A Higher-Order Quantitative Logic

Giorgio Bacci
Aalborg University, Denmark

Rasmus Ejlers Møgelberg
IT University of Copenhagen, Denmark

Abstract

We present a sensitivity-aware lambda-calculus for programming in a category of complete 1-bounded metric spaces and a quantitative higher-order logic for reasoning about programs. The calculus has a monad for distributions as well as a guarded fixed point operator interpreted using the Banach fixed point theorem. The logic is valued in the unit interval, and equality is interpreted as distance between points. The logic has a recursion principle for distributions and a guarded recursion principle. We show how these can be used in combination to reason about bisimilarity distance between Markov processes. We also show how to encode a version of the Kantorovich distance in the logic which can be proved equivalent to equality. This encoding can be used for reasoning using couplings.

2012 ACM Subject Classification Theory of computation → Logic and verification; Theory of computation → Random walks and Markov chains; Theory of computation → Higher order logic

Keywords and phrases Sensitivity aware lambda calculus, quantitative logic, guarded recursion, Markov processes, bisimilarity distance, couplings

Funding *Rasmus Ejlers Møgelberg*: This work was partially funded by the Independent Research Fund Denmark, grant number 2032-00134B.

1 Introduction

Quantitative reasoning about programs is concerned with program distances. The goal is often to establish upper bounds to program distances, reasoning about the sensitivity of program outputs to program inputs, or to show convergence of sequences of programs. There has been a lot of research recently in developing logics for quantitative reasoning, but much of this has focused on developing program logics for imperative languages [5, 11, 1, 2] with the aim of reasoning about increasingly advanced examples. Here we take a step back and study the basic principles needed for quantitative reasoning, in particular, induction and recursion principles.

At its core, quantitative logic is a logic of metric spaces. There are two approaches to quantitative logic: One is Quantitative Equational Logic [9], a Boolean valued logic with predicates of the form $x =_\epsilon y$, to be interpreted as 'the distance between x and y is at most ϵ'. Note that only equalities carry quantitative information. The other approach is to work with a single equality predicate to be interpreted as the distance between elements. The result is a logic valued in the positive reals in which 0 is true and the judgement of implication is interpreted as the $\geq$ relation. Upper limits to distances between programs can then be encoded as equality in contexts. For example $c \vdash t = u$ is interpreted as 'the distance between t and u is at most c'. The former approach seems to have received most attention, although the latter has been the subject of some papers lately [3, 7]. In this talk we take the latter approach, because we believe it is elegant and because it allows for a powerful guarded recursion principle.

Here are two immediate observations: The first is that quantitative logic is naturally an affine logic. The reason is that in order to have transitivity $x = y, y = z \vdash x = z$, one needs to interpret comma as sum, which then reduces transitivity to the triangle inequality. Recall that in affine logic weakening is an admissible rule, but contraction is not. The second

observation is that the standard form of congruence ($u = s \vdash t[u/x] = t[s/x]$) is not true. Take for example t to be $2x$. Since this term doubles the distances between its inputs, it satisfies $2(x = y) \vdash 2x = 2y$. As a consequence, we need to account for sensitivity of terms in variables to express congruence.

The talk therefore describes the following: A sensitivity-aware lambda calculus for terms interpreted in a category of metric spaces, as well as a logic for reasoning about these terms. The talk is based on the recent manuscript [4].

2 A Category of Metric Spaces

We work in a category **CMet** of complete 1-bounded metric spaces and non-expansive maps (i.e., functions f that do not increase distances: $d(f(x), f(y)) \leq d(x, y)$). The restriction of 1-boundedness means that the distance $d(x, y)$ between any two elements in a metric space is at most 1. One reason for this restriction is that it allows us to consider sets as discrete metric spaces, where all distances are either 0 or 1. Any map out of a discrete space is non-expansive, and as a consequence, one can give a recursion principle for discrete natural numbers in **CMet**, and similarly for other recursive types.

The category **CMet** is complete, but more importantly, it carries a monoidal structure $\otimes$, where the underlying set of $X \otimes Y$ is the Cartesian product, and

$$d_{X\otimes Y}((x, y), (x', y')) = \min\{d(x, x') + d(y, y'), 1\}\,.$$

This monoidal structure is moreover closed, meaning that the functor $- \otimes X$ of forming the monoidal product with X has a right adjoint $X \multimap -$. The underlying set of $X \multimap Y$ is the set of non-expansive functions from X to Y.

$$d_{X\multimap Y}(f, g) = \sup_{x\in X} d(f(x), d(g(x)))\,.$$

There is also an operation of scaling a metric space by a factor. The underlying set of cX is X with metric

$$d_{cX}(x, x') = \min\{cd(x, x'), 1\}$$

This operation is well-defined for $c \in (0, \infty)$ and can be extended to $[0, \infty]$ by defining $0X$ to be the one point set for $X \neq \emptyset$ and $0 \cdot \emptyset = \emptyset$, and ∞X to be the discrete space on X. The Banach fixed point theorem can then be expressed as follows.

▶ **Theorem 1.** *Let X be a complete non-empty metric space, and let $f : cX \to X$ be non-expansive for $c < 1$. Then f has a unique fixed point. Moreover, if $g : (1 - c)Y \otimes cX \to X$ is non-expansive, the map mapping y to the fixed point of $g(y, -)$ is a non-expansive map $Y \to X$.*

The above formulation of the Banach fixed point theorem is related to the type of a guarded fixed point operator $(\triangleright X \to X) \to X$, as modelled in the topos of trees [6], or using ultra-metric spaces [8], where $\triangleright$ as here, is scaling by a factor c. To our knowledge, this has not been considered for general metric spaces.

Finally, we consider the set $\mathcal{D}X$ of Radon probability measures on a metric space X. This can be equipped with the Kantorovich metric to give a monad on **CMet**, called *Radon probability monad*, with functor acting on morphisms as the pushforward measure along the given function. The unit is the Dirac measure $\delta_X : X \to \mathcal{D}X$, but rather than describing the multiplication, we recall that this monad has an algebraic presentation as the free complete *interpolative barycentric algebra* [9, 10].

▶ **Definition 2** (IB Algebra). *A (complete)* interpolative barycentric algebra *is a complete metric space X with morphisms $\oplus_p \colon pX \otimes (1-p)X \to X$, for all $p \in (0,1)$, such that*

$$x \oplus_p x = x \tag{IDEM}$$
$$x \oplus_p y = y \oplus_{1-p} x \tag{COMM}$$
$$(x \oplus_p y) \oplus_q z = x \oplus_{pq} (y \oplus_{\frac{q-pq}{1-pq}} z) \tag{ASSOC}$$

A homomorphism of IB algebras is a non-expansive map such that $f(x \oplus_p y) = f(x) \oplus_p f(y)$ holds for all $p \in (0,1)$.

The axioms are those of barycentric algebras (a.k.a., convex algebras), axiomatizing probabilistic choice through binary convex combination operations $x \oplus_p y$. For the Radon monad, the operation $\oplus_p : p\mathcal{D}X \otimes (1-p)\mathcal{D}X \to \mathcal{D}X$ is simply the convex combination of probability distributions. The fact that $\mathcal{D}X$ is the free complete IB algebra provides us with the following principle of structural induction on Radon distributions on complete metric spaces.

▶ **Proposition 3.** *If $f \colon \Gamma \otimes rX \to Y$ (with $r < \infty$) and Y is an IB algebra, there exists a unique $\overline{f} \colon \Gamma \otimes r\mathcal{D}X \to Y$ which is a homomorphism in its second argument, satisfying $f = \overline{f} \circ (\Gamma \otimes r\delta_X)$.*

The condition $r < \infty$ implies that f is continuous in the second argument, which is required to guarantee the existence of the homomorphic extension $\overline{f}$. We shall see that this principle is the basis for our logical induction rule on Radon distributions over complete metric spaces.

3 A Sensitivity Aware λ-Calculus

We define a sensitivity-aware λ-calculus for programming in **CMet**. The syntax, summarised below, is based on a λ-calculus with product, with a few modifications.

$$\begin{aligned} t, u ::= {} & x \mid \lambda x.t \mid tu \mid \langle t, u\rangle \mid \pi_1 t \mid \pi_2 t \\ & \mid (t,u) \mid \mathsf{let}\ (x,y) = u\ \mathsf{in}\ t \\ & \mid \delta t \mid t \oplus_p u \mid \mathsf{let}\ x = u\ \mathsf{in}\ t \\ & \mid \mathsf{fix}\ x.t \end{aligned}$$

There are two pairs constructors, $\langle t, u\rangle$ and (t,u), corresponding to the Cartesian and monoidal product, respectively. The first one is eliminated using the projections $\pi_i t$, whereas the second one is eliminated using case analysis ($\mathsf{let}\ (x,y) = u\ \mathsf{in}\ t$). The constructors δt and $t \oplus_p u$ are used to form distributions by means of Dirac (delta) distributions and convex combinations; Probability distributions are then sampled using ($\mathsf{let}\ x = u\ \mathsf{in}\ t$). $\mathsf{fix}\ x.t$ is the "Banach" fixed point combinator.

Terms are typed with judgments of the form $\Gamma \vdash t : A$, where Γ is a typing context and A is a type. The types are as follows:

$$A, B ::= b \mid A \times B \mid A \,{}_p\!\otimes_q B \mid A \multimap_p B \mid \mathcal{D}A$$

essentially, corresponding to the constructions of the previous section, where b ranges over a collection of base types. These are interpreted in metric spaces as

$$[\![A \times B]\!] \triangleq [\![A]\!] \times [\![B]\!]\,, \qquad [\![\mathcal{D}A]\!] \triangleq \mathcal{D}[\![A]\!]\,,$$
$$[\![A \,{}_p\!\otimes_q B]\!] \triangleq p[\![A]\!] \otimes q[\![B]\!]\,, \qquad [\![A \multimap_p B]\!] \triangleq p[\![A]\!] \multimap [\![B]\!]\,.$$

$$\frac{p \geq 1}{\Gamma, x :^p A, \Gamma' \vdash x : A} \qquad \frac{\Gamma, x :^p A \vdash t : B}{\Gamma \vdash \lambda x.t : A \multimap_p B} \qquad \frac{\Gamma \vdash t : A \multimap_p B \quad \Gamma' \vdash u : A}{\Gamma + p\Gamma' \vdash t\, u : B}$$

$$\frac{\Gamma \vdash t : A \quad \Gamma \vdash u : B}{\Gamma \vdash \langle t, u \rangle : A \times B} \qquad \frac{\Gamma \vdash t : A_1 \times A_2}{\Gamma \vdash \pi_i t : A_i}$$

$$\frac{\Gamma \vdash t : A \quad \Gamma' \vdash u : B}{p\Gamma + q\Gamma' + \Gamma'' \vdash (t, u) : A \,{}_p\!\otimes_q B} \qquad \frac{\Gamma, x :^p A, y :^q B \vdash t : C \quad \Gamma' \vdash u : A \,{}_p\!\otimes_q B}{\Gamma + \Gamma' \vdash \mathsf{let}\ (x, y) = u\ \mathsf{in}\ t : C}$$

$$\frac{\Gamma \vdash t : A}{\Gamma \vdash \delta t : \mathcal{D}A} \qquad \frac{\Gamma \vdash t : \mathcal{D}A \quad \Gamma' \vdash u : \mathcal{D}A \quad p \in (0,1)}{p\Gamma + (1-p)\Gamma' \vdash t \oplus_p u : \mathcal{D}A}$$

$$\frac{\Gamma, x :^r A \vdash t : E \quad \Gamma' \vdash u : \mathcal{D}A \quad E \text{ IB algebra} \quad r < \infty}{\Gamma + r\Gamma' \vdash \mathsf{let}\ x = u\ \mathsf{in}\ t : E} \qquad \frac{(1-p)\Gamma, x :^p A \vdash t : A \quad p < 1}{\Gamma \vdash \mathsf{fix}\ x.t : A}$$

Figure 1 Typing rules.

Although rescaling of metric spaces played a central role in the previous section, it is not a primitive type former in the calculus. Instead, it is part of the tensor type $A \,{}_p\!\otimes_q B$ and function type $A \multimap_p B$ constructors. This choice was made to minimize the bookkeeping necessary for scalars in terms.

The sensitivity of each variable used in a term is tracked by sensitivity annotations of the form $x :^p A$ in typing contexts Γ. A binding $x :^p A$ in a context Γ means that the variable x has type A under Γ and that terms typed under Γ are p-sensitive with respect to x. In this sense, the language is strongly related to Fuzz [12]. Formally, typing contexts are formed according to the rules below, with expected interpretation in metric spaces:

$$\frac{}{\langle\rangle :: \mathsf{ctx}}\,, \qquad \frac{\Gamma :: \mathsf{ctx} \quad x \notin \Gamma \quad p \in [0, \infty]}{\Gamma, x :^p A :: \mathsf{ctx}}\,; \qquad [\![\langle\rangle]\!] \triangleq \mathbf{1}\,, \quad [\![\Gamma, x :^p A]\!] \triangleq [\![\Gamma]\!] \otimes p[\![A]\!]\,.$$

Terms in context then define non-expansive maps

$$[\![\Gamma \vdash t : A]\!] : [\![\Gamma]\!] \to [\![A]\!]\,.$$

The typing rules shown in Figure 1 reflect the semantic results mentioned above, where scaling and sum of contexts are defined using point-wise scaling and sum of sensitivity factors. Observe that $\Gamma + \Gamma'$ is only defined when Γ and Γ' agree on the order and the types of all variable bindings —for example, $(x :^p A, y :^q B) + (y :^q B, x :^p A)$ is not defined.

The rule for variable introduction reflects that projection can be given any Lipschitz factor $p \geq 1$. We allow all such p, and not just $p = 1$, to incorporate weakening directly into the typing rules. In the rule for lambda abstraction, the sensitivity on the function type matches that of the variable to be bound. Indeed, a term of type $A \multimap_p B$ denotes a function with Lipschitz factor p. When applying such a term to an argument in the rule for function application, the sensitivity factor for the argument must be scaled accordingly. In the rule for tensor introduction, Γ'' is used merely to incorporate weakening in the rule, also in the case where both p and q are 0. The use of Γ and Γ' is more interesting: they are combined linearly in the conclusion because so are the distances in the tensor type. Compare this to the rules for Cartesian product, where the context in the conclusion is required to be the same as in the premise. The elimination rule for $\mathcal{D}$ uses the judgment of a type being an IB algebra and reflects Proposition 3. The type of the fixed point combinator reflects Theorem 1.

4 The logic

We next define a logic for reasoning about terms of the language described above. Recall that this logic should be real valued and equality should be interpreted as distance. Since the metric spaces considered are 1-bounded, we take as semantic universe of propositions the closed unit interval $[0,1]$. Since this is an object of **CMet**, we will use the lambda calculus to specify the notion of well-formed proposition. Simply add a type Prop to the language interpreted as $[0,1]$ as well as operations for forming propositions such as

$$\frac{\Gamma \vdash t : A \quad \Gamma' \vdash s : A}{\Gamma + \Gamma' \vdash t = s : \mathsf{Prop}} \qquad \frac{\Gamma \vdash \varphi : \mathsf{Prop} \quad \Gamma' \vdash \psi : \mathsf{Prop}}{\Gamma + \Gamma' \vdash \varphi \bullet \psi : \mathsf{Prop}} \qquad \frac{\Gamma \vdash \varphi : \mathsf{Prop} \quad \Gamma' \vdash \psi : \mathsf{Prop}}{\Gamma + \Gamma' \vdash \varphi \multimap \psi : \mathsf{Prop}}$$

where

$$[\![t = s]\!] = d([\![t]\!], [\![s]\!]), \quad [\![\varphi \bullet \psi]\!] = \min\{[\![\varphi]\!] + [\![\psi]\!], 1\}, \quad [\![\varphi \multimap \psi]\!] = \max\{[\![\varphi]\!] - [\![\psi]\!], 0\},$$

plus rules for scaling propositions, disjunction, conjunction, and universal as well as existential quantification.

The logical judgement $\Delta \mid \Psi \vdash \phi$ states that the sequence of logical formulas Ψ implies ϕ, presupposing that Ψ and ϕ are well-formed (sequences of) terms of type Prop in context Δ. Here Δ is assumed to be a context in which all sensitivities are set to ∞. The reason for this is that sensitivity factors for terms variables in logical predicates are irrelevant for logical judgments, and keeping track of these in logical judgements adds unnecessary complications to the logic. Note that, ∞ is the most general sensitivity annotation possible: If $\Delta \vdash t : A$ then also $\Delta' \vdash t : A$ where Δ' is obtained from Δ by setting all sensitivity annotations to ∞. In logical judgements, we use the notation $\Delta, x : A$ as shorthand for the rigorous $\Delta, x :^{\infty} A$. (This notation is justified by the fact that discrete contexts act essentially as ordinary set-contexts).

Key rules include those for equality

$$\frac{\Delta \vdash t : A}{\Delta \mid \Psi \vdash t = t} \qquad \frac{\Delta, x :^{p} A \vdash \varphi : \mathsf{Prop} \quad \Delta \vdash s, t : A \quad \Delta \mid \Psi \vdash \varphi[t/x] \quad \Delta \mid \Psi' \vdash p(t = s)}{\Delta \mid \Psi, \Psi' \vdash \varphi[s/x]}$$

of which the second is essentially due to Dagnino and Pasquali [7]. We observe that these are strong enough to prove that equality is a congruence relation, in the sensitivity-aware sense.

Another key rule is the induction rule for distributions

$$\frac{\begin{array}{cc} \Delta \vdash t : \mathcal{D}A & \Delta, y : A \mid \Psi \vdash \varphi[\delta y/x] \\ \Delta, x :^{r} \mathcal{D}A \vdash \varphi : \mathsf{Prop} & \Delta, \mu : \mathcal{D}A, \nu : \mathcal{D}A \mid p\varphi[\mu/x], (1-p)\varphi[\nu/x] \vdash \varphi[\mu \oplus_p \nu/x] \end{array} \quad r < \infty}{\Delta \mid \Psi \vdash \varphi[t/x]}$$

Intuitively, the rule states that to prove that $\varphi[t/x]$ holds for any Radon distribution $t : \mathcal{D}A$, it suffices to show the proposition holds for t generic Dirac distributions (the base case), and that property is preserved by convex combinations of distributions (the inductive step). This rule is sound by Proposition 3 as Prop is an IB algebra.

Finally, the logic contains a guarded recursion principle

$$\frac{\Delta \mid (1-p)\Psi, p\varphi \vdash \varphi \quad p \in (0,1)}{\Delta \mid \Psi \vdash \varphi}$$

Note the similarity between this rule and the typing rule for fixed points.

5 Examples

We start by showing how the fixed point combinator can be used to define recursive Markov processes such as one satisfying

$$m \equiv a; (\delta(m) \oplus_{\frac{1}{3}} \delta(z)) \tag{1}$$

The syntax is to be read as 'emit label a, then continue as m with probability $\frac{1}{3}$ and as process z with probability $\frac{2}{3}$.

Following [13], Markov processes of this type coalgebras of type $S \to A \otimes c\mathcal{D}(S)$ in **CMet**, where $c \in (0, 1]$ is some discount factor and A a metric space of labels. The behaviour of a Markov process can be abstractly characterised as an element of the final coalgebra, corresponding to the coinductive solution $\mathbb{P}_c$ to the functorial equation $\mathbb{P}_c \cong A \otimes c\mathcal{D}(\mathbb{P}_c)$. The behavioural distance is just the distance in $\mathbb{P}_c$ between behaviours [14].

In order to program with $\mathbb{P}_c$ we add to the calculus the basic types $\mathbb{P}_c$ and A and terms

$$\mathsf{ufld} : \mathbb{P}_c \multimap_1 A \,{}_1\!\otimes_c \mathcal{D}(\mathbb{P}_c) \qquad\qquad \mathsf{fld} : A \,{}_1\!\otimes_c \mathcal{D}(\mathbb{P}_c) \multimap_1 \mathbb{P}_c$$

We will write $a; m$ for $\mathsf{fld}\,(a, m)$.

The recursive definition (1) is productive in the sense that it only calls itself with probability $\frac{1}{3}$. Therefore, it can be defined as a term of type $\mathbb{P}_1$. Precisely, because

$$z :^{\frac{2}{3}} \mathbb{P}_1, m :^{\frac{1}{3}} \mathbb{P}_1 \vdash a; (\delta(m) \oplus_{\frac{1}{3}} \delta(z)) : \mathbb{P}_1$$

we can define $z :^1 \mathbb{P}_1 \vdash m : \mathbb{P}_1$ as $m \triangleq \mathsf{fix}\ m.a; (\delta(m) \oplus_{\frac{1}{3}} \delta(z))$.

Next we consider a simple example of reasoning in our logic. Consider the two recursive definitions of Markov processes

$$m \equiv a; (\delta(m) \oplus_{\frac{1}{3}} \delta(z)) \qquad\qquad n \equiv a; (\delta(n) \oplus_{\frac{1}{2}} \delta(z))$$

which can both be defined using the fixed point operator of the language. We show that the distance between these is at most $\frac{1}{4}$, which can be expressed in our logic as

$$\frac{1}{4}\mathsf{ff} \vdash m = n$$

This is done by guarded recursion by showing

$$\frac{1}{3}(m = n), \left(\frac{2}{3} \cdot \frac{1}{4}\right) \mathsf{ff} \vdash m = n$$

Since

$$m \equiv a; (\delta(m) \oplus_{\frac{1}{3}} (\delta(z) \oplus_{\frac{1}{4}} \delta(z))) \qquad\qquad n \equiv a; (\delta(n) \oplus_{\frac{1}{3}} (\delta(n) \oplus_{\frac{1}{4}} \delta(z)))$$

by the congruence property, this boils down to showing

$$\frac{1}{3}(m = n), \left(\frac{2}{3} \cdot \frac{1}{4}\right) \mathsf{ff} \vdash \frac{1}{3}(m = n) \bullet \frac{2}{3}\left(\frac{1}{4}(n = z) \bullet \frac{3}{4}(z = z)\right)$$

which is easy.

5.1 Internalising the Kantorovich Distance

If $\phi : A \multimap \mathsf{Prop}$ is a predicate with sensitivity 1, and $\mu : \mathcal{D}A$ we can consider $\mathcal{D}\phi(\mu) : \mathcal{D}(\mathsf{Prop})$, which is essentially a random variable. We can compute the mean of this internally in the language as

$$E_{x\sim\mu}[\phi] \triangleq \mathsf{let}\ x = \mu\ \mathsf{in}\ \phi(x)$$

The use of the word 'mean' is justified by the following equations

$$E_{x\sim\delta(y)}[\phi] \equiv \phi(y)$$
$$E_{x\sim\mu\oplus_p\mu'}[\phi] \equiv p(E_{x\sim\mu}[\phi]) \bullet (1-p)(E_{x\sim\mu'}[\phi])$$

If $\omega : \mathcal{D}(A\,{}_1\otimes_1 A)$, the predicate of ω being a coupling between distributions μ and ν can be expressed internally in the logic as

$$\omega \in \mathsf{Cpl}(\mu,\nu) \triangleq (\mathcal{D}(\pi_1)\omega = \mu) \bullet (\mathcal{D}(\pi_2)\omega = \nu)$$

Note that this is a quantitative statement, but it is globally true (i.e., $\mathsf{tt} \vdash \omega \in \mathsf{Cpl}(\mu,\nu)$) iff ω is a coupling in the usual sense. We can now define the Kantorovich distance as

$$K(\mu,\nu) \triangleq \exists\omega.\omega \in \mathsf{Cpl}(\mu,\nu) \bullet E_{(x,y)\sim\omega}[x{=}y]$$

▶ **Theorem 4.** *The predicates $K(\mu,\nu)$ and $\mu = \nu$ are equivalent.*

In the talk, we will sketch how this can be used to reason about programs by use of the coupling method in our logic. The concrete example is a random walk on an N-dimensional hypercube $\mathsf{Pos} \triangleq \mathsf{Bool}^N$, where the distance between $p, q \in \mathsf{Pos}$ is $\frac{1}{N}$ times the number of positions where p and q differ. We show that

$$\frac{N-1}{N+1}(p = q) \vdash \mathsf{hwalk}\,p = \mathsf{hwalk}\,q \tag{2}$$

where $\mathsf{hwalk} : \mathsf{Pos} \multimap_1 \mathcal{D}(\mathsf{Pos})$ is one step of the random walk. By the Banach fixed point theorem, this implies that the random walk converges to a fixed distribution, regardless of the initial position.

Showing this requires relating the random choices made in $\mathsf{hwalk}\,p$ to those of $\mathsf{hwalk}\,q$ by means of a coupling. This is done by constructing a term $\omega : \mathcal{D}(\mathsf{Pos}\,{}_1\otimes_1\mathsf{Pos})$ and showing that

$$\frac{N-1}{N+1}(p = q) \vdash \exists\omega.\omega \in \mathsf{Cpl}(\mathsf{hwalk}\,p, \mathsf{hwalk}\,q) \bullet E_{(x,y)\sim\omega}[x{=}y]$$

which by Theorem 4 implies (2).

The manuscript [4] shows the following further examples: An upper bound on the distance between processes describing a biased coin toss and a fair one, equivalence of bisimulation and equality for a type of Markov processes with a discounting factor below 1, as well as convergence for a temporal difference learning algorithm.

References

1 Alejandro Aguirre, Gilles Barthe, Justin Hsu, Benjamin Lucien Kaminski, Joost-Pieter Katoen, and Christoph Matheja. A pre-expectation calculus for probabilistic sensitivity. *Proc. ACM Program. Lang.*, 5(POPL):1–28, 2021. doi:10.1145/3434333.

2 Martin Avanzini, Gilles Barthe, Davide Davoli, and Benjamin Grégoire. A quantitative probabilistic relational hoare logic. *Proc. ACM Program. Lang.*, 9(POPL):1167–1195, 2025. doi:10.1145/3704876.

3 Giorgio Bacci, Radu Mardare, Prakash Panangaden, and Gordon D. Plotkin. Propositional logics for the lawvere quantale. In Marie Kerjean and Paul Blain Levy, editors, *Proceedings of the 39th Conference on the Mathematical Foundations of Programming Semantics, MFPS XXXIX, Indiana University, Bloomington, IN, USA, June 21-23, 2023*, volume 3 of *EPTICS*. EpiSciences, 2023. URL: https://doi.org/10.46298/entics.12292, doi:10.46298/ENTICS.12292.

4 Giorgio Bacci and Rasmus Ejlers Møgelberg. Induction and recursion principles in a higher-order quantitative logic. *CoRR*, abs/2501.18275, 2025. URL: https://doi.org/10.48550/arXiv.2501.18275, arXiv:2501.18275, doi:10.48550/ARXIV.2501.18275.

5 Gilles Barthe, Benjamin Grégoire, and Santiago Zanella-Béguelin. Formal certification of code-based cryptographic proofs. In Zhong Shao and Benjamin C. Pierce, editors, *Proceedings of the 36th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL 2009, Savannah, GA, USA, January 21-23, 2009*, pages 90–101. ACM, 2009. doi:10.1145/1480881.1480894.

6 Lars Birkedal, Rasmus Ejlers Møgelberg, Jan Schwinghammer, and Kristian Støvring. First steps in synthetic guarded domain theory: step-indexing in the topos of trees. *Log. Methods Comput. Sci.*, 8(4), 2012. doi:10.2168/LMCS-8(4:1)2012.

7 Francesco Dagnino and Fabio Pasquali. Logical foundations of quantitative equality. In Christel Baier and Dana Fisman, editors, *LICS '22: 37th Annual ACM/IEEE Symposium on Logic in Computer Science, Haifa, Israel, August 2 - 5, 2022*, pages 16:1–16:13. ACM, 2022. doi:10.1145/3531130.3533337.

8 Erik P. de Vink and Jan J. M. M. Rutten. Bisimulation for probabilistic transition systems: A coalgebraic approach. *Theor. Comput. Sci.*, 221(1-2):271–293, 1999. doi:10.1016/S0304-3975(99)00035-3.

9 Radu Mardare, Prakash Panangaden, and Gordon D. Plotkin. Quantitative algebraic reasoning. In Martin Grohe, Eric Koskinen, and Natarajan Shankar, editors, *Proceedings of the 31st Annual ACM/IEEE Symposium on Logic in Computer Science, LICS '16, New York, NY, USA, July 5-8, 2016*, pages 700–709. ACM, 2016. doi:10.1145/2933575.2934518.

10 Radu Mardare, Prakash Panangaden, and Gordon D. Plotkin. Free complete wasserstein algebras. *Log. Methods Comput. Sci.*, 14(3), 2018. doi:10.23638/LMCS-14(3:19)2018.

11 Federico Olmedo, Benjamin Lucien Kaminski, Joost-Pieter Katoen, and Christoph Matheja. Reasoning about recursive probabilistic programs. In Martin Grohe, Eric Koskinen, and Natarajan Shankar, editors, *Proceedings of the 31st Annual ACM/IEEE Symposium on Logic in Computer Science, LICS '16, New York, NY, USA, July 5-8, 2016*, pages 672–681. ACM, 2016. doi:10.1145/2933575.2935317.

12 Jason Reed and Benjamin C. Pierce. Distance makes the types grow stronger: a calculus for differential privacy. In Paul Hudak and Stephanie Weirich, editors, *Proceeding of the 15th ACM SIGPLAN international conference on Functional programming, ICFP 2010, Baltimore, Maryland, USA, September 27-29, 2010*, pages 157–168. ACM, 2010. doi:10.1145/1863543.1863568.

13 Franck van Breugel. A behavioural pseudometric for metric labelled transition systems. In *CONCUR*, volume 3653 of *Lecture Notes in Computer Science*, pages 141–155. Springer, 2005.

14 Franck van Breugel, Claudio Hermida, Michael Makkai, and James Worrell. An accessible approach to behavioural pseudometrics. In *ICALP*, volume 3580 of *Lecture Notes in Computer Science*, pages 1018–1030. Springer, 2005.