Dexter Kozen's Influence on the Theory of Markov Processes

Prakash Panangaden

Ithaca 1985

- Department of Computer Science hired two new profs that year:
- Dexter Kozen and me.
- Dexter was famous for the so-called BKR algorithm but

Semantics and Logic

- also for modal logic (PDL)
- mu-calculus and
- probabilistic programs.

Two key papers

- Semantics of probabilistic programs, JCSS, 1981
- A probabilistic PDL, JCSS, 1985.

New ideas

- Take probability theory on continuous state spaces seriously.
- Hence, work with sigma-algebras, measure theory and integration.
- A new Stone-type duality.

What is Stone duality?

- Stone representation theorem: every boolean algebra is isomorphic to a concrete boolean algebra of sets.
- But we know "elements are a hack!"

- If we look at all the boolean algebras and the maps between them then
- it looks exactly like a certain collection of topological spaces and the maps between them
- going backwards!

- This kind of thing happens several times in mathematics:
- C^* algebras and compact Hausdorff spaces (Gelfand duality)
- Finite-dimensional vector spaces and themselves (self duality)

- Happens in computer science too:
- We can define what a program does by going forwards: St x Act --> St
- or backwards:
- precondition <-- predicate x Act.

- Dexter found that this works for probabilistic programs too:
- Forwards semantics: measure transformers (Markov kernels)
- Backward semantics: transformers of random variables (value functions).

What is a probabilistic relation?

What is a probabilistic relation?

Answer: Markov kernels!

What is a probabilistic relation?

Answer: Markov kernels!

$k(x, A)$: conditional probability of landing in a set A if one starts at x.

What is a probabilistic relation?

Answer: Markov kernels!

$k(x, A)$: conditional probability of landing in a set A if one starts at x.

Crucial for dealing with continuous state spaces.

What is a probabilistic relation?

Answer: Markov kernels!

$k(x, A)$: conditional probability of landing in a set A if one starts at x.

Crucial for dealing with continuous state spaces.

$k(x, \cdot)$: a measure and $k(\cdot, A)$ is a measurable function.

What is a probabilistic relation?

Answer: Markov kernels!

$k(x, A)$: conditional probability of landing in a set A if one starts at x.

Crucial for dealing with continuous state spaces.

$k(x, \cdot)$: a measure and $k(\cdot, A)$ is a measurable function.

What does this have to do with relations?

How does one compose relations?

How does one compose relations?

$$x(R \circ S)y = \exists z(xRz \wedge zSy).$$

How does one compose relations?

$$x(R \circ S)y = \exists z(xRz \wedge zSy).$$

If we try to think of a "probabilistic relation" as a joint measure on the product space we cannot compose them.

How does one compose relations?

$$x(R \circ S)y = \exists z(xRz \wedge zSy).$$

If we try to think of a "probabilistic relation" as a joint measure on the product space we cannot compose them.

But Markov kernels are just the ticket!

How does one compose relations?

$$x(R \circ S)y = \exists z(xRz \wedge zSy).$$

If we try to think of a "probabilistic relation" as a joint measure on the product space we cannot compose them.

But Markov kernels are just the ticket!

$h : X \to Y$ and $k : Y \to Z$

How does one compose relations?

$$x(R \circ S)y = \exists z(xRz \wedge zSy).$$

If we try to think of a "probabilistic relation" as a joint measure on the product space we cannot compose them.

But Markov kernels are just the ticket!

$$h : X \to Y \text{ and } k : Y \to Z$$

$$(k \circ h)(x, C) = \int k(y, C)h(x, \mathrm{d}y).$$

How does one compose relations?

$$x(R \circ S)y = \exists z(xRz \wedge zSy).$$

If we try to think of a "probabilistic relation" as a joint measure on the product space we cannot compose them.

But Markov kernels are just the ticket!

$$h : X \to Y \text{ and } k : Y \to Z$$

$$(k \circ h)(x, C) = \int k(y, C)h(x, \mathrm{d}y).$$

The category of Markov processes (1999), ENTCS, P.

Dexter's Probabilistic Language

$$S ::== x_i := f(\vec{x}) | S_1; S_2 | if\ \mathbf{B}\ then\ S_1\ else\ S_2 | while\ \mathbf{B}\ do\ S.$$

Dexter's Probabilistic Language

$$S ::== x_i := f(\vec{x}) | S_1; S_2 | if\ \mathbf{B}\ then\ S_1\ else\ S_2 | while\ \mathbf{B}\ do\ S.$$

Here f could be assigning a value randomly according to some distribution.

Dexter's Probabilistic Language

$$S ::== x_i := f(\vec{x}) | S_1 ; S_2 | if\ \mathbf{B}\ then\ S_1\ else\ S_2 | while\ \mathbf{B}\ do\ S.$$

Here f could be assigning a value randomly according to some distribution.

Dexter gave measure transformer semantics in terms of Markov kernels.

Dexter's Probabilistic Language

$$S ::== x_i := f(\vec{x}) | S_1; S_2 | if\ \mathbf{B}\ then\ S_1\ else\ S_2 | while\ \mathbf{B}\ do\ S.$$

Here f could be assigning a value randomly according to some distribution.

Dexter gave measure transformer semantics in terms of Markov kernels.

He also gave *backward semantics* in terms of how random variables transform.

Suppose that we want to describe the effect of `x := 1729`.

Suppose that we want to describe the effect of `x := 1729`.

Forwards: modify the description of the state so that `x` is now bound to 1729.

Suppose that we want to describe the effect of `x := 1729`.

Forwards: modify the description of the state so that `x` is now bound to 1729.

Backwards: If P is true *after* the command then P with every x replaced by 1729 must have been true *before*.

Suppose that we want to describe the effect of `x := 1729`.

Forwards: modify the description of the state so that `x` is now bound to 1729.

Backwards: If P is true *after* the command then P with every x replaced by 1729 must have been true *before*.

In Dexter's probabilistic setting:

Suppose that we want to describe the effect of `x := 1729`.

Forwards: modify the description of the state so that `x` is now bound to 1729.

Backwards: If P is true *after* the command then P with every x replaced by 1729 must have been true *before*.

In Dexter's probabilistic setting:

If f is the value of a random variable after a command described by a Markov kernel h, then $\int f(x')h(x, dx')$ is the value before.

In terms of categories

In terms of categories

The category of Markov kernels is **SRel**.

In terms of categories

The category of Markov kernels is **SRel**.

The category of probabilistic predicate transformers is called **SPT**.

In terms of categories

The category of Markov kernels is **SRel**.

The category of probabilistic predicate transformers is called **SPT**.

Duality: $\mathbf{SRel} \equiv \mathbf{SPT}^{op}$.

Why the categorical language?

After all, dealing with arrows can be painful!

Why the categorical language?

After all, dealing with arrows can be painful!

I was greatly influenced by a line from the famous song by Dexter: *Categories Uber Alles.*

I was greatly influenced by a line from the famous song by Dexter: *Categories Uber Alles.*

We know that elements are just a hack.

In the late 1990s I, working with Josée Desharnais, Abbas Edalat, Radha Jagadeesan and Vineet Gupta were developing the theory of Labelled Markov Processes (MDPs).

A key concept was *bisimulation*: when do two processes behave *exactly* the same?

Logical characterization: when they agree on all the formulas of a simple (modal) logic.

But equivalences are suspect, one should work with metrics.

Intuition: processes are *close* if the *shortest* formula distinguishing them is big.

Intuition: processes are *close* if the *shortest* formula distinguishing them is big.

How does one make this precise?

The Kozen Analogy

Logic	Probability
State s	Distribution μ
Predicate ϕ	Random variable f
Satisfaction $s \models \phi$	Pairing $\int f \mathrm{d}\mu$

We defined a class of functional expressions $\mathcal{F}$ and interpreted them just like a logic.

We defined a class of functional expressions $\mathcal{F}$ and interpreted them just like a logic.

$$d(s, s') = \sup_{f \in \mathcal{F}} |f(s) - f(s')|.$$

We defined a class of functional expressions $\mathcal{F}$ and interpreted them just like a logic.

$$d(s, s') = \sup_{f \in \mathcal{F}} |f(s) - f(s')|.$$

The study of behavioural metrics is now flourishing with many papers every year in concurrency theory and also in machine learning and algorithms.

But it all started with a
random walk
through
Markov processes.

Happy Birthday Dexter!