

$$\frac{M : \mathbf{nat} \quad N : \tau \quad P : \tau}{\mathbf{if}\ M = 0\ \mathbf{then}\ N\ \mathbf{else}\ P : \tau}$$

Types

- PCF terms are **typed**: $\sigma, \tau, \ldots ::= \mathbf{nat} \mid \sigma \to \tau$

- Semantics of types: $[\![\tau]\!]$ will be a pointed dcpo
  - $[\![\mathbf{nat}]\!] \triangleq \mathbb{N}_\perp$
    — add a fresh $\perp$,
    representing **non-termination**



$$\frac{}{x_\sigma : \sigma} \qquad \frac{M : \tau}{\lambda x_\sigma . M : \tau}$$

$$\frac{M : \sigma \to \tau \quad N : \sigma}{MN : \tau}$$

$$\frac{M : \tau \to \tau}{\mathbf{rec}(M) : \tau}$$

$$\frac{}{0 : \mathbf{nat}} \quad \frac{}{1 : \mathbf{nat}} \quad \ldots$$

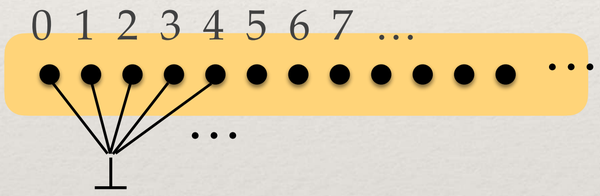
$$\frac{M : \mathbf{nat}}{\mathbf{s}(M) : \mathbf{nat}} \qquad \frac{M : \mathbf{nat}}{\mathbf{p}(M) : \mathbf{nat}}$$

$$\frac{M : \mathbf{nat} \quad N : \tau \quad P : \tau}{\mathbf{if}\ M = 0\ \mathbf{then}\ N\ \mathbf{else}\ P : \tau}$$

Types

- PCF terms are **typed**: $\sigma, \tau, \ldots ::= \mathbf{nat} \mid \sigma \to \tau$

- Semantics of types: $[\![\tau]\!]$ will be a pointed dcpo
  - $[\![\mathbf{nat}]\!] \triangleq \mathbb{N}_\perp$
    — add a fresh $\perp$,
    representing **non-termination**



  - $[\![\sigma \to \tau]\!] \triangleq [[\![\sigma]\!] \to [\![\tau]\!]]$
    — space of Scott-continuous maps from $[\![\sigma]\!]$ to $[\![\tau]\!]$

$$\frac{}{x_\sigma : \sigma} \qquad \frac{M : \tau}{\lambda x_\sigma . M : \tau}$$

$$\frac{M : \sigma \to \tau \quad N : \sigma}{MN : \tau}$$

$$\frac{M : \tau \to \tau}{\mathbf{rec}(M) : \tau}$$

$$\frac{}{0 : \mathbf{nat}} \quad \frac{}{1 : \mathbf{nat}} \quad \ldots$$

$$\frac{M : \mathbf{nat}}{\mathbf{s}(M) : \mathbf{nat}} \qquad \frac{M : \mathbf{nat}}{\mathbf{p}(M) : \mathbf{nat}}$$

$$\frac{M : \mathbf{nat} \quad N : \tau \quad P : \tau}{\mathbf{if}\ M = 0\ \mathbf{then}\ N\ \mathbf{else}\ P : \tau}$$

A denotational semantics for PCF

- Design (**denotational**) semantics $[\![M]\!]$ of terms $M : \tau$ so that $[\![M]\!]\rho \in [\![\tau]\!]$ for every environment ρ mapping variables to values (of the right types)

A denotational semantics for PCF

- Design (**denotational**) semantics $[\![M]\!]$ of terms $M : \tau$ so that $[\![M]\!]\rho \in [\![\tau]\!]$ for every environment ρ mapping variables to values (of the right types)

On **nat**:

- $[\![0]\!]\rho \mathrel{\hat{=}} 0$, $[\![1]\!]\rho \mathrel{\hat{=}} 1$, etc.
- $[\![\mathbf{s}(M)]\!]\rho \mathrel{\hat{=}} [\![M]\!]\rho{+}1$,
  $[\![\mathbf{p}(M)]\!]\rho \mathrel{\hat{=}} [\![M]\!]\rho{-}1$ if $[\![M]\!]\rho{\neq}0$, $\perp$
  $\perp$ otherwise
- $[\![\mathbf{if}\ M = 0\ \mathbf{then}\ N\ \mathbf{else}\ P]\!]\rho \mathrel{\hat{=}}$
  $[\![N]\!]\rho$ if $[\![M]\!]\rho{=}0$
  $[\![P]\!]\rho$ if $[\![M]\!]\rho{\neq}0, \perp$
  $\perp$ if $[\![M]\!]\rho{=}\perp$

A denotational semantics for PCF

- Design (**denotational**) semantics $[\![M]\!]$ of terms $M : \tau$ so that $[\![M]\!]\rho \in [\![\tau]\!]$ for every environment ρ mapping variables to values (of the right types)

On **nat**:

- $[\![0]\!]\rho \mathrel{\hat{=}} 0$, $[\![1]\!]\rho \mathrel{\hat{=}} 1$, etc.
- $[\![\mathbf{s}(M)]\!]\rho \mathrel{\hat{=}} [\![M]\!]\rho{+}1$,
  $[\![\mathbf{p}(M)]\!]\rho \mathrel{\hat{=}} [\![M]\!]\rho{-}1$ if $[\![M]\!]\rho{\neq}0$, $\bot$
  $\bot$ otherwise
- $[\![\mathbf{if}\ M = 0\ \mathbf{then}\ N\ \mathbf{else}\ P]\!]\rho \mathrel{\hat{=}}$
  $[\![N]\!]\rho$ if $[\![M]\!]\rho{=}0$
  $[\![P]\!]\rho$ if $[\![M]\!]\rho{\neq}0, \bot$
  $\bot$ if $[\![M]\!]\rho{=}\bot$

Lambda-calculus:

- $[\![x]\!]\rho \mathrel{\hat{=}} \rho(x)$
- $[\![MN]\!]\rho \mathrel{\hat{=}} [\![M]\!]\rho([\![N]\!]\rho)$
- $[\![\lambda x\,.\,M]\!]\rho \mathrel{\hat{=}} (V \mapsto [\![M]\!](\rho[x \mapsto V]))$
- $[\![\mathbf{rec}(M)]\!]\rho \mathrel{\hat{=}} \mathit{lfp}([\![M]\!]\rho)$

A denotational semantics for PCF

- Design (**denotational**) semantics $[\![M]\!]$ of terms $M : \tau$ so that $[\![M]\!]\rho \in [\![\tau]\!]$ for every environment ρ mapping variables to values (of the right types)

On **nat**:

- $[\![0]\!]\rho \mathrel{\hat{=}} 0$, $[\![1]\!]\rho \mathrel{\hat{=}} 1$, etc.
- $[\![\mathbf{s}(M)]\!]\rho \mathrel{\hat{=}} [\![M]\!]\rho{+}1$,
  $[\![\mathbf{p}(M)]\!]\rho \mathrel{\hat{=}} [\![M]\!]\rho{-}1$ if $[\![M]\!]\rho{\neq}0, \bot$
  $\bot$ otherwise
- $[\![\mathbf{if}\ M = 0\ \mathbf{then}\ N\ \mathbf{else}\ P]\!]\rho \mathrel{\hat{=}}$
  $[\![N]\!]\rho$ if $[\![M]\!]\rho{=}0$
  $[\![P]\!]\rho$ if $[\![M]\!]\rho{\neq}0, \bot$
  $\bot$ if $[\![M]\!]\rho{=}\bot$

Lambda-calculus:

- $[\![x]\!]\rho \mathrel{\hat{=}} \rho(x)$
- $[\![MN]\!]\rho \mathrel{\hat{=}} [\![M]\!]\rho([\![N]\!]\rho)$
- $[\![\lambda x\,.\,M]\!]\rho \mathrel{\hat{=}} (V \mapsto [\![M]\!](\rho[x \mapsto V]))$
- $[\![\mathbf{rec}(M)]\!]\rho \mathrel{\hat{=}} \mathit{lfp}([\![M]\!]\rho)$

Theorem 1. On a pointed dcpo X, every Scott-continuous map $f : X \to X$ has a least fixed point.

A denotational semantics for PCF

- Design (**denotational**) semantics $[\![M]\!]$ of terms $M : \tau$ so that $[\![M]\!]\rho \in [\![\tau]\!]$ for every environment ρ mapping variables to values (of the right types)

On **nat**:

- $[\![0]\!]\rho \mathrel{\hat{=}} 0$, $[\![1]\!]\rho \mathrel{\hat{=}} 1$, etc.
- $[\![\mathbf{s}(M)]\!]\rho \mathrel{\hat{=}} [\![M]\!]\rho{+}1$,
  $[\![\mathbf{p}(M)]\!]\rho \mathrel{\hat{=}} [\![M]\!]\rho{-}1$ if $[\![M]\!]\rho{\neq}0$, $\bot$
  $\bot$ otherwise
- $[\![\mathbf{if}\ M = 0\ \mathbf{then}\ N\ \mathbf{else}\ P]\!]\rho \mathrel{\hat{=}}$
  $[\![N]\!]\rho$ if $[\![M]\!]\rho{=}0$
  $[\![P]\!]\rho$ if $[\![M]\!]\rho{\neq}0, \bot$
  $\bot$ if $[\![M]\!]\rho{=}\bot$

Lambda-calculus:

- $[\![x]\!]\rho \mathrel{\hat{=}} \rho(x)$
- $[\![MN]\!]\rho \mathrel{\hat{=}} [\![M]\!]\rho([\![N]\!]\rho)$
- $[\![\lambda x\,.\,M]\!]\rho \mathrel{\hat{=}} (V \mapsto [\![M]\!](\rho[x \mapsto V]))$
- $[\![\mathbf{rec}(M)]\!]\rho \mathrel{\hat{=}} \mathit{lfp}([\![M]\!]\rho)$

Theorem 1. On a pointed dcpo X, every Scott-continuous map $f : X \to X$ has a least fixed point.

- Expressions have **transparent** semantics
  (functions are functions, application is application, etc.)
- **compositional** semantics: $[\![M]\!]\rho$ defined from the semantics of immediate **subterms** of M
- No **execution** mechanism involved

An operational semantics for PCF

- An **abstract machine** (à la Krivine) = a transition relation between **configurations** C, M

Contexts $C ::= \_ \mid C[\_N] \mid C[\mathbf{s}(\_)] \mid C[\mathbf{p}(\_)] \mid \mid C[\mathbf{if}\ \_ = 0\ \mathbf{then}\ N\ \mathbf{else}\ P]$

An operational semantics for PCF

- An **abstract machine** (à la Krivine) = a transition relation between **configurations** C, M

Contexts $C ::= \_ \mid C[\_N] \mid C[\mathbf{s}(\_)] \mid C[\mathbf{p}(\_)] \mid \mid C[\mathbf{if}\ \_ = 0\ \mathbf{then}\ N\ \mathbf{else}\ P]$

<u>Exploration rules</u> (looking for redexes)

$$
\begin{aligned}
C, MN &\to C[\_N], M \\
C, \mathbf{s}(M) &\to C[\mathbf{s}(\_)], M \\
C, \mathbf{p}(M) &\to C[\mathbf{p}(\_)], M \\
C, \mathbf{if}\ M = 0\ \mathbf{then}\ N\ \mathbf{else}\ P &\to C[\mathbf{if}\ \_ = 0\ \mathbf{then}\ N\ \mathbf{else}\ P], M
\end{aligned}
$$

An operational semantics for PCF

- An **abstract machine** (à la Krivine) = a transition relation between **configurations** C, M

Contexts $C ::= \_ \mid C[\_N] \mid C[\mathbf{s}(\_)] \mid C[\mathbf{p}(\_)] \mid \mid C[\mathbf{if}\ \_ = 0\ \mathbf{then}\ N\ \mathbf{else}\ P]$

<u>Exploration rules</u> (looking for redexes)

$$
\begin{aligned}
C, MN &\to C[\_N], M \\
C, \mathbf{s}(M) &\to C[\mathbf{s}(\_)], M \\
C, \mathbf{p}(M) &\to C[\mathbf{p}(\_)], M \\
C, \mathbf{if}\ M = 0\ \mathbf{then}\ N\ \mathbf{else}\ P &\to C[\mathbf{if}\ \_ = 0\ \mathbf{then}\ N\ \mathbf{else}\ P], M
\end{aligned}
$$

<u>Computation rules</u>

$$
\begin{aligned}
C[\_N], \lambda x\,.\,M &\to C, M[x := N] \\
C[\mathbf{s}(\_)], n &\to C, n+1 \\
C[\mathbf{p}(\_)], n+1 &\to C, n \\
C[\mathbf{if}\ \_ = 0\ \mathbf{then}\ N\ \mathbf{else}\ P], 0 &\to C, N \\
C[\mathbf{if}\ \_ = 0\ \mathbf{then}\ N\ \mathbf{else}\ P], n+1 &\to C, P \\
C, \mathbf{rec}(M) &\to C, M(\mathbf{rec}(M))
\end{aligned}
$$

… i.e., an interpreter

```
type term = V of string                    (* variables *)
          | A of term * term               (* applications *)
          | L of string * term             (* abstractions *)
          | R of term                      (* rec(M) *)
          | C of int                       (* 0, 1, 2, ... *)
          | S of term                      (* s(M) *)
          | P of term                      (* p(M) *)
          | I of term * term * term        (* if *)
```

… i.e., an interpreter

```
type term = V of string                    (* variables *)
          | A of term * term               (* applications *)
          | L of string * term             (* abstractions *)
          | R of term                      (* rec(M) *)
          | C of int                       (* 0, 1, 2, ... *)
          | S of term                      (* s(M) *)
          | P of term                      (* p(M) *)
          | I of term * term * term        (* if *)
```

```
type ctx = Ec                              (* _ *)
         | Ac of ctx *term                 (* C[_N] *)
         | Sc of ctx                       (* C[s(_)] *)
         | Pc of ctx                       (* C[p(_)] *)
         | Ic of ctx * term * term         (* C[if ...] *)
```

… i.e., an interpreter

```
type term = V of string                    (* variables *)
          | A of term * term               (* applications *)
          | L of string * term             (* abstractions *)
          | R of term                      (* rec(M) *)
          | C of int                       (* 0, 1, 2, ... *)
          | S of term                      (* s(M) *)
          | P of term                      (* p(M) *)
          | I of term * term * term        (* if *)
```

```
type ctx = Ec                              (* _ *)
         | Ac of ctx *term                 (* C[_N] *)
         | Sc of ctx                       (* C[s(_)] *)
         | Pc of ctx                       (* C[p(_)] *)
         | Ic of ctx * term * term         (* C[if ...] *)
```

```
let rec exec (c:ctx) (t:term) =
    match t with
      (* exploration *)
      A(m,n) ->   exec (Ac(c,n),m)
    | S(m) ->     exec (Sc(c),m)
    | P(m) ->     exec (Pc(c),m)
    | I(m,n,p) -> exec (Ic(c,n,p),m)
      (* computation *)
    | _ ->        compute c t
```

$$C, MN \to C[\_N], M$$
$$C, \mathbf{s}(M) \to C[\mathbf{s}(\_)], M$$
$$C, \mathbf{p}(M) \to C[\mathbf{p}(\_)], M$$
$$C, \mathbf{if}\ M = 0\ \mathbf{then}\ N\ \mathbf{else}\ P \to C[\mathbf{if}\ \_ = 0\ \mathbf{then}\ N\ \mathbf{else}\ P], M$$

… i.e., an interpreter

```
type term = V of string                    (* variables *)
          | A of term * term               (* applications *)
          | L of string * term             (* abstractions *)
          | R of term                      (* rec(M) *)
          | C of int                       (* 0, 1, 2, ... *)
          | S of term                      (* s(M) *)
          | P of term                      (* p(M) *)
          | I of term * term * term        (* if *)
```

```
type ctx = Ec                              (* _ *)
         | Ac of ctx *term                 (* C[_N] *)
         | Sc of ctx                       (* C[s(_)] *)
         | Pc of ctx                       (* C[p(_)] *)
         | Ic of ctx * term * term         (* C[if ...] *)
```

```
let rec exec (c:ctx) (t:term) =
    match t with
      (* exploration *)
      A(m,n) ->   exec (Ac(c,n),m)
    | S(m) ->     exec (Sc(c),m)
    | P(m) ->     exec (Pc(c),m)
    | I(m,n,p) -> exec (Ic(c,n,p),m)
      (* computation *)
    | _ ->        compute c t
```

```
and compute (c:ctx) (t:term) =
    match c, t with
      Ac(c',n), L(x,m)          ->   exec c' (subst m x n)
    | Sc(c'), I(i)              ->   exec c' (I(i+1))
    | Pc(c'), I(i) when i>0     ->   exec c' (I(i-1))
    | Ic(c',n,p), I(0)          ->   exec c' n
    | Ic(c',n,p), I(i) when i>0 ->   exec c' p
    | _, R(m)                   ->   exec c (A(m,t))
    | Ec, result                ->   result (* finished! *)
    | _, _ -> failwith « stumped »
```

$$C, MN \to C[\_N], M$$
$$C, \mathbf{s}(M) \to C[\mathbf{s}(\_)], M$$
$$C, \mathbf{p}(M) \to C[\mathbf{p}(\_)], M$$
$$C, \mathbf{if}\ M = 0\ \mathbf{then}\ N\ \mathbf{else}\ P \to C[\mathbf{if}\ \_ = 0\ \mathbf{then}\ N\ \mathbf{else}\ P], M$$

$$C[\_N], \lambda x\,.\,M \to C, M[x := N]$$
$$C[\mathbf{s}(\_)], n \to C, n+1$$
$$C[\mathbf{p}(\_)], n+1 \to C, n$$
$$C[\mathbf{if}\ \_ = 0\ \mathbf{then}\ N\ \mathbf{else}\ P], 0 \to C, N$$
$$C[\mathbf{if}\ \_ = 0\ \mathbf{then}\ N\ \mathbf{else}\ P], n+1 \to C, P$$
$$C, \mathbf{rec}(M) \to C, M(\mathbf{rec}(M))$$

… which really runs

```
goubault@macbook-pro-de-jean Topics2023 % gimml
GimML comes with ABSOLUTELY NO WARRANTY (See file COPYRIGHT).
GimML for Darwin 20.5.0, by Jean Goubault-Larrecq (c) 2021.
> use "pcf.ml"
>
```

… which really runs

```
goubault@macbook-pro-de-jean Topics2023 % gimml
GimML comes with ABSOLUTELY NO WARRANTY (See file COPYRIGHT).
GimML for Darwin 20.5.0, by Jean Goubault-Larrecq (c) 2021.
> use "pcf.ml"
>
```

$$\mathit{Plus} \triangleq \mathbf{rec}(\lambda \mathit{plus} \,.\, \lambda m \,.\, \lambda n \,.\, \mathbf{if}\ n = 0\ \mathbf{then}\ m\ \mathbf{else}\ \mathit{plus}\ m\ (\mathbf{p}(n)))$$

$$\mathit{Times} \triangleq \mathbf{rec}(\lambda \mathit{times} \,.\, \lambda m \,.\, \lambda n \,.\, \mathbf{if}\ n = 0\ \mathbf{then}\ 0\ \mathbf{else}\ \mathit{Plus}\ m\ (\mathit{times}\ m\ (\mathbf{p}(n))))$$

$$\mathit{Fact} \triangleq \mathbf{rec}(\lambda \mathit{fact} \,.\, \lambda n \,.\, \mathbf{if}\ n = 0\ \mathbf{then}\ 1\ \mathbf{else}\ \mathit{Times}\ n\ (\mathit{fact}\ (\mathbf{p}(n))))$$

… which really runs

```
goubault@macbook-pro-de-jean Topics2023 % gimml
GimML comes with ABSOLUTELY NO WARRANTY (See file COPYRIGHT).
GimML for Darwin 20.5.0, by Jean Goubault-Larrecq (c) 2021.
> use "pcf.ml"
>
```

$$\mathit{Plus} \triangleq \mathbf{rec}(\lambda \mathit{plus} \,.\, \lambda m \,.\, \lambda n \,.\, \mathbf{if}\ n = 0\ \mathbf{then}\ m\ \mathbf{else}\ \mathit{plus}\ m\ (\mathbf{p}(n)))$$

$$\mathit{Times} \triangleq \mathbf{rec}(\lambda \mathit{times} \,.\, \lambda m \,.\, \lambda n \,.\, \mathbf{if}\ n = 0\ \mathbf{then}\ 0\ \mathbf{else}\ \mathit{Plus}\ m\ (\mathit{times}\ m\ (\mathbf{p}(n))))$$

$$\mathit{Fact} \triangleq \mathbf{rec}(\lambda \mathit{fact} \,.\, \lambda n \,.\, \mathbf{if}\ n = 0\ \mathbf{then}\ 1\ \mathbf{else}\ \mathit{Times}\ n\ (\mathit{fact}\ (\mathbf{p}(n))))$$

The two semantics are related

- **Theorem (soundness).** If $C, M \to^* C', M'$ then $[\![C[M]]\!]\rho = [\![C'[M']]\!]\rho$

The two semantics are related

- **Theorem (soundness).** If $C, M \to^* C', M'$ then $[\![C[M]]\!]\rho = [\![C'[M']]\!]\rho$
- In particular, if $\_, M \to^* \_, n$ ($n \in \mathbb{N}$), then $[\![M]\!]\rho = n$

The two semantics are related

- **Theorem (soundness).** If $C, M \to^* C', M'$ then $[\![C[M]]\!]\rho = [\![C'[M']]\!]\rho$
- In particular, if $\_, M \to^* \_, n$ ($n \in \mathbb{N}$), then $[\![M]\!]\rho = n$
- **Theorem (adequacy).** If $[\![M]\!]\rho = n \in \mathbb{N}$ ($\neq \bot$),
  then the machine **terminates**: $\_, M \to^* \_, n$

The two semantics are related

- **Theorem (soundness).** If $C, M \to^* C', M'$ then $[\![C[M]]\!]\rho = [\![C'[M']]\!]\rho$
- In particular, if $\_, M \to^* \_, n$ ($n \in \mathbb{N}$), then $[\![M]\!]\rho = n$
- **Theorem (adequacy).** If $[\![M]\!]\rho = n \in \mathbb{N}$ ($\neq \bot$),
  then the machine **terminates**: $\_, M \to^* \_, n$
- That is trickier to prove, and uses **logical relations** [Plotkin 77].

The two semantics are related

- **Theorem (soundness).** If $C, M \to^* C', M'$ then $[\![C[M]]\!]\rho = [\![C'[M']]\!]\rho$
- In particular, if $\_, M \to^* \_, n$ ($n \in \mathbb{N}$), then $[\![M]\!]\rho = n$
- **Theorem (adequacy).** If $[\![M]\!]\rho = n \in \mathbb{N}$ ($\neq \bot$),
  then the machine **terminates**: $\_, M \to^* \_, n$
- That is trickier to prove, and uses **logical relations** [Plotkin 77].
- **Note:** only works at the **observable** type **nat**
  (although some variants can be made to work at function types, too [BloomRiecke 88])

Observational equivalence

- Let $M \cong N$ if and only if
  M and N are (operationally) **indistinguishable**
  by any context of **observable** type

Observational equivalence

- Let $M \cong N$ if and only if
  M and N are (operationally) **indistinguishable**
  by any context of **observable** type

I.e., you can replace M by N
(or conversely)
in any program,
and you will never see the difference
(except for timing, e.g.)

Observational equivalence

- Let $M \cong N$ if and only if
  M and N are (operationally) **indistinguishable**
  by any context of **observable** type
- Formally: $M \cong N$ (where M, N are closed) iff
  for every context $C : \mathbf{nat}$, for every $n \in \mathbb{N}$,
  $C, M \to^* \_, n$ if and only if $C, N \to^* \_, n$

I.e., you can replace M by N
(or conversely)
in any program,
and you will never see the difference
(except for timing, e.g.)

Observational equivalence

- Let $M \cong N$ if and only if
  M and N are (operationally) **indistinguishable**
  by any context of **observable** type
- Formally: $M \cong N$ (where M, N are closed) iff
  for every context C : **nat**, for every $n \in \mathbb{N}$,
  $C, M \to^* \_, n$ if and only if $C, N \to^* \_, n$
- **Fact.** If $[\![M]\!]=[\![N]\!]$ then $M \cong N$.

I.e., you can replace M by N
(or conversely)
in any program,
and you will never see the difference
(except for timing, e.g.)

where M, N are closed
(no ρ necessary in $[\![M]\!]\rho$)

Observational equivalence

- Let $M \cong N$ if and only if
  M and N are (operationally) **indistinguishable**
  by any context of **observable** type
- Formally: $M \cong N$ (where M, N are closed) iff
  for every context C : **nat**, for every $n \in \mathbb{N}$,
  $C, M \to^* \_, n$ if and only if $C, N \to^* \_, n$

I.e., you can replace M by N
(or conversely)
in any program,
and you will never see the difference
(except for timing, e.g.)

- **Fact.** If $[\![M]\!]=[\![N]\!]$ then $M \cong N$.

where M, N are closed
(no ρ necessary in $[\![M]\!]\rho$)

- *Proof.* If $C, M \to^* \_, n$ then $[\![C[M]]\!] = n$ (soundness)
  so $[\![C[N]]\!] = n$ (denotational semantics is **compositional**)
  so $C, N \to^* \_, n$ (adequacy) $\square$

Full abstraction

- **Fact.** If $[\![M]\!]=[\![N]\!]$ then $M \cong N$.

Full abstraction

- **Fact.** If $[\![M]\!]=[\![N]\!]$ then $M \cong N$.
- **Full abstraction** is when this is an equivalence…
  and it fails in PCF [Sazonov76, Plotkin 77]:
  some murky issue with **parallel or**
  (a function that exists **semantically** but that not in the **syntax**)
  … which I will ignore here

Probabilistic PCF

Probabilistic choice

- Add instruction $M \oplus N$: do M or N, with probability $\frac{1}{2}$

Probabilistic choice

- Add instruction $M \oplus N$: do M or N, with probability $\frac{1}{2}$
- Operational semantics with extra **labeled** transitions

$$C, M \oplus N \xrightarrow{\frac{1}{2}} C, M$$

$$C, M \oplus N \xrightarrow{\frac{1}{2}} C, N$$

Probabilistic choice

- Add instruction $M \oplus N$: do M or N, with probability $\frac{1}{2}$
- Operational semantics with extra **labeled** transitions

$$C, M \oplus N \xrightarrow{\frac{1}{2}} C, M \qquad C, M \oplus N \xrightarrow{\frac{1}{2}} C, N$$

- Denotational semantics is now (as a first approach) a (sub)**probability distribution** over possible outcomes

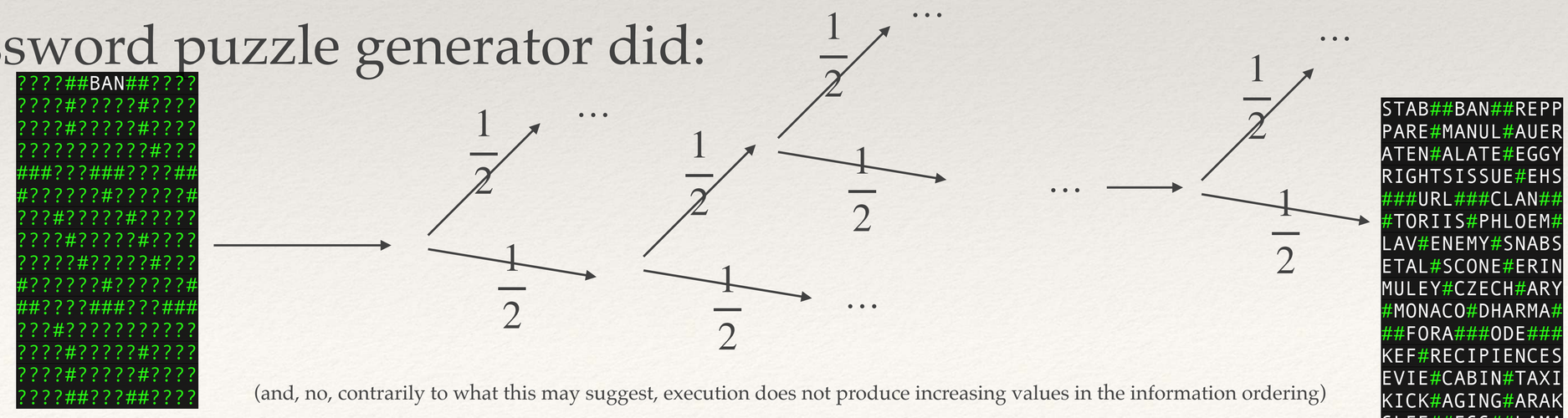
Probabilistic choice

- Add instruction $M \oplus N$: do M or N, with probability $\frac{1}{2}$
- Operational semantics with extra **labeled** transitions

$$C, M \oplus N \xrightarrow{\frac{1}{2}} C, M \qquad C, M \oplus N \xrightarrow{\frac{1}{2}} C, N$$

- Denotational semantics is now (as a first approach) a (sub)**probability distribution** over possible outcomes
- What our crossword puzzle generator did:

(more or less)



(and, no, contrarily to what this may suggest, execution does not produce increasing values in the information ordering)

Probabilistic PCF

$M, N, P, \ldots ::= \ldots$	(as in PCF)
$\mid M \oplus N$	probabilistic choice
$\mid \mathbf{ret}\ M$	monad unit
$\mid \mathbf{do}\ x_\sigma = M; N$	sequential composition

Probabilistic PCF

- $M, N, P, \ldots ::= \ldots$ (as in PCF)
  - $\mid M \oplus N$ probabilistic choice
  - $\mid \mathbf{ret}\ M$ monad unit
  - $\mid \mathbf{do}\ x_\sigma = M; N$ sequential composition
- Types: $\sigma, \tau, \ldots ::= \mathbf{nat} \mid \mathbf{unit} \mid \sigma \to \tau \mid \mathbf{T}\tau$

  $\mathbf{T}\tau$: monadic types [Moggi 91]

$$\frac{M : \mathbf{T}\tau \qquad N : \mathbf{T}\tau}{M \oplus N : \mathbf{T}\tau} \qquad \frac{}{* : \mathbf{unit}}$$

$$\frac{M : \tau}{\mathbf{ret}\ M : \mathbf{T}\tau} \qquad \frac{M : \mathbf{T}\sigma \qquad N : \mathbf{T}\tau}{\mathbf{do}\ x_\sigma = M; N : \mathbf{T}\tau}$$

$\mathbf{T}\tau$ = type of **(first-class)** distributions

Probabilistic PCF

- $M, N, P, \ldots ::= \ldots$ (as in PCF)
  - $\mid M \oplus N$ probabilistic choice
  - $\mid \mathbf{ret}\ M$ monad unit
  - $\mid \mathbf{do}\ x_\sigma = M; N$ sequential composition

$$\frac{M : \mathbf{T}\tau \qquad N : \mathbf{T}\tau}{M \oplus N : \mathbf{T}\tau} \qquad \frac{}{* : \mathbf{unit}}$$

$$\frac{M : \tau}{\mathbf{ret}\ M : \mathbf{T}\tau} \qquad \frac{M : \mathbf{T}\sigma \qquad N : \mathbf{T}\tau}{\mathbf{do}\ x_\sigma = M; N : \mathbf{T}\tau}$$

- Types: $\sigma, \tau, \ldots ::= \mathbf{nat} \mid \mathbf{unit} \mid \sigma \to \tau \mid \mathbf{T}\tau$

  $\mathbf{T}\tau$ = type of **(first-class)** distributions

  $\mathbf{T}\tau$: monadic types [Moggi 91]

- New operational rules:

Exploration rules

$$C, \mathbf{do}\ x = M; N \to C[\mathbf{do}\ x = \_; N], M$$
$$\_, \mathbf{ret}\ M \to \mathbf{ret}\ \_, M$$

Computation rules

$$C[\mathbf{do}\ x = \_; N], \mathbf{ret}\ M \to C, N[x := M]$$
$$C, M \oplus N \to^{1/2} M$$
$$C, M \oplus N \to^{1/2} N$$

An extended interpreter

```
type term = V of string                    (* variables *)
          | A of term * term               (* applications *)
          | ...
          | Flip of term * term            (* M⊕N *)
          | Ret of term                    (* ret M *)
          | Do of string * term
                * term                     (* do x=M;N *)
```

An extended interpreter

```
type term = V of string                    (* variables *)
          | A of term * term               (* applications *)
          | ...
          | Flip of term * term            (* M⊕N *)
          | Ret of term                    (* ret M *)
          | Do of string * term
                * term                     (* do x=M;N *)
```

```
type ctx = Ec                              (* _ *)
         | Ac of ctx *term                 (* C[_N] *)
         | ...
         | Rc                              (* ret _ *)
         | Dc of ctx * string
                * term                     (* C[do x=_;N] *)
```

An extended interpreter

```
type term = V of string                    (* variables *)
          | A of term * term               (* applications *)
          | ...
          | Flip of term * term            (* M⊕N *)
          | Ret of term                    (* ret M *)
          | Do of string * term
                * term                     (* do x=M;N *)
```

```
type ctx = Ec                              (* _ *)
         | Ac of ctx *term                 (* C[_N] *)
         | ...
         | Rc                              (* ret _ *)
         | Dc of ctx * string
                * term                     (* C[do x=_;N] *)
```

```
let rec exec (c:ctx) (t:term) =
    match t with
      (* exploration *)
      ...
    | Do(x,m,n) ->  exec (Dc(c,x,n),m)
    | Ret(m) when c==Ec
                  -> exec (Rc,m)
      (* computation *)
    | _ ->            compute c t
```

An extended interpreter

```
type term = V of string                    (* variables *)
          | A of term * term               (* applications *)
          | ...
          | Flip of term * term            (* M⊕N *)
          | Ret of term                    (* ret M *)
          | Do of string * term
                * term                     (* do x=M;N *)
```

```
type ctx = Ec                              (* _ *)
         | Ac of ctx *term                 (* C[_N] *)
         | ...
         | Rc                              (* ret _ *)
         | Dc of ctx * string
                * term                     (* C[do x=_;N] *)
```

```
let rec exec (c:ctx) (t:term) =
    match t with
      (* exploration *)
      ...
    | Do(x,m,n) ->  exec (Dc(c,x,n),m)
    | Ret(m) when c==Ec
                  -> exec (Rc,m)
      (* computation *)
    | _ ->          compute c t
```

$$C, \mathbf{do}\ x = M; N \to C[\mathbf{do}\ x = \_; N], M$$
$$\_, \mathbf{ret}\ M \to \mathbf{ret}\ \_, M$$

```
and compute (c:ctx) (t:term) =
    match c, t with
      ...
    | Dc(c',x,n), Ret(m)   ->     exec c' (subst n x m)
    | _, Flip(m,n)         ->
      if Random.bool() then      exec c m
                       else      exec c n
    | Rc, result           ->    result (* finished! *)
    | Ec, result           ->    result (* finished! *)
    | _, _ -> failwith « stumped »
```

$$C[\mathbf{do}\ x = \_; N], \mathbf{ret}\ M \to C, N[x := M]$$
$$C, M \oplus N \to^{1/2} M$$
$$C, M \oplus N \to^{1/2} N$$

Let’s run it, on another example

```
goubault@macbook-pro-de-jean Topics2023 % gimml
GimML comes with ABSOLUTELY NO WARRANTY (See file COPYRIGHT).
GimML for Darwin 20.5.0, by Jean Goubault-Larrecq (c) 2021.
> use "prob_pcf.ml"
>
```

Let's run it, on another example

```
goubault@macbook-pro-de-jean Topics2023 % gimml
GimML comes with ABSOLUTELY NO WARRANTY (See file COPYRIGHT).
GimML for Darwin 20.5.0, by Jean Goubault-Larrecq (c) 2021.
> use "prob_pcf.ml"
>
```

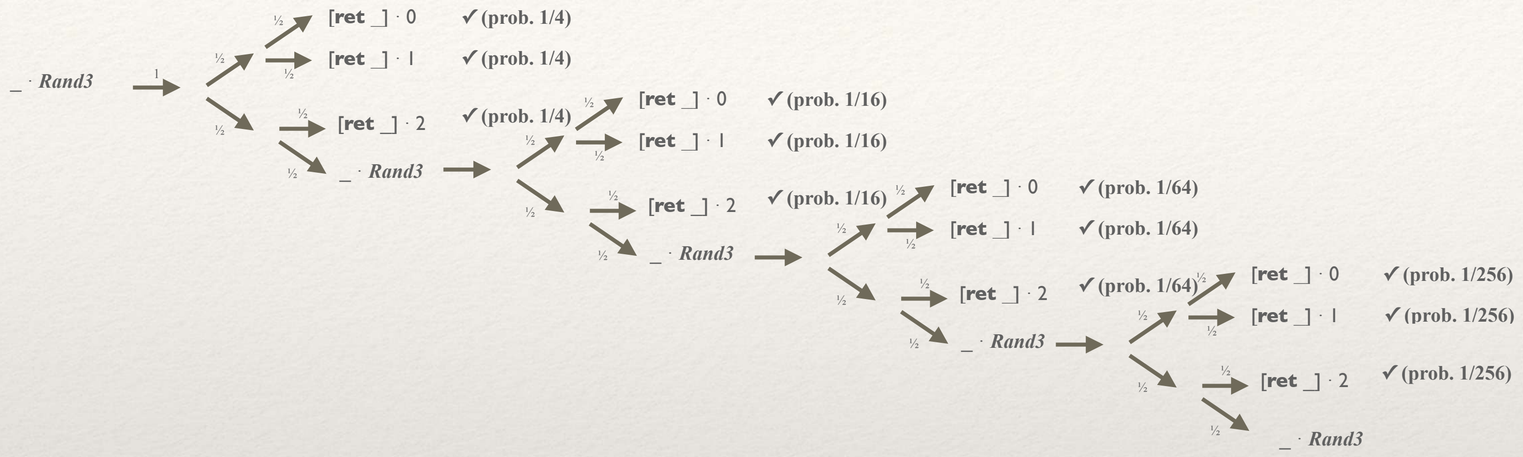
$$Rand3 \triangleq \mathbf{rec}(\lambda r\,.\ (\mathbf{ret}\ 0 \oplus \mathbf{ret}\ 1) \oplus (\mathbf{ret}\ 2 \oplus r))$$

What does Rand3 do?

_ · *Rand3*

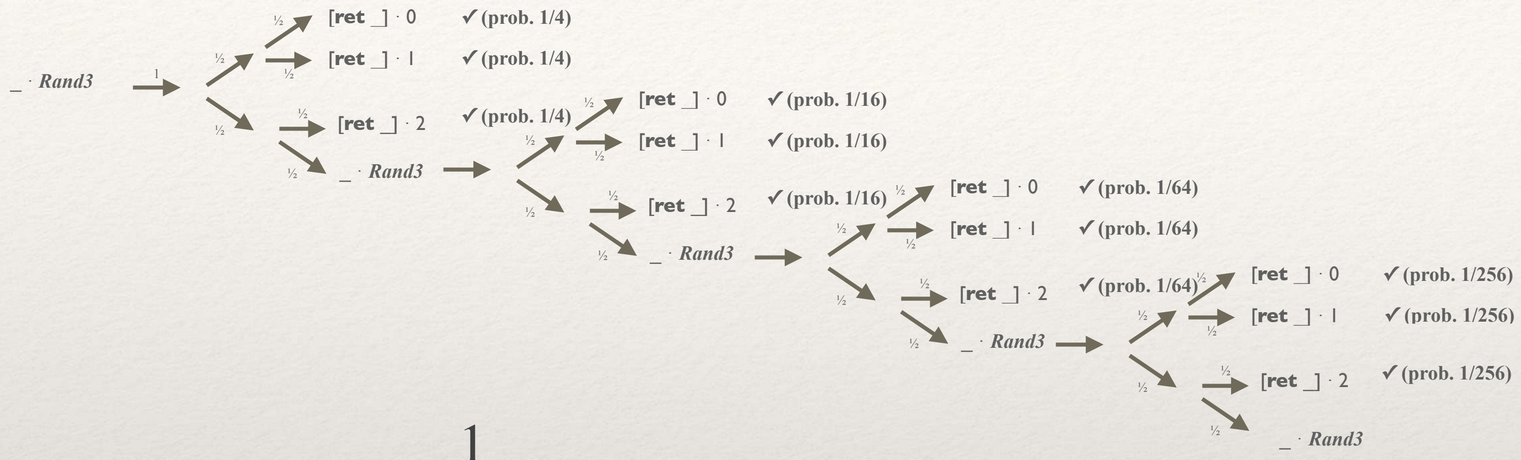
$$
\begin{aligned}
\mathit{Rand3} \triangleq \mathbf{rec}(\lambda r\,.\ & \\
& (\mathbf{ret}\ 0 \oplus \mathbf{ret}\ 1) \\
& \oplus (\mathbf{ret}\ 2 \oplus r))
\end{aligned}
$$

What does Rand3 do?



$$\mathit{Rand3} \triangleq \mathbf{rec}(\lambda r\,.\ (\mathbf{ret}\ 0 \oplus \mathbf{ret}\ 1) \oplus (\mathbf{ret}\ 2 \oplus r))$$

What does Rand3 do?

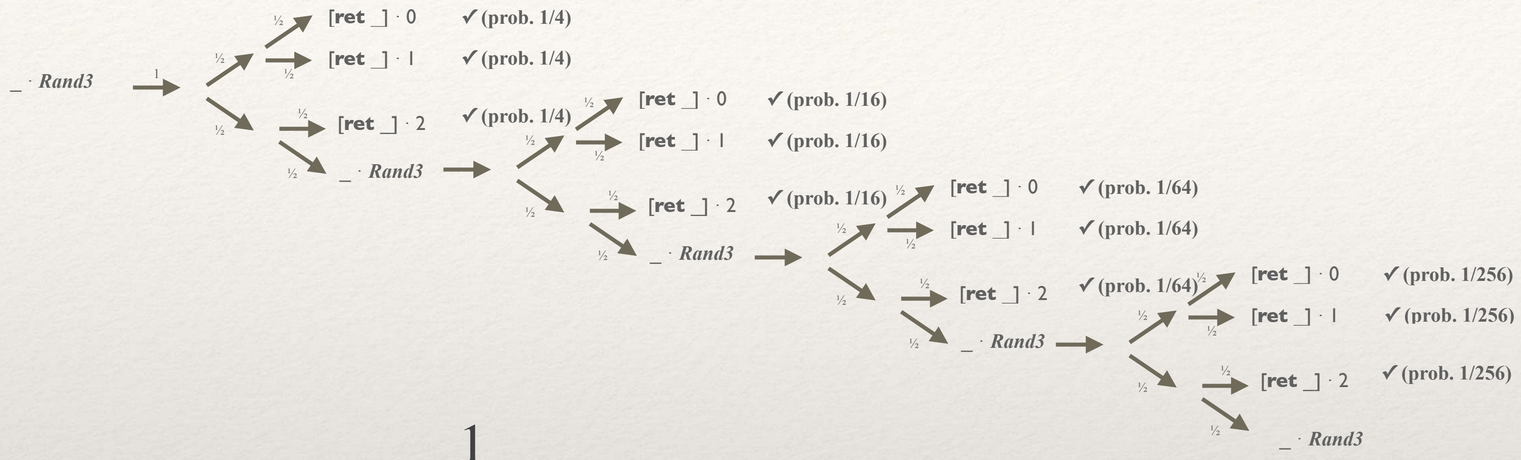


$$Rand3 \triangleq \mathbf{rec}(\lambda r\,.\ (\mathbf{ret}\ 0 \oplus \mathbf{ret}\ 1) \oplus (\mathbf{ret}\ 2 \oplus r))$$

Rejection sampling

- $\Pr[\_\,.\,Rand3 \downarrow n] = \dfrac{1}{3}$ for every $n \in \{0,1,2\}$: proof?

What does Rand3 do?

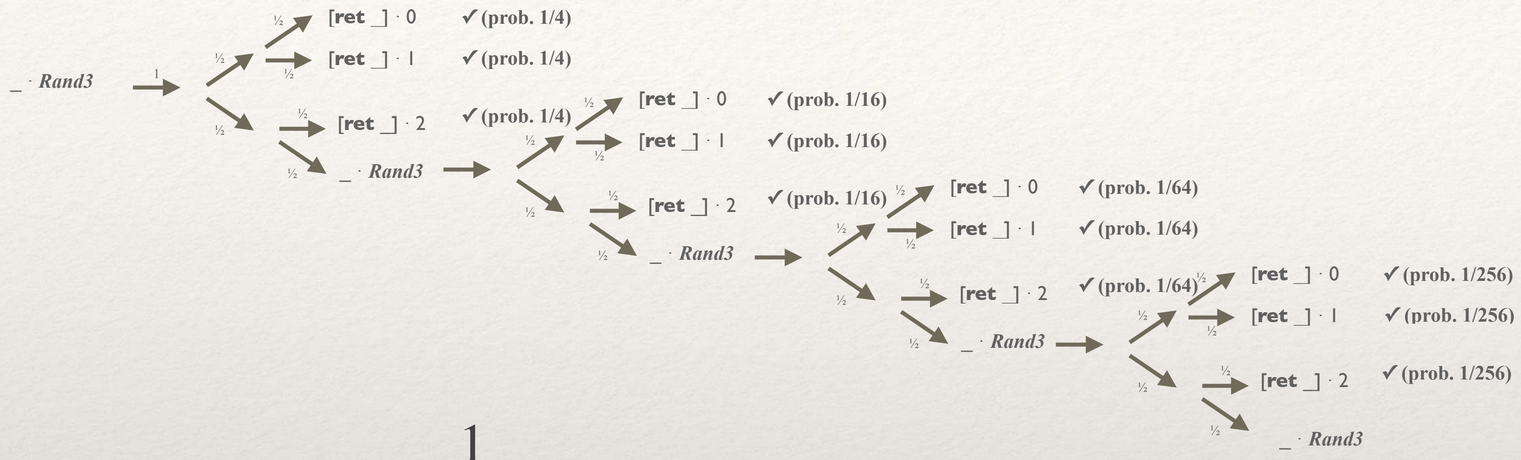


$$Rand3 \triangleq \mathbf{rec}(\lambda r\,.\ (\mathbf{ret}\ 0 \oplus \mathbf{ret}\ 1) \oplus (\mathbf{ret}\ 2 \oplus r))$$

Rejection sampling

- $\Pr[\_\,.\,Rand3 \downarrow n] = \dfrac{1}{3}$ for every $n \in \{0,1,2\}$: proof?
- **Operationally**: $\dfrac{1}{4} + \dfrac{1}{16} + \dfrac{1}{64} + \dfrac{1}{256} + \cdots = \dfrac{1}{3}$

What does Rand3 do?



$$Rand3 \triangleq \mathbf{rec}(\lambda r\,.\ (\mathbf{ret}\ 0 \oplus \mathbf{ret}\ 1) \oplus (\mathbf{ret}\ 2 \oplus r))$$

Rejection sampling

- $\Pr[\_\,.\,Rand3 \downarrow n] = \dfrac{1}{3}$ for every $n \in \{0,1,2\}$: proof?
- **Operationally**: $\dfrac{1}{4} + \dfrac{1}{16} + \dfrac{1}{64} + \dfrac{1}{256} + \cdots = \dfrac{1}{3}$
- **Denotationally**: $[\![Rand3]\!]\rho(\{n\})$ (denotational measure of getting n) $= x$

  where $\dfrac{1}{4} + \dfrac{1}{4}\,.\,x = x$ (**rec** defines a **fixed point**)

Denotational semantics for probabilistic PCF

- Introduced in Claire Jones' PhD thesis [Jones 90]
- $[\![\mathbf{T}\tau]\!] \triangleq \mathbf{V}_{\leq 1}([\![\tau]\!])$ dcpo of **subprobability valuations** on $[\![\tau]\!]$
  (~ think « subprobability measures »)

Denotational semantics for probabilistic PCF

- Introduced in Claire Jones' PhD thesis [Jones 90]
- $[\![\mathbf{T}\tau]\!] \triangleq \mathbf{V}_{\leq 1}([\![\tau]\!])$ dcpo of **subprobability valuations** on $[\![\tau]\!]$
  (~ think « subprobability measures »)
- $[\![M \oplus N]\!]\rho \triangleq \frac{1}{2}[\![M]\!]\rho + \frac{1}{2}[\![N]\!]\rho$

$$\frac{M : \mathbf{T}\tau \qquad N : \mathbf{T}\tau}{M \oplus N : \mathbf{T}\tau}$$

$$\frac{M : \tau}{\mathbf{ret}\ M : \mathbf{T}\tau} \qquad \frac{M : \mathbf{T}\sigma \qquad N : \mathbf{T}\tau}{\mathbf{do}\ x_\sigma = M; N : \mathbf{T}\tau}$$

Denotational semantics for probabilistic PCF

- Introduced in Claire Jones' PhD thesis [Jones 90]
- $[\![\mathbf{T}\tau]\!] \triangleq \mathbf{V}_{\leq 1}([\![\tau]\!])$ dcpo of **subprobability valuations** on $[\![\tau]\!]$
  (~ think « subprobability measures »)
- $[\![M \oplus N]\!]\rho \triangleq \frac{1}{2}[\![M]\!]\rho + \frac{1}{2}[\![N]\!]\rho$
- $[\![\mathbf{ret}\ M]\!]\rho \triangleq \delta_{[\![M]\!]\rho}$

  $[\![\mathbf{do}\ x_\sigma = M; N]\!]\rho \triangleq (U \mapsto \int_x [\![N]\!]\rho(U)\, d[\![M]\!]\rho)$

$$\frac{M : \mathbf{T}\tau \qquad N : \mathbf{T}\tau}{M \oplus N : \mathbf{T}\tau}$$

$$\frac{M : \tau}{\mathbf{ret}\ M : \mathbf{T}\tau} \qquad \frac{M : \mathbf{T}\sigma \qquad N : \mathbf{T}\tau}{\mathbf{do}\ x_\sigma = M; N : \mathbf{T}\tau}$$

Denotational semantics for probabilistic PCF

- Introduced in Claire Jones' PhD thesis [Jones 90]
- $[\![\mathbf{T}\tau]\!] \triangleq \mathbf{V}_{\leq 1}([\![\tau]\!])$ dcpo of **subprobability valuations** on $[\![\tau]\!]$
  (~ think « subprobability measures »)
- $[\![M \oplus N]\!]\rho \triangleq \frac{1}{2}[\![M]\!]\rho + \frac{1}{2}[\![N]\!]\rho$
- $[\![\mathbf{ret}\ M]\!]\rho \triangleq \delta_{[\![M]\!]\rho}$

  $[\![\mathbf{do}\ x_\sigma = M; N]\!]\rho \triangleq (U \mapsto \int_x [\![N]\!]\rho(U)\, d[\![M]\!]\rho)$

Let us define all that first!

$$\frac{M : \mathbf{T}\tau \qquad N : \mathbf{T}\tau}{M \oplus N : \mathbf{T}\tau}$$

$$\frac{M : \tau}{\mathbf{ret}\ M : \mathbf{T}\tau} \qquad \frac{M : \mathbf{T}\sigma \qquad N : \mathbf{T}\tau}{\mathbf{do}\ x_\sigma = M; N : \mathbf{T}\tau}$$

Continuous valuations

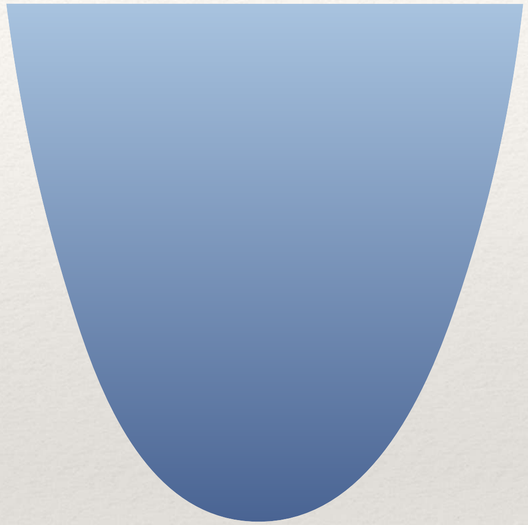
… We come to some topology at last

- Measures give mass to **measurable** subsets
- Following a remark by [Smyth 83] that testable property=open, Jones remarks that it is more natural to measure **open** subsets
- Hence what are the **open** subsets of a dcpo?

  Let me introduce the **Scott topology.**

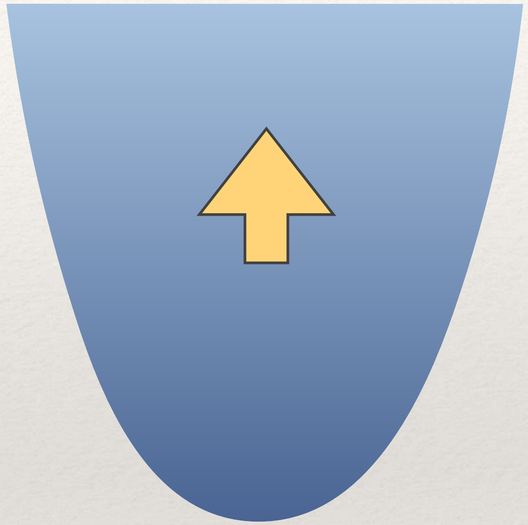
The Scott topology

- A subset U of a dcpo X is **Scott-open** iff:



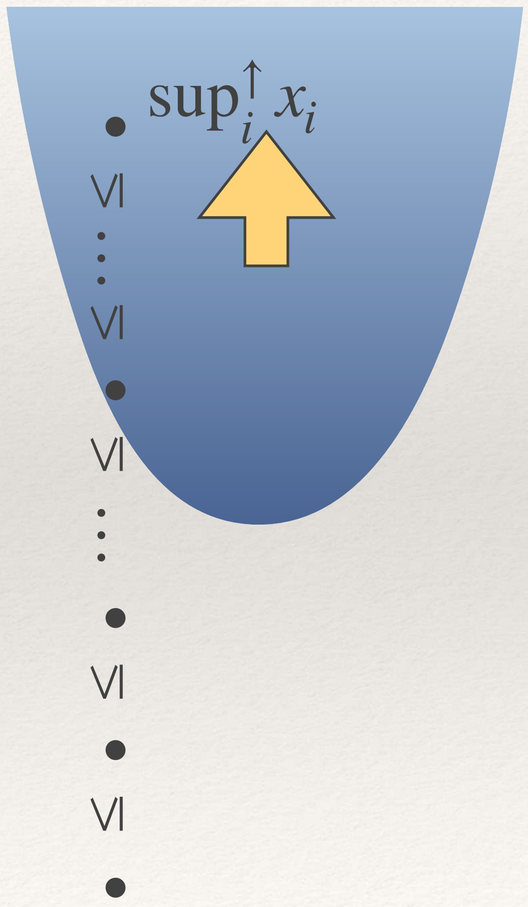
The Scott topology

- A subset U of a dcpo X is **Scott-open** iff:
  - U is **upwards closed** $(x \in U, x \leq y \Rightarrow y \in U)$



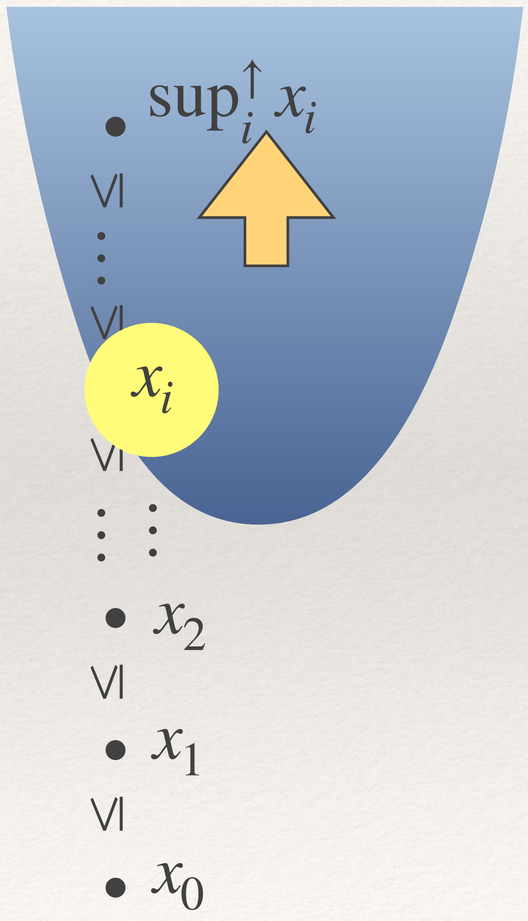
The Scott topology

- A subset U of a dcpo X is **Scott-open** iff:
  - U is **upwards closed** ($x \in U, x \leq y \Rightarrow y \in U$)
  - if $\sup_i^{\uparrow} x_i \in U$ then some $x_i \in U$ already



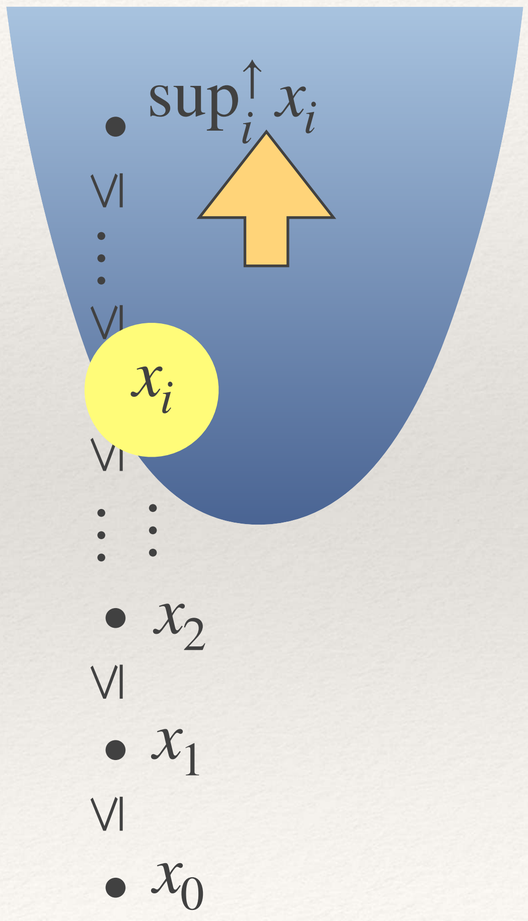
The Scott topology

- A subset U of a dcpo X is **Scott-open** iff:
  - U is **upwards closed** ($x \in U, x \leq y \Rightarrow y \in U$)
  - if $\sup_i^{\uparrow} x_i \in U$ then some $x_i \in U$ already



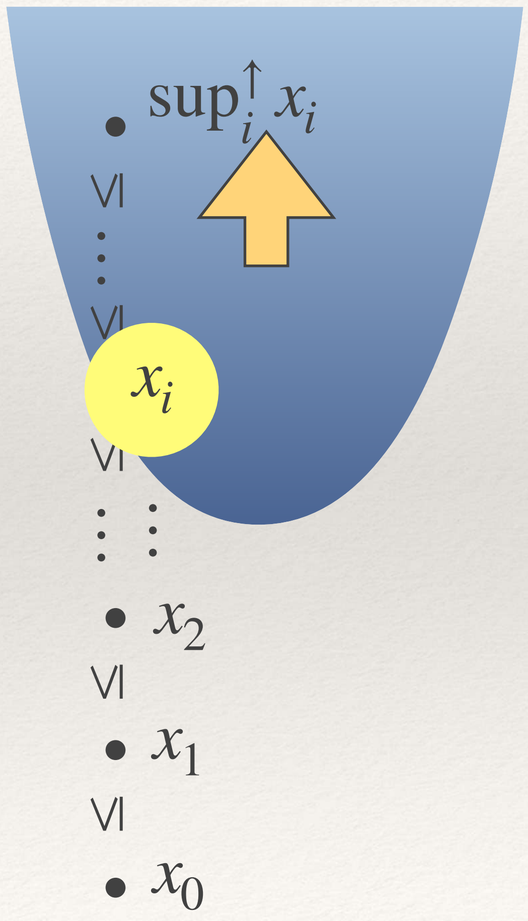
The Scott topology

- A subset U of a dcpo X is **Scott-open** iff:
  - U is **upwards closed** ($x \in U, x \leq y \Rightarrow y \in U$)
  - if $\sup_i^{\uparrow} x_i \in U$ then some $x_i \in U$ already
- Equivalently, iff its characteristic map $\chi_U \colon X \to \mathbb{S}$ is Scott-continuous, where $\mathbb{S} \mathrel{\hat{=}} \{0 < 1\}$



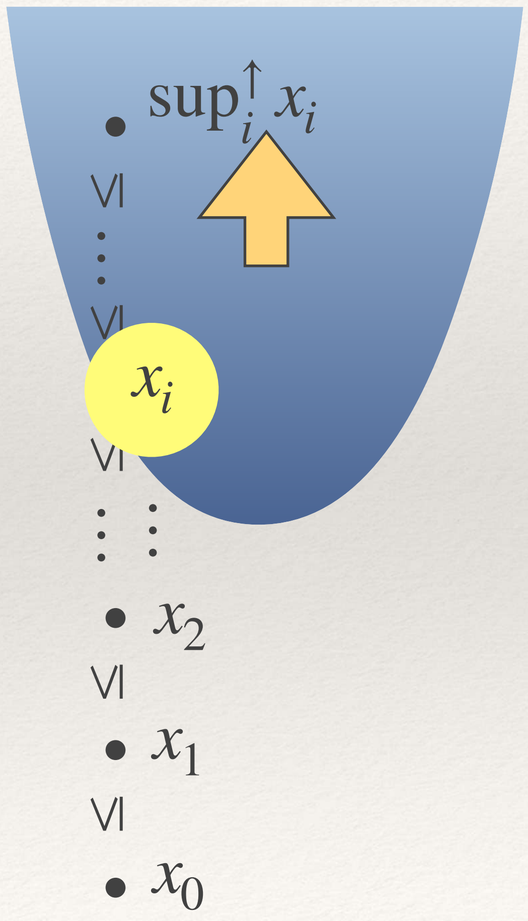
The Scott topology

- A subset U of a dcpo X is **Scott-open** iff:
  - U is **upwards closed** ($x \in U, x \leq y \Rightarrow y \in U$)
  - if $\sup_i^{\uparrow} x_i \in U$ then some $x_i \in U$ already
- Equivalently, iff its characteristic map $\chi_U \colon X \to \mathbb{S}$ is Scott-continuous, where $\mathbb{S} \mathrel{\hat{=}} \{0 < 1\}$
- The collection of Scott-open sets forms a topology: the **Scott topology**



The Scott topology

- A subset U of a dcpo X is **Scott-open** iff:
  - U is **upwards closed** ($x \in U, x \leq y \Rightarrow y \in U$)
  - if $\sup_i^{\uparrow} x_i \in U$ then some $x_i \in U$ already
- Equivalently, iff its characteristic map $\chi_U \colon X \to \mathbb{S}$ is Scott-continuous, where $\mathbb{S} \mathrel{\hat{=}} \{0 < 1\}$
- The collection of Scott-open sets forms a topology: the **Scott topology**
- The **Scott-continuous** maps are exactly the **continuous** maps for the Scott topologies



Continuous valuations

- First studied by [SahebDjahromi 80]: gives mass to **Scott-open** subsets
- Makes sense on **every topological space**, not just dcpos with the Scott topology
- Let $\mathcal{O}X$ denote the lattice of open subsets of a space X
- **Definition.** A **valuation** ν on X is a map $\nu\colon \mathcal{O}X \to \overline{\mathbb{R}}_+$ satisfying:

Continuous valuations

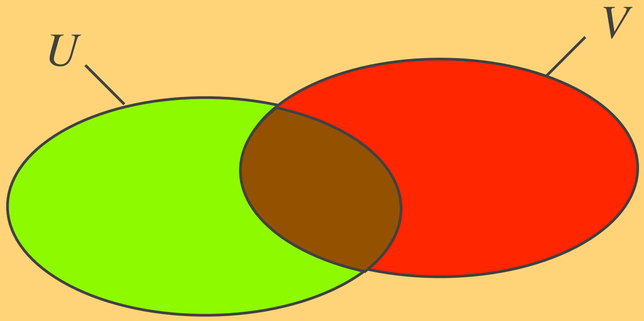
- First studied by [SahebDjahromi 80]: gives mass to **Scott-open** subsets
- Makes sense on **every topological space**, not just dcpos with the Scott topology
- Let $\mathcal{O}X$ denote the lattice of open subsets of a space X
- **Definition.** A **valuation** ν on X is a map $\nu\colon \mathcal{O}X \to \overline{\mathbb{R}}_+$ satisfying:
  - **strictness**: $\nu(\emptyset) = 0$

Continuous valuations

- First studied by [SahebDjahromi 80]: gives mass to **Scott-open** subsets
- Makes sense on **every topological space**, not just dcpos with the Scott topology
- Let $\mathcal{O}X$ denote the lattice of open subsets of a space X
- **Definition.** A **valuation** ν on X is a map $\nu\colon \mathcal{O}X \to \overline{\mathbb{R}}_+$ satisfying:
  - **strictness**: $\nu(\emptyset) = 0$
  - **monotonicity**: $\nu(U) \leq \nu(V)$ if $U \subseteq V$

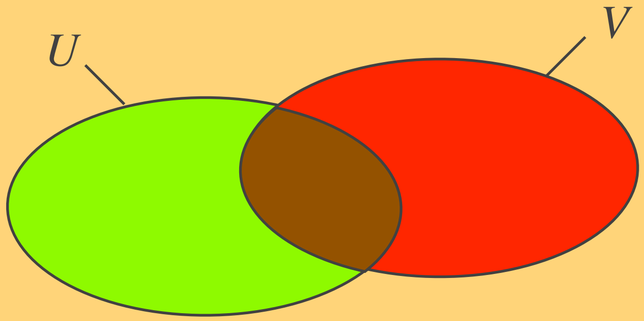
Continuous valuations

- First studied by [SahebDjahromi 80]: gives mass to **Scott-open** subsets
- Makes sense on **every topological space**, not just dcpos with the Scott topology
- Let $\mathcal{O}X$ denote the lattice of open subsets of a space X
- **Definition.** A **valuation** ν on X is a map $\nu\colon \mathcal{O}X \to \overline{\mathbb{R}}_+$ satisfying:
  - **strictness**: $\nu(\emptyset) = 0$
  - **monotonicity**: $\nu(U) \leq \nu(V)$ if $U \subseteq V$
  - **modularity**: $\nu(U \cup V) + \nu(U \cap V) = \nu(U) + \nu(V)$



Continuous valuations

- First studied by [SahebDjahromi 80]: gives mass to **Scott-open** subsets
- Makes sense on **every topological space**, not just dcpos with the Scott topology
- Let $\mathcal{O}X$ denote the lattice of open subsets of a space X
- **Definition.** A **valuation** ν on X is a map $\nu\colon \mathcal{O}X \to \overline{\mathbb{R}}_+$ satisfying:
  - **strictness**: $\nu(\emptyset) = 0$
  - **monotonicity**: $\nu(U) \leq \nu(V)$ if $U \subseteq V$
  - **modularity**: $\nu(U \cup V) + \nu(U \cap V) = \nu(U) + \nu(V)$



- A **continuous valuation** is not just monotonic, but Scott-continuous: $\nu(\sup_i^{\uparrow} U_i) = \sup_i^{\uparrow} \nu(U_i)$.

Simple valuations

- **Definition.** The **Dirac valuation** δ_x:

$$\delta_x(U) \triangleq \begin{cases} 1 & \text{if } x \in U \\ 0 & \text{otherwise} \end{cases}$$

  is a continuous valuation.

- If you draw at random with respect to δ_x, you will get x **all the time**.

Simple valuations

- **Definition.** The **Dirac valuation** δ_x:
$$\delta_x(U) \triangleq \begin{cases} 1 & \text{if } x \in U \\ 0 & \text{otherwise} \end{cases}$$
is a continuous valuation.

- If you draw at random with respect to δ_x, you will get x **all the time**.

- **Definition.** A **simple valuation** is $\sum_{i=1}^{n} a_i \delta_{x_i}$, where $a_i \in \mathbb{R}_+$

Simple valuations

- **Definition.** The **Dirac valuation** δ_x:
$$\delta_x(U) \triangleq \begin{cases} 1 & \text{if } x \in U \\ 0 & \text{otherwise} \end{cases}$$
is a continuous valuation.

- If you draw at random with respect to δ_x, you will get x **all the time**.

- **Definition.** A **simple valuation** is $\sum_{i=1}^{n} a_i \delta_{x_i}$, where $a_i \in \mathbb{R}_+$

- … draws each x_i with probability a_i (assuming x_i pairwise distinct)

Simple valuations

- **Definition.** The **Dirac valuation** δ_x:
$$\delta_x(U) \triangleq \begin{cases} 1 & \text{if } x \in U \\ 0 & \text{otherwise} \end{cases}$$
is a continuous valuation.
- If you draw at random with respect to δ_x, you will get x **all the time**.
- **Definition.** A **simple valuation** is $\sum_{i=1}^{n} a_i \delta_{x_i}$, where $a_i \in \mathbb{R}_+$
- … draws each x_i with probability a_i (assuming x_i pairwise distinct)
- always Scott-continuous

Other continuous valuations

- All (bounded) valuations are simple on a finite space
- Our dcpos $[\![\tau]\!]$ will **not** be finite

Other continuous valuations

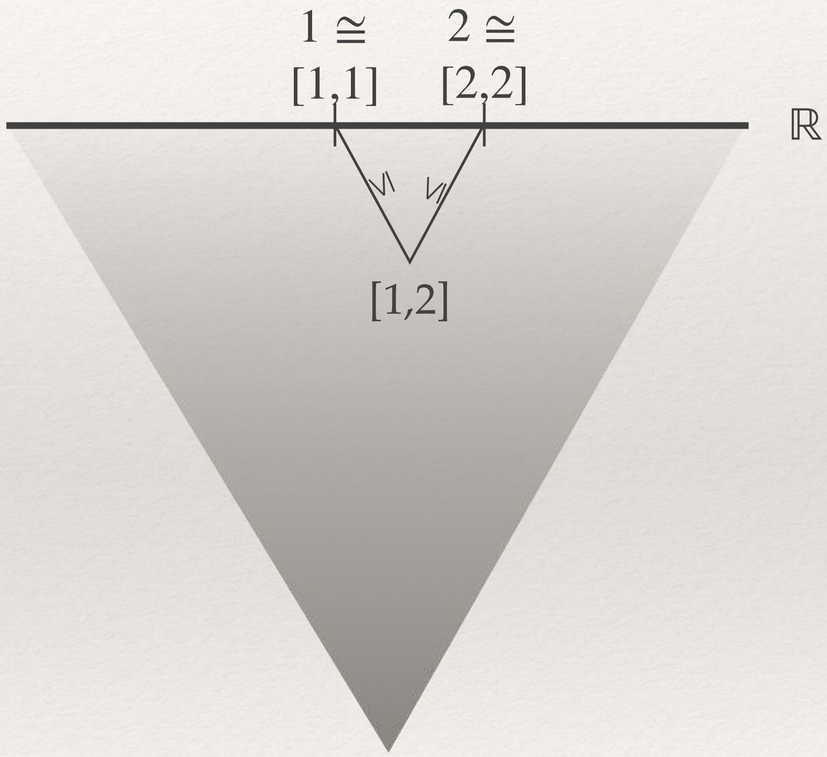
- All (bounded) valuations are simple on a finite space
- Our dcpos $[\![\tau]\!]$ will **not** be finite
- In general, not all continuous valuations are simple.
  E.g., **Lebesgue measure** λ on $\mathbb{R}$ (restricted to its open subsets)
  is a continuous valuation that is not simple

Other continuous valuations

- All (bounded) valuations are simple on a finite space
- Our dcpos $[\![\tau]\!]$ will **not** be finite
- In general, not all continuous valuations are simple.
  E.g., **Lebesgue measure** λ on $\mathbb{R}$ (restricted to its open subsets)
  is a continuous valuation that is not simple
- Here is an example on a **dcpo**…

The dcpo $\mathbb{IR}$

- Elements of $\mathbb{IR}$: intervals $[a, b]$ with $a \leq b \in \mathbb{R}$
  We equate $[a, a]$ with $a \in \mathbb{R}$, so $\mathbb{R} \subseteq \mathbb{IR}$
- Ordered by **reverse** inclusion:
  $[a, b] \leq [c, d]$ iff $[a, b] \supseteq [c, d]$

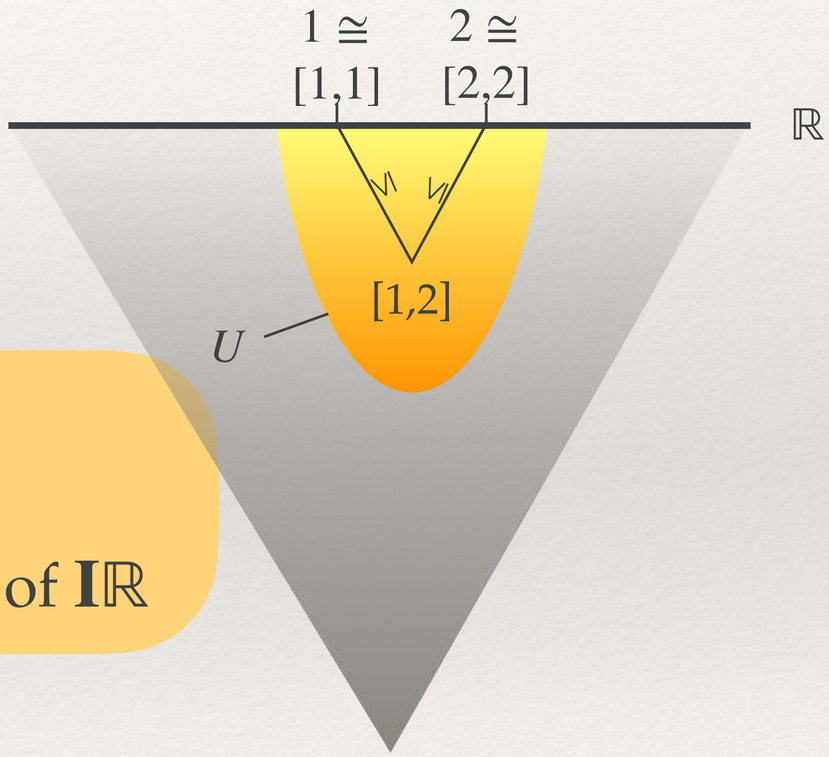


The dcpo $\mathbf{I}\mathbb{R}$

- Elements of $\mathbf{I}\mathbb{R}$: intervals $[a, b]$ with $a \leq b \in \mathbb{R}$
  We equate $[a, a]$ with $a \in \mathbb{R}$, so $\mathbb{R} \subseteq \mathbf{I}\mathbb{R}$

- Ordered by **reverse** inclusion:
  $[a, b] \leq [c, d]$ iff $[a, b] \supseteq [c, d]$

- **Fact.** The open subsets of $\mathbb{R}$ (with its usual metric topology)
  are exactly the sets $U \cap \mathbb{R}$,
  where U ranges over the **Scott-open** subsets of $\mathbf{I}\mathbb{R}$

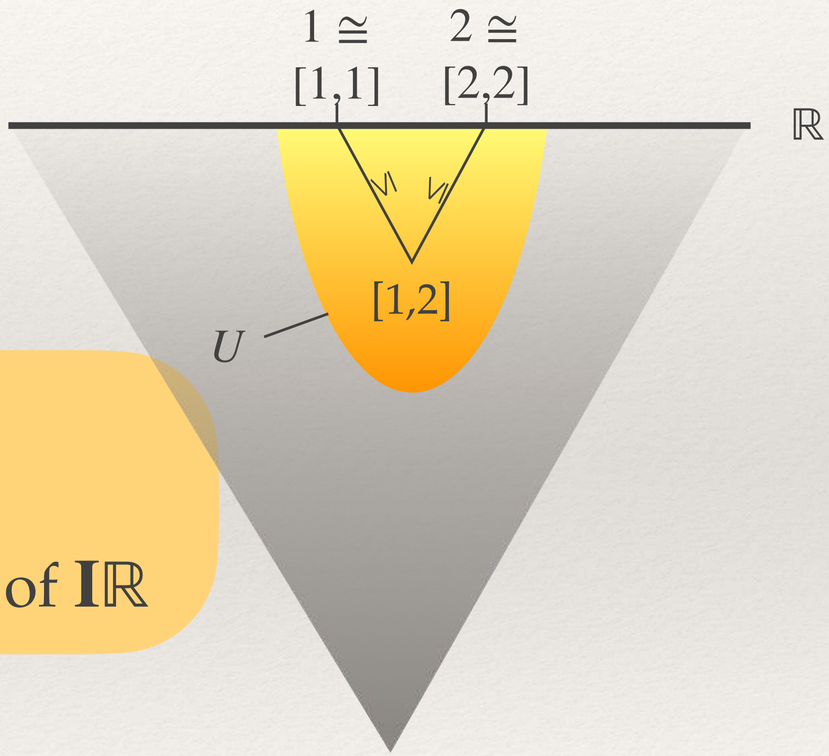


The dcpo $\mathbf{I}\mathbb{R}$

- Elements of $\mathbf{I}\mathbb{R}$: intervals $[a, b]$ with $a \leq b \in \mathbb{R}$
  We equate $[a, a]$ with $a \in \mathbb{R}$, so $\mathbb{R} \subseteq \mathbf{I}\mathbb{R}$

- Ordered by **reverse** inclusion:
  $[a, b] \leq [c, d]$ iff $[a, b] \supseteq [c, d]$

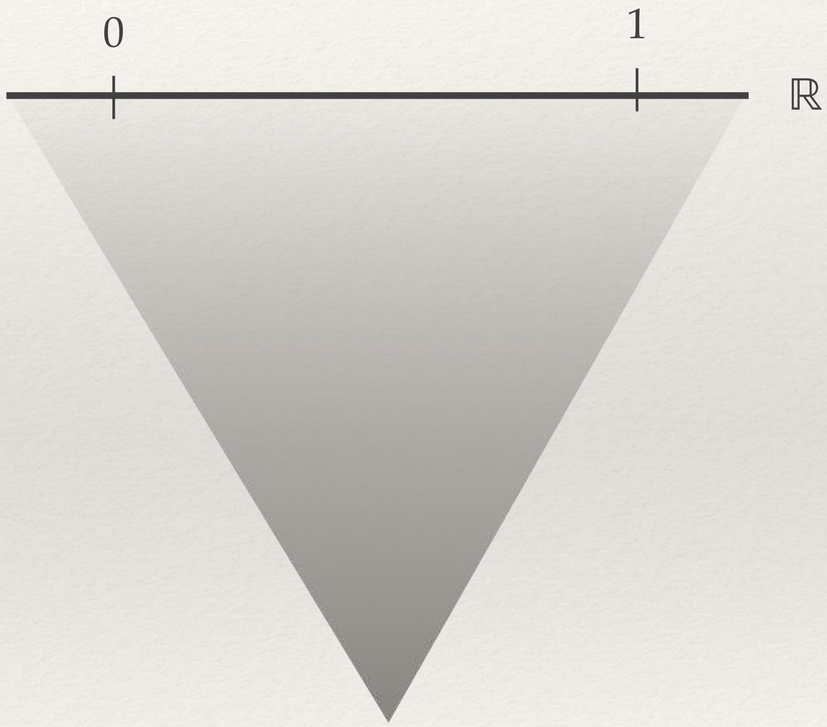
- **Fact.** The open subsets of $\mathbb{R}$ (with its usual metric topology)
  are exactly the sets $U \cap \mathbb{R}$,
  where U ranges over the **Scott-open** subsets of $\mathbf{I}\mathbb{R}$



- Now let $\bar{\lambda}(U) \triangleq \lambda(U \cap \mathbb{R})$, for every $U \in \mathcal{O}(\mathbf{I}\mathbb{R})$:
  $\bar{\lambda}$ is a continuous valuation on the dcpo $\mathbf{I}\mathbb{R}$
  that is **not a simple valuation** (Note: $\bar{\lambda}$ is supported on $\mathbb{R}$)

Beyond simple valuations

- Maybe we only need **simple** valuations in computer science?
- No: simple valuations do not form a **dcpo**

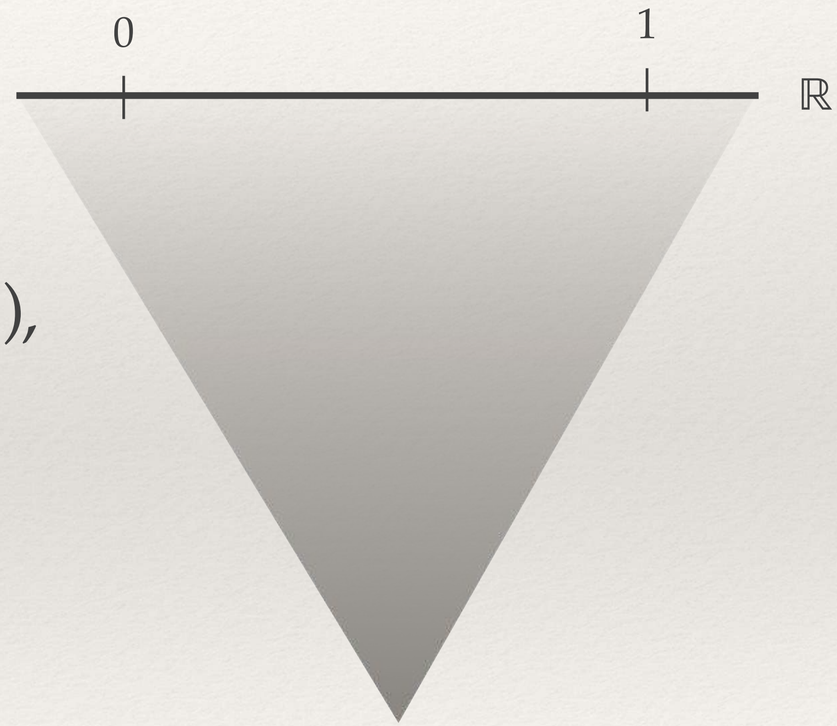


Beyond simple valuations

- Maybe we only need **simple** valuations in computer science?
- No: simple valuations do not form a **dcpo**
- E.g., $\bar{\lambda}_{|[0,1]} = \sup^{\uparrow}_{n \in \mathbb{N}} \bar{\lambda}_n$ (uniform measure on [0,1]),

  where $\bar{\lambda}_n \mathrel{\hat{=}} \sum_{i=1}^{2^n} \frac{1}{2^n} \delta_{[\frac{i-1}{2^n}, \frac{i}{2^n}]}$

  (there is a similar formula for $\bar{\lambda}$ itself, but I wish to show you a **probability** valuation)

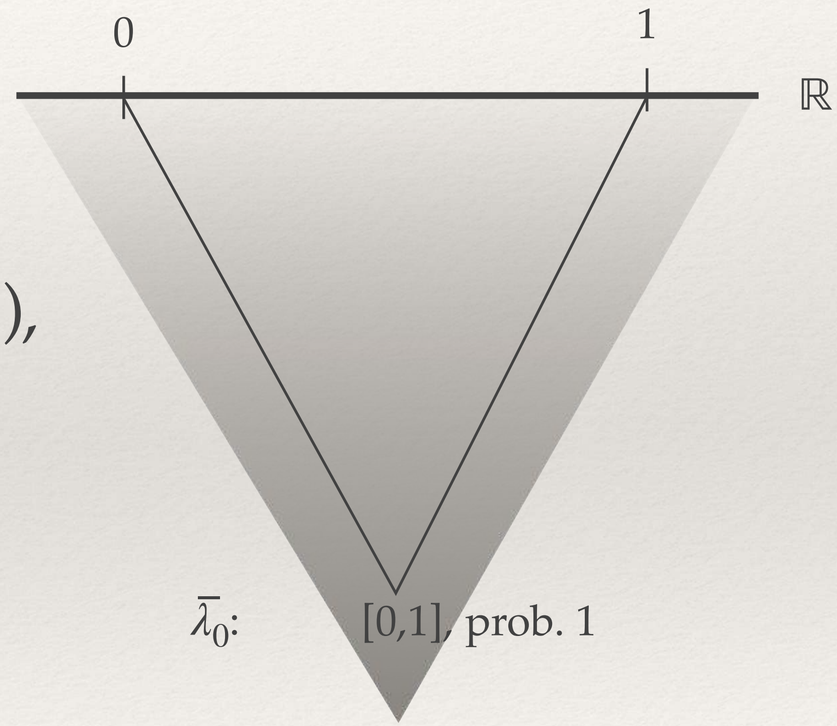


Beyond simple valuations

- Maybe we only need **simple** valuations in computer science?
- No: simple valuations do not form a **dcpo**
- E.g., $\bar{\lambda}_{|[0,1]} = \sup^{\uparrow}_{n \in \mathbb{N}} \bar{\lambda}_n$ (uniform measure on [0,1]),

  where $\bar{\lambda}_n \mathrel{\hat{=}} \sum_{i=1}^{2^n} \frac{1}{2^n} \delta_{[\frac{i-1}{2^n}, \frac{i}{2^n}]}$

  (there is a similar formula for $\bar{\lambda}$ itself, but I wish to show you a **probability** valuation)

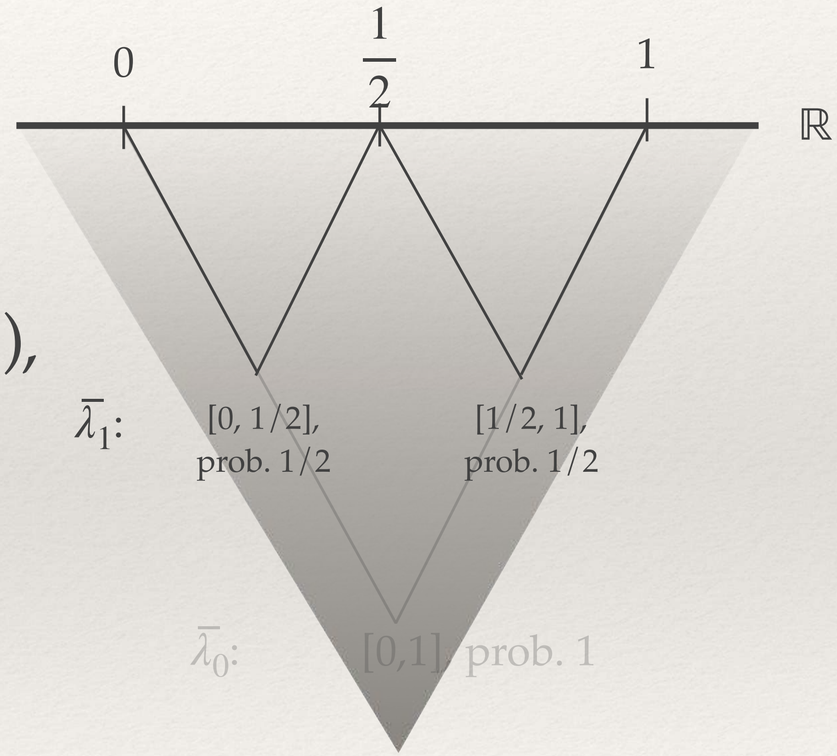


Beyond simple valuations

- Maybe we only need **simple** valuations in computer science?
- No: simple valuations do not form a **dcpo**
- E.g., $\bar{\lambda}_{|[0,1]} = \sup^{\uparrow}_{n \in \mathbb{N}} \bar{\lambda}_n$ (uniform measure on [0,1]),

  where $\bar{\lambda}_n \triangleq \sum_{i=1}^{2^n} \frac{1}{2^n} \delta_{[\frac{i-1}{2^n}, \frac{i}{2^n}]}$

  (there is a similar formula for $\bar{\lambda}$ itself, but I wish to show you a **probability** valuation)

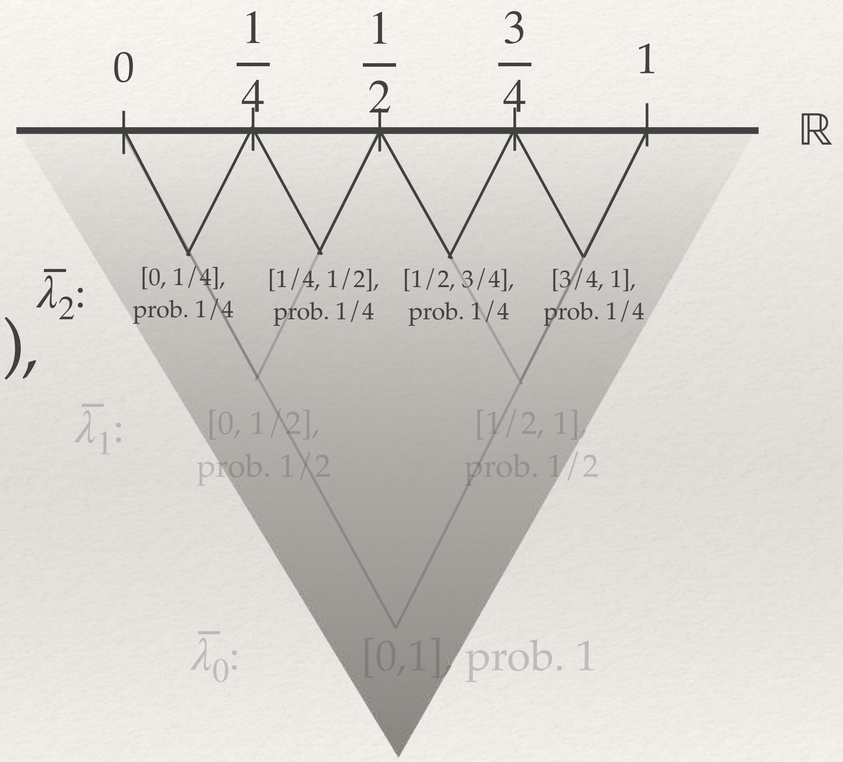


Beyond simple valuations

- Maybe we only need **simple** valuations in computer science?
- No: simple valuations do not form a **dcpo**
- E.g., $\bar{\lambda}_{|[0,1]} = \sup^{\uparrow}_{n \in \mathbb{N}} \bar{\lambda}_n$ (uniform measure on [0,1]),

  where $\bar{\lambda}_n \triangleq \sum_{i=1}^{2^n} \frac{1}{2^n} \delta_{[\frac{i-1}{2^n}, \frac{i}{2^n}]}$

  (there is a similar formula for $\bar{\lambda}$ itself, but I wish to show you a **probability** valuation)



Continuous valuations and measures

Continuous valuation $\nu : \mathcal{O}X \to \overline{\mathbb{R}}_+$:
- **strict**: $\nu(\emptyset) = 0$
- **Scott-continuous,**
- **modular**: $\nu(U \cup V) + \nu(U \cap V) = \nu(U) + \nu(V)$

- **Theorem [Adamski 77].** Given any **Borel measure** on a hereditarily Lindelöf space, (in particular, a second-countable space) its restriction to open sets is a **continuous valuation**.
  (i.e., every Borel measure on such a space is *τ-smooth.*)

- **Theorem [de Brecht, JGL, Jia, Lyu 2019].** Every **continuous valuation** on an LCS-complete space (a G_δ subspace of a locally compact sober space) extends to a **Borel measure**.

- A sweet spot: de Brecht's **quasi-Polish** spaces (=2nd countable LCS-complete)
  Those include all **ω-continuous dcpos** + all **Polish** spaces

Continuous valuations and measures

- **Theorem [Adamski 77].** Given any
  on a hereditarily Lindelöf
  its restriction to open sets

$\overline{\mathbb{R}}_+$:

$) = \nu(U) + \nu(V)$

Just remember that
continuous valuations~measures,
in most cases (not all, though)

- **Theorem [de Brecht, JGL, Jia, Lyu 2019].** Every **continuous valuation**
  on an LCS-complete space (a G_δ subspace of a locally compact sober space)
  extends to a **Borel measure**.

- A sweet spot: de Brecht's **quasi-Polish** spaces (=2nd countable LCS-complete)
  Those include all ω**-continuous dcpos** + all **Polish** spaces

Integration

- For every **lower semicontinuous** map $f\colon X \to \overline{\mathbb{R}}_+$ and every continuous valuation μ on X,

  there is an **integral** $\displaystyle\int_x f(x)d\mu$ (or $\displaystyle\int f\, d\mu$ for short)

  [Jones 90, Tix 95]

I.e., continuous from X to ($\overline{\mathbb{R}}_+$, Scott topology)

Integration

- For every **lower semicontinuous** map $f\colon X \to \overline{\mathbb{R}}_+$ and every continuous valuation μ on X,

  there is an **integral** $\int_x f(x)d\mu$ (or $\int f\, d\mu$ for short)

  [Jones 90, Tix 95]

Integration

I.e., continuous from X to ($\overline{\mathbb{R}}_+$, Scott topology)

If X is a dcpo, this just means **Scott-continuous**, but the definition works for any topological space X

- For every **lower semicontinuous** map $f\colon X \to \overline{\mathbb{R}}_+$ and every continuous valuation μ on X,

  there is an **integral** $\int_x f(x)d\mu$ (or $\int f\, d\mu$ for short)

  [Jones 90, Tix 95]

The probabilistic powerdomain

- The **probabilistic powerdomain** $\mathbf{V}_{\leq 1}X$ of a space X is the dcpo of all subprobability (continuous) valuations ν on X

The probabilistic powerdomain

- The **probabilistic powerdomain** $\mathbf{V}_{\leq 1}X$ of a space X is the dcpo of all subprobability (continuous) valuations ν on X

i.e., $\nu(X) \leq 1$

The probabilistic powerdomain

- The **probabilistic powerdomain** $\mathbf{V}_{\leq 1}X$ of a space X is the dcpo of all subprobability (continuous) valuations ν on X

  i.e., $\nu(X) \leq 1$

- ordered by $\mu \leq \nu$ iff for every $U \in \mathcal{O}X$, $\mu(U) \leq \nu(U)$

The probabilistic powerdomain

- The **probabilistic powerdomain** $\mathbf{V}_{\leq 1}X$ of a space X is the dcpo of all subprobability (continuous) valuations ν on X

  i.e., $\nu(X) \leq 1$

- ordered by $\mu \leq \nu$ iff for every $U \in \mathcal{O}X$, $\mu(U) \leq \nu(U)$

- We can now define (as promised):
  — $[\![\mathbf{T}\tau]\!] \triangleq \mathbf{V}_{\leq 1}([\![\tau]\!])$ first-class subprobability distributions
  — $[\![M \oplus N]\!]\rho \triangleq \frac{1}{2}[\![M]\!]\rho + \frac{1}{2}[\![N]\!]\rho$
  — $[\![\mathbf{ret}\ M]\!]\rho \triangleq \delta_{[\![M]\!]\rho}$
  — $[\![\mathbf{do}\ x_\sigma = M; N]\!]\rho \triangleq (U \in \mathcal{O}([\![\sigma]\!]) \mapsto \int_x [\![N]\!]\rho(U)\, d[\![M]\!]\rho)$

The probabilistic powerdomain

- The **probabilistic powerdomain** $\mathbf{V}_{\leq 1}X$ of a space X is the dcpo of all subprobability (continuous) valuations ν on X

  i.e., $\nu(X) \leq 1$

- ordered by $\mu \leq \nu$ iff for every $U \in \mathcal{O}X$, $\mu(U) \leq \nu(U)$

- We can now define (as promised):
  — $[\![\mathbf{T}\tau]\!] \triangleq \mathbf{V}_{\leq 1}([\![\tau]\!])$ first-class subprobability distributions
  — $[\![M \oplus N]\!]\rho \triangleq \frac{1}{2}[\![M]\!]\rho + \frac{1}{2}[\![N]\!]\rho$
  — $[\![\mathbf{ret}\ M]\!]\rho \triangleq \delta_{[\![M]\!]\rho}$
  — $[\![\mathbf{do}\ x_\sigma = M; N]\!]\rho \triangleq (U \in \mathcal{O}([\![\sigma]\!]) \mapsto \int_x [\![N]\!]\rho(U)\, d[\![M]\!]\rho)$

  Really defines a **strong monad** on **Dcpo**

Soundness, adequacy

- Operationally, let $\Pr[C, M \downarrow n] \mathrel{\hat{=}}$ the (sub)probability that $C, M \to^* \mathbf{ret}\ \_, n$
  — sum over all executions $C, M \to^* \mathbf{ret}\ \_, n$
  — of their weights, defined as the product of their probability labels
- **Prop (soundness).** $\Pr[C, M \downarrow n] \leq [\![C[M]]\!]\rho(\{n\})$ (at type **nat**)

Soundness, adequacy

- Operationally, let $\Pr[C, M \downarrow n] \mathrel{\hat{=}}$ the (sub)probability that $C, M \to^* \mathbf{ret}\ \_, n$
  — sum over all executions $C, M \to^* \mathbf{ret}\ \_, n$
  — of their weights, defined as the product of their probability labels
- **Prop (soundness).** $\Pr[C, M \downarrow n] \leq [\![C[M]]\!]\rho(\{n\})$ (at type **nat**)
- **Theorem (adequacy).** $\Pr[C, M \downarrow n] = [\![C[M]]\!]\rho(\{n\})$ (at type **nat**)

Soundness, adequacy

- Operationally, let $\Pr[C, M \downarrow n] \mathrel{\hat{=}}$ the (sub)probability that $C, M \to^* \mathbf{ret}\ \_, n$
  — sum over all executions $C, M \to^* \mathbf{ret}\ \_, n$
  — of their weights, defined as the product of their probability labels
- **Prop (soundness).** $\Pr[C, M \downarrow n] \leq [\![C[M]]\!]\rho(\{n\})$ (at type **nat**)
- **Theorem (adequacy).** $\Pr[C, M \downarrow n] = [\![C[M]]\!]\rho(\{n\})$ (at type **nat**)
- Soundness is a bit harder than before: requires $[\![C]\!]$ **linear**
- Adequacy: by a suitable form of logical relations

No full abstraction (1/2)

- Fact. If $[\![M]\!]=[\![N]\!]$ then $M \cong N$. (as in PCF, consequence of adequacy)
- Just like PCF, probabilistic PCF is **not** fully abstract

No full abstraction (1/2)

- Fact. If $[\![M]\!]=[\![N]\!]$ then $M \cong N$. (as in PCF, consequence of adequacy)
- Just like PCF, probabilistic PCF is **not** fully abstract
- **Parallel or** is a problem, as before

No full abstraction (1/2)

- Fact. If $[\![M]\!]=[\![N]\!]$ then $M \cong N$. (as in PCF, consequence of adequacy)
- Just like PCF, probabilistic PCF is **not** fully abstract
- **Parallel or** is a problem, as before
- There is a second obstacle to full abstraction here:
  the **statistical testers** $[V > r] \in [\![\mathbf{T}\ \tau \to \mathbf{T}\ \mathbf{unit}]\!]$
  mapping $\nu \in [\![\mathbf{T}\ \tau]\!]$ to δ_* if $\nu(V) > r$
  to 0 otherwise
  are **not definable** in probabilistic PCF
  … leading to another cause of failure of full abstraction

No full abstraction (2/2)

- Fact. If $[\![M]\!]=[\![N]\!]$ then $M \cong N$.
- Even if we added primitive parallel or and statistical testers, we do not have the **technology** to prove the converse.
- The problem is that we need all dcpos $[\![\tau]\!]$ to be at least

  continuous dcpos (a.k.a. **domains**)

  to hope to obtain a proof

Continuous dcpos, a.k.a. domains

Continuous dcpos, a.k.a. domains

- Motto: the continuous dcpos are the **nice** dcpos,
  where (almost) every property you wish for is true

Continuous dcpos, a.k.a. domains

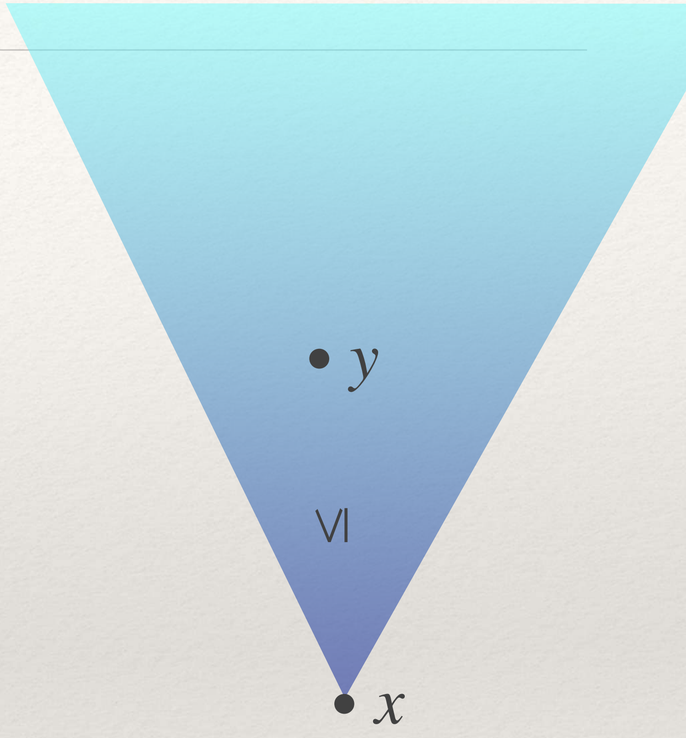
- Motto: the continuous dcpos are the **nice** dcpos,
  where (almost) every property you wish for is true

- Let $x \ll y$ (x **way-**below y) iff
  $y \leq \sup_i^{\uparrow} z_i$ implies $\exists i, y \leq z_i$

Continuous dcpos, a.k.a. domains

- Motto: the continuous dcpos are the **nice** dcpos,
  where (almost) every property you wish for is true

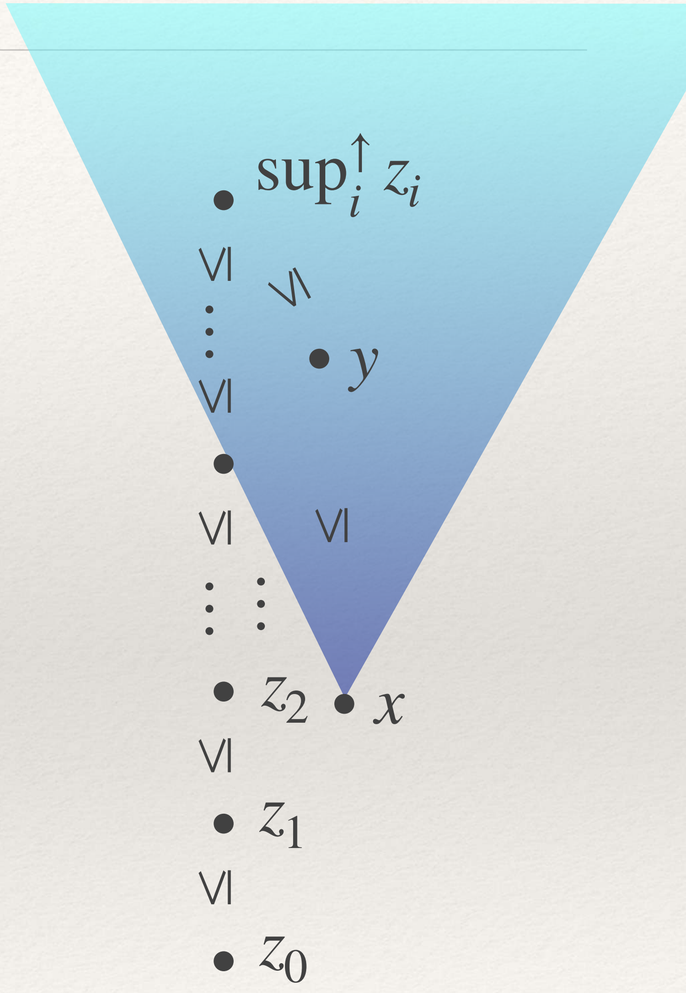
- Let $x \ll y$ (x **way-**below y) iff
  $y \leq \sup_i^{\uparrow} z_i$ implies $\exists i, y \leq z_i$



Continuous dcpos, a.k.a. domains

- Motto: the continuous dcpos are the **nice** dcpos,
  where (almost) every property you wish for is true

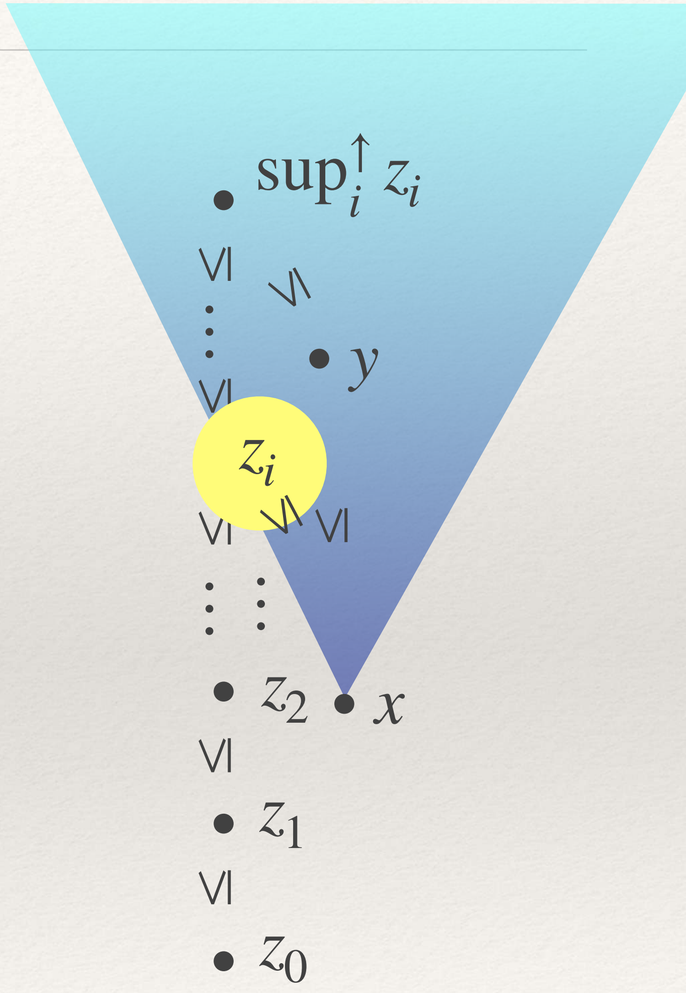
- Let $x \ll y$ (x **way-**below y) iff
  $y \leq \sup_i^{\uparrow} z_i$ implies $\exists i, y \leq z_i$



Continuous dcpos, a.k.a. domains

- Motto: the continuous dcpos are the **nice** dcpos,
  where (almost) every property you wish for is true

- Let $x \ll y$ (x **way-**below y) iff
  $y \leq \sup_i^{\uparrow} z_i$ implies $\exists i, y \leq z_i$

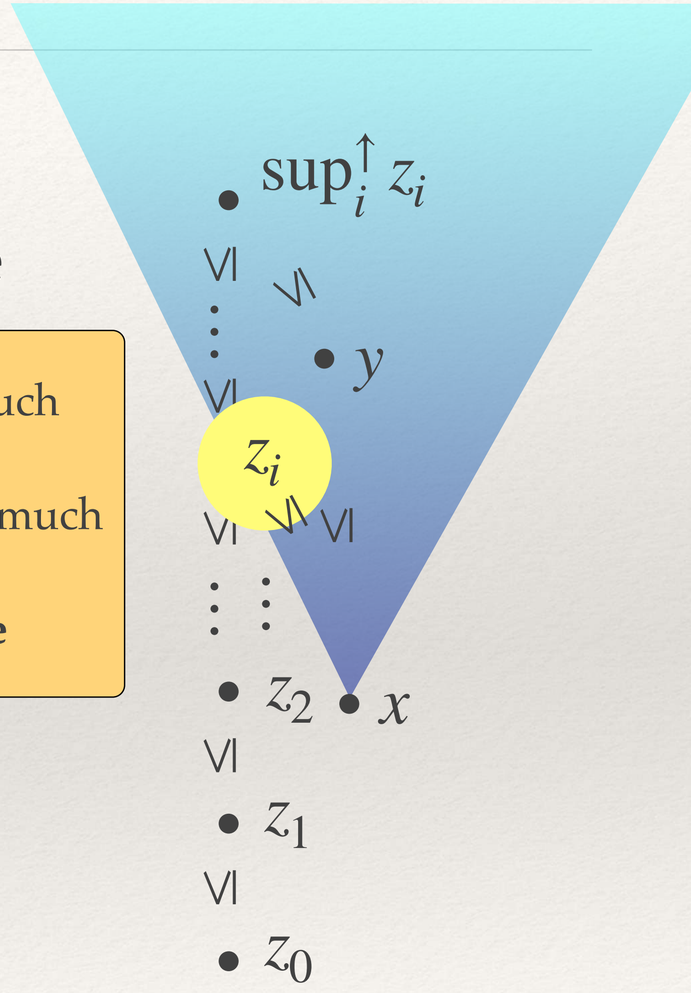


Continuous dcpos, a.k.a. domains

- Motto: the continuous dcpos are the **nice** dcpos,
  where (almost) every property you wish for is true

- Let $x \ll y$ (x **way-**below y) iff
  $y \leq \sup_i^{\uparrow} z_i$ implies $\exists i, y \leq z_i$

in order to get at least as much information as y,
you must get at least as much information as x
at some **finite stage**



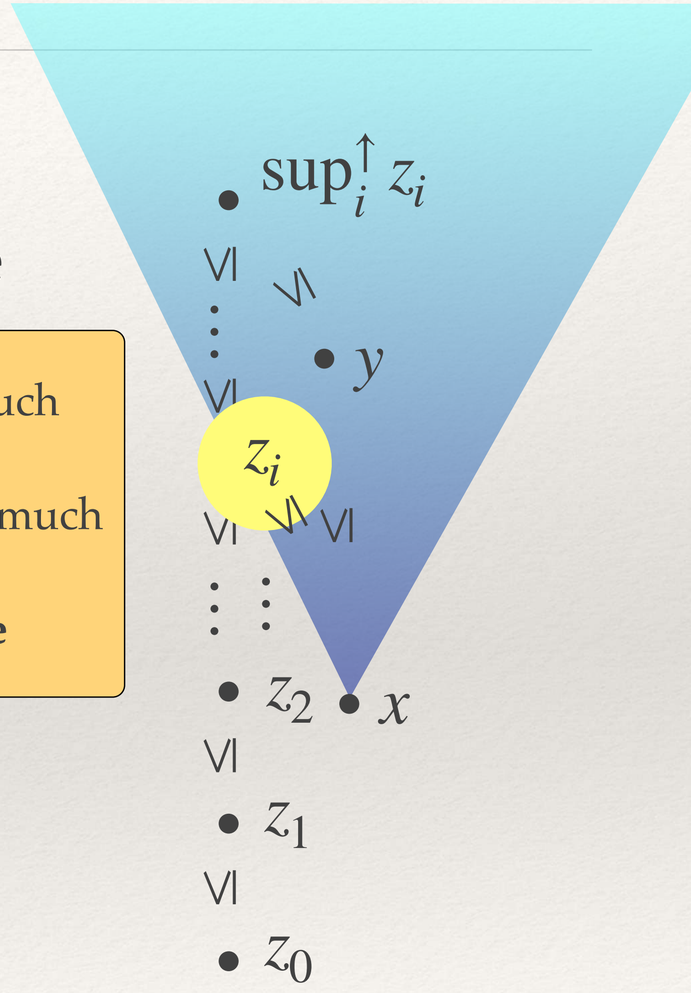
Continuous dcpos, a.k.a. domains

- Motto: the continuous dcpos are the **nice** dcpos,
  where (almost) every property you wish for is true

- Let $x \ll y$ (x **way-**below y) iff
  $y \leq \sup_i^{\uparrow} z_i$ implies $\exists i, y \leq z_i$

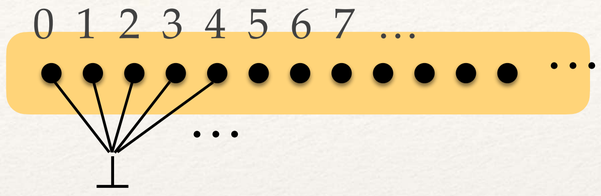
in order to get at least as much information as y,
you must get at least as much information as x
at some **finite stage**

- **Definition.** A dcpo is **continuous** iff
  every point x is the supremum
  of some directed family of points **way-below** x.



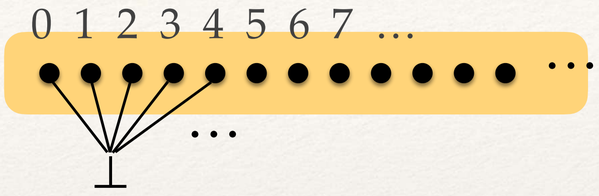
Examples, counterexamples

- 😀 $\mathbb{N}_\perp$, the semantics of **nat** in PCF is a continuous dcpo



Examples, counterexamples

- 😀 $\mathbb{N}_\perp$, the semantics of **nat** in PCF is a continuous dcpo

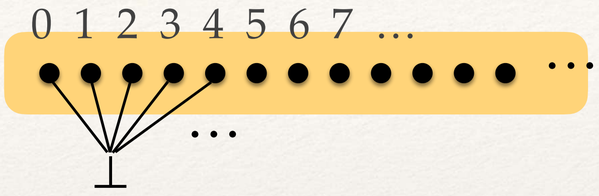


- **Theorem [Jones 90].** For every continuous dcpo X, 😃 $\mathbf{V}_{\leq 1}X$ is a continuous dcpo

  Hence $[\![\tau]\!]$ continuous dcpo $\Rightarrow$ $[\![\mathbf{T}\tau]\!]$ continuous dcpo in probabilistic PCF

Examples, counterexamples

- 😀 $\mathbb{N}_\perp$, the semantics of **nat** in PCF is a continuous dcpo



- **Theorem [Jones 90].** For every continuous dcpo X,
  😃 $\mathbf{V}_{\leq 1}X$ is a continuous dcpo

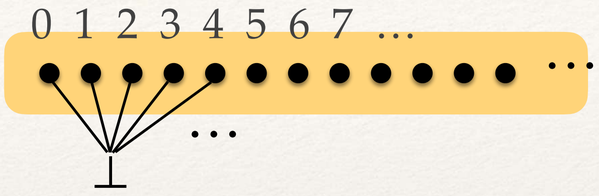
  Hence $[\![\tau]\!]$ continuous dcpo $\Rightarrow$ $[\![\mathbf{T}\tau]\!]$ continuous dcpo in probabilistic PCF

- 😢 **Fact.** In general, $[X \to Y]$ is not continuous, even if X and Y are.

  (Consider $X \mathrel{\hat{=}} Y \mathrel{\hat{=}} \mathbb{Z}^-$.)

Examples, counterexamples

- 😀 $\mathbb{N}_\perp$, the semantics of **nat** in PCF is a continuous dcpo



- **Theorem [Jones 90].** For every continuous dcpo X, 😃 $\mathbf{V}_{\leq 1}X$ is a continuous dcpo

  Hence $[\![\tau]\!]$ continuous dcpo $\Rightarrow$ $[\![\mathbf{T}\tau]\!]$ continuous dcpo in probabilistic PCF

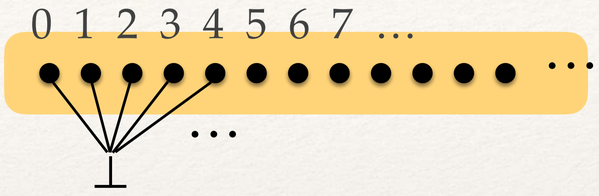
- 😢 **Fact.** In general, $[X \to Y]$ is not continuous, even if X and Y are.

  (Consider $X \mathrel{\hat{=}} Y \mathrel{\hat{=}} \mathbb{Z}^-$.)

Cont is not Cartesian-closed

Examples, counterexamples

- 😀 $\mathbb{N}_\perp$, the semantics of **nat** in PCF is a continuous dcpo



- **Theorem [Jones 90].** For every continuous dcpo X, 😃 $\mathbf{V}_{\leq 1}X$ is a continuous dcpo

  Hence $[\![\tau]\!]$ continuous dcpo $\Rightarrow$ $[\![\mathbf{T}\tau]\!]$ continuous dcpo in probabilistic PCF

- 😢 **Fact.** In general, $[X \to Y]$ is not continuous, even if X and Y are.

  (Consider $X \mathrel{\hat{=}} Y \mathrel{\hat{=}} \mathbb{Z}^-$.)

Cont is not Cartesian-closed

So $[\![\tau]\!]$ may fail to be in **Cont** for types τ of high enough order

Cartesian-closed categories of domains 1: bc-domains

- A **bc-domain** is a continuous dcpo that:
  — is pointed (has a $\perp$)
  — any two elements x, y with an upper bound
  have a **least** upper bound $x \vee y$

Cartesian-closed categories of domains 1: bc-domains

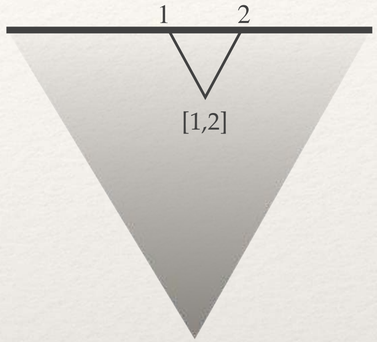
- A **bc-domain** is a continuous dcpo that:
  — is pointed (has a $\perp$)
  — any two elements x, y with an upper bound
  have a **least** upper bound $x \vee y$

- $\mathbb{N}_\perp$ is a bc-domain

Cartesian-closed categories of domains 1: bc-domains

- A **bc-domain** is a continuous dcpo that:
  — is pointed (has a $\perp$)
  — any two elements x, y with an upper bound
  have a **least** upper bound $x \vee y$

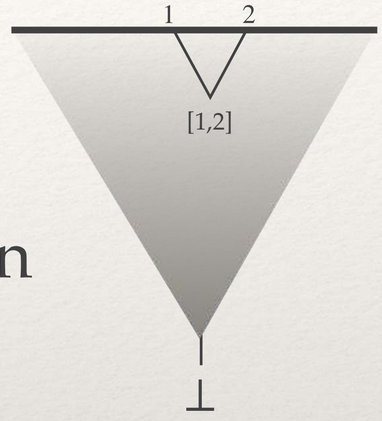
- $\mathbb{N}_\perp$ is a bc-domain
- not $\mathbb{IR}$ (no $\perp$)



Cartesian-closed categories of domains 1: bc-domains

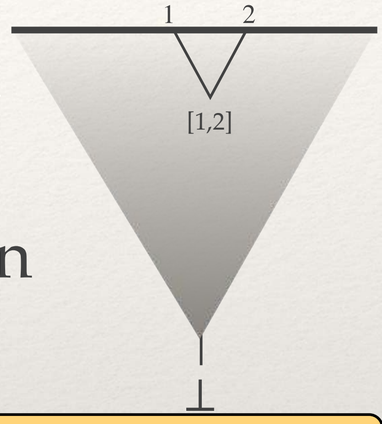
- A **bc-domain** is a continuous dcpo that:
  — is pointed (has a $\perp$)
  — any two elements x, y with an upper bound have a **least** upper bound $x \vee y$

- $\mathbb{N}_\perp$ is a bc-domain
- not $\mathbb{IR}$ (no $\perp$)
- $\mathbb{IR}_\perp$ is a bc-domain



Cartesian-closed categories of domains 1: bc-domains

- A **bc-domain** is a continuous dcpo that:
  — is pointed (has a $\perp$)
  — any two elements x, y with an upper bound have a **least** upper bound $x \vee y$
- $\mathbb{N}_\perp$ is a bc-domain
- not $\mathbb{IR}$ (no $\perp$)
- $\mathbb{IR}_\perp$ is a bc-domain



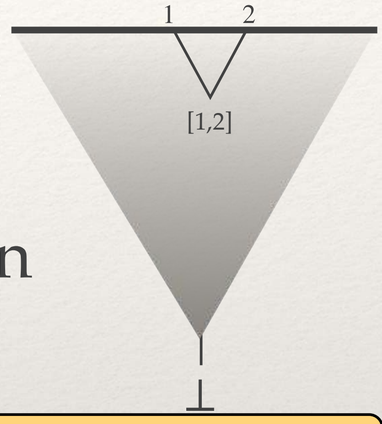
- 😀 **Fact.** For any two bc-domains X, Y, $[X \to Y]$ is a bc-domain.

BCDom is Cartesian-closed

Cartesian-closed categories of domains 1: bc-domains

- A **bc-domain** is a continuous dcpo that:
  — is pointed (has a $\bot$)
  — any two elements x, y with an upper bound have a **least** upper bound $x \vee y$

- $\mathbb{N}_\bot$ is a bc-domain
- not $\mathbb{IR}$ (no $\bot$)
- $\mathbb{IR}_\bot$ is a bc-domain



- 😀 **Fact.** For any two bc-domains X, Y, $[X \to Y]$ is a bc-domain.

BCDom is Cartesian-closed

- 😕 **Fact [Jones 90].** $\mathbf{V}_{\leq 1}X$ is not a bc-domain in general, even if X is.

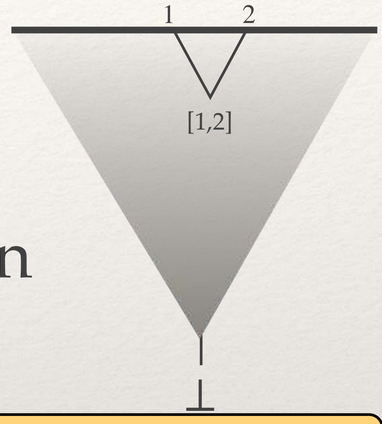
(In $X \triangleq$

, $\frac{1}{2}\delta_a$ and $\frac{1}{2}\delta_b$ have **uncountably many** minimal upper bounds.)

Cartesian-closed categories of domains 1: bc-domains

- A **bc-domain** is a continuous dcpo that:
  — is pointed (has a $\bot$)
  — any two elements x, y with an upper bound have a **least** upper bound $x \vee y$

- $\mathbb{N}_\bot$ is a bc-domain
- not $\mathbb{IR}$ (no $\bot$)
- $\mathbb{IR}_\bot$ is a bc-domain



- 😀 **Fact.** For any two bc-domains X, Y, $[X \to Y]$ is a bc-domain.

BCDom is Cartesian-closed

- 😕 **Fact [Jones 90].** $\mathbf{V}_{\leq 1}X$ is not a bc-domain in general, even if X is.

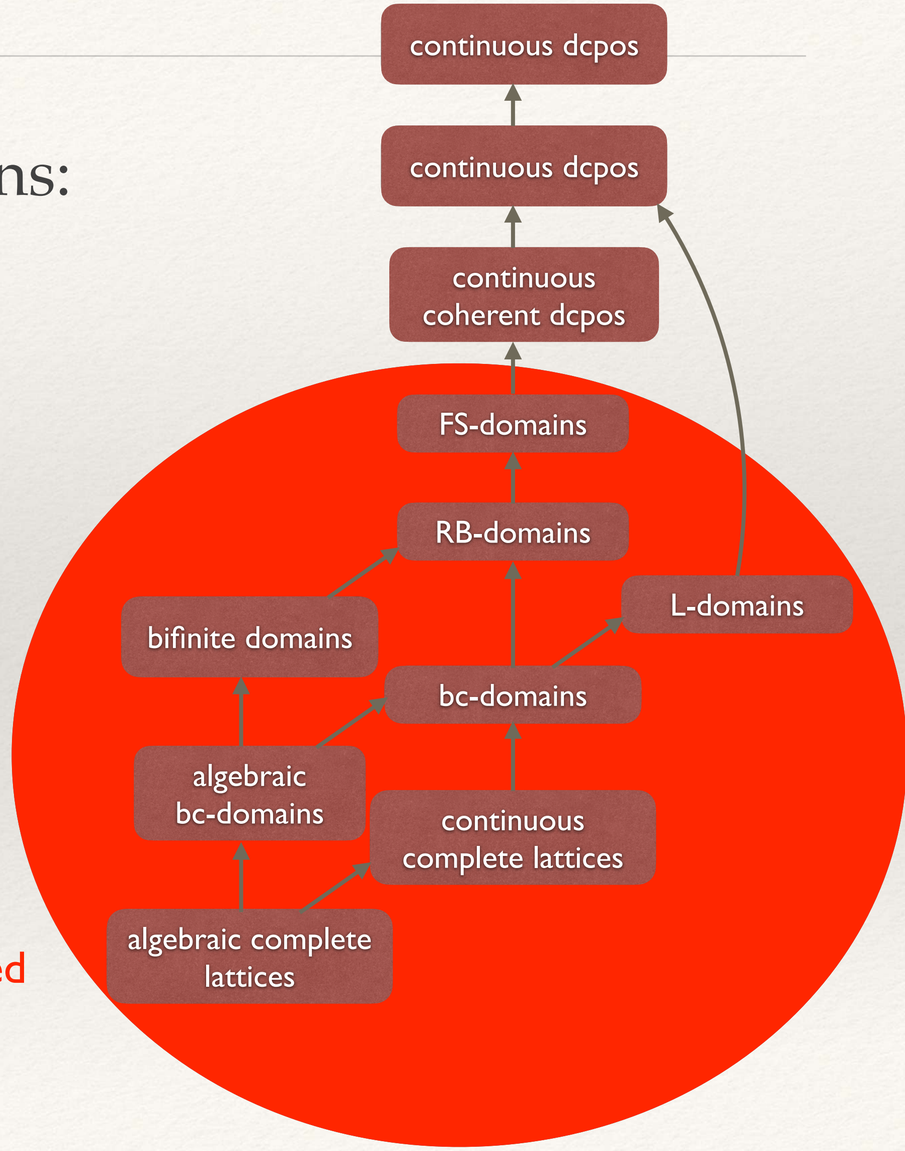
So $[\![\tau]\!]$ may fail to be in **BCDom** for complex enough types τ

(In $X \triangleq$ 

, $\frac{1}{2}\delta_a$ and $\frac{1}{2}\delta_b$ have **uncountably many** minimal upper bounds.)

The state of the art

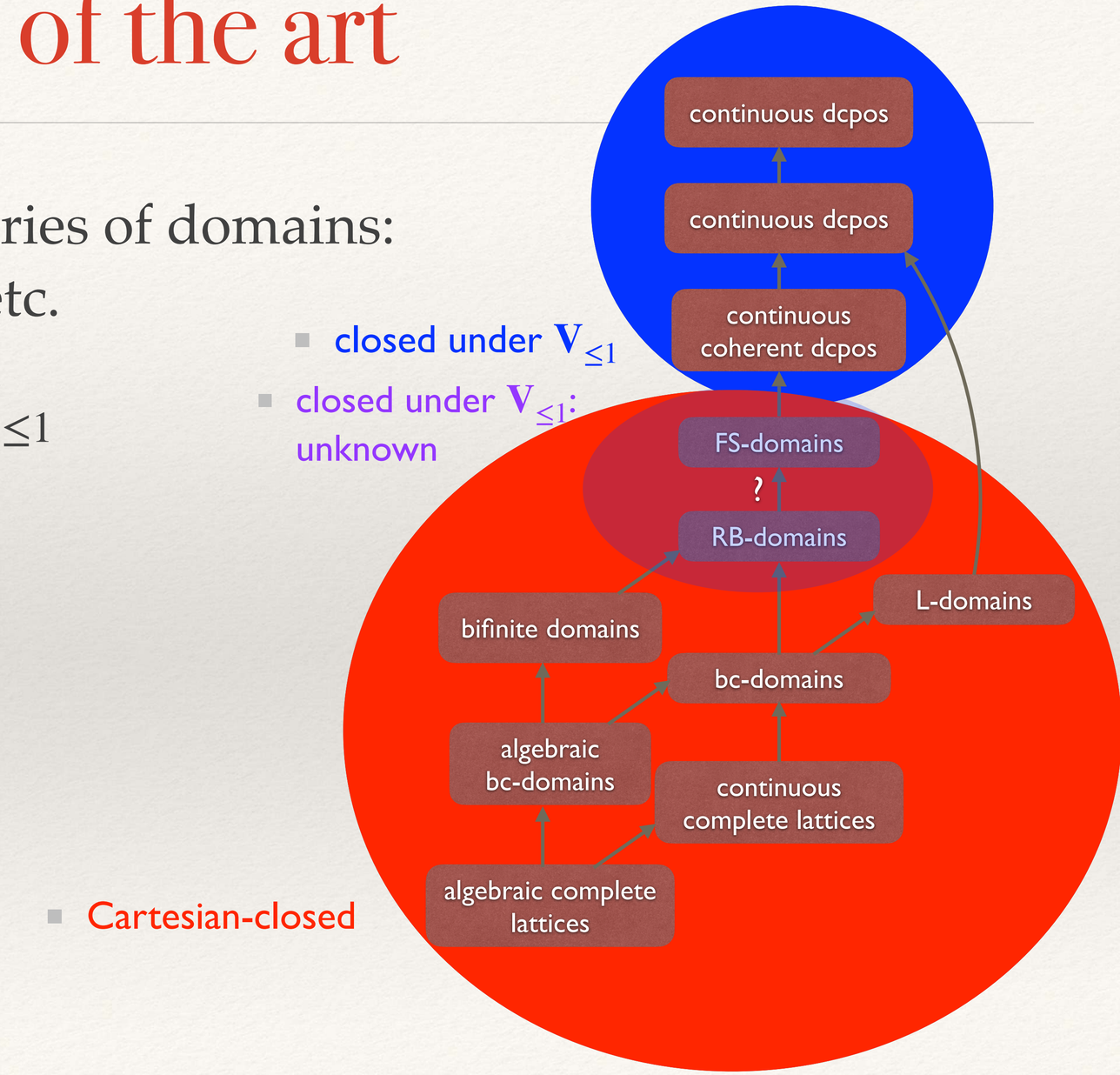
- There are other Cartesian-closed categories of domains: L-domains, RB-domains, FS-domains, etc.



- Cartesian-closed

The state of the art

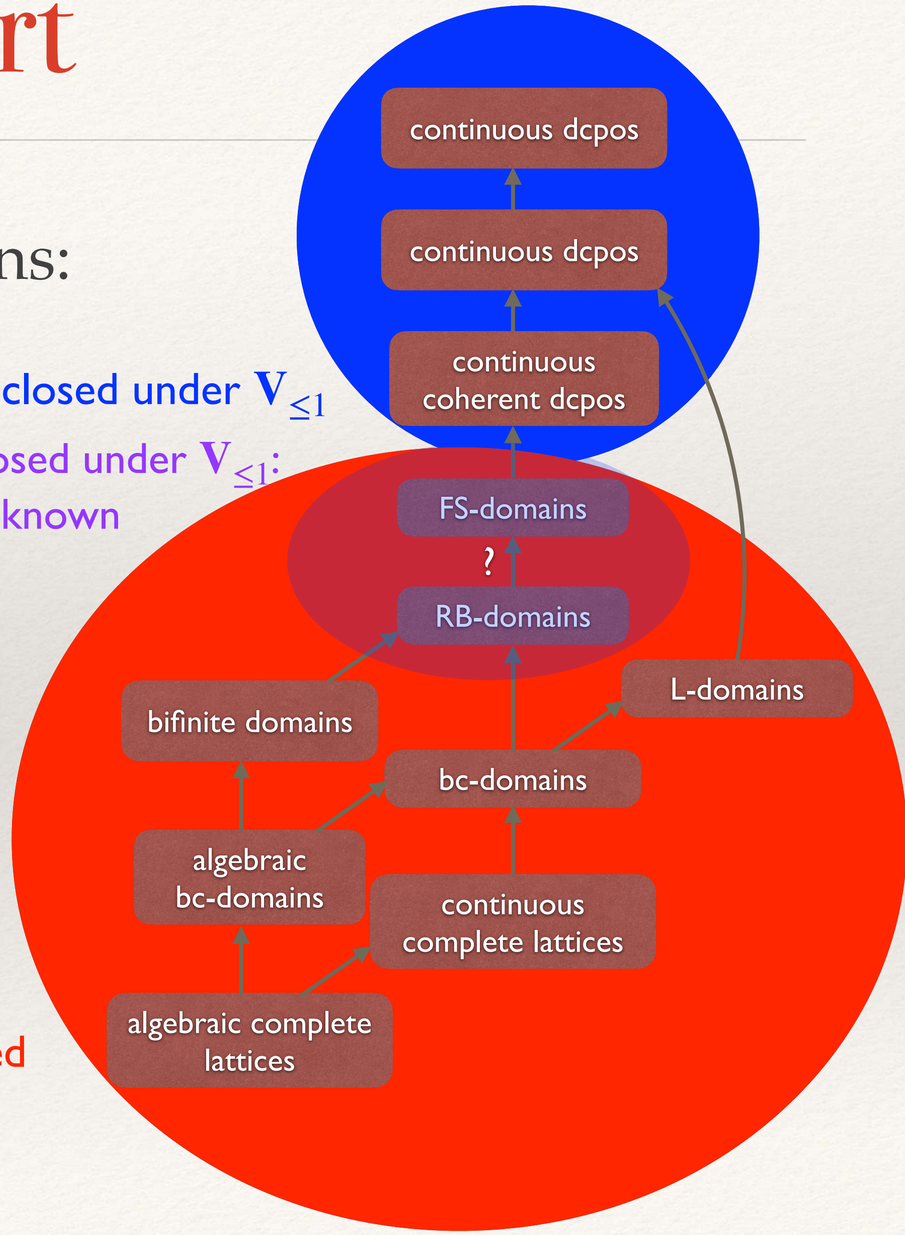
- There are other Cartesian-closed categories of domains: L-domains, RB-domains, FS-domains, etc.
- 😕 None is known to be closed under $\mathbf{V}_{\leq 1}$



The state of the art

- There are other Cartesian-closed categories of domains: L-domains, RB-domains, FS-domains, etc.
- 🙁 None is known to be closed under $\mathbf{V}_{\leq 1}$
- As of 2023, the best results are still those of [Jung,Tix 98]

apart from [JGL 12] ($\mathbf{V}_{\leq 1}$(QRB-domain) is a QRB-domain)
or [Mislove 20] ($\mathbf{V}_{\leq 1}$(chain) is a continuous lattice)
or [JGL 22] ($\mathbf{V}_{\leq 1}$(quasi-cont. dcpo) is quasi-continuous)



Statistical programming languages

Statistical programming languages

- Meanwhile, statisticians were inventing their own languages in order to describe **probabilistic processes:**
  - — Church [Goodman,Mansinghka,Roy,Bonawitz,Tenenbaum 08]
  - — Anglican [Wood,van de Meent,Mansinghka 14]
  - — WebPPL [Wood,Stuhlmüller 14]
  - — Venture [Mansinghka,Selsam,Perov 14]
  - — Hakaru [Shan, Ramsey 17]
  - — PPCF [Ehrhard, Pagani, Tasson 18]
  - — PCFSS [Dal Lago, Hoshino 19]
  - — etc.

```
(define (DP alpha proc)
  (let ((sticks (mem (lambda x (beta 1.0 alpha))))
        (atoms  (mem (lambda x (proc)))))
    (lambda () (atoms (pick-a-stick sticks 1)))))

(define (pick-a-stick sticks J)
  (if (< (random) (sticks J))
      J
      (pick-a-stick sticks (+ J 1))))

(define (DPmem alpha proc)
  (let ((dps (mem (lambda args
                    (DP alpha
                        (lambda () (apply proc args
                        )))))
    (lambda argsin ((apply dps argsin))) ))
```

Dirichlet processes in Church

```
[assume initial-state-dist (list (/ 1 3) (/ 1 3) (/ 1 3))]
[assume get-state-transition-dist (lambda (s)
  (cond ((= s 0) (list .1 .5 .4)) ((= s 1) (list .2 .2 .6))
        ((= s 2) (list .15 .15 .7))))]
[assume transition (lambda (prev-state)
  (discrete (get-state-transition-dist prev-state)))]
[assume get-state (mem (lambda (index)
  (if (<= index 0) (discrete initial-state-dist)
                   (transition (get-state (- index 1))))))]
[assume get-state-observation-mean (lambda (s)
        (cond ((= s 0) -1) ((= s 1) 1) ((= s 2) 0)))]
[observe (normal (get-state-obs-mean (get-state 1)) 1) .9]
[observe (normal (get-state-obs-mean (get-state 2)) 1) .8]
⋮
[observe (normal (get-state-obs-mean (get-state 16)) 1) -1]
[predict (get-state 0)]
[predict (get-state 1)]
⋮
[predict (get-state 16)]
⋮
```

Hidden Markov models in Venture

Statistical programming languages

- Meanwhile, statisticians were inventing their own languages in order to describe **probabilistic processes:**
  — Church [Goodman,Mansinghka,Roy,Bonawitz,Tenenbaum 08]
  — Anglican [Wood,van de Meent,Mansinghka 14]
  — WebPPL [Wood,Stuhlmüller 14]
  — Venture [Mansinghka,Selsam,Perov 14]
  — Hakaru [Shan, Ramsey 17]
  — PPCF [Ehrhard, Pagani, Tasson 18]
  — PCFSS [Dal Lago, Hoshino 19]
  — etc.

- Basically pure higher-order functional languages,

```
(define (DP alpha proc)
  (let ((sticks (mem (lambda x (beta 1.0 alpha))))
        (atoms  (mem (lambda x (proc)))))
    (lambda () (atoms (pick-a-stick sticks 1)))))

(define (pick-a-stick sticks J)
  (if (< (random) (sticks J))
      J
      (pick-a-stick sticks (+ J 1))))

(define (DPmem alpha proc)
  (let ((dps (mem (lambda args
                    (DP alpha
                        (lambda () (apply proc args
                        )))))
    (lambda argsin ((apply dps argsin))) ))
```

Dirichlet processes in Church

```
[assume initial-state-dist (list (/ 1 3) (/ 1 3) (/ 1 3))]
[assume get-state-transition-dist (lambda (s)
  (cond ((= s 0) (list .1 .5 .4)) ((= s 1) (list .2 .2 .6))
        ((= s 2) (list .15 .15 .7))))]
[assume transition (lambda (prev-state)
  (discrete (get-state-transition-dist prev-state)))]
[assume get-state (mem (lambda (index)
  (if (<= index 0) (discrete initial-state-dist)
                   (transition (get-state (- index 1))))))]
[assume get-state-observation-mean (lambda (s)
        (cond ((= s 0) -1) ((= s 1) 1) ((= s 2) 0)))]
[observe (normal (get-state-obs-mean (get-state 1)) 1) .9]
[observe (normal (get-state-obs-mean (get-state 2)) 1) .8]
⋮
[observe (normal (get-state-obs-mean (get-state 16)) 1) -1]
[predict (get-state 0)]
[predict (get-state 1)]
⋮
[predict (get-state 16)]
⋮
```

Hidden Markov models in Venture

Statistical programming languages

- Meanwhile, statisticians were inventing their own languages in order to describe **probabilistic processes:**
  — Church [Goodman,Mansinghka,Roy,Bonawitz,Tenenbaum 08]
  — Anglican [Wood,van de Meent,Mansinghka 14]
  — WebPPL [Wood,Stuhlmüller 14]
  — Venture [Mansinghka,Selsam,Perov 14]
  — Hakaru [Shan, Ramsey 17]
  — PPCF [Ehrhard, Pagani, Tasson 18]
  — PCFSS [Dal Lago, Hoshino 19]
  — etc.

- Basically pure higher-order functional languages,
  - plus computation **over the reals**

```
(define (DP alpha proc)
  (let ((sticks (mem (lambda x (beta 1.0 alpha))))
        (atoms  (mem (lambda x (proc)))))
    (lambda () (atoms (pick-a-stick sticks 1)))))

(define (pick-a-stick sticks J)
  (if (< (random) (sticks J))
      J
      (pick-a-stick sticks (+ J 1))))

(define (DPmem alpha proc)
  (let ((dps (mem (lambda args
                    (DP alpha
                        (lambda () (apply proc args
                        )))))
    (lambda argsin ((apply dps argsin))) ))
```

Dirichlet processes in Church

reals

```
[assume initial-state-dist (list (/ 1 3) (/ 1 3) (/ 1 3))]
[assume get-state-transition-dist (lambda (s)
  (cond ((= s 0) (list .1 .5 .4)) ((= s 1) (list .2 .2 .6))
        ((= s 2) (list .15 .15 .7))))]
[assume transition (lambda (prev-state)
  (discrete (get-state-transition-dist prev-state)))]
[assume get-state (mem (lambda (index)
  (if (<= index 0) (discrete initial-state-dist)
                   (transition (get-state (- index 1))))))]
[assume get-state-observation-mean (lambda (s)
        (cond ((= s 0) -1) ((= s 1) 1) ((= s 2) 0)))]
[observe (normal (get-state-obs-mean (get-state 1)) 1) .9]
[observe (normal (get-state-obs-mean (get-state 2)) 1) .8]
⋮
[observe (normal (get-state-obs-mean (get-state 16)) 1) -1]
[predict (get-state 0)]
[predict (get-state 1)]
⋮
[predict (get-state 16)]
⋮
```

Hidden Markov models in Venture

Statistical programming languages

- Meanwhile, statisticians were inventing their own languages in order to describe **probabilistic processes:**
  — Church [Goodman,Mansinghka,Roy,Bonawitz,Tenenbaum 08]
  — Anglican [Wood,van de Meent,Mansinghka 14]
  — WebPPL [Wood,Stuhlmüller 14]
  — Venture [Mansinghka,Selsam,Perov 14]
  — Hakaru [Shan, Ramsey 17]
  — PPCF [Ehrhard, Pagani, Tasson 18]
  — PCFSS [Dal Lago, Hoshino 19]
  — etc.

- Basically pure higher-order functional languages,
  - plus computation **over the reals**
  - with a rich set of distributions **over the reals**

distributions over the reals

```
(define (DP alpha proc)
  (let ((sticks (mem (lambda x (beta 1.0 alpha))))
        (atoms  (mem (lambda x (proc)))))
    (lambda () (atoms (pick-a-stick sticks 1)))))

(define (pick-a-stick sticks J)
  (if (< (random) (sticks J))
      J
      (pick-a-stick sticks (+ J 1))))

(define (DPmem alpha proc)
  (let ((dps (mem (lambda args
                    (DP alpha
                        (lambda () (apply proc args
                        )))))
    (lambda argsin ((apply dps argsin))) ))
```

Dirichlet processes in Church

reals

```
[assume initial-state-dist (list (/ 1 3) (/ 1 3) (/ 1 3))]
[assume get-state-transition-dist (lambda (s)
  (cond ((= s 0) (list .1 .5 .4)) ((= s 1) (list .2 .2 .6))
        ((= s 2) (list .15 .15 .7))))]
[assume transition (lambda (prev-state)
  (discrete (get-state-transition-dist prev-state)))]
[assume get-state (mem (lambda (index)
  (if (<= index 0) (discrete initial-state-dist)
                   (transition (get-state (- index 1))))))]
[assume get-state-observation-mean (lambda (s)
        (cond ((= s 0) -1) ((= s 1) 1) ((= s 2) 0)))]
[observe (normal (get-state-obs-mean (get-state 1)) 1) .9]
[observe (normal (get-state-obs-mean (get-state 2)) 1) .8]
⋮
[observe (normal (get-state-obs-mean (get-state 16)) 1) -1]
[predict (get-state 0)]
[predict (get-state 1)]
⋮
[predict (get-state 16)]
⋮
```

Hidden Markov models in Venture

Statistical programming languages

distributions over the reals

- Meanwhile, statisticians were inventing their own languages in order to describe **probabilistic processes:**
  — Church [Goodman,Mansinghka,Roy,Bonawitz,Tenenbaum 08]
  — Anglican [Wood,van de Meent,Mansinghka 14]
  — WebPPL [Wood,Stuhlmüller 14]
  — Venture [Mansinghka,Selsam,Perov 14]
  — Hakaru [Shan, Ramsey 17]
  — PPCF [Ehrhard, Pagani, Tasson 18]
  — PCFSS [Dal Lago, Hoshino 19]
  — etc.

```
(define (DP alpha proc)
  (let ((sticks (mem (lambda x (beta 1.0 alpha))))
        (atoms  (mem (lambda x (proc)))))
    (lambda () (atoms (pick-a-stick sticks 1)))))

(define (pick-a-stick sticks J)
  (if (< (random) (sticks J))
      J
      (pick-a-stick sticks (+ J 1))))

(define (DPmem alpha proc)
  (let ((dps (mem (lambda args
                    (DP alpha
                        (lambda () (apply proc args
                        )))))
    (lambda argsin ((apply dps argsin))) ))
```

Dirichlet processes in Church

- Basically pure higher-order functional languages,
  - plus computation **over the reals**
  - with a rich set of distributions **over the reals**
  - plus computation of **conditional probabilities**

reals

conditioning

```
[assume initial-state-dist (list (/ 1 3) (/ 1 3) (/ 1 3))]
[assume get-state-transition-dist (lambda (s)
  (cond ((= s 0) (list .1 .5 .4)) ((= s 1) (list .2 .2 .6))
        ((= s 2) (list .15 .15 .7))))]
[assume transition (lambda (prev-state)
  (discrete (get-state-transition-dist prev-state)))]
[assume get-state (mem (lambda (index)
  (if (<= index 0) (discrete initial-state-dist)
                   (transition (get-state (- index 1))))))]
[assume get-state-observation-mean (lambda (s)
        (cond ((= s 0) -1) ((= s 1) 1) ((= s 2) 0)))]
[observe (normal (get-state-obs-mean (get-state 1)) 1) .9]
[observe (normal (get-state-obs-mean (get-state 2)) 1) .8]
⋮
[observe (normal (get-state-obs-mean (get-state 16)) 1) -1]
[predict (get-state 0)]
[predict (get-state 1)]
⋮
[predict (get-state 16)]
⋮
```

Hidden Markov models in Venture

Statistical programming languages

- Meanwhile, statisticians were inventing their own languages in order to describe **probabilistic processes:**
  - — Church [Goodman,Mansinghka,Roy,Bonawitz,Tenenbaum 08]
  - — Anglican [Wood,van de Meent,Mansinghka 14]
  - — WebPPL [Wood,Stuhlmüller 14]
  - — Venture [Mansinghka,Selsam,Perov 14]
  - — Hakaru [Shan, Ramsey 17]
  - — PPCF [Ehrhard, Pagani, Tasson 18]
  - — PCFSS [Dal Lago, Hoshino 19]
  - — etc.
- Basically pure higher-order functional languages,
  - plus computation **over the reals**
  - with a rich set of distributions **over the reals**
  - plus computation of **conditional probabilities**

distributions over the reals

```
(define (DP alpha proc)
  (let ((sticks (mem (lambda x (beta 1.0 alpha))))
        (atoms  (mem (lambda x (proc)))))
    (lambda () (atoms (pick-a-stick sticks 1)))))

(define (pick-a-stick sticks J)
  (if (< (random) (sticks J))
      J
      (pick-a-stick sticks (+ J 1))))

(define (DPmem alpha proc)
  (let ((dps (mem (lambda args
                    (DP alpha
                        (lambda () (apply proc args
                        )))))
    (lambda argsin ((apply dps argsin))) ))
```

Dirichlet processes in Church

Usually implemented as **floating-point**…
but we can do better
(from a formal
point of view)

reals

conditioning

```
[assume initial-state-dist (list (/ 1 3) (/ 1 3) (/ 1 3))]
[assume get-state-transition-dist (lambda (s)
  (cond ((= s 0) (list .1 .5 .4)) ((= s 1) (list .2 .2 .6))
        ((= s 2) (list .15 .15 .7))))]
[assume transition (lambda (prev-state)
  (discrete (get-state-transition-dist prev-state)))]
[assume get-state (mem (lambda (index)
  (if (<= index 0) (discrete initial-state-dist)
                   (transition (get-state (- index 1))))))]
[assume get-state-observation-mean (lambda (s)
        (cond ((= s 0) -1) ((= s 1) 1) ((= s 2) 0)))]
[observe (normal (get-state-obs-mean (get-state 1)) 1) .9]
[observe (normal (get-state-obs-mean (get-state 2)) 1) .8]
⋮
[observe (normal (get-state-obs-mean (get-state 16)) 1) -1]
[predict (get-state 0)]
[predict (get-state 1)]
⋮
[predict (get-state 16)]
⋮
```

Hidden Markov models in Venture

Semantics

- The difficulty of handling **continuous** distributions and **real numbers** spurred the invention of new semantical domains:
  - quasi-Borel spaces/predomains [Heunen,Kammar,Staton,Yang 17; Vákár, Kammar, Staton 21]
  - measurable cones [Ehrhard,Pagani, Tasson 17]
  - measurable space/geometry of interaction [Dal Lago, Hoshino 19]

Semantics

- The difficulty of handling **continuous** distributions and **real numbers** spurred the invention of new semantical domains:
  - quasi-Borel spaces/predomains [Heunen,Kammar,Staton,Yang 17; Vákár, Kammar, Staton 21]
  - measurable cones [Ehrhard,Pagani, Tasson 17]
  - measurable space/geometry of interaction [Dal Lago, Hoshino 19]
- … especially since [Jung, Tix 98] was **misinterpreted** as a failure of domain theory — we will give a dcpo semantics

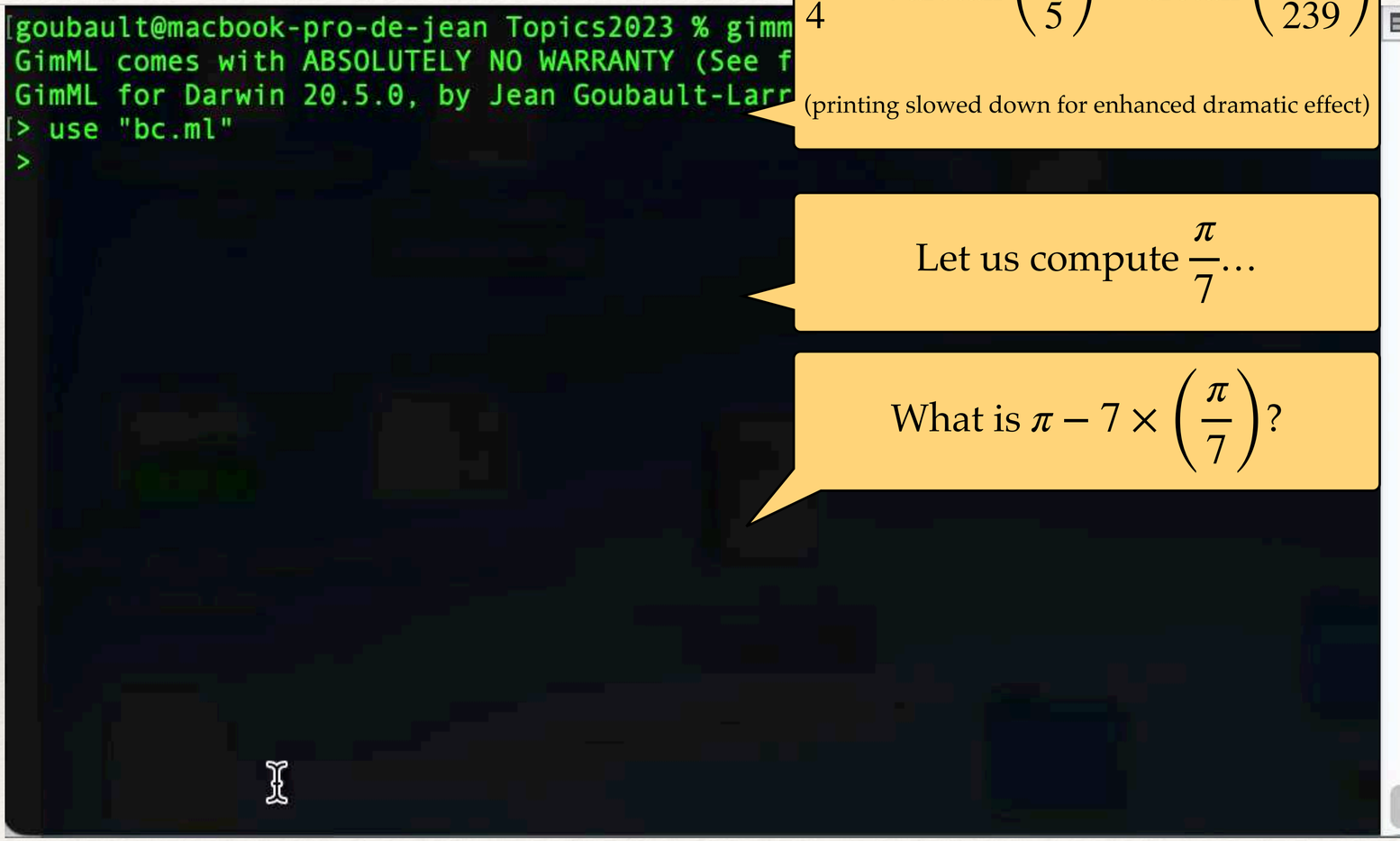
Exact real arithmetic

- **Exact real arithmetic** can be implemented in many ways
- E.g., [Boehm 87] encodes a real number x by a **function** $n \in \mathbb{N} \mapsto x_n \in \mathbb{Z}$

  such that $\left| x - \frac{x_n}{4^n} \right| \leq \frac{1}{4^n}$

Exact real arithmetic

- **Exact real arithmetic** can be implemented in many ways
- E.g., [Boehm 87] encodes a real number x by a **function** $n \in \mathbb{N} \mapsto x_n \in \mathbb{Z}$ such that $\left| x - \frac{x_n}{4^n} \right| \leq \frac{1}{4^n}$
- x is really $\sup^{\uparrow}_{n \in \mathbb{N}} \left[\frac{x_n}{4^n}, \frac{x_n + 1}{4^n} \right]$ in $\mathbb{IR}_{\perp}$, as we now demonstrate…



Eg., compute π by Machin's formula

$$\frac{\pi}{4} = 4 \arctan\left(\frac{1}{5}\right) - \arctan\left(\frac{1}{239}\right)$$

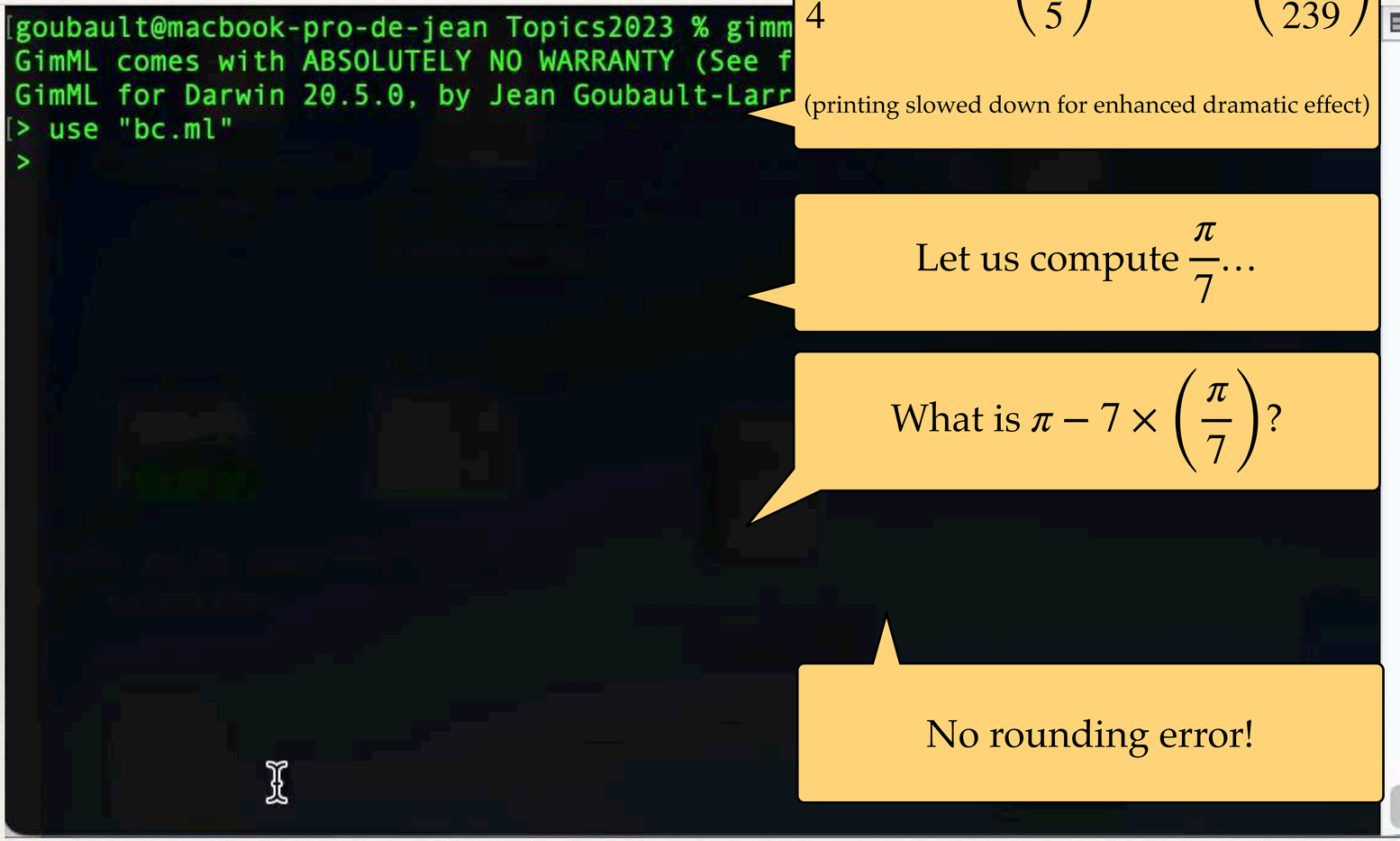
(printing slowed down for enhanced dramatic effect)

Let us compute $\frac{\pi}{7}$…

What is $\pi - 7 \times \left(\frac{\pi}{7}\right)$?

Exact real arithmetic

- **Exact real arithmetic** can be implemented in many ways
- E.g., [Boehm 87] encodes a real number x by a **function** $n \in \mathbb{N} \mapsto x_n \in \mathbb{Z}$ such that $\left| x - \frac{x_n}{4^n} \right| \leq \frac{1}{4^n}$
- x is really $\sup^{\uparrow}_{n \in \mathbb{N}} \left[\frac{x_n}{4^n}, \frac{x_n + 1}{4^n} \right]$ in $\mathbb{IR}_{\perp}$, as we now demonstrate…



Eg., compute π by Machin's formula

$$\frac{\pi}{4} = 4 \arctan\left(\frac{1}{5}\right) - \arctan\left(\frac{1}{239}\right)$$

(printing slowed down for enhanced dramatic effect)

Let us compute $\frac{\pi}{7}$…

What is $\pi - 7 \times \left(\frac{\pi}{7}\right)$?

No rounding error!

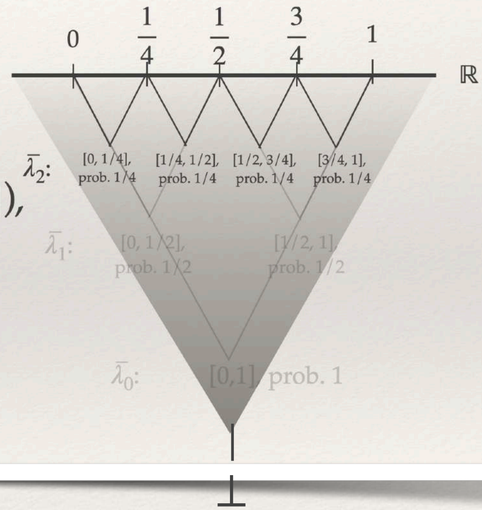
Lebesgue is implementable

- Remember $\bar{\lambda}_{|[0,1]}$?
- This is also implementable, as a **program** that draws **exact real numbers** at random in [0, 1]…

Beyond simple valuations

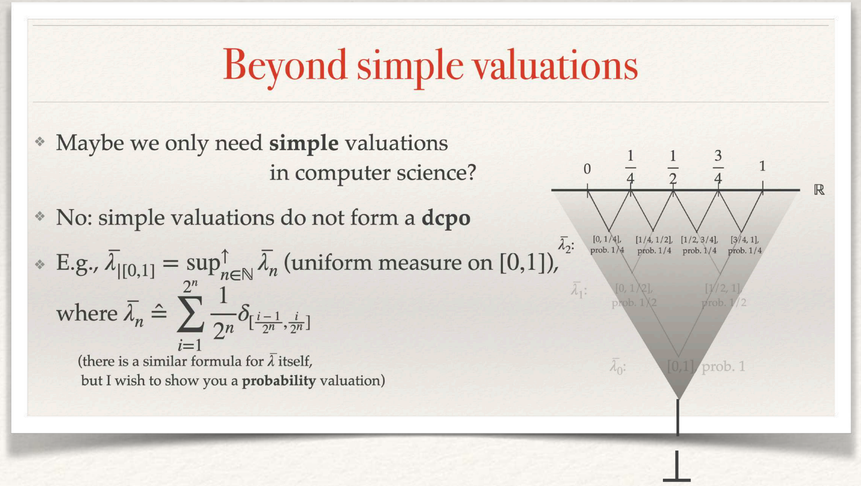
- Maybe we only need **simple** valuations in computer science?
- No: simple valuations do not form a **dcpo**
- E.g., $\bar{\lambda}_{|[0,1]} = \sup^{\uparrow}_{n \in \mathbb{N}} \bar{\lambda}_n$ (uniform measure on [0,1]), where $\bar{\lambda}_n \mathrel{\hat{=}} \sum_{i=1}^{2^n} \frac{1}{2^n} \delta_{[\frac{i-1}{2^n}, \frac{i}{2^n}]}$

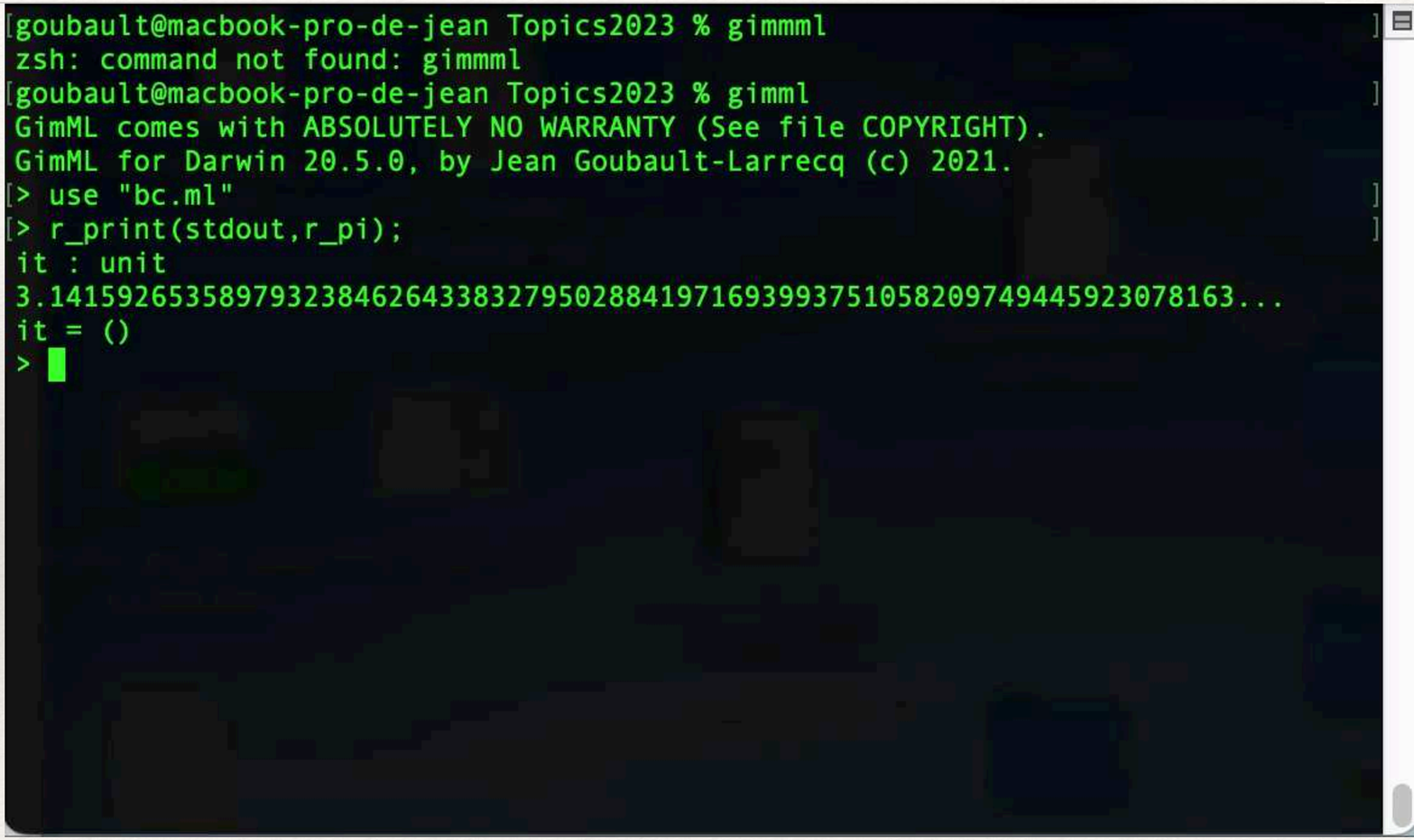
  (there is a similar formula for $\bar{\lambda}$ itself, but I wish to show you a **probability** valuation)



Lebesgue is implementable

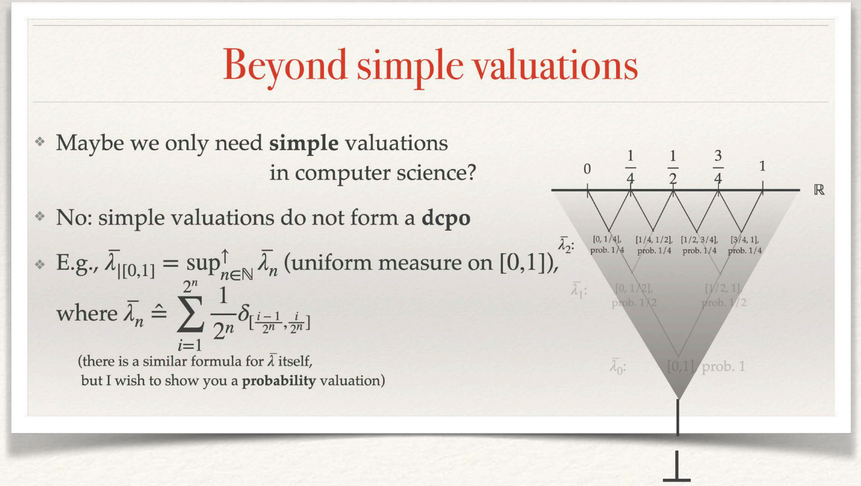
- Remember $\overline{\lambda}_{|[0,1]}$?
- This is also implementable, as a **program** that draws **exact real numbers** at random in [0, 1]…

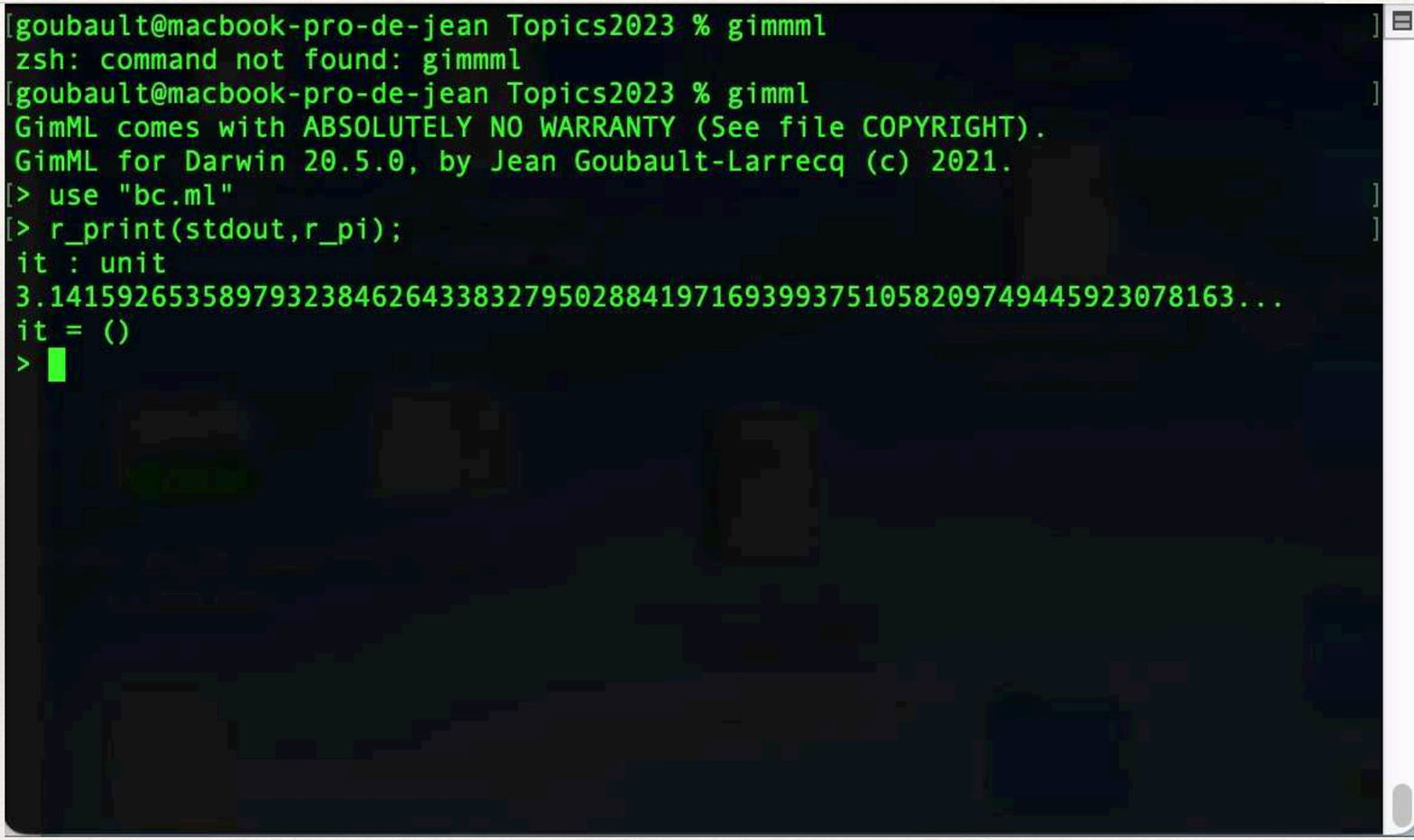




Lebesgue is implementable

- Remember $\overline{\lambda}_{|[0,1]}$?
- This is also implementable, as a **program** that draws **exact real numbers** at random in [0, 1]…





Interval Statistical PCF (ISPCF), v1

- $$
\begin{aligned}
M, N, P, \ldots ::= \; & \ldots && \text{(as in PCF)} \\
| \; & \mathbf{ret}\ M && \text{monad unit} \\
| \; & \mathbf{do}\ x_\sigma = M; N && \text{sequential composition} \\
| \; & \cancel{M \oplus N} && \cancel{\text{probabilistic choice}} \\
| \; & \mathbf{sample}[0,1] && (\bar{\lambda}_{|[0,1]}) \\
| \; & \underline{r} && (\text{real constants, } r \in \mathbb{R}) \\
| \; & \underline{f}(M_1, \cdots, M_n) && (f \in \{ +, -, >, \cdots \})
\end{aligned}
$$

(subsumed by **sample**[0,1])

- Types: $\sigma, \tau, \ldots ::= \mathbf{nat} \mid \mathbf{unit} \mid \mathbf{real} \mid \sigma \to \tau \mid \mathbf{T}\tau$

Interval Statistical PCF (ISPCF), v1

- $$
\begin{aligned}
M, N, P, \ldots ::= \; & \ldots && \text{(as in PCF)} \\
| \; & \mathbf{ret}\ M && \text{monad unit} \\
| \; & \mathbf{do}\ x_\sigma = M; N && \text{sequential composition} \\
| \; & \cancel{M \oplus N} && \cancel{\text{probabilistic choice}} \\
| \; & \mathbf{sample}[0,1] && (\bar{\lambda}_{|[0,1]}) \\
| \; & \underline{r} && (\text{real constants, } r \in \mathbb{R}) \\
| \; & \underline{f}(M_1, \cdots, M_n) && (f \in \{+, -, >, \cdots\})
\end{aligned}
$$

(subsumed by **sample**[0,1])

Distribution on **exact reals**

- Types: $\sigma, \tau, \ldots ::= \mathbf{nat} \mid \mathbf{unit} \mid \mathbf{real} \mid \sigma \to \tau \mid \mathbf{T}\tau$

Interval Statistical PCF (ISPCF), v1

$$
\begin{aligned}
M, N, P, \ldots ::= \; & \ldots && \text{(as in PCF)} \\
| \; & \mathbf{ret}\ M && \text{monad unit} \\
| \; & \mathbf{do}\ x_\sigma = M; N && \text{sequential composition} \\
| \; & \cancel{M \oplus N} && \cancel{\text{probabilistic choice}} \\
| \; & \mathbf{sample}[0,1] && (\bar{\lambda}_{|[0,1]}) \\
| \; & \underline{r} && (\text{real constants, } r \in \mathbb{R}) \\
| \; & \underline{f}(M_1, \cdots, M_n) && (f \in \{+, -, >, \cdots\})
\end{aligned}
$$

(subsumed by **sample**[0,1])

Distribution on **exact reals**

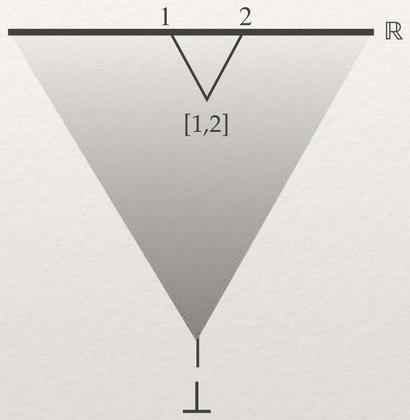
Exact real arithmetic

- Types: $\sigma, \tau, \ldots ::= \mathbf{nat} \mid \mathbf{unit} \mid \boxed{\mathbf{real}} \mid \sigma \to \tau \mid \mathbf{T}\tau$

The denotational semantics of ISPCF v1

- $[\![\mathbf{nat}]\!] \triangleq \mathbb{N}_\perp$, $[\![\mathbf{unit}]\!] \triangleq \{\perp, *\}$, $[\![\sigma \to \tau]\!] \triangleq [[\![\sigma]\!] \to [\![\tau]\!]]$,
  $[\![\mathbf{T}\tau]\!] \triangleq \mathbf{V}_{\leq 1}[\![\tau]\!]$ as before

  plus $[\![\mathbf{real}]\!] \triangleq \mathbf{I}\mathbb{R}_\perp$



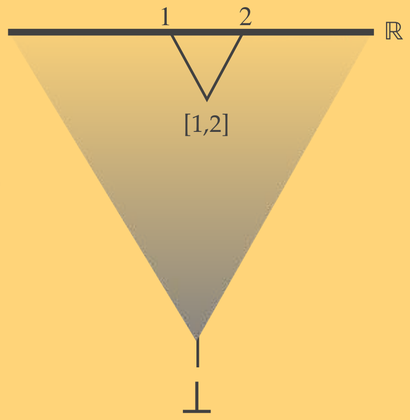
- $[\![\mathbf{sample}[0,1]]\!]\rho \triangleq \bar{\lambda}_{|[0,1]}$
  $[\![\underline{r}]\!]\rho \triangleq [r, r]$
  $[\![\underline{f}(M_1, \cdots, M_n)]\!]\rho \triangleq \check{f}([\![M_1]\!]\rho, \cdots, [\![M_n]\!]\rho)$

  where $\check{f} \triangleq$ largest Scott-continuous map $\leq f \ldots$

The denotational semantics of ISPCF v1

- $[\![\mathbf{nat}]\!] \triangleq \mathbb{N}_\perp$, $[\![\mathbf{unit}]\!] \triangleq \{\perp, *\}$, $[\![\sigma \to \tau]\!] \triangleq [[\![\sigma]\!] \to [\![\tau]\!]]$,
  $[\![\mathbf{T}\tau]\!] \triangleq \mathbf{V}_{\leq 1}[\![\tau]\!]$ as before

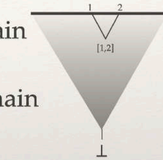
plus $[\![\mathbf{real}]\!] \triangleq \mathbf{I}\mathbb{R}_\perp$



- $[\![\mathbf{sample}[0,1]]\!]\rho \triangleq \bar{\lambda}_{|[0,1]}$
  $[\![\underline{r}]\!]\rho \triangleq [r, r]$
  $[\![\underline{f}(M_1, \cdots, M_n)]\!]\rho \triangleq \check{f}([\![M_1]\!]\rho, \cdots, [\![M_n]\!]\rho)$
  where $\check{f} \triangleq$ largest Scott-continuous map $\leq f \ldots$

The extension theorem for bc-domains

- A **bc-domain** is a continuous dcpo that:
  — is pointed (has a $\perp$)
  — any two elements x, y with an upper bound have a **least** upper bound $x \vee y$
- $\mathbb{N}_\perp$ is a bc-domain
- not $\mathbb{IR}$ (no $\perp$)
- $\mathbb{IR}_\perp$ is a bc-domain



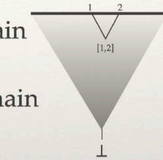
- **Theorem.** For every (arbitrary) function $f\colon X \to Z$ from a topological space X to a bc-domain Z, there is a **largest continuous** map $\check{f}\colon X \to Z$ such that $\forall x \in X, \check{f}(x) \leq f(x)$.

- *Proof.* $\check{f}(x) = \sup\{z \in Z \mid x \in int(f^{-1}(\uparrow z))\}$ $\left(= \sup\{\inf_U f \mid U \in \mathcal{O}X \text{ s.t. } x \in U\}\right)$.

  The sup exists because Z is a bc-domain. $\square$

The extension theorem for bc-domains

- A **bc-domain** is a continuous dcpo that:
  — is pointed (has a $\perp$)
  — any two elements x, y with an upper bound have a **least** upper bound $x \vee y$
- $\mathbb{N}_\perp$ is a bc-domain
- not $\mathbb{IR}$ (no $\perp$)
- $\mathbb{IR}_\perp$ is a bc-domain



- **Theorem.** For every (arbitrary) function $f\colon X \to Z$ from a topological space X to a bc-domain Z, there is a **largest continuous** map $\check{f}\colon X \to Z$ such that $\forall x \in X, \check{f}(x) \leq f(x)$.

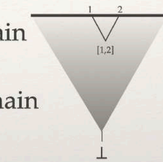
a **best possible** approximation

- *Proof.* $\check{f}(x) = \sup\{z \in Z \mid x \in int(f^{-1}(\uparrow z))\}$ $\left(= \sup\{\inf_U f \mid U \in \mathcal{O}X \text{ s.t. } x \in U\}\right)$.

  The sup exists because Z is a bc-domain. $\square$

The extension theorem for bc-domains

- A **bc-domain** is a continuous dcpo that:
  — is pointed (has a $\perp$)
  — any two elements x, y with an upper bound have a **least** upper bound $x \vee y$
- $\mathbb{N}_\perp$ is a bc-domain
- not $\mathbb{IR}$ (no $\perp$)
- $\mathbb{IR}_\perp$ is a bc-domain



- **Theorem.** For every (arbitrary) function $f\colon X \to Z$ from a topological space X to a bc-domain Z, there is a **largest continuous** map $\check{f}\colon X \to Z$ such that $\forall x \in X, \check{f}(x) \leq f(x)$.

a **best possible** approximation

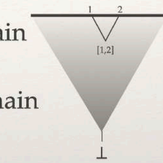
- *Proof.* $\check{f}(x) = \sup\{z \in Z \mid x \in int(f^{-1}(\uparrow z))\}$ $(= \sup\{\inf_U f \mid U \in \mathcal{O}X \text{ s.t. } x \in U\})$.

  The sup exists because Z is a bc-domain. $\square$

- $\check{+}$ maps $([a, b], [c, d])$ to $([a + c, b + d])$

The extension theorem for bc-domains

- A **bc-domain** is a continuous dcpo that:
  — is pointed (has a $\perp$)
  — any two elements x, y with an upper bound have a **least** upper bound $x \vee y$
- $\mathbb{N}_\perp$ is a bc-domain
- not $\mathbb{IR}$ (no $\perp$)
- $\mathbb{IR}_\perp$ is a bc-domain



- **Theorem.** For every (arbitrary) function $f\colon X \to Z$ from a topological space X to a bc-domain Z, there is a **largest continuous** map $\check{f}\colon X \to Z$ such that $\forall x \in X, \check{f}(x) \leq f(x)$.

a **best possible** approximation

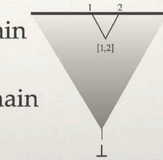
- *Proof.* $\check{f}(x) = \sup\{z \in Z \mid x \in int(f^{-1}(\uparrow z))\}$ $(= \sup\{\inf_U f \mid U \in \mathcal{O}X \text{ s.t. } x \in U\})$.

  The sup exists because Z is a bc-domain. $\square$

- $\check{+}$ maps $([a, b], [c, d])$ to $([a + c, b + d])$
- $\check{\text{sgn}}$ maps $[a, b]$ to -1 if $b < 0$, $+1$ if $a > 0$, $\perp$ otherwise

The extension theorem for bc-domains

- A **bc-domain** is a continuous dcpo that:
  — is pointed (has a $\perp$)
  — any two elements x, y with an upper bound have a **least** upper bound $x \vee y$
- $\mathbb{N}_\perp$ is a bc-domain
- not $\mathbb{IR}$ (no $\perp$)
- $\mathbb{IR}_\perp$ is a bc-domain



- **Theorem.** For every (arbitrary) function $f\colon X \to Z$ from a topological space X to a bc-domain Z, there is a **largest continuous** map $\check{f}\colon X \to Z$ such that $\forall x \in X, \check{f}(x) \leq f(x)$.

a **best possible** approximation

- *Proof.* $\check{f}(x) = \sup\{z \in Z \mid x \in int(f^{-1}(\uparrow z))\}$ $(= \sup\{\inf_U f \mid U \in \mathcal{O}X \text{ s.t. } x \in U\})$.

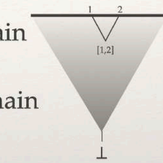
  The sup exists because Z is a bc-domain. $\square$

- $\check{+}$ maps $([a, b], [c, d])$ to $([a + c, b + d])$
- $\check{\text{sgn}}$ maps $[a, b]$ to -1 if $b < 0$, $+1$ if $a > 0$, $\perp$ otherwise

$\perp$ here is unavoidable: $\check{\text{sgn}}$ is **best** possible (alternatively, $\mathbb{R}$ is **connected**, so every non-trivial continuous predicate on $\mathbb{R}$ is **constant**)

The extension theorem for bc-domains

- A **bc-domain** is a continuous dcpo that:
  — is pointed (has a $\perp$)
  — any two elements x, y with an upper bound have a **least** upper bound $x \vee y$
- $\mathbb{N}_\perp$ is a bc-domain
- not $\mathbb{IR}$ (no $\perp$)
- $\mathbb{IR}_\perp$ is a bc-domain



- **Theorem.** For every (arbitrary) function $f\colon X \to Z$ from a topological space X to a bc-domain Z, there is a **largest continuous** map $\check{f}\colon X \to Z$ such that $\forall x \in X, \check{f}(x) \leq f(x)$.

a **best possible** approximation

- *Proof.* $\check{f}(x) = \sup\{z \in Z \mid x \in int(f^{-1}(\uparrow z))\}$ $(= \sup\{\inf_U f \mid U \in \mathcal{O}X \text{ s.t. } x \in U\})$.

  The sup exists because Z is a bc-domain. $\square$

- $\check{+}$ maps $([a,b],[c,d])$ to $([a+c, b+d])$
- $\check{\mathrm{sgn}}$ maps $[a,b]$ to -1 if $b < 0$, $+1$ if $a > 0$, $\perp$ otherwise
- $\check{>}$ maps $([a,b],[c,d])$ to true (0) if $a > d$, false (1) if $b < c$, $\perp$ otherwise

$\perp$ here is unavoidable: $\check{\mathrm{sgn}}$ is **best** possible (alternatively, $\mathbb{R}$ is **connected**, so every non-trivial continuous predicate on $\mathbb{R}$ is **constant**)

The operational semantics of ISPCF v1

Exploration rules

$$C, \mathbf{do}\ x = M; N \to C[\mathbf{do}\ x = \_; N], M$$
$$\_, \mathbf{ret}\ M \to \mathbf{ret}\ \_, M$$

Computation rules

$$C[\mathbf{do}\ x = \_; N], \mathbf{ret}\ M \to C, N[x := M]$$
$$C[\mathbf{sample}[0,1]] \to C[\mathbf{ret}\ \underline{r}] \qquad (r \leftarrow_{\mathrm{rand}} \lambda_{|[0,1]})$$
$$C[\underline{f}(\underline{r}_1, \cdots, \underline{r}_n)] \to C[\underline{f(r_1, \cdots, r_n)}]$$

- **Theorem (soundness, adequacy).**
  $\Pr[C, M \downarrow n] = [\![C[M]]\!]\rho$
  (at type **nat**)

The operational semantics of ISPCF v1

Exploration rules

$$C, \mathbf{do}\ x = M; N \to C[\mathbf{do}\ x = \_; N], M$$
$$\_, \mathbf{ret}\ M \to \mathbf{ret}\ \_, M$$

Computation rules

$$C[\mathbf{do}\ x = \_; N], \mathbf{ret}\ M \to C, N[x := M]$$
$$C[\mathbf{sample}[0,1]] \to C[\mathbf{ret}\ \underline{r}] \qquad (r \leftarrow_{\mathrm{rand}} \lambda_{|[0,1]})$$
$$C[\underline{f}(\underline{r}_1, \cdots, \underline{r}_n)] \to C[\underline{f(r_1, \cdots, r_n)}]$$

Transition relation is now really a **Markov kernel** on a **topological space** of configurations:

$$\Pr[C[\mathbf{sample}[0,1]] \to \_ \in U] = \lambda(\{r \in [0,1] \mid C[\mathbf{ret}\ \underline{r}] \in U\})$$
$$\Pr[C[\underline{f}(\underline{r}_1, \cdots, \underline{r}_n) \to \_ \in U] = \delta_{C[\underline{f(r_1, \cdots, r_n)}]}(U) \qquad \text{if } f(r_1, \cdots, r_n) \text{ defined}$$
$$= 0 \qquad \text{otherwise}$$

- **Theorem (soundness, adequacy).** $\Pr[C, M \downarrow n] = [\![C[M]]\!]\rho$ (at type **nat**)

The operational semantics of ISPCF v1

Exploration rules

$$C, \mathbf{do}\ x = M; N \to C[\mathbf{do}\ x = \_; N], M$$
$$\_, \mathbf{ret}\ M \to \mathbf{ret}\ \_, M$$

Computation rules

$$C[\mathbf{do}\ x = \_; N], \mathbf{ret}\ M \to C, N[x := M]$$
$$C[\mathbf{sample}[0,1]] \to C[\mathbf{ret}\ \underline{r}] \qquad (r \leftarrow_{\mathrm{rand}} \lambda_{|[0,1]})$$
$$C[\underline{f}(\underline{r}_1, \cdots, \underline{r}_n)] \to C[\underline{f(r_1, \cdots, r_n)}]$$

Transition relation is now really a
Markov kernel on a
topological space of configurations:

$$\Pr[C[\mathbf{sample}[0,1]] \to \_ \in U] = \lambda(\{r \in [0,1] \mid C[\mathbf{ret}\ \underline{r}] \in U\})$$
$$\Pr[C[\underline{f}(\underline{r}_1, \cdots, \underline{r}_n) \to \_ \in U] = \delta_{C[\underline{f(r_1, \cdots, r_n)}]}(U) \qquad \text{if } f(r_1, \cdots, r_n) \text{ defined}$$
$$= 0 \qquad \text{otherwise}$$

with **actual** real numbers (in $\mathbb{R}$)
not simply **exact** real numbers (in $\mathbf{I}\mathbb{R}$)

- **Theorem (soundness, adequacy).**
  $\Pr[C, M \downarrow n] = [\![C[M]]\!]\rho$
  (at type **nat**)

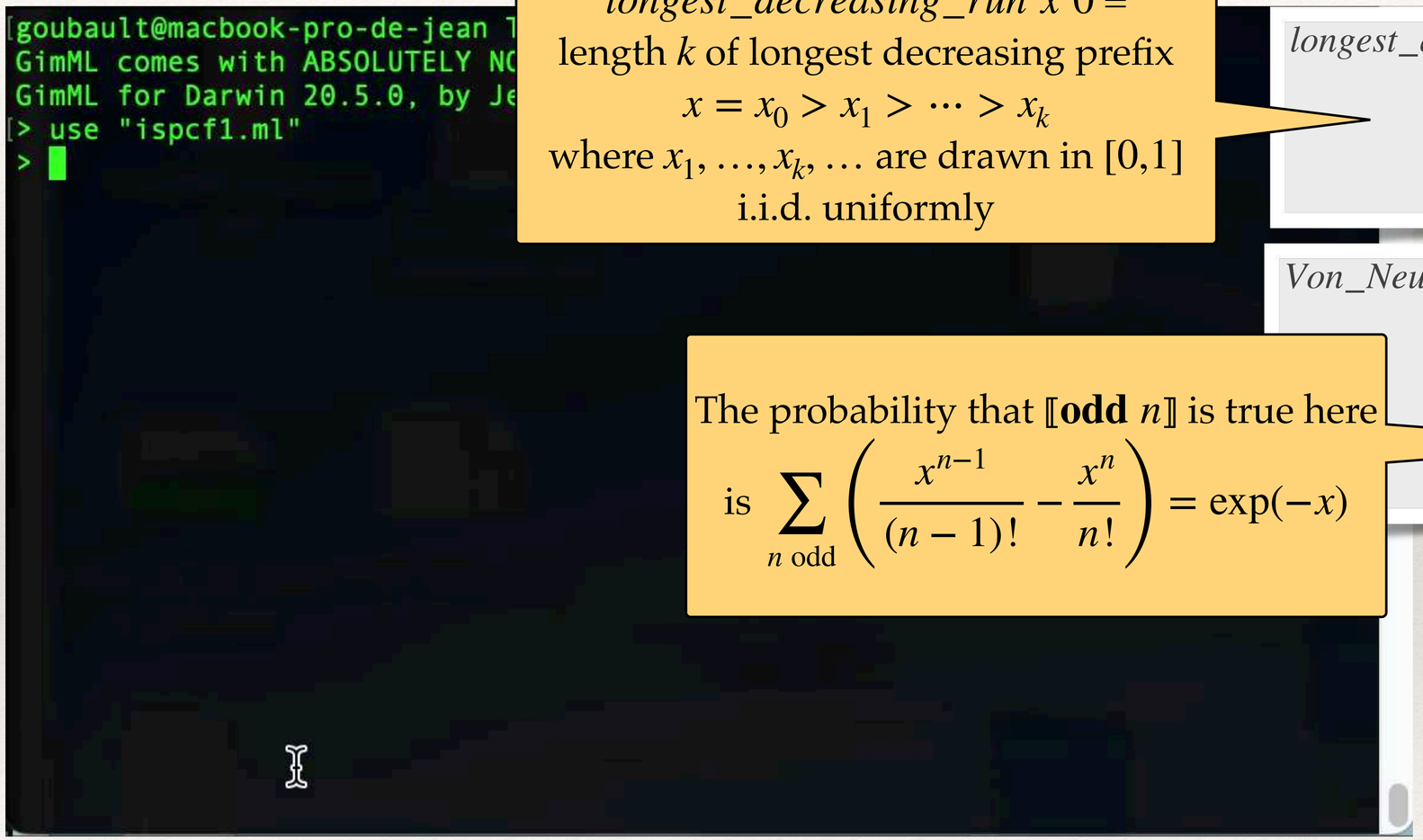
What does this compute?

```
goubault@macbook-pro-de-jean Topics2023 % gimml
GimML comes with ABSOLUTELY NO WARRANTY (See file COPYRIGHT).
GimML for Darwin 20.5.0, by Jean Goubault-Larrecq (c) 2021.
> use "ispcf1.ml"
>
```

$$
\begin{aligned}
\mathit{longest\_decreasing\_run} \mathrel{\hat{=}} \mathbf{rec}(\lambda f_{\mathbf{real}\to\mathbf{int}\to\mathbf{T}\ \mathbf{int}} \cdot \lambda x_{\mathbf{real}} \cdot \lambda n_{\mathbf{int}} \cdot\ & \\
&\mathbf{do}\ u = \mathbf{sample}[0,1]; \\
&\mathbf{if}\ u > x\ \mathbf{then}\ \mathbf{ret}\ n \\
&\qquad \mathbf{else}\ f\ u\ (\mathbf{s}(n)))
\end{aligned}
$$

$$
\begin{aligned}
\mathit{Von\_Neumann} \mathrel{\hat{=}} \mathbf{do}\ x = \mathbf{sample}[0,1]; \\
\mathbf{rec}(\lambda f_{\mathbf{real}\to\mathbf{T}\ \mathbf{real}} \cdot \lambda \ell_{\mathbf{real}} \cdot \\
&\mathbf{do}\ n = \mathit{longest\_decreasing\_run}\ x\ 0; \\
&\mathbf{if}\ \mathbf{odd}\ n\ \mathbf{then}\ f(\ell \underline{+} \underline{1.0}) \\
&\quad \mathbf{else}\ \mathbf{ret}\ \ell)\ x
\end{aligned}
$$

What does this compute?



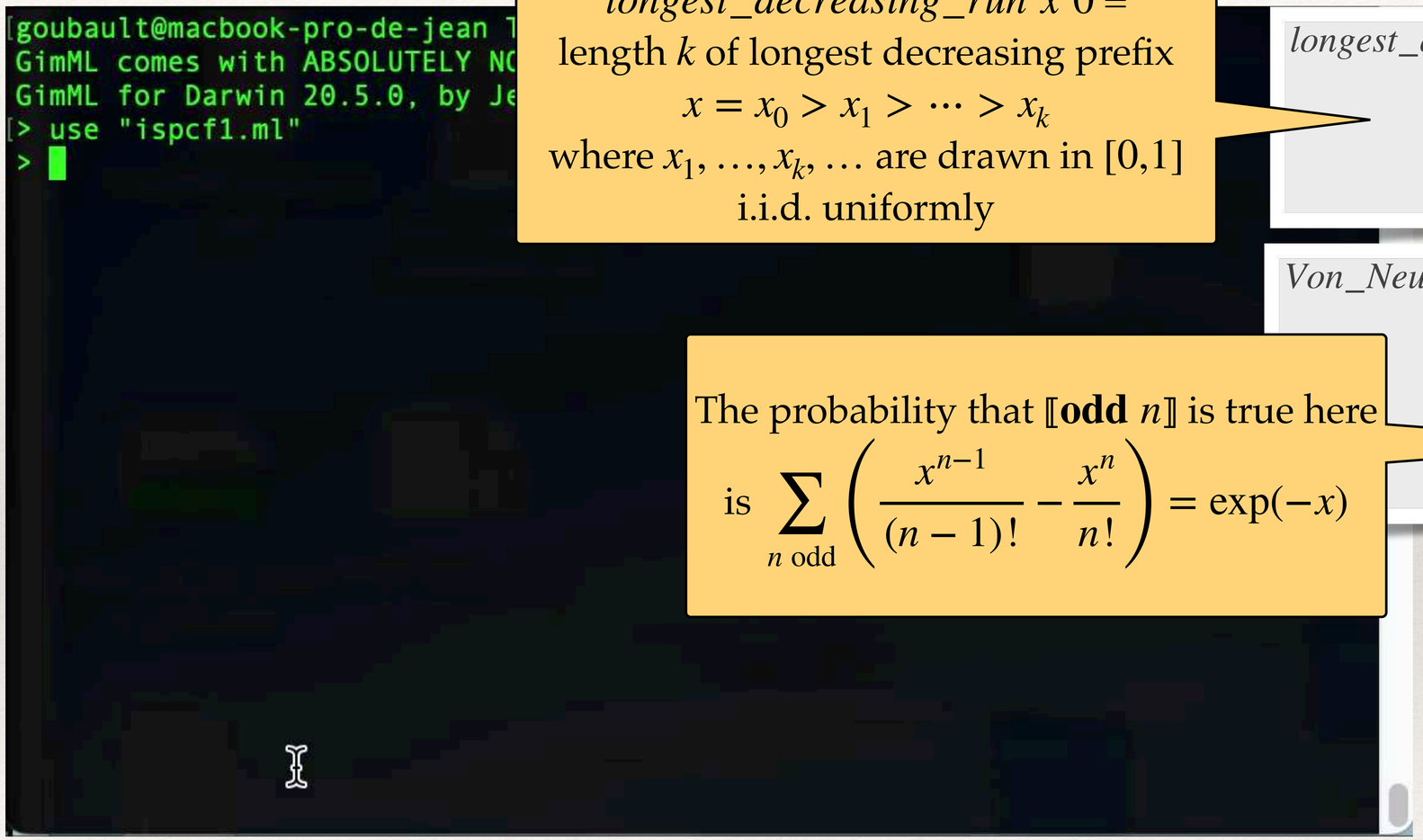
longest_decreasing_run x 0 =
length k of longest decreasing prefix
$x = x_0 > x_1 > \cdots > x_k$
where $x_1, \ldots, x_k, \ldots$ are drawn in [0,1]
i.i.d. uniformly

$$\mathit{longest\_decreasing\_run} \mathrel{\hat{=}} \mathbf{rec}(\lambda f_{\mathbf{real}\to\mathbf{int}\to\mathbf{T\ int}} \cdot \lambda x_{\mathbf{real}} \cdot \lambda n_{\mathbf{int}} \cdot \mathbf{do}\ u = \mathbf{sample}[0,1];\ \mathbf{if}\ u > x\ \mathbf{then\ ret}\ n\ \mathbf{else}\ f\ u\ (\mathbf{s}(n)))$$

$$\mathit{Von\_Neumann} \mathrel{\hat{=}} \mathbf{do}\ x = \mathbf{sample}[0,1];\ \mathbf{rec}(\lambda f_{\mathbf{real}\to\mathbf{T\ real}} \cdot \lambda \ell_{\mathbf{real}} \cdot \mathbf{do}\ n = \mathit{longest\_decreasing\_run}\ x\ 0;\ \mathbf{if\ odd}\ n\ \mathbf{then}\ f(\ell \underline{+} \underline{1.0})\ \mathbf{else\ ret}\ \ell)\ x$$

The probability that $[\![\mathbf{odd}\ n]\!]$ is true here is $\displaystyle\sum_{n \text{ odd}} \left(\frac{x^{n-1}}{(n-1)!} - \frac{x^n}{n!} \right) = \exp(-x)$

What does this compute?



longest_decreasing_run x 0 =
length k of longest decreasing prefix
$x = x_0 > x_1 > \cdots > x_k$
where $x_1, \ldots, x_k, \ldots$ are drawn in [0,1]
i.i.d. uniformly

$$\begin{aligned}
\textit{longest_decreasing_run} \mathrel{\hat{=}} \textbf{rec}(\lambda f_{\textbf{real}\to\textbf{int}\to\textbf{T int}} \cdot \lambda x_{\textbf{real}} \cdot \lambda n_{\textbf{int}} \cdot \\
&\textbf{do } u = \textbf{sample}[0,1]; \\
&\textbf{if } u > x \textbf{ then ret } n \\
&\qquad \textbf{else } f\ u\ (\mathbf{s}(n)))
\end{aligned}$$

$$\begin{aligned}
\textit{Von_Neumann} \mathrel{\hat{=}} \textbf{do } x = \textbf{sample}[0,1]; \\
&\textbf{rec}(\lambda f_{\textbf{real}\to\textbf{T real}} \cdot \lambda \ell_{\textbf{real}} \cdot \\
&\quad \textbf{do } n = \textit{longest_decreasing_run}\ x\ 0; \\
&\quad \textbf{if odd } n \textbf{ then } f(\ell \underline{+} \underline{1.0}) \\
&\qquad \textbf{else ret } \ell)\ x
\end{aligned}$$

The probability that $[\![\textbf{odd } n]\!]$ is true here is $\displaystyle\sum_{n \text{ odd}} \left(\frac{x^{n-1}}{(n-1)!} - \frac{x^n}{n!} \right) = \exp(-x)$

- … the exponential distribution $\exp(-x)\ d\lambda$ [von Neumann 49]

What does this compute?

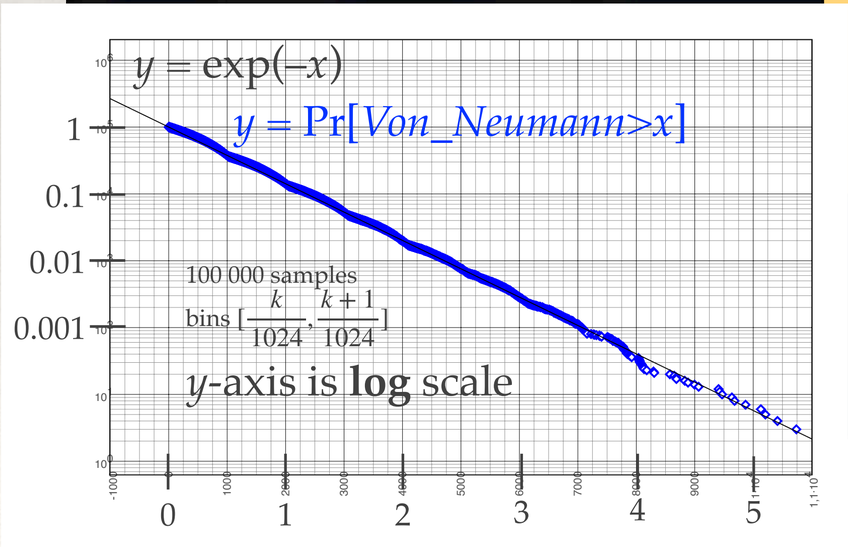
```
[goubault@macbook-pro-de-jean T
GimML comes with ABSOLUTELY NO
GimML for Darwin 20.5.0, by Je
[> use "ispcf1.ml"
>
```

longest_decreasing_run x 0 =
length k of longest decreasing prefix
$x = x_0 > x_1 > \cdots > x_k$
where $x_1, \ldots, x_k, \ldots$ are drawn in [0,1]
i.i.d. uniformly

$$\begin{aligned}
\textit{longest_decreasing_run} \mathrel{\hat{=}} \mathbf{rec}(\lambda f_{\mathbf{real}\to\mathbf{int}\to\mathbf{T\ int}} \cdot \lambda x_{\mathbf{real}} \cdot \lambda n_{\mathbf{int}} \cdot \\
&\mathbf{do}\ u = \mathbf{sample}[0,1]; \\
&\mathbf{if}\ u > x\ \mathbf{then\ ret}\ n \\
&\qquad \mathbf{else}\ f\ u\ (\mathbf{s}(n)))
\end{aligned}$$

$$\begin{aligned}
\textit{Von_Neumann} \mathrel{\hat{=}} \mathbf{do}\ x = \mathbf{sample}[0,1]; \\
&\mathbf{rec}(\lambda f_{\mathbf{real}\to\mathbf{T\ real}} \cdot \lambda \ell_{\mathbf{real}} \cdot \\
&\quad \mathbf{do}\ n = \textit{longest_decreasing_run}\ x\ 0; \\
&\quad \mathbf{if\ odd}\ n\ \mathbf{then}\ f(\ell \underline{+} \underline{1.0}) \\
&\qquad \mathbf{else\ ret}\ \ell)\ x
\end{aligned}$$

The probability that $[\![\mathbf{odd}\ n]\!]$ is true here is $\displaystyle\sum_{n \text{ odd}} \left(\frac{x^{n-1}}{(n-1)!} - \frac{x^n}{n!} \right) = \exp(-x)$



- … the exponential distribution $\exp(-x)\ d\lambda$ [von Neumann 49]

What does this compute?

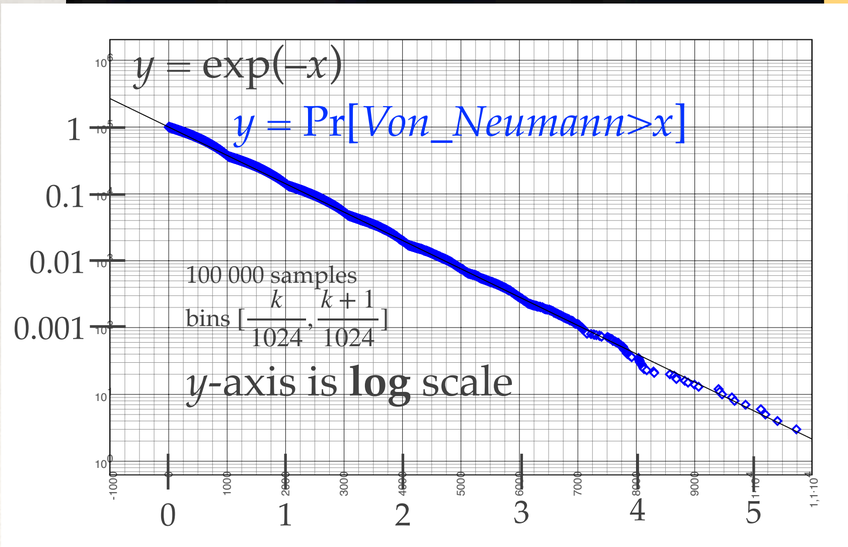
```
[goubault@macbook-pro-de-jean T
GimML comes with ABSOLUTELY NO
GimML for Darwin 20.5.0, by Je
[> use "ispcf1.ml"
>
```

longest_decreasing_run x 0 =
length k of longest decreasing prefix
$x = x_0 > x_1 > \cdots > x_k$
where $x_1, \ldots, x_k, \ldots$ are drawn in [0,1]
i.i.d. uniformly

$$\begin{aligned}
\textit{longest_decreasing_run} \mathrel{\hat{=}} \mathbf{rec}(\lambda f_{\mathbf{real}\to\mathbf{int}\to\mathbf{T\ int}} \cdot \lambda x_{\mathbf{real}} \cdot \lambda n_{\mathbf{int}} \cdot \\
&\mathbf{do}\ u = \mathbf{sample}[0,1]; \\
&\mathbf{if}\ u > x\ \mathbf{then\ ret}\ n \\
&\qquad \mathbf{else}\ f\ u\ (\mathbf{s}(n)))
\end{aligned}$$

$$\begin{aligned}
\textit{Von_Neumann} \mathrel{\hat{=}} \mathbf{do}\ x = \mathbf{sample}[0,1]; \\
&\mathbf{rec}(\lambda f_{\mathbf{real}\to\mathbf{T\ real}} \cdot \lambda \ell_{\mathbf{real}} \cdot \\
&\quad \mathbf{do}\ n = \textit{longest_decreasing_run}\ x\ 0; \\
&\quad \mathbf{if\ odd}\ n\ \mathbf{then}\ f(\ell \underline{+} \underline{1.0}) \\
&\qquad \mathbf{else\ ret}\ \ell)\ x
\end{aligned}$$

The probability that $[\![\mathbf{odd}\ n]\!]$ is true here is $\displaystyle\sum_{n \text{ odd}} \left(\frac{x^{n-1}}{(n-1)!} - \frac{x^n}{n!} \right) = \exp(-x)$



- … the exponential distribution $\exp(-x)\ d\lambda$ [von Neumann 49]

Soft conditioning

- A new primitive: **score** : **real** → **T unit**
- **score** $\underline{r}$ « multiplies probability
  of current execution branch by r »

Soft conditioning

- A new primitive: **score** : **real** → **T unit**
- **score** $\underline{r}$ « multiplies probability of current execution branch by r »

requires going beyond subprobability measures
e.g., consider **score** $\underline{3.0}$

Soft conditioning

- A new primitive: **score** : **real** → **T unit**
- **score** $\underline{r}$ « multiplies probability of current execution branch by r »

requires going beyond subprobability measures
e.g., consider **score** $\underline{3.0}$

One should then **normalize** the result
Smallish issue: normalization (= conditioning) is **not** computable [Ackermann, Freer, Roy 11]
(just do not normalize)

Soft conditioning

- A new primitive: **score** : **real** → **T unit**
- **score** $\underline{r}$ « multiplies probability of current execution branch by r »
- Conditioning on a property P:
  if $\neg P$ **then score** $\underline{0.0}$

requires going beyond subprobability measures
e.g., consider **score** $\underline{3.0}$

One should then **normalize** the result
Smallish issue: normalization (= conditioning) is **not** computable [Ackermann, Freer, Roy 11]
(just do not normalize)

Soft conditioning

- A new primitive: **score** : **real** → **T unit**
- **score** $\underline{r}$ « multiplies probability of current execution branch by r »
- Conditioning on a property P:
  if $\neg P$ **then score** $\underline{0.0}$
- **Soft** conditioning [Vákár, Kammar, Staton 21]: argument to **score** is a **density**
  (here, some Gaussian noise around the [formally incompatible] conditions $a = 1.1$, $2a = 1.9$, $3a = 2.7$)

requires going beyond subprobability measures
e.g., consider **score** $\underline{3.0}$

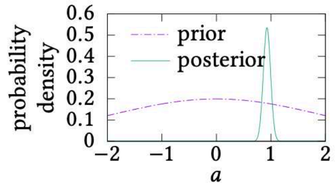
One should then **normalize** the result
Smallish issue: normalization (= conditioning) is **not** computable [Ackermann, Freer, Roy 11]
(just do not normalize)

let a = *normal-rng* $(0, 2)$ in
score(*normal-pdf*$(1.1 \mid a * 1, 0.25)$);
score(*normal-pdf*$(1.9 \mid a * 2, 0.25)$);
score(*normal-pdf*$(2.7 \mid a * 3, 0.25)$);
a

(a)

Prior: $a \sim N(0, 2)$
Observations:
$1.1 \sim N(1a, \frac{1}{4})$
$1.9 \sim N(2a, \frac{1}{4})$
$2.7 \sim N(3a, \frac{1}{4})$

(b)



(c)

Fig. 1. Bayesian linear regression as (a) a first-order probabilistic program, (b) an informal specification, and (c) a plot of the prior and posterior distributions. Here $N(\mu, \sigma)$ is the normal (Gaussian) distribution with density *normal-pdf*$(x \mid \mu, \sigma) = (2\pi\sigma^2)^{-\frac{1}{2}} e^{\frac{-(x-\mu)^2}{2\sigma^2}}$ and random number generator *normal-rng*(μ, σ).

The semantics of **score**

- **score** : **real** → **T unit**
- $[\![\mathbf{score}\ M]\!] \mathrel{\hat{=}} |[\![M]\!]| \,.\, \delta_*$,
  where $|\_|$ is largest continuous map : $\mathbf{I}\mathbb{R} \to \overline{\mathbb{R}}_+$
  below absolute value : $\mathbb{R} \to \overline{\mathbb{R}}_+$ on $\mathbb{R}$
  $|[a,b]| \mathrel{\hat{=}} a$ if $a > 0$, $-b$ if $b < 0$, 0 if $a \leq 0 \leq b$

The semantics of **score**

- **score** : **real** → **T unit**
- $[\![\mathbf{score}\ M]\!] \mathrel{\hat{=}} |[\![M]\!]| \,.\, \delta_*$,
  where $|\_|$ is largest continuous map : $\mathbf{I}\mathbb{R} \to \overline{\mathbb{R}}_+$
  below absolute value : $\mathbb{R} \to \overline{\mathbb{R}}_+$ on $\mathbb{R}$
  $|[a,b]| \mathrel{\hat{=}} a$ if $a > 0$, $-b$ if $b < 0$, 0 if $a \leq 0 \leq b$
- We must also redefine $[\![\mathbf{T}\ \tau]\!]$ as
  $\mathbf{V}([\![\tau]\!])$ = dcpo of **all** continuous valuations on $[\![\tau]\!]$
  (not just the subprobability valuations)

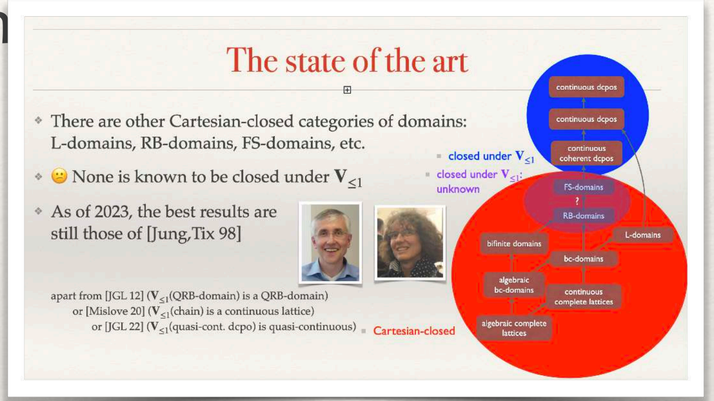
The semantics of **score**

- **score** : **real** $\to$ **T unit**
- $[\![\mathbf{score}\ M]\!] \mathrel{\hat{=}} |[\![M]\!]| \,.\, \delta_*$,
  where $|\_|$ is largest continuous map : $\mathbf{I}\mathbb{R} \to \overline{\mathbb{R}}_+$
  below absolute value : $\mathbb{R} \to \overline{\mathbb{R}}_+$ on $\mathbb{R}$
  $|[a,b]| \mathrel{\hat{=}} a$ if $a > 0$, $-b$ if $b < 0$, 0 if $a \leq 0 \leq b$
- We must also redefine $[\![\mathbf{T}\ \tau]\!]$ as
  $\mathbf{V}([\![\tau]\!])$ = dcpo of **all** continuous valuations on $[\![\tau]\!]$
  (not just the subprobability valuations)
- Soundness, adequacy… as usual [JGL, Jia 23]
  … but operational semantics uses **unbounded** distributions

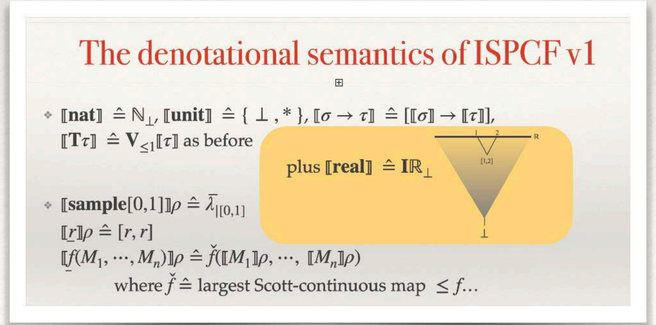
Commutativity

The catch

- I once told you that we needed at least **continuous** dcpos if we aim to prove a form of full abstraction
- … but that leads to difficult problems



- Then I showed you a semantics of ISPCF in (**non**-continuous) dcpos that had all sorts of nice properties
- The catch is that…



Commutativity

- The (rather silly) catch is that we cannot prove that

$$[\![\mathbf{do}\ x = M; \mathbf{do}\ y = N; P]\!] = [\![\mathbf{do}\ y = N; \mathbf{do}\ x = M; P]\!]$$

(*x* not free in *N*, *y* not free in *M*)

i.e., that you can draw *x* and *y* independently **in any order**

Commutativity

- The (rather silly) catch is that we cannot prove that

$$[\![\mathbf{do}\ x = M; \mathbf{do}\ y = N; P]\!] = [\![\mathbf{do}\ y = N; \mathbf{do}\ x = M; P]\!]$$

(*x* not free in *N*, *y* not free in *M*)

i.e., that you can draw *x* and *y* independently **in any order**

- although $\mathbf{do}\ x = M; \mathbf{do}\ y = N; P \quad \cong \quad \mathbf{do}\ y = N; \mathbf{do}\ x = M; P$

(this will be a consequence of what we will do next)

Commutativity

- The (rather silly) catch is that we cannot prove that

$$[\![\mathbf{do}\ x = M; \mathbf{do}\ y = N; P]\!] = [\![\mathbf{do}\ y = N; \mathbf{do}\ x = M; P]\!]$$

(*x* not free in *N*, *y* not free in *M*)

i.e., that you can draw *x* and *y* independently **in any order**

- although $\mathbf{do}\ x = M; \mathbf{do}\ y = N; P \quad \cong \quad \mathbf{do}\ y = N; \mathbf{do}\ x = M; P$

(this will be a consequence of what we will do next)

- I.e., the **V** monad on **Dcpo** is not known to be commutative

(but it is on **Top** and on **Cont**)

Commutativity

- The (rather silly) catch is that we cannot prove that

$$[\![\mathbf{do}\ x = M; \mathbf{do}\ y = N; P]\!] = [\![\mathbf{do}\ y = N; \mathbf{do}\ x = M; P]\!]$$

(*x* not free in *N*, *y* not free in *M*)

i.e., that you can draw *x* and *y* independently **in any order**

- although $\mathbf{do}\ x = M; \mathbf{do}\ y = N; P \quad \cong \quad \mathbf{do}\ y = N; \mathbf{do}\ x = M; P$

(this will be a consequence of what we will do next)

- I.e., the **V** monad on **Dcpo** is not known to be commutative

(but it is on **Top** and on **Cont**)

- I.e., the integral interchange (**Fubini-Tonelli**) theorem is not known to hold for continuous valuations on **Dcpo**

Commutativity

- The (rather silly) catch is that we cannot prove that

$$[\![\mathbf{do}\ x = M; \mathbf{do}\ y = N; P]\!] = [\![\mathbf{do}\ y = N; \mathbf{do}\ x = M; P]\!]$$

(*x* not free in *N*, *y* not free in *M*)

i.e., that you can draw *x* and *y* independently **in any order**

- although $\mathbf{do}\ x = M; \mathbf{do}\ y = N; P \quad \cong \quad \mathbf{do}\ y = N; \mathbf{do}\ x = M; P$

(this will be a consequence of what we will do next)

- I.e., the **V** monad on **Dcpo** is not known to be commutative

(but it is on **Top** and on **Cont**)

- I.e., the integral interchange (**Fubini-Tonelli**) theorem is not known to hold for continuous valuations on **Dcpo**

There is a very subtle issue here … as Fubini-Tonelli holds on the **larger** category **Top** 😖

Fubini-Tonelli for continuous valuations

- **Theorem [folklore; JGL, Xia 23].** Let X, Y be arbitrary topological spaces.

$$\int_x \left(\int_y h(x,y) d\nu \right) d\mu = \int_y \left(\int_x h(x,y) d\mu \right) d\nu$$

for every lower semicontinuous map $h \colon X \times Y \to \overline{\mathbb{R}}_+$

Fubini-Tonelli for continuous valuations

- **Theorem [folklore; JGL, Xia 23].** Let X, Y be arbitrary topological spaces.

$$\int_x \left(\int_y h(x,y) d\nu \right) d\mu = \int_y \left(\int_x h(x,y) d\mu \right) d\nu$$

for every lower semicontinuous map $h : X \times Y \to \overline{\mathbb{R}}_+$

Remember that
lower semicontinuous maps
from a dcpo to $\overline{\mathbb{R}}_+$
= Scott-continuous

Fubini-Tonelli for continuous valuations

- **Theorem [folklore; JGL, Xia 23].** Let X, Y be arbitrary topological spaces.

$$\int_x \left(\int_y h(x,y) d\nu \right) d\mu = \int_y \left(\int_x h(x,y) d\mu \right) d\nu$$

for every lower semicontinuous map $h \colon X \times Y \to \overline{\mathbb{R}}_+$

Remember that
lower semicontinuous maps
from a dcpo to $\overline{\mathbb{R}}_+$
= Scott-continuous

- **Unknown.** Let X, Y be arbitrary dcpos.

$$\int_x \left(\int_y h(x,y) d\nu \right) d\mu \stackrel{?}{=} \int_y \left(\int_x h(x,y) d\mu \right) d\nu$$

for every Scott-continuous map $h \colon X \times Y \to \overline{\mathbb{R}}_+$

Fubini-Tonelli for continuous valuations

- **Theorem [folklore; JGL, Xia 23].** Let X, Y be arbitrary topological spaces.

$$\int_x \left(\int_y h(x,y) d\nu \right) d\mu = \int_y \left(\int_x h(x,y) d\mu \right) d\nu$$

for every lower semicontinuous map $h \colon X \times Y \to \overline{\mathbb{R}}_+$

Remember that
lower semicontinuous maps
from a dcpo to $\overline{\mathbb{R}}_+$
= Scott-continuous

- **Unknown.** Let X, Y be arbitrary dcpos.

$$\int_x \left(\int_y h(x,y) d\nu \right) d\mu \stackrel{?}{=} \int_y \left(\int_x h(x,y) d\mu \right) d\nu$$

for every Scott-continuous map $h \colon X \times Y \to \overline{\mathbb{R}}_+$

The issue: products differ
in **Top** and in **Dcpo**
(the $\times$ symbol is ambiguous)

Fubini-Tonelli for continuous valuations

- **Theorem [folklore; JGL, Xia 23].** Let X, Y be arbitrary topological spaces.

$$\int_x \left(\int_y h(x,y)d\nu \right) d\mu = \int_y \left(\int_x h(x,y)d\mu \right) d\nu$$

for every lower semicontinuous map $h\colon X \times Y \to \overline{\mathbb{R}}_+$

in **Top**

Remember that lower semicontinuous maps from a dcpo to $\overline{\mathbb{R}}_+$ = Scott-continuous

- **Unknown.** Let X, Y be arbitrary dcpos.

$$\int_x \left(\int_y h(x,y)d\nu \right) d\mu =^? \int_y \left(\int_x h(x,y)d\mu \right) d\nu$$

for every Scott-continuous map $h\colon X \times Y \to \overline{\mathbb{R}}_+$

The issue: products differ in **Top** and in **Dcpo** (the $\times$ symbol is ambiguous)

Fubini-Tonelli for continuous valuations

- **Theorem [folklore; JGL, Xia 23].** Let X, Y be arbitrary topological spaces.

$$\int_x \left(\int_y h(x,y)d\nu \right) d\mu = \int_y \left(\int_x h(x,y)d\mu \right) d\nu$$

for every lower semicontinuous map $h\colon X \times Y \to \overline{\mathbb{R}}_+$

in **Top**

Remember that lower semicontinuous maps from a dcpo to $\overline{\mathbb{R}}_+$ = Scott-continuous

- **Unknown.** Let X, Y be arbitrary dcpos.

$$\int_x \left(\int_y h(x,y)d\nu \right) d\mu =^? \int_y \left(\int_x h(x,y)d\mu \right) d\nu$$

for every Scott-continuous map $h\colon X \times Y \to \overline{\mathbb{R}}_+$

in **Dcpo**

The issue: products differ in **Top** and in **Dcpo** (the $\times$ symbol is ambiguous)

Products in **Top**, products in **Dcpo**

- In **Top**: $X \times Y$ has the **product topology**,

  open sets = unions of **open rectangles** $U \times V$, $U \in \mathcal{O}X$, $V \in \mathcal{O}Y$

- In **Dcpo**: $X \times Y$ has the **Scott topology** of $\leq \times \leq$

$$
\begin{tikzcd}
\begin{array}{c}\text{dcpos}\\ X, Y\end{array} \arrow[r, "\begin{array}{c}\text{take Scott}\\ \text{topologies}\end{array}"] \arrow[d, "\begin{array}{c}\text{form}\\ \text{product}\\ \text{in } \mathbf{Dcpo}\end{array}"'] & \begin{array}{c}\text{spaces}\\ X_\sigma, Y_\sigma\end{array} \arrow[d, "\begin{array}{c}\text{form}\\ \text{product}\\ \text{in } \mathbf{Top}\end{array}"] \\
\begin{array}{c}\text{dcpo}\\ X \times Y\end{array} \arrow[r, "\begin{array}{c}\text{take Scott}\\ \text{topology}\end{array}"'] & \begin{array}{c}X_\sigma \times Y_\sigma\\ (X \times Y)_\sigma\end{array}
\end{tikzcd}
$$

Products in **Top**, products in **Dcpo**

- In **Top**: $X \times Y$ has the **product topology**,
  open sets = unions of **open rectangles** $U \times V$, $U \in \mathcal{O}X$, $V \in \mathcal{O}Y$
- In **Dcpo**: $X \times Y$ has the **Scott topology** of $\leq \times \leq$

$$
\begin{tikzcd}
\begin{array}{c}\text{dcpos}\\ X, Y\end{array} \arrow[r, "\text{take Scott topologies}"] \arrow[d, "\text{form product in } \mathbf{Dcpo}"'] & \begin{array}{c}\text{spaces}\\ X_\sigma, Y_\sigma\end{array} \arrow[d, "\text{form product in } \mathbf{Top}"] \\
\begin{array}{c}\text{dcpo}\\ X \times Y\end{array} \arrow[r, "\text{take Scott topology}"'] & \begin{array}{c}X_\sigma \times Y_\sigma\\ (X \times Y)_\sigma\end{array}
\end{tikzcd}
$$

Note: $X_\sigma \times Y_\sigma = (X \times Y)_\sigma$
if X or Y **continuous dcpo**

Products in **Top**, products in **Dcpo**

- In **Top**: $X \times Y$ has the **product topology**,
  open sets = unions of **open rectangles** $U \times V$, $U \in \mathcal{O}X$, $V \in \mathcal{O}Y$
- In **Dcpo**: $X \times Y$ has the **Scott topology** of $\leq \times \leq$

$$
\begin{tikzcd}
\begin{array}{c}\text{dcpos}\\ X, Y\end{array} \arrow[r, "\text{take Scott topologies}"] \arrow[d, "\text{form product in } \mathbf{Dcpo}"'] & \begin{array}{c}\text{spaces}\\ X_\sigma, Y_\sigma\end{array} \arrow[d, "\text{form product in } \mathbf{Top}"] \\
\begin{array}{c}\text{dcpo}\\ X \times Y\end{array} \arrow[r, "\text{take Scott topology}"'] & \begin{array}{c}X_\sigma \times Y_\sigma\\ (X \times Y)_\sigma\end{array}
\end{tikzcd}
$$

Note: $X_\sigma \times Y_\sigma = (X \times Y)_\sigma$
if X or Y **continuous dcpo**

Products in **Top**, products in **Dcpo**

- In **Top**: $X \times Y$ has the **product topology**,
  open sets = unions of **open rectangles** $U \times V$, $U \in \mathcal{O}X$, $V \in \mathcal{O}Y$
- In **Dcpo**: $X \times Y$ has the **Scott topology** of $\leq \times \leq$

$$
\begin{tikzcd}
\text{dcpos } X, Y \arrow[r, "\text{take Scott topologies}"] \arrow[d, "\text{form product in } \mathbf{Dcpo}"'] & \text{spaces } X_\sigma, Y_\sigma \arrow[d, "\text{form product in } \mathbf{Top}"] \\
\text{dcpo } X \times Y \arrow[r, "\text{take Scott topology}"'] & X_\sigma \times Y_\sigma \\
 & (X \times Y)_\sigma
\end{tikzcd}
$$

Note: $X_\sigma \times Y_\sigma = (X \times Y)_\sigma$
if X or Y **continuous dcpo**

More generally:
— if X_σ or Y_σ is core-compact [Gierz,Hofmann,Keimel,Lawson,Mislove 03]
— if X_σ and Y_σ are first-countable [de Brecht, priv. comm., 19]

Minimal valuations

- Let $\mathbf{V}_{\mathrm{fin}}X$ be the subset of $\mathbf{V}X$ consisting of simple valuations
- The smallest subdcpo $\mathbf{M}X$ of $\mathbf{V}X$ containing $\mathbf{V}_{\mathrm{fin}}X$ is the dcpo of **minimal valuations**

Minimal valuations

- Let $\mathbf{V}_{\mathrm{fin}}X$ be the subset of $\mathbf{V}X$ consisting of simple valuations
- The smallest subdcpo $\mathbf{M}X$ of $\mathbf{V}X$ containing $\mathbf{V}_{\mathrm{fin}}X$
  is the dcpo of **minimal valuations**
- Explicitly, a minimal valuation is
  a directed supremum of directed suprema of … of simple valuations
  (iterated transfinitely)

Minimal valuations

- Let $\mathbf{V}_{\mathrm{fin}}X$ be the subset of $\mathbf{V}X$ consisting of simple valuations
- The smallest subdcpo $\mathbf{M}X$ of $\mathbf{V}X$ containing $\mathbf{V}_{\mathrm{fin}}X$
  is the dcpo of **minimal valuations**
- Explicitly, a minimal valuation is
  a directed supremum of directed suprema of … of simple valuations
  (iterated transfinitely)
- **Prop [Jia,Lindenhovius,Mislove,Zamdzhiev 21; JGL, Jia 23].**
  Fubini-Tonelli holds on **Dcpo** if one of the valuations is minimal.
- *Proof sketch*: Integration commutes with directed suprema.
  This reduces the question to the case of simple valuations,
  where commutation is easy.

ISPCF, v2

- $[\![\mathbf{nat}]\!] \triangleq \mathbb{N}_\perp$, $[\![\mathbf{unit}]\!] \triangleq \{\perp, *\}$, $[\![\sigma \to \tau]\!] \triangleq [[\![\sigma]\!] \to [\![\tau]\!]]$, $[\![\mathbf{real}]\!] \triangleq \mathbf{I}\mathbb{R}_\perp$

  $[\![\mathbf{T}\tau]\!] \triangleq \mathbf{M}[\![\tau]\!]$

  Instead of $\mathbf{V}[\![\tau]\!]$
  (this is the only change!)

- $[\![\mathbf{sample}[0,1]]\!]\rho \triangleq \bar{\lambda}_{|[0,1]}$

  $[\![\underline{r}]\!]\rho \triangleq [r, r]$ $\qquad$ $[\![\underline{f}(M_1, \cdots, M_n)]\!]\rho \triangleq \check{f}([\![M_1]\!]\rho, \cdots, [\![M_n]\!]\rho)$

ISPCF, v2

- $[\![\mathbf{nat}]\!] \triangleq \mathbb{N}_\perp$, $[\![\mathbf{unit}]\!] \triangleq \{\perp, *\}$, $[\![\sigma \to \tau]\!] \triangleq [[\![\sigma]\!] \to [\![\tau]\!]]$, $[\![\mathbf{real}]\!] \triangleq \mathbf{I}\mathbb{R}_\perp$

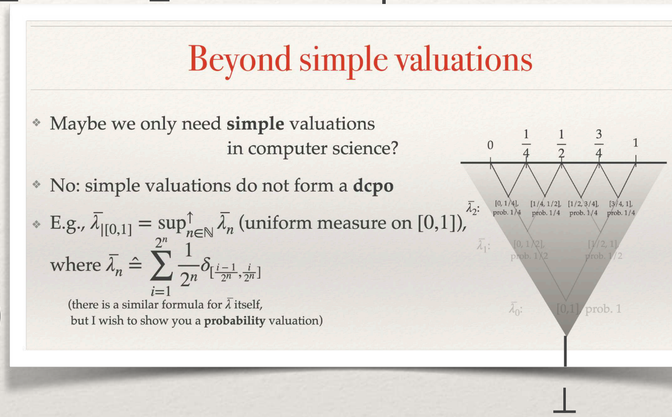
  $[\![\mathbf{T}\tau]\!] \triangleq \mathbf{M}[\![\tau]\!]$

Instead of $\mathbf{V}[\![\tau]\!]$
(this is the only change!)

- $[\![\mathbf{sample}[0,1]]\!]\rho \triangleq \bar{\lambda}_{|[0,1]}$

That **is** a minimal valuation

  $[\![\underline{r}]\!]\rho \triangleq [r, r]$ $\qquad$ $[\![\underline{f}(M_1, \cdots, M_n)]\!]\rho \triangleq f([\![M_1]\!]\rho, \cdots, [\![M_n]\!]\rho)$



ISPCF, v2

- $[\![\mathbf{nat}]\!] \triangleq \mathbb{N}_\perp$, $[\![\mathbf{unit}]\!] \triangleq \{\perp, *\}$, $[\![\sigma \to \tau]\!] \triangleq [[\![\sigma]\!] \to [\![\tau]\!]]$, $[\![\mathbf{real}]\!] \triangleq \mathbf{I}\mathbb{R}_\perp$

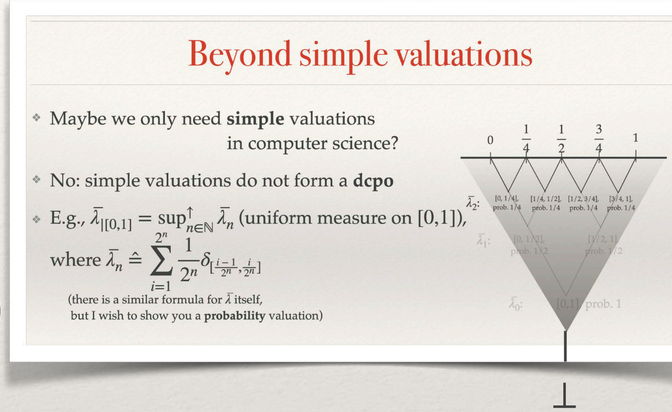
  $[\![\mathbf{T}\tau]\!] \triangleq \mathbf{M}[\![\tau]\!]$

Instead of $\mathbf{V}[\![\tau]\!]$
(this is the only change!)

- $[\![\mathbf{sample}[0,1]]\!]\rho \triangleq \bar{\lambda}_{|[0,1]}$

That **is** a minimal valuation

  $[\![\underline{r}]\!]\rho \triangleq [r, r]$ $\qquad$ $[\![\underline{f}(M_1, \cdots, M_n)]\!]\rho \triangleq f([\![M_1]\!]\rho, \cdots, [\![M_n]\!]\rho)$



- **Theorem (soundness, adequacy).**
  $\Pr[C, M \downarrow n] = [\![C[M]]\!]\rho$
  (at type **nat**)

Operational semantics
is **unchanged**

ISPCF, v2

- $[\![\mathbf{nat}]\!] \triangleq \mathbb{N}_\perp$, $[\![\mathbf{unit}]\!] \triangleq \{\perp, *\}$, $[\![\sigma \to \tau]\!] \triangleq [[\![\sigma]\!] \to [\![\tau]\!]]$, $[\![\mathbf{real}]\!] \triangleq \mathbf{I}\mathbb{R}_\perp$

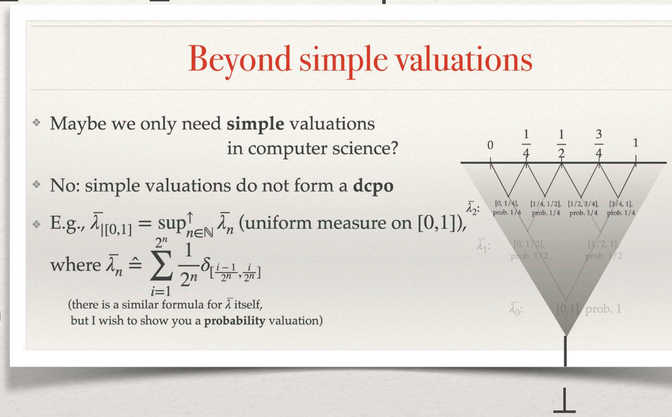
  $[\![\mathbf{T}\tau]\!] \triangleq \mathbf{M}[\![\tau]\!]$

Instead of $\mathbf{V}[\![\tau]\!]$
(this is the only change!)

- $[\![\mathbf{sample}[0,1]]\!]\rho \triangleq \bar{\lambda}_{|[0,1]}$

That **is** a minimal valuation

  $[\![\underline{r}]\!]\rho \triangleq [r, r]$ $\qquad$ $[\![\underline{f}(M_1, \cdots, M_n)]\!]\rho \triangleq f([\![M_1]\!]\rho, \cdots, [\![M_n]\!]\rho)$



- **Theorem (soundness, adequacy).**
  $\Pr[C, M \downarrow n] = [\![C[M]]\!]\rho$
  (at type **nat**)

Operational semantics
is **unchanged**

- $[\![\mathbf{do}\ x = M; \mathbf{do}\ y = N; P]\!]$
  $= [\![\mathbf{do}\ y = N; \mathbf{do}\ x = M; P]\!]$
  (x not free in N, y not free in M)

ISPCF, v2

- $[\![\mathbf{nat}]\!] \triangleq \mathbb{N}_\perp$, $[\![\mathbf{unit}]\!] \triangleq \{\perp, *\}$, $[\![\sigma \to \tau]\!] \triangleq [[\![\sigma]\!] \to [\![\tau]\!]]$, $[\![\mathbf{real}]\!] \triangleq \mathbf{I}\mathbb{R}_\perp$

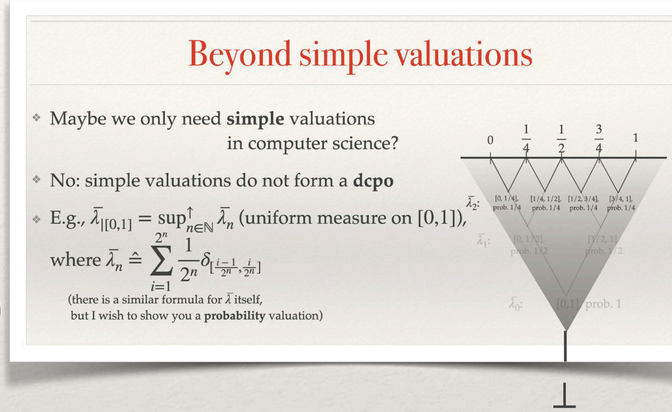
  $[\![\mathbf{T}\tau]\!] \triangleq \mathbf{M}[\![\tau]\!]$

Instead of $\mathbf{V}[\![\tau]\!]$
(this is the only change!)

- $[\![\mathbf{sample}[0,1]]\!]\rho \triangleq \bar{\lambda}_{|[0,1]}$

That **is** a minimal valuation

  $[\![\underline{r}]\!]\rho \triangleq [r, r]$ $\qquad$ $[\![\underline{f}(M_1, \cdots, M_n)]\!]\rho \triangleq f([\![M_1]\!]\rho, \cdots, [\![M_n]\!]\rho)$



- **Theorem (soundness, adequacy).**
  $\Pr[C, M \downarrow n] = [\![C[M]]\!]\rho$
  (at type **nat**)

Operational semantics
is **unchanged**

- $[\![\mathbf{do}\ x = M; \mathbf{do}\ y = N; P]\!]$
  $= [\![\mathbf{do}\ y = N; \mathbf{do}\ x = M; P]\!]$
  (x not free in N, y not free in M)

Problem **solved**.
In particular, $\mathbf{do}\ x = M; \mathbf{do}\ y = N; P$
$\cong$ $\mathbf{do}\ y = N; \mathbf{do}\ x = M; P$

Hierarchies of continuous valuations

Simple valuations

$$\sum_{i=1}^{n} a_i \delta_{x_i}$$

$\subseteq$ Minimal valuations

dcpo-closure of simple valuations

$\subseteq$ Point-continuous valuations

sobrification of simple valuations [Heckmann 97]

$\subseteq$ Central valuations

center of the **V** monad on **Dcpo** [Jia, Mislove, Zamdzhiev 21]

$\subseteq$ Continuous valuations

[SahebDjahromi 80, Jones 90]

All those satisfy Fubini-Tonelli on **Dcpo**

(they form commutative monads on **Dcpo**)

(Fubini-Tonelli? unknown)

Hierarchies of continuous valuations

Simple valuations

$$\sum_{i=1}^{n} a_i \delta_{x_i}$$

$\subseteq$ Minimal valuations

dcpo-closure of simple valuations

$\subseteq$ Point-continuous valuations

sobrification of simple valuations [Heckmann 97]

$\subseteq$ Central valuations

center of the **V** monad on **Dcpo** [Jia, Mislove, Zamdzhiev 21]

$\subseteq$ Continuous valuations

[SahebDjahromi 80, Jones 90]

All those satisfy Fubini-Tonelli on **Dcpo**

(they form commutative monads on **Dcpo**)

(Fubini-Tonelli? unknown)

- Is any of those inclusions **strict**?
- If every continuous valuation on a dcpo is minimal/point-continuous/central, then we would obtain Fubini-Tonelli on **Dcpo**.

Hierarchies of continuous valuations

Simple valuations

$$\sum_{i=1}^{n} a_i \delta_{x_i}$$

$\subsetneq$

Minimal valuations

dcpo-closure of simple valuations

$\subseteq$

Point-continuous valuations

sobrification of simple valuations [Heckmann 97]

$\subseteq$

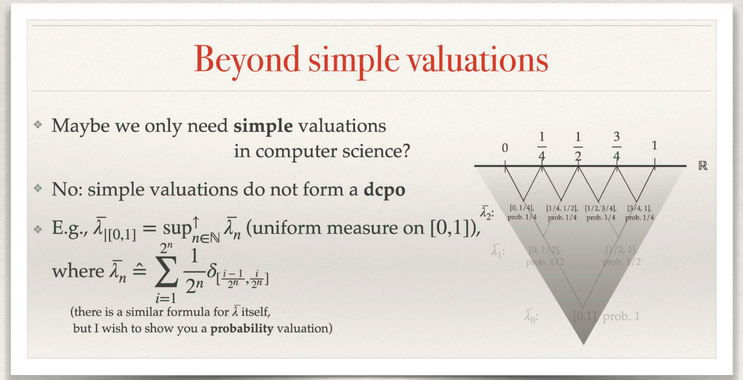
Central valuations

center of the **V** monad on **Dcpo** [Jia, Mislove, Zamdzhiev 21]

$\subseteq$

Continuous valuations

[SahebDjahromi 80, Jones 90]



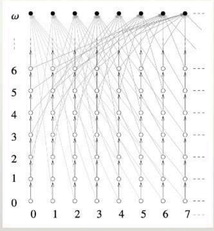
Hierarchies of continuous valuations

Simple valuations $\sum_{i=1}^{n} a_i \delta_{x_i}$ $\subsetneq$ Minimal valuations (dcpo-closure of simple valuations) $\subsetneq$ Point-continuous valuations (sobrification of simple valuations [Heckmann 97]) $\subseteq$ Central valuations (center of the **V** monad) $\subseteq$ Continuous valuations [SahebDjahromi 80]

- **Theorem [JGL, Jia 21].** On Johnstone's dcpo **J**,
  — every continuous valuation is point-continuous
  — there is a continuous valuation μ that is **not minimal**.
- *Proof:* pretty elaborate, uses that every bounded discrete valuation extends to a measure [Alvarez-Manilla, Edalat, Saheb-Djahromi 00], plus a detailed analysis of valuations of the form $\theta + r \,.\, \mu$ with θ bounded discrete. □

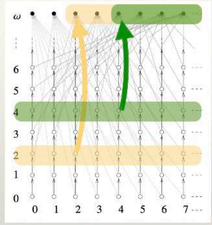
The Johnstone dcpo J

- Johnstone's dcpo **J** (1981): a well-known non-sober dcpo
  — Points = pairs (m, n) in $\mathbb{N} \times (\mathbb{N} \cup \{\omega\})$
  — $(m, n) \leq (m', n')$ iff
    — $m = m'$ and $n \leq n'$
    — or $n \leq m'$ and $n' = \omega$



A funny valuation on J

- On Johnstone's dcpo **J**, there is a continuous valuation μ defined by:
  $\mu(U) = 1$ for every non-empty Scott-open set U
  $\mu(\emptyset) = 0$
- Modularity $\mu(U \cup V) + \mu(U \cap V) = \mu(U) + \mu(V)$ comes from the fact that $\mathbf{J}_\sigma$ is **hyperconnected**:
  any two non-empty open sets intersect.
  (Check it! Observe that every non-empty open set contains all points (m, ω) for m large enough.)
- We will show that μ is not minimal.



Hierarchies of continuous valuations

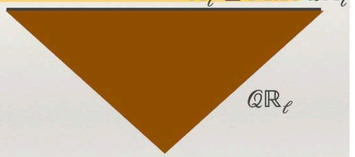
The Sorgenfrey line $\mathbb{R}_\ell$

- A famous counterexample in topology:
  Sorgenfrey topology on $\mathbb{R}$ generated by basic open sets $[a, b[$
  (topology of convergence from the right)

Nice	Not so nice
paracompact, T_4	product with itself **not** T_4
zero-dimensional	**not** locally compact
Choquet-complete, hence Baire	**not** consonant
first-countable, with countable dense subset ($\mathbb{Q}$)	
hereditarily Lindelöf	**not** second-countable
completely quasi-metrizable	**not** metrizable

The Smyth powerdomain of $\mathbb{R}_\ell$

- **Proposition.** $\mathcal{Q}\mathbb{R}_\ell$ is a dcpo model of $\mathbb{R}_\ell$:
  through $x \mapsto \{x\}$, $\mathbb{R}_\ell$ embeds as a **topological subspace** of $\mathcal{Q}\mathbb{R}_\ell$.
- Explicitly, the open subsets of $\mathbb{R}_\ell$
  are the intersections $\mathcal{U} \cap \mathbb{R}_\ell$
  where $\mathcal{U}$ is Scott-open in $\mathcal{Q}\mathbb{R}_\ell$
- The Lebesgue valuation λ on $\mathbb{R}_\ell$ induces
  a continuous valuation $\bar{\lambda}$ on $\mathcal{Q}\mathbb{R}_\ell$ by
  $\bar{\lambda}(\mathcal{U}) = \lambda(\mathcal{U} \cap \mathbb{R}_\ell)$
  (the image continuous valuation through the embedding $\mathbb{R}_\ell \to \mathcal{Q}\mathbb{R}_\ell$)



Point-continuous valuations
sobrification
of simple valuations
[Heckmann 97]

$\subseteq$

Central valuations
center of the **V** monad
on **Dcpo**
[Jia, Mislove, Zamdzhiev 21]

$\subsetneq$

Continuous valuations
[SahebDjahromi 80, Jones 90]

Theorem [JGL, Jia 21]. On the dcpo $\mathcal{Q}(\mathbb{R}_\ell)$ $\bar{\lambda}$ is **not point-continuous**.

Proof: pretty involved, uses Adamski's theorem, results by [He, Li, Xi, Zhao 19] on the Smyth powerdomain of first-countable spaces, and a complete characterization of the compact subsets of the Sorgenfrey line $\mathbb{R}_\ell$. $\square$

Concluding remarks

Conclusion, past, and future

- A short tour of **semantics** of probabilistic programming languages
- This is just the tip of the iceberg…
  in fact I have barely touched the subject of **topology**
  — and only through domain theory
- Let me quickly tell you some of the other things…

Full abstraction

- Probabilistic PCF + (angelic) **non-determinism**
  + statistical termination testers $[V > r]$
  is **fully abstract** [JGL 15]

- Denotationally, redefine $[\![\mathbf{T}\tau]\!]$ as $\mathbf{P}_{\mathrm{A}}[\![\tau]\!]$,
  domain of **Hoare previsions** on $[\![\tau]\!]$ [JGL 07]

- The key to full abstraction is theorems on
  coincidence of topologies
  [JGL 15]

$M \precsim^{\mathrm{may}} N$ *iff* $[\![M]\!]_S \leq [\![N]\!]_S$, *at all types*.

Definition

A *continuous upper prevision* F is a functional from $\langle X \to \mathbb{R}^+\rangle$ to $\mathbb{R}^+$ such that:

- F is (Sco[illegible]$_{Hoare}$ tinuous: $F(\sup^{\uparrow}_{i\in I} h_i) = \sup^{\uparrow}_{i\in I} F(h_i)$;
- F is positively homogeneous: $F(ah) = aF(h)$ $(a \in \mathbb{R}^+)$;
- F is convex: $F(h + h') \leq F(h) + F(h')$.

This models probabilistic + angelic non-determinism

(Exercise: Check that sup is indeed upper.)

Fortunately:

Lemma

For every type τ, $[\![\tau]\!]_S$ is a bc-domain.

(One of the nice CCCs of continuous domains.)

Proposition (Key result — *coincidence of topologies*)

If X and Y are bc-domains, then:

- *Scott topology on $[X \to Y]$ = pointwise convergence*
  Subbasis: $[a \in V] = \{f \mid f(a) \in V\}$, $a \in X$, V open in Y
- *Scott topology on previsions on X = weak topology*
  Subbasis: $[h > r] = \{F \mid F(h) > r\}$, $h \in [X \to I]$, $r \in \mathbb{Q}$

Previsions

- For a continuous valuation ν, $h \mapsto \int_X h d\nu$ is continuous and **linear**

  This is in fact an isomorphism $\mathbf{V}X \cong [[X \to \overline{\mathbb{R}}_+] \to_{\mathrm{lin}} \overline{\mathbb{R}}_+]$

- **Previsions** relax the linearity requirement

  e.g., Hoare previsions are **sublinear**: $F(h_1 + h_2) \leq F(h_1) + F(h_2)$ (and $F(r \,.\, h) = r \,.\, F(h)$)

- **Isomorphisms** between spaces of previsions on X

  and convex hyperspaces over $\mathbf{V}X$,

  (as used by [Mislove 00], [McIver, Morgan 01], [Tix 99; Tix, Keimel, Plotkin 09])

  [JGL 08]: on continuous dcpos

  [JGL 17]: on large families of **topological spaces**

A few open questions

- Full abstraction for ISPCF+angelic non-determinism+statistical testers? [Yes]
  — any connection between statistical testers and conditioning?
- Fubini-Tonelli on **Dcpo**?
  — equivalently, is every continuous valuation central on **Dcpo**?
- Can the Jung-Tix dream be realized?
  — a Cartesian-closed category of continuous dcpos, closed under **V**?
- Can we trust **pseudo-random** number generators
  instead of true random number generators
  in the implementation of **sample**[0,1]?