Structure in Machine Learning

Prakash Panangaden

27th June 2022

Abstract

I give some examples of instances of ideas from semantics being useful in machine learning.

1 Introduction

I am delighted to be able to contribute this short piece to celebrate yet another milestone in Samson Abramsky's illustrious career. Samson is known for many things and I don't want to compile another list of accomplishments here. Rather, I would like to focus on one issue where he has been a forceful advocate: bringing together the two wings of theoretical computer science. These are often referred to as Theory A (algorithms, complexity, combinatorics) and Theory B (logic, semantics, verification). There are some points of contact between the two, finite model theory is a notable example, and some areas that have drifted from one wing to the other, perhaps automata theory is an example. However, culturally and socially, there is a big gap as can be seen at once from a glance at the accepted papers at STOC, FOCS, LICS, POPL and other conferences.

Samson was one of the prime movers in the Simons Institute thematic semester in the Autumn of 2016 where, along with Anuj Dawar and Phokion Kolaitis, we organised a programme specifically aimed at bringing the two branches of theoretical computer science together. Apart from these organisational initiatives he took a leading rôle in pushing research topics which resulted in papers that gave a structural account of important topics in combinatorics [ADW17, AW17, AS18, ABKM19].

Another theme that has been pursued by Samson and his collaborators is the use of topological ideas in explaining features of quantum mechan-

ics [AMB11, AB11]. Thus, one understands contextuality as a manifestation of the nonexistence of global sections of an appropriate sheaf, ideas like had appeared earlier [IB98], but the paper by Abramsky and Brandenberger [AB11] made this particularly clear in a simple setting. It turned out that this viewpoint could be brought to bear in other areas like logic or the theory of relational databases [Abr14, ABK$^+$15]. While topology is not geometry, it does lead to visual thinking and intuition.

The most important influence on the present article is a recent paper called "Whither[1] semantics" [Abr20] which appeared in a tribute volume to Maurice Nivat. This paper is essentially a call to arms to those of us working with structural approaches to use our methods to attack "hard" problems: here "hard" means concrete quantitative problems. I am not going to summarise that paper here but instead I will take up the challenge and talk about structural methods in machine learning.

The siren song of machine learning has been seducing people from all sorts of intellectual disciplines: probability theory, statistics, optimization, convexity theory and other areas of mathematics, but also algorithms, complexity theory, pattern recognition and numerical linear algebra from computer science and even many branches of physics: Hamiltonian mechanics, statistical mechanics and even quantum field theory.

I cannot resist recounting an incident that happened during the FLoC conference in Oxford in 2018. I was in a pub and bumped into a friend from graduate school days whom I had not seen in 30 years. We were both physics graduate students at Chicago in the 1970s and he, unlike me, stayed in physics and had become a well-known condensed-matter physicist. He was with a group of Oxford physicists who graciously invited me to join them for a drink. I was asked about my area of research and I replied, "machine learning", whereupon one of the physicists exclaimed, "oh yes, we invented that!"

2 Probabilistic programming languages

One of the earliest papers to consider a probabilistic programming language is a study of probabilistic LCF by Saheb-Djahromi [SD78]. This led to work on the interplay between probability theory and domain theory [SD80]

[1] Spelled with two h's.

and ultimately to probabilistic powerdomains [JP89] and integration on domains [Eda95]. Shortly after Saheb-Djahromi, Dexter Kozen [Koz81] considered a typical imperative programming augmented with probabilistic choice and gave a formal semantics in terms of measure theory.

In the world of verification and process algebra papers appeared in the 1980s [Var85] but the most significant step was a paper by Larsen and Skou [LS91] which appeared as a conference paper in 1989. They introduced a modal logic and proved a logical characterisation theorem. The latter was subsequently refined and extended to continuous state spaces by Desharnais *et al.* [DEP02].

The idea of modelling a stochastic process as a program in a programming language is due to Gupta *et al.* [GJP99]. In this paper the language was in the family of concurrent constraint programming languages [Sar89]. What was significant here is the important role played by conditional probability in the semantics.

It took about 10 years before these ideas really flowered. This happened in a spectacular result due to Ackerman, Freer and Roy [AFR11] where they showed that one could have a computable probability distribution and impose computable constraints but the resulting conditional distribution is *non-computable*. This raised the question about what could be the included in a probabilistic programming language. Clearly one needs some kind of control on conditioning.

But why were machine learning people interested in probabilistic programming languages? For precisely the same reasons that the semantics community had been advocating all along: *compositionality*. It is clear that graphical formalisms like Bayes nets will get hopelessly clumsy as they get larger. Perhaps this lesson has not sunk in to the wider machine learning community as graphical models are widely used and taught but at least the idea of compositional construction of models has indeed entered the subject and the explicit role played by programming language theory is acknowledged.

At present one cannot say that the main stream of machine learning has embraced these ideas, but there are flourishing groups at the intersection of programming languages and machine learning that are developing higher-order probabilistic languages [GMR$^+$08, TvdMYW16] and the theory needed to understand the combination of probability distributions, higher-type programs and conditioning [SYW$^+$17, HKSY17, CJ19]

3 Reinforcement learning and fixed-point theory

In statistical machine learning one is typically trying to learn some structural information from randomly sampled data. A typical example is where one has an unknown labelling function *i.e.* a map from $L : \mathcal{X} \to \mathcal{Y}$, where $\mathcal{X}$ is the space from which samples are drawn and $\mathcal{Y}$ is the space of labels. The machine-learning algorithm is presented with m samples, correctly labelled, and tries to "learn" L. Reinforcement learning (RL), by contrast, is a more active process.

The basic set up is a Markov decision process (MDP). One has an agent that interacts with this process and is rewarded (or penalised) as actions are chosen. The agent tries to learn a policy that will optimise the cumulative reward. We formalise this a bit more precisely now. We write $\mathcal{D}(S)$ for the set of probability distributions on a set S.

Definition 3.1. A **Markov decision process** M is a 4-tuple

$$(S, \mathcal{A}, \forall a \in \mathcal{A}\ \tau_a : S \to \mathcal{D}(S), R : S \times \mathcal{A} \to \mathcal{D}(\mathbf{R})),$$

where, S is a finite set of *states*, $\mathcal{A}$ is a set of *actions*, τ_a is a *transition probability function* and R is the *reward*.

All kinds of minor variations may be seen in the literature. If the system is in a state s and the action a is chosen, there is a transition to a new state governed by the probability distribution $\tau_a(s)$ and a reward is assigned according to the probability distribution $R(s, a)$. A numerical parameter $\gamma \in (0, 1)$, called the *discount factor* is often given as part of the description of the MDP. In RL, the agent does not know the internal structure or dynamics of the MDP and by choosing actions tries to learn a *policy*, a map $\pi : S \to \mathcal{D}(\mathcal{A})$, which will maximise the expected discounted reward:

$$\sum_{t=0}^{\infty} \gamma^t r_t,$$

where t represents the time step and r_t is the reward at time t. Rewards in the future are discounted, which ensures that the sum is finite if R is reasonably well behaved. The basic paradigm is due to Bellman [Bel54] who invented the basic algorithms, namely dynamic programming, to determine optimal policies. A concise modern exposition is given in [Sze10] including new algorithms invented since that time.

This subject is not an example of the semantics community influencing the machine learning community, rather it is a case of two communities converging onto a common mathematical paradigm: fixed-point theory. For semanticists, fixed-point theory is a basic and beloved tool imported into programming language theory by Dana Scott, Jaco de Bakker and others from its roots in computability (Kleene) and lattice theory (Tarski). In RL the existence of an optimal policy, *and an algorithm to compute it* is based on the Banach fixed-point theorem.

Returning to the world of MDPs, we define a *value function* as a map from states to expected rewards. The value function taken as parameter the policy π. More precisely, we define $V^\pi : S \longrightarrow \mathbf{R}$, for a policy π by

$$V^\pi(s) = \mathbb{E}_{a\sim\pi(s)}[\mathbb{E}_{r\sim R(s,a)}[r] + \gamma \sum_{s'\in S} \tau_a(s)(s')V^\pi(s').]$$

Here the notation $\mathbb{E}_{x\sim D}$ means the expectation value when x is sampled according to the distribution D. This can clearly be seen to be a fixed-point equation for the *function* V; so it is even a higher-type entity.

The basic theorem that ensures the existence of V is the Banach fixed-point theorem.

Theorem 3.2. Let (X, d) be a *complete* metric space and let $f : X \longrightarrow X$ be a contractive function: $\exists\gamma \in (0, 1),\ \forall x, y \in X,\ d(f(x), f(y)) \leq \gamma d(x, y)$. Then there is a *unique* fixed point for f: a point x_0 such that $f(x_0) = x_0$.

The proof is easy, start from any point and keep iterating f. The contractiveness condition ensures that the sequence is Cauchy and the completeness ensures that the sequence has a limit. It is easy to verify that this limit is the unique fixed point. Indeed this iterative approach is the basis of many of the algorithms for finding optimal policies. Here we have only talked about the value function associated with a policy, but it is not hard to define an optimal value function and show that it too satisfies a fixed-point equation.

Here is a case of a parallel evolution of similar ideas. The use of fixed-point theory in these cases is more than a coincidence. The communities did not interact and there was no cross fertilisation. It is also the case that fixed-point theory was used in various areas of mathematics and economics and there were many other fixed-point theorems available. However, next we shall see a case where there was interaction with tangible results.

4 A role for bisimulation metrics

Bisimulation is one of the triumphs of semantics. Its origins in process algebra have been well described by Sangiorgi [San09] and the books [San11, SR11] give an up-to-date account of the state of the art including probabilistic bisimulation introduced by Larsen and Skou [LS91]. The machine learning community working with MDPs independently invented probabilistic bisimulation [GDG03] though they did acknowledge the priority of Larsen and Skou; unfortunately, everyone else in machine learning seems to be unaware of the work of Larsen and Skou.

Shortly after the introduction of probabilistic bisimulation, it became clear that the concept was not robust [GJS90]. Bisimulation is a binary relation: it holds or not, whereas the parameters on which it depends vary continuously. The idea of a *metric* measuring behavioural equivalence was mooted by Giacalone et al. Unfortunately, it is not easy to define such a concept: a naive "up to ε" definition does not work. The definition was finally given by Desharnais et al. [DGJP99, DGJP04] based on an earlier logical characterisation result [DEP98]. A fixed-point definition more in the spirit of the bisimulation relation as a greatest-fixed-point was then given by van Breugel and Worrell [vBW01b, vBW01a]. This version of the definition is what has spurred most of the subsequent developments and in particular the use of ideas from optimal transport theory and Kantorovich-Rubinstein duality [Vil08].

These metrics were defined for labelled Markov processes: transition systems with no notion of "reward" or "optimal policy." A bisimulation metric for MDPs was forthcoming shortly thereafter [FPP04, FPP05]. In these papers the definition of bisimulation is given by adding a term that reflects the reward. This tiny change had a significant impact: it was proven that the difference in the optimal value function between two different states: $|V(s) - V(s')|$ is bounded by the bisimulation metric $d(s, s')$. This showed a very tight relation between the bisimulation metric and the quantities of interest in reinforcement learning. Indeed later Ferns and Precup [FP14] showed that bisimulation metrics *are* optimal value functions for an appropriate MDP.

5 A recent success story

Recently a striking use of metric arguments is in a recent paper in The 23rd International Conference on Artificial Intelligence and Statistics (AISTATS 2020) conference [APPB20]. A number of algorithms in RL proceed by sampling from the MDP: so the assumption is that the dynamics of the MDP are not known but one can obtain samples of trajectories from the MDP. The algorithms then proceed by iteratively computing an approximation to the fixed point. There are many such algorithms, the general class of such algorithms is called *stochastic approximation algorithms*.

Proofs of convergence of these algorithms can be quite difficult; see the references in [APPB20] for details. In the AISTATS paper however, we took the perspective that one should view the algorithms themselves as Markov processes of *higher type*: that means the state space is the space of distributions. Then using coupling techniques we showed that many of the algorithms define contractive maps on the space of distributions equipped with the Kantorovich-Wasserstein[2] metric. Thus many of the difficult proofs found in the literature can be done in a simple and more uniform way. One has to find a coupling; this may be more or less difficult, but all our examples (there are seven algorithms that we treat) are dealt with with simple-minded couplings. This is, in my opinion, a pleasing application of ideas from the semantics world to RL and one which is appreciated by the RL community.

We proceed to some of the details of this work. The first innovation, which is due to the last-named author of [APPB20] and other co-workers [BDM17], is to view reinforcement learning as working with the distribution of possible rewards rather than the expected reward. This is called "distributional reinforcement learning" and there seems to be experimental support that one gets better performance using this type of learning.

In [APPB20] the basic idea is to try to show that the update rule defined on the distributions is a contraction in suitable metric. This metric is the Kantorovich metric obtained by lifting the metric induced by the infinity norm.

Given an MDP as defined above with a discount factor γ, we introduce the

[2]This should just be called the Kantorovich metric. It is usually called just the Wasserstein metric; this is a historical mistake.

Bellman operator T^π which is dependent on a choice of policy π by:

$$T^\pi(V) = \lambda s. \mathbb{E}_{a\sim\pi(s), r\sim R(s,a)}[r + \gamma \mathbb{E}_{s'\sim P(s,a)}[V(s')]]$$

The fixed-point of T^π is the value function V^π described above. The value function of the optimal policy π^* is the fixed-point of what is called the *Bellman optimality operator*:

$$T(V) = \lambda s. \max_a \mathbb{E}_{r\sim R(s,a)}[r + \gamma \mathbb{E}_{s'\sim P(s,a)}[V(s')]].$$

These operators are contractions on the space $\mathbf{R}^{|S|}$ with the metric induced by the infinity norm.

We now give a more explicit description of the Kantorovich metric [Vil08]. Suppose that we wish to compare two probability measures P, Q defined on a metric space (X, ρ) with the Borel sets induced by the metric. What can one "see" about a measure? Integrals! So we can think of something like

$$\sup_f |\int f \mathrm{d}P - \int f \mathrm{d}Q|.$$

But over what class of functions should one take the sup? If we don't restrict the functions, it is easy to see that the sup can be made infinite always. Kantorovich restricted to *non-expansive* functions or 1-Lipschitz functions: $(f(x) - f(y))| \leq \rho(x, y)$. This is where the metric ρ plays a crucial rôle. We call the set of such functions $1 - \mathrm{Lip}$. Then we define

$$K(P, Q) = \sup_{f\in 1-\mathrm{Lip}} |\int f \mathrm{d}P - \int f \mathrm{d}Q|.$$

There is a beautiful duality theorem known as Kantorovich-Rubinstein duality, which gives a description of K as a minimum.

Definition 5.1. A **coupling** between probability distributions P, Q on a space (X, ρ) is a probability measure γ defined on $X \times X$ such that the marginals coincide with P and Q. This means that for any Borel set $A \subset X$ we have $\gamma(A \times X) = P(A)$ and $\gamma(X \times A) = Q(A)$.

Intuitively, one thinks of this as a transport plan. Think of the measures as defining piles of sand. One wishes to move the sand so that the pile P becomes the pile Q. The quantity $\gamma(A \times B)$ defines how much must be moved from region A to region B. The cost of moving the sand depends on how far the sand is to be moved. Thus we are led to define the total cost as

$$\int \rho(x, y) \mathrm{d}\pi.$$

Let us write $\Gamma(P, Q)$ for the set of all possible couplings between P and Q. The Kantorovich-Rubinstein duality theorem states:

$$W_1(P, Q) = \inf_{\gamma \in \Gamma(P,Q)} \int_{X \times X} \rho(x, y) \mathrm{d}\gamma = K(P, Q).$$

The notation W_1 is much more common and in view of the duality theorem it is the same as K. The key point is that the W_1 metric is defined using optimal couplings. Thus any old coupling gives an upper bound on the W_1 distance.

Now let us consider a typical RL algorithm, called $TD(0)$ [Sze10]. The TD stands for "temporal difference" and the 0 need not concern us. It has a simple update rule for the value function. If the value function after n steps is written as V_n we have

$$V_{n+1}(s) := (1 - \alpha)V_n(s) + \alpha(R(s, a) + \gamma V_n(s'))$$

which should be viewed as a distributional equation in our setting. Here $R(s, a)$ is the distribution over the possible rewards defined by the MDP, a is sampled from the actions according to the distribution defined by the policy and s' is the new state sampled according to the MDP dynamics. The parameter α, called the "step size", tells us the extent we use the old value function as opposed to the one generated after one transition. The coinductive character of these kind of value function updates makes it a natural to compare it with a bisimulation metric and, indeed, in [FPP04] such bounds are given.

We take the view that a learning algorithm is itself a Markov chain whose state space is the space of probability distributions over the underlying MDP. The update rule for a particular learning algorithm like $TD(0)$ then defines a Markov kernel. A Markov kernel K can be viewed as a map from a probability distribution P to a new distribution $K(P)$ given by

$$K(P) = A \mapsto \int_X K(x, A) \mathrm{d}P(x)$$

where A is a Borel set. In our paper [APPB20] we proved that the $TD(0)$ kernel, call it K_α for step size α, defines a contractive map:

$$W_1(K_\alpha(P), K_\alpha(Q)) \leq (1 - \alpha + \alpha\gamma)W_1(P, Q)$$

for any step size $\alpha \in (0, 1]$.

The idea of the proof is to start with an optimal coupling of the initial distributions and then we use this to construct a coupling for the updated distributions. This latter coupling may not be optimal but it suffices to produce an upper bound and prove contractiveness.

Using essentially the same method we were able to cover 6 more algorithms in the same way. The technique depends on one's ability to find suitable couplings for the updates. What is particularly pleasing is that the construction of the couplings was *ridiculously easy*. Of course, we then encountered examples where it was not at all clear that we could find suitable couplings for the updates. However, the example that we did do were all notable examples in the literature and had been proved to converge earlier by much more painful methods. A little structural thinking can go a long way.

6 Conclusions

This short note is a response to a question raised by Samson in "Whither Semantics": do we want to lead or follow? I would like to suggest that indeed there is a role for semantics-based ideas in machine learning. I hope that the examples described above will give the semantics community some motivation to interact with the machine learning community. I must take this opportunity to say that in my case machine learning people, Doina Precup and Joelle Pineau, came knocking on my door and sought me out rather than the other way around.

There is plenty more to be done. One of the big mysteries of machine learning is why do neural networks work so well? All the theory of optimization is geared towards convex situations but in deep neural networks are very non-convex. They are also massively overparametrised so why do they seem to generalise so well? People are invoking ideas from physics and high-dimensional geometry to try to explain these things: most of these ideas are in flux and largely experimental so it would not do much good to cite a lot of papers here. As far as I can see, any decent ideas are worth investigating: higher-order programming, monoidal categories and the algebra of string diagrams, ideas from causality and many other things.

References

[AB11] Samson Abramsky and Adam Brandenburger. A unified sheaf-theoretic account of non-locality and contextuality. *CoRR*, abs/1102.0264, 2011.

[ABK+15] Samson Abramsky, Rui Soares Barbosa, Kohei Kishida, Raymond Lal, and Shane Mansfield. Contextuality, cohomology and paradox. In Stephan Kreutzer, editor, *24th EACSL Annual Conference on Computer Science Logic, CSL 2015, September 7-10, 2015, Berlin, Germany*, volume 41 of *LIPIcs*, pages 211–228. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2015.

[ABKM19] Samson Abramsky, Rui Soares Barbosa, Martti Karvonen, and Shane Mansfield. A comonadic view of simulation and quantum resources. In *34th Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2019, Vancouver, BC, Canada, June 24-27, 2019*, pages 1–12. IEEE, 2019.

[Abr14] Samson Abramsky. Contextual semantics: From quantum mechanics to logic, databases, constraints, and complexity. *CoRR*, abs/1406.7386, 2014.

[Abr20] Samson Abramsky. Whither semantics? *Theor. Comput. Sci.*, 807:3–14, 2020.

[ADW17] Samson Abramsky, Anuj Dawar, and Pengming Wang. The pebbling comonad in finite model theory. In *32nd Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2017, Reykjavik, Iceland, June 20-23, 2017*, pages 1–12. IEEE Computer Society, 2017.

[AFR11] Nathanael L. Ackerman, Cameron E. Freer, and Daniel M. Roy. Noncomputable conditional distributions. In *Logic in Computer Science (LICS), 2011 26th Annual IEEE Symposium on*, pages 107–116. IEEE, 2011.

[AMB11] Samson Abramsky, Shane Mansfield, and Rui Soares Barbosa. The cohomology of non-locality and contextuality. In Bart Jacobs, Peter Selinger, and Bas Spitters, editors, *Proceedings 8th International Workshop on Quantum Physics*

and Logic, QPL 2011, Nijmegen, Netherlands, October 27-29, 2011, volume 95 of *EPTCS*, pages 1–14, 2011.

[APPB20] Philip Amortila, Doina Precup, Prakash Panangaden, and Marc Bellemare. A distributional analysis of sampling-based reinforcement learning algorithms. In *The 23rd International Conference on Artificial Intelligence and Statistics*, 2020.

[AS18] Samson Abramsky and Nihil Shah. Relating structure and power: Comonadic semantics for computational resources. In Dan R. Ghica and Achim Jung, editors, *27th EACSL Annual Conference on Computer Science Logic, CSL 2018, September 4-7, 2018, Birmingham, UK*, volume 119 of *LIPIcs*, pages 2:1–2:17. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2018.

[AW17] Samson Abramsky and Viktor Winschel. Coalgebraic analysis of subgame-perfect equilibria in infinite games without discounting. *Mathematical Structures in Computer Science*, 27(5):751–761, 2017.

[BDM17] Marc G. Bellemare, Will Dabney, and Rémi Munos. A distributional perspective on reinforcement learning. In *Proceedings of the 34th International Conference on Machine Learning, Volume 70*, pages 449–458. JMLR. org, 2017.

[Bel54] Richard E. Bellman. The theory of dynamic programming. *Bulletin of the American Mathematical Society*, 60(6):503–516, 1954.

[CJ19] Kenta Cho and Bart Jacobs. Disintegration and bayesian inversion via string diagrams. *Mathematical Structures in Computer Science*, 29(7):938–971, 2019.

[DEP98] J. Desharnais, A. Edalat, and P. Panangaden. A logical characterization of bisimulation for labelled Markov processes. In *proceedings of the 13th IEEE Symposium On Logic In Computer Science, Indianapolis*, pages 478–489. IEEE Press, June 1998.

[DEP02] J. Desharnais, A. Edalat, and P. Panangaden. Bisimulation for labeled Markov processes. *Information and Computation*, 179(2):163–193, Dec 2002.

[DGJP99] J. Desharnais, V. Gupta, R. Jagadeesan, and P. Panangaden. Metrics for labeled Markov systems. In *Proceedings of CONCUR99*, number 1664 in Lecture Notes in Computer Science. Springer-Verlag, 1999.

[DGJP04] Josée Desharnais, Vineet Gupta, Radhakrishnan Jagadeesan, and Prakash Panangaden. A metric for labelled Markov processes. *Theoretical Computer Science*, 318(3):323–354, June 2004.

[Eda95] Abbas Edalat. Domain theory and integration. *Theoretical Computer Science*, 151:163–193, 1995.

[FP14] Norm Ferns and Doina Precup. Bisimulation metrics are optimal value functions. In *Proceedings of the 30th Conference on Uncertainty in Artificial Intelligence*, 2014. Article 67.

[FPP04] Norm Ferns, Prakash Panangaden, and Doina Precup. Metrics for finite Markov decision precesses. In *Proceedings of the 20th Conference on Uncertainty in Artificial Intelligence*, pages 162–169, July 2004.

[FPP05] Norm Ferns, Prakash Panangaden, and Doina Precup. Metrics for Markov decision processes with infinite state spaces. In *Proceedings of the 21st Conference on Uncertainty in Artificial Intelligence*, pages 201–208, July 2005.

[GDG03] Robert Givan, Thomas Dean, and Matthew Greig. Equivalence notions and model minimization in Markov decision processes. *Artificial Intelligence*, 147(1-2):163–223, 2003.

[GJP99] V. Gupta, R. Jagadeesan, and P. Panangaden. Stochastic processes as concurrent constraint programs. In *Proceedings of the 26th Proceedings Of The Annual ACM Symposium On Principles Of Programming Languages*, pages 189–202, 1999.

[GJS90] A. Giacalone, C. Jou, and S. Smolka. Algebraic reasoning for probabilistic concurrent systems. In *Proceedings of the Working Conference on Programming Concepts and Methods*, IFIP TC2, 1990.

[GMR+08] Noah Goodman, Vikash Mansingkha, Daniel Roy, Keith Bonawitz, and Joshua Tenenbaum. Church: a language for

generative models. In *Proceedings of the 24th Conference on Uncertainty in Artificial Intelligence*, pages 220–229, 2008.

[HKSY17] Chris Heunen, Ohad Kammar, Sam Staton, and Hongseok Yang. A convenient category for higher-order probability theory. In *Proceedings of the Thirty-second Annual ACM-IEEE Symposium on Logic in Computer Science*, 2017.

[IB98] Chris J. Isham and Jeremy Butterfield. Topos perspective on the kochen-specker theorem: I. quantum states as generalized valuations. *International journal of theoretical physics*, 37(11):2669–2733, 1998.

[JP89] C. Jones and G. D. Plotkin. A probabilistic powerdomain of evaluations. In *Proceedings of the Fourth Annual IEEE Symposium On Logic In Computer Science*, pages 186–195, 1989.

[Koz81] D. Kozen. Semantics of probabilistic programs. *Journal of Computer and Systems Sciences*, 22:328–350, 1981.

[LS91] K. G. Larsen and A. Skou. Bisimulation through probablistic testing. *Information and Computation*, 94:1–28, 1991.

[San09] Davide Sangiorgi. On the origins of bisimulation and coinduction. *ACM Transactions on Programming Languages and Systems (TOPLAS)*, 31(4):15, 2009.

[San11] Davide Sangiorgi. *Introduction to Bisimulation and Coinduction*. Cambridge University Press, 2011.

[Sar89] Vijay A. Saraswat. *Concurrent Constraint Programming Languages*. PhD thesis, Carnegie-Mellon University, January 1989.

[SD78] N. Saheb-Djahromi. Probabilistic LCF. In *Mathematical Foundations Of Computer Science*, number 64 in Lecture Notes In Computer Science. Springer-Verlag, 1978.

[SD80] N. Saheb-Djahromi. Cpos of measures for nondeterminism. *Theoretical Computer Science*, 12(1):19–37, 1980.

[SR11] Davide Sangiorgi and Jan Rutten, editors. *Advanced topics in bisimulation and coinduction*. Number 52 in Cambridge

Tracts in Theoretical Computer Science. Cambridge University Press, 2011.

[SYW+17] Sam Staton, Hongseok Yang, Frank Wood, Chris Heunen, and Ohad Kammar. Semantics for probabilistic programming: higher-order functions, continuous distributions, and soft constraints. In *Proceedings of the 31st Annual ACM-IEEE Symposium On Logic In Computer Science*, pages 525–534, 2017.

[Sze10] Csaba Szepesvári. Algorithms for reinforcement learning. *Synthesis lectures on artificial intelligence and machine learning*, 4(1):1–103, 2010.

[TvdMYW16] David Tolpin, Jan-Willem van de Meent, Hongseok Yang, and Frank Wood. Design and implementation of probabilistic programming language Anglican. In *Proceedings of the 28th Symposium on the Implementation and Application of Functional Programming Languages*, page 6. ACM, 2016.

[Var85] Moshe Vardi. Automatic verification of probabilistic concurrent finite-state programs. In *26th IEEE Symposium On Foundations Of Computer Science*, pages 327–338, 1985.

[vBW01a] Franck van Breugel and James Worrell. An algorithm for quantitative verification of probabilistic systems. In K. G. Larsen and M. Nielsen, editors, *Proceedings of CONCUR'01*, number 2154 in Lecture Notes In Computer Science, pages 336–350. Springer-Verlag, 2001.

[vBW01b] Franck van Breugel and James Worrell. Towards quantitative verification of probabilistic systems. In *Proceedings of the Twenty-eighth International Colloquium on Automata, Languages and Programming*. Springer-Verlag, July 2001.

[Vil08] Cédric Villani. *Optimal transport: old and new*. Springer-Verlag, 2008.