Linear Models of Computation and Program Learning

Michael Bukatin[1] and Steve Matthews[2]

[1] Nokia Corporation
Burlington, Massachusetts, USA
bukatin@cs.brandeis.edu
[2] Department of Computer Science
University of Warwick
Coventry, UK
Steve.Matthews@warwick.ac.uk

Abstract. We consider two classes of computations which admit taking linear combinations of execution runs: probabilistic sampling and generalized animation. We argue that the task of program learning should be more tractable for these architectures than for conventional deterministic programs. We look at the recent advances in the "sampling the samplers" paradigm in higher-order probabilistic programming. We also discuss connections between partial inconsistency, non-monotonic inference, and vector semantics.

Keywords: Probabilistic programming, fuzzy sampling, generalized animation, negative probability, bilattices, bitopology, extended interval numbers

1 Introduction

One of the key challenges of program learning is that software tends to be too brittle and insufficiently robust with respect to minor variation.

Biological systems tend to be much more flexible and adaptive with respect to variation. In particular, biological cells are capable of functioning at wide ranges of the level of expressions of various proteins, which are machines working in parallel. Regulation of the level of expression of specific proteins is a key element of flexibility of biological systems. It is argued in evolutionary developmental biology that the flexible architrecture together with conservation of core mechanisms is crucial for the observed rate of biological evolution [9, 15]. It is suggested that morphology evolves largely by altering the expression of functionally conserved proteins [6].

To incorporate regulation of expression into a system of genetic programming one might evolve programs describing systems of parallel computational processes. Then one might take the CPU allocation and other computational resources given to a particular computational process as computational equivalent of the level of expression of a particular protein.

Of course, many of the architectures for parallel computations are brittle as well, with delicate mechanisms of writing to shared memory and locks. To achieve flexibility one should use parallel architectures which minimize those delicate interdependencies.

Computational architectures which admit the notion of linear combination of execution runs are particularly attractive in this sense. Then one can regulate the system simply by controlling coefficients in a linear combination of its components.

In this paper we consider two computational architectures which admit linear combinations of execution runs.

One such architecture is probabilistic sampling. If one has two samplers generating points of two distributions with a uniform speed, so that the notion of "a number of points generated per unit of time" is well defined for both of them, one can obtain a sampler generating a linear combination of those two distributions with arbitrary positive coefficients simply by running those two samplers in parallel with appropriate relative speeds.

We argue that it should be easier to learn probabilistic programs than to learn deterministic programs due to the fact that probabilistic programs admit linear combinations of execution runs. We also discuss the techniques to allow negative coefficients in those linear combinations later in the text.

There is a lot of affinity between methods of evolutionary programming and methods of probabilistic sampling. Evolutionary schemas can be considered as particular sampling methods, while many sampling schemas have strong evolutionary flavor (more details in Section 2).

This means that instead of thinking in terms of genetic programming for probabilistic programs one might think about program learning in terms of the "sampling the samplers" paradigm, namely in terms of probabilistic programs sampling other probabilistic programs (generative models producing other generative models as their points). This "sampling the samplers" paradigm manifests itself, in particular, in recent work on learning probabilistic programs by Yura Perov and Frank Wood [25] and also in recent advances in compositional concept learning obtained by Brenden Lake [17]. We review these and some other recent advances in higher-order probablistic programming in Section 3.

1.1 Negative Coefficients

Another computational architecture which admit linear combinations of execution runs is generalized animation. We define generalized (monochrome) image as a map from a set (called the set of points) to reals. We define generalized (monochrome) animation as a map from time (discrete or continuous) to generalized images. Linear combination of images is defined point-wise. The secondary structure on the set of points might vary. To obtain a conventional monochrome image the secondary structure typically assigns coordinates from a discretized rectangle to points. For display purposes, zero is normally associated with gray level in the middle between the most dark and the most bright possible values. This approach allows the use of both positive and negative coefficients in the linear combinations of generalized images and animations.

Conventional color images and music are examples of generalized animations, and the use of linear combinations of those is standard in video and audio mixing software.

One feature shared between animations and probabilistic programs is that complex behaviors can often be expressed by very short programs. Many software experiments including our own work with simulated reflection in water waves demonstrate that interesting and expressive dynamics can result from simple programs.

Another feature animations seem to share with sampling architecture is that they tend to be non-brittle, and that their mutations and crossover tend to produce meaningful results in evolutionary setting [7]. This architecture provides a direct way to incorporate aesthetic criteria into software systems. This architecture can also leverage existing animations, digital and physical (such as light refections and refractions in water), as computational oracles.

A lot of expressive power of this architecture comes from the ability to have non-standard secondary structures on the set of points. Points can be associated with vertices or edges of a graph, grammar rules, etc. One should be able to formulate mechanisms of higher-order animation programming via variable illumination of elements of such structures.

1.2 Negative Probability

In order to enable both positive and negative coefficients for probabilistic sampling one can allow sampling via two parallel sampling channels: a positive one and a negative one.

There is evidence that signed functions are sampled via parallel positive and negative channels in neural system, for example, in retina (see pages 65 and 173 of [21]). The idea that some of the brain functioning might be understood as Markov Chain Monte Carlo sampling was developed in recent years and led to fruitful applications to the computational schemes robust with respect to noise and benefitting from presence of noise and thus suitable for implementation in low-powered circuits (see [19] and references therein). The combination of this idea and of the evidence for sampling via parallel positive and negative channels is suggestive.

One way to understand and formalize this situation is via allowing negative values for probabilities and probability densities. Quasiprobability distributions allowing both positive and negative probability values have long history. Their first prominent use comes in phase space formulation of quantum mechanics via Wigner quasiprobability distribution in 1940s [23, 11]. The intuition behind the notion of negative probability is discussed in detail in [8]. More recently, negative probabilities are finding use in quantum algorithms [22].

In denotational semantics of probabilistic programs, Dexter Kozen found it fruitful to replace the space of probability distributions with the space of signed measures [16]. This allowed him to express denotations of probabilistic programs as continuous linear operators with finite norms. The probabilistic powerdomain was embedded into the positive cone of the resulting Banach lattice.

1.3 Partial Inconsistency, Non-monotonic Inference, and Vector Semantics

Addition of the elements expressing partial degrees of contradiction results in an embedding of an approximation domain into a vector space in yet another important case, the interval numbers, by extending them with *pseudosegments* $[a,b]$ with the contradictory property that $b < a$.

The resulting spaces tend to be equipped with two Scott topologies dual to each other, which enables both upwards and downwards computable inference steps, and thus facilitates non-monotonic reasoning.

The resulting mathematical landscape is a field directly adjacent to the main topic of this paper. We review this field and present some of our own results there in Section 4.

2 Parallels between Methods of Evolutionary Programming and Probabilistic Sampling

The connections between probabilistic programming and genetic programming are much tighter than it is usually acknowledged.

Many variants of MCMC are evolutionary in spirit. Acceptance/rejection of the samples corresponds to selection. Production of new samples via modifications of the accepted ones corresponds to mutations to produce offspring from the survivors.

Bayesian Optimization Algorithm changes the procedure of producing the next generation in genetic algorithms from pairwise crossover to resampling from the estimated distribution of the individuums selected for fitness [24]. This scheme of crossover is used by the seminal MOSES system [18]. This is similar in spirit to population-based methods of sampling.

3 Some Recent Advances in Higher-Order Probabilistic Programming

We are seeing a very rapid progress in probabilistic programming in recent years.

What particularly catches our attention is a series of results solving various computer vision problems as Bayesian inverse problems to computer graphics rendering, starting with [20].

For an example of a powerful model learning scheme for probabilistic programs using matrix decomposition and a context-free grammar of models see [12].

The term "higher-order probabilistic programming" usually means a higher-order functional programming language implementing sampling semantics. Recently we are seeing examples of research implementing higher-order sampling schemas in a more narrow and focused sense of the word: samplers which generate other samplers, probabilistic programs sampling the space of probabilistic programs.This is a particularly important development for program learning.

A recent work on learning probabilistic programs within the "sampling the samplers" paradigm by Perov and Wood allows, in particular, to "compile" probabilistic programs so that the resulting samplers just sample the posterior directly without sampling the whole joint distribution (another possible name for this procedure which comes to mind is "partial evaluation", although neither term is quite adequate for this novel procedure). The work is done using the new Anglican engine which implements a probabilistic programming language similar to Venture, but is written in Clojure (which should enable better parallelization and better performance scaling with more hardware) and uses a higher-order PMCMC ("Particle Markov Chain Monte Carlo") sampling scheme, where efficient high-dimensional proposal distributions for MCMC are generated by particle filters. The work follows earlier successes of Maddison and Tarlow in capturing frequent context-dependent syntactic patterns of code from open source repositories within generative models (see [25] and references therein).

Another important work done with the use of multilevel "generative models emitting generative models" architecture is the research by Lake in compositional concept learning [17]. The typical tasks performed are learning the letters of synthetic alphabet and spoken Japanese-like words. The author claims that this is the first time when a machine learning system combines learning from one or a few examples (rather than from big data corpora) with learning rich conceptual representations.

4 Partial Inconsistency, Non-monotonic Inference, and Vector Semantics

The traditional mathematical view is that there is only one kind of contradiction and that all contradictions imply each other and everything else. However, there is also rich tradition of studying various kinds of graded or partial contradictions.

There are a number of common motives appearing multiple times in various studies of graded inconsistency. These common motives link a variety of independently done studies together and serve as focal elements of what we call the *partial inconsistency landscape* [3]. We list many of these common motives and some of their interplay.

An especially important motive is that in the presence of partial inconsistency many otherwise impoverished algebraic structures become groups and vector spaces. In particular, domains for denotational semantics tend to acquire group and vector space structure when partial inconsistency is present.

Known applications include handling of inconsistent information and non-monotonic and anti-monotonic inference. Perhaps even more importantly for the advanced AI, vector semantics is likely to offer new powerful schemes for program learning, as we are arguing in this paper.

We only overview a small slice of this field here and present some of our results. For more details, see [4] and references therein.

4.1 Focal Elements of the Partial Inconsistency Landscape

- Various forms of *negative measure* (negative length and distance, negative probability and signed measures, negative membership and signed multisets)
- Bilattices
- Bitopology
- Domains with group and vector space structures
- Modal and paraconsistent logic and possible world models
- Bicontinuous domains
- The domain of arrows, $D^{Op} \times D$ or $C^{Op} \times D$
- Non-monotonic and anti-monotonic inference
- Modal and paraconsistent logic and possible world models
- Hahn-Jordan decomposition or "bilattice pattern":
  $x = (x \wedge 0) + (x \vee 0)$ or $x = (x \wedge \perp) \sqcup (x \vee \perp)$

4.2 Partially Inconsistent Interval Numbers

Interval numbers are segments $[a,b]$ on the real line where $a \leq b$. One can extend interval numbers by adding *pseudosegments* $[a,b]$ with the contradictory property that $b < a$. This structure was independently discovered many times and is known under various names including Kaucher interval arithmetic, directed interval arithmetic, generalized interval arithmetic, and modal interval arithmetic (a comprehensive repository of literature on the subject is maintained by Evgenija Popova [26]). Our group tends to call it *partially inconsistent interval numbers*.

There are two partial orders on partially inconsistent interval numbers. The *informational order*, $\sqsubseteq$, is defined by reverse inclusion on interval numbers: $[a,d] \sqsubseteq [b,c]$ iff $a \leq b$ and $c \leq d$. The same formula is used for partially inconsistent interval numbers. The *material order* is component-wise: $[a,b] \leq [c,d]$ iff $a \leq c$ and $b \leq d$.

Addition on interval numbers (and partially inconsistent interval numbers) is defined component-wise: $[a_1,b_1]+[a_2,b_2]=[a_1+a_2,b_1+b_2]$.

The operation of *weak minus* is defined as $-[a,b]=[-b,-a]$. Addition and weak minus are *monotonic* with respect to $\sqsubseteq$.

Consider $-[a,b]+[a,b]=[-b,-a]+[a,b]=[a-b,b-a]$. If $a<b$, then the strict inequality, $[a-b,b-a] \sqsubset [0,0]$, holds. So if $a<b$, $-[a,b]+[a,b]$ approximates $[0,0]$, but is not equal to it, hence interval numbers with weak minus don't form a group.

If one allows pseudosegments, one can define the component-wise *true minus*: $-[a,b]=[-a,-b]$. Partially inconsistent interval numbers with the component-wise addition and the true minus form a group (and a 2D vector space over reals). The true minus maps precisely defined numbers, $[a,a]$, to precisely defined numbers, $[-a,-a]$. Other than that, the true minus maps segments to pseudosegments and maps pseudosegments to segments. The true minus is *anti-monotonic* with respect to $\sqsubseteq$.

4.3 Bilattices

A bilattice is a set equipped with two lattice structures defining two partial orders, the *material order*, $\leq$, and the *informational order*, $\sqsubseteq$, and an involution monotonic with respect to $\sqsubseteq$, antimonotonic with respect to $\leq$, and preserving appropriate lattice structures. Additional axioms are often imposed.

Bilattices were introduced by Matthew Ginsberg [10] to provide a unified framework a variety of inferences schemes used in AI, such as *non-monotonic inference*, inference with uncertainty, etc. They are now ubiquitous in the studies of partial and graded inconsistency.

The simplest example of a bilattice is the four-valued logic: $f<\bot<t, f<\top<t$, $\bot \sqsubset f \sqsubset \top, \bot \sqsubset t \sqsubset \top$.

Partially inconsistent interval numbers form a bilattice. Sometimes one wants both orders to form complete lattices. This can be achieved by allowing a and b to also take $-\infty$ and $+\infty$ as values, or by confining a and b within a segment $[A,B]$, in both cases sacrificing the property of partially inconsistent interval numbers being a group.

4.4 Bitopology and Non-monotonic Inference

Asymmetric topology such as Scott topology generated by a partial order $\sqsubseteq$ is often used in computer science to encode monotonic inference and limits of monotonic inference. For example, the upper topology on the real line consists of the open rays $(a,+\infty)$ (take $a=-\infty$ and $a=+\infty$ to represent the whole space and the empty set). This topology encodes the processes generating monotonically non-descreasing sequences of reals, $x_1 \leq x_2 \leq \ldots$, and their limits.

Scott continuous functions are functions respecting this structure. More specifically, Scott continuous functions between two spaces with Scott topologies are monotonic functions preserving appropriately defined limits.

If there are two Scott topologies pointing into opposite directions, one can infer both upwards and downwards, thus enabling non-monotonic inference.

In our example, one can also consider the lower topology on the real line consisting of the open rays $(-\infty,b)$, and this is the second Scott topology, encoding the processes generating sequences $y_1 \geq y_2 \geq \ldots$. Switching between these two topologies one can encode non-monotonic sequences.

A space with two topologies is called a bitopological space, and a space with Scott topologies generated by $\sqsubseteq$ and $\sqsupseteq$ with certain additional properties is called a bicontinuous domain [14].

4.5 Order Reversal and the Domain of Arrows

If we have a bicontinuous domain $(X,\sqsubseteq)$, then the dual space $X^{Op}=X^*=(X,\sqsupseteq)$ is also a bicontinuous domain.

If we think informally about an arrow from space X to space Y, then our intuition tells us that the arrow is greater if it "points more upwards", that is, if its right end is higher, and its left end is lower.

Formalizing this intuition we define the space of arrows from X to Y as $X^* \times Y$.

If we consider real numbers $\mathbb{R}$ with the standard order, $\sqsubseteq=\leq$, then partially inconsistent interval numbers are a space of arrows pointing from the right ends of the segments to the left ends of the segments, $\mathbb{R} \times \mathbb{R}^*$.

If $\mathbb{R}$ is modified to become a domain, R (by adding $-\infty$ and $+\infty$ as values or by taking a finite segment), we call $R \times R^*$ a domain of arrows.

There are two ways to describe Scott topology in terms of generalized distances. One is via asymmetric quasi-metrics, with $d(x,y)=0$ if and only if $x \sqsubseteq y$. Another is via dropping the $d(x,x)=0$ requirement which leads to relaxed and partial metrics. Quasi-metrics of this kind are monotonic with respect to one of the variables and anti-monotonic with respect of another variable. So the only way to have these generalized distances to be Scott continuous as functions from $X \times X$ to the domain representing distances is via the route of relaxed and partial metrics [5].

In the bicontinuous situation, quasi-metrics can be understood as Scott continuous functions from the domain of arrows, $X^* \times X$, to the domain representing distances.

Order-reversing involutions ($x \sqsubseteq y \Leftrightarrow f(y) \sqsubseteq f(x)$ and $f(f(x))=x$) play a prominent role in this context. From the viewpoint of domain theory, order-reversing involutions should be thought of as functions $X \to X^*$ (or $X^* \to X$). Hence order-reversing involutions are functions $X^* \times X \to X \times X^*$ (and vice versa) on the domain of arrows.

4.6 Further Mathematical Aspects

There are at least three ways bitopologies occur in studies of partial inconsistency. The connections between partial inconsistency and bitopological Stone duality via the notion of d-frame are explored in [13]. A fuzzy bitopology valued in lattice L is a fuzzy topology valued in the bilattice L^2 (in particular, an ordinary bitopology is a topology valued in the four-valued logic) [28]. Finally, in the context of bitopological groups and anti-monotonic inverse the following situation is typical: two topologies, T and T^{-1}, are group dual of each other, the multiplication is continuous with respect to both topologies, and the inverse is a bicontinuous map from (X,T,T^{-1}) to its bitopological dual, (X,T^{-1},T) [1].

In particular, partially inconsistent interval numbers over reals extended with $\pm\infty$ are isomorphic to the d-frame of the (lower, upper) bitoplogy on the reals, anti-monotonic inverses on the reals and on the partially inconsistent numbers themselves play prominent roles, and any real-valued fuzzy bitopology can be represented as fuzzy topology valued in partially inconsistent interval numbers (see [4] for details).

In general, the paraconsistent equivalent of real-valued fuzzy mathematics is mathematics valued in partially inconsistent interval numbers.

It is natural to ascribe negative length $b-a$ to pseudosegments and to associate with them generalized characteristic functions taking the value -1 in the (b,a) interval and 0 outside of that interval (signed multisets allowing negative degree of membership).

4.7 Computational Models with Involutions

We are currently looking at various computational models involving involutions.

Given a domain of arrows $X \times X^*$, a sequence $(x_1,y_1),(x_2,y_2),\ldots$ is called a monotonic sequence with involutive steps, if for any $n \in \mathbb{N}$ either $x_n \sqsubseteq x_{n+1}$ and $y_n \sqsupseteq y_{n+1}$ (in which case the step n is called monotonic) or $x_n = y_{n+1}$ and $y_n = x_{n+1}$ (in which case the step n is called an involution).

One can define a notion of convergence robust with respect to the insertion of pairs of involutive steps and prove that if (x,y) is a limit under this notion, then $x = y$.

Architectures based on involutive steps are rather prominent in the context of reversible and quantum computations. For example, the well-known Grover's quantum algorithm can be described as a sequence of reflections of subsets of a plane [27].

Architectures where the state of an abstract machine is an image on the plane, and an involutive computational step selects a line on this plane and a subset symmetric with respect to this line, and performs a reflection of the image within this subset, seem to be quite attractive in the context of classical computations as well.

5 Conclusion

It is possible to talk about linear combinations of probabilistic programs when their semantics is expressed as linear operators [16].

For $0 < \alpha < 1$ and **random** being a generator of uniformly distributed reals between 0 and 1, the linear operator corresponding to the program **if random** $< \alpha$ **then P else Q** is a linear combination of the linear operators corresponding to programs **P** and **Q** with coefficients α and $1-\alpha$.

However, when one aims for better schemes of program learning, the situations where one can consider linear combinations of single execution runs rather than linear combinations of the overall program meanings should be especially attractive. In this paper we consider two such architectures, probabilistic sampling and generalized animation, and the recent progress in this field looks very promising.

We give an overview of mathematical material tightly connected to linear models of computations via the partial inconsistency landscape, because this material is likely to grow in relevance as uses of linear models of computations are further explored.

We would like to conclude by describing a possible hybrid approach to program learning. Instead of implementing everything in terms of architectures admitting linear combinations of single execution runs one can use a hybrid approach, mixing these architectures and traditional software. In this context we might be inspired by hybrid hardware connecting live neural tissue and electronic circuits.

One might decide to use large existing software components and try to automate the process of connecting them together using flexible probabilistic connectors. Here one should note the progress in automated generation of test suites for software systems.

Another possible approach is to attempt the use of small inflexible components inside the flexible "tissue" of linear models.

Acknowledgments. We would like to thank Ralph Kopperman, Leon Peshkin, Josh Tenenbaum, and Levy Ulanovsky for helpful discussions.

References

1. Andima, S., Kopperman, R., Nickolas, P.: An Asymmetric Ellis Theorem. Topology and Its Applications 155, 146–160 (2007)
2. Berlinet, A., Thomas-Agnan, C.: Reproducing Kernel Hilbert Spaces in Probability and Statistics. Kluwer Academic Publishers, Boston (2001)
3. Bukatin, M., Kopperman, R., Matthews, S.: Partial Inconsistency Landscape: an Overview. 28th Summer Conference on Topology and Its Applications, Nipissing University, July 2013, `http://at.yorku.ca/cgi-bin/abstract/cbgy-77`
4. Bukatin, M., Kopperman, R., Matthews, S.: Progress Report on Partial Inconsistency, Bitopology, and Vector Semantics. Conference on Computational Topology and Its Applications, Kent State University, Ohio, November 2014, `http://www.cs.brandeis.edu/~bukatin/PartialInconsistencyProgressNov2014.pdf`
5. Bukatin, M., Scott, J.: Towards Computing Distances Between Programs via Scott Domains. In: Adian, S., Nerode, A. (eds.), Logical Foundations of Computer Science, LNCS, vol. 1234, pp.33–43. Springer, Heidelberg (1997)
6. Carroll, S.: Evo-Devo and an Expanding Evolutionary Synthesis: A Genetic Theory of Morphological Evolution. Cell 134, 25–36 (2008)
7. Draves, S.: The Electric Sheep Screen-Saver: A Case Study in Aesthetic Evolution. In: Rothlauf, F. et al (eds.), Applications of Evolutionary Computing, LNCS, vol. 3449, pp.458–467, Springer, Heidelberg (2005), `http://draves.org/evomusart05/`
8. Feynman, R.: Negative Probability. In: Peat, F., Hiley, B. (eds.), Quantum Implications : Essays in Honour of David Bohm, pp. 235–248, Routledge and Kegan Paul, London and New York (1987)

9. Gerhart, J., Kirschner, M.: The Theory of Facilitated Variation, Proc. Natl. Acad. Sci. 104(Suppl 1), 8582–8589 (2007)
10. Ginsberg, M.: Multivalued Logics: a Uniform Approach to Inference in Artifcial Intelligence. Computational Intelligence 4(3), 256–316 (1992)
11. Groenewold, H.: On the Principles of Elementary Quantum Mechanics, Physica 12, 405–460 (1946)
12. Grosse, R., Salakhutdinov, R., Freeman, W., Tenenbaum, J.: Exploiting Compositionality to Explore a Large Space of Model Structures. Conference on Uncertainty in Artificial Intelligence (2012). Arxiv preprint at `http://arxiv.org/abs/1210.4856`
13. Jung, A., Moshier, M.A.: On the Bitopological Nature of Stone Duality. Technical Report CSR-06-13. School of Computer Science, University of Birmingham (2006)
14. Keimel, K.: Bicontinuous Domains and Some Old Problems in Domain Theory. Electronic Notes in Theoretical Computer Science 257, 35–54 (2009)
15. Kirschner, M., Gerhart, J.: The Plausibility of Life: Resolving Darwin's Dilemma. Yale University Press (2005)
16. Kozen, D.: Semantics of Probabilistic Programs. Journal of Computer and System Sciences 22 (3), 328–350 (1981)
17. Lake, B.: Towards More Human-like Concept Learning in Machines: Compositionality, Causality, and Learning-to-learn. PhD Thesis, MIT (2014), `http://cims.nyu.edu/~brenden/LakePhDThesis.pdf`
18. Looks, M.: Competent Program Evolution. PhD Thesis, Washington University in St. Louis (2006), http://metacog.org/doc.html
19. Maass, W.: Noise as a Resource for Computation and Learning in Networks of Spiking Neurons. Special Issue of the Proc. of the IEEE on "Engineering Intelligent Electronic Systems based on Computationa Neuroscience" 102(5), 860–880 (2014)
20. Mansinghka, V., Kulkarni, T., Perov, Yu., Tenenbaum, J.: Approximate Bayesian Image Interpretation using Generative Probabilistic Graphics Programs (2013). `http://probcomp.csail.mit.edu/gpgp`
21. Marr, D.: Vision. W. H. Freeman and Company, New York (1982)
22. Miquel, C., Paz, J., Saraceno M.: Quantum Computers in Phase Space, Phys. Rev. A 65, 062309 (2002). Arxiv preprint at `http://arxiv.org/abs/quant-ph/0204149`
23. Moyal, J.: Quantum Mechanics as a Statistical Theory, Proceedings of the Cambridge Philosophical Society 45, 99–124 (1949)
24. Pelikan, M.: Bayesian Optimization Algorithm: from Single Level to Hierarchy. PhD Thesis, University of Illinois at Urbana-Champaign (2002), http://www.medal-lab.org/files/2002023.pdf
25. Perov, Yu., Wood, F.: Learning Probabilistic Programs, Preprint (2014), `http://arxiv.org/abs/1407.2646`
26. Popova, E.: The Arithmetic on Proper & Improper Intervals (a Repository of Literature on Interval Algebraic Extensions), `http://www.math.bas.bg/~epopova/directed.html`
27. Rieffel. E., Polak, W.: Quantum Computing: A Gentle Introduction. MIT Press (2011)
28. Rodabaugh, S.: Functorial Comparisons of Bitopology with Topology and the Case for Redundancy of Bitopology in Lattice-valued Mathematics. Applied General Topology 9(1), 77–108 (2008)