Relational Expectation Properties by Probabilistic Coupling

Abstract

Relational properties describe how two program executions are related, while *expectation* properties describe average-case behavior of probabilistic programs. We investigate formal verification techniques for *relational expectation properties*. This class includes key technical properties modeling stability in machine learning, and properties associated with fast mixing of Markov chains.

Technically, we design a relational program logic $\mathbb{E}$pRHL that is inspired by the logic pRHL, a powerful tool for proving relational properties by reasoning about probabilistic couplings. We enhance pRHL with an orthogonal, compositional reasoning principle based on premetrics; roughly, the expected distance between the outputs should be bounded as a function of the distance between the inputs.

We demonstrate our logic on three classes of examples: uniform stability of variants of the Stochastic Gradient Method used in machine learning, fast mixing for a Markov chain modeling population dynamics, and fast mixing for a Markov chain from statistical physics, using the path coupling method.

1. Introduction

Probabilistic programs are a classical and powerful tool in computer science. By taking random samples, such programs can model complex distributions and use computational resources more efficiently. By now probabilistic programs have found applications in numerous fields, including machine learning, statistical physics and even quantitative biology.

Given the broad range of applications, researchers have proposed a multitude of probabilistic properties reflecting interesting features of probabilistic programs. For instance, a program may have a high probability of producing an accurate answer, or a program may satisfy a statistical notion of privacy. An important subclass of probabilistic properties involve *expected values*, used to describe average-case behavior. These *expectation properties* have long been studied from a formal verification perspective; seminal works, including PPDL [26] and pGCL [29], sparked continuing lines of research (see, e.g., [19, 24]).

However, existing systems consider properties that describe a probabilistic execution on a single input, so-called *non-relational* properties. In contrast, there has been comparatively little work on *relational* expectation properties, which relate the average-case behavior of two probabilistic programs on two related inputs. As a simple example, consider a probabilistic version of a sensitivity condition: if two numeric inputs differ by 1, then the average numerical outputs should differ by at most k. This notion of expected sensitivity generalizes the usual notion used to model robustness of computations (e.g., Chaudhuri et al. [15]).

Probabilistic relational properties—and not just expectation properties—pose broad challenges for formal verification. In a nutshell, the central issue is reasoning about two distinct sources of randomness. Since relational properties describe separate executions, the random choices in one run are unrelated to random choices in the other run. Even if the programs share the same code, the two executions might have completely different control flow.

To address this issue, attention has turned to *probabilistic couplings* as a powerful tool for verifying relational properties. Couplings are a clean abstraction for reasoning about pairs of probabilistic processes, with a solid history in mathematics and probability theory. The key idea is that even though the two executions are unrelated, for many common relational properties it suffices to prove that the outputs are related *assuming that the random samples are correlated in a particular way*. Probabilistic couplings, then, describe how to correlate random samples. Often, the correlation deterministically maps one sampled value to another, e.g. by forcing two coin tosses to take opposite values. In these cases, couplings also abstract away all reasoning about probabilities—the key data is the relation between the two samples, rather than the probability of the particular draws.

Crucially for formal verification, couplings can be cleanly composed to reason about the necessary correlations. Recent work interprets the program logic pRHL [1] as a logic for constructing couplings, and verifies challenging properties like convergence of probabilistic properties [4, 11]; using an approximate variant of pRHL, the coupling idea can be extended to verify differential privacy [7, 9]. While these systems represent significant progress in verifying relational properties, they are currently limited to binary, more qualitative guarantees. There are many interesting probabilistic properties of this style: stating that two probabilities are equal, or one probability is larger than another, etc. On the other hand, expectation properties often have a more quantitative aspect, for instance bounding the difference between two expected values. These expectation properties lie beyond the reach of existing techniques.

$\mathbb{E}$PRHL*: relational expectation properties by coupling.* To address this shortcoming we propose a quantitative reasoning principle for proving expectation properties—complementary to couplings—and realize it in a new program logic $\mathbb{E}$PRHL for proving relational expectation properties. There are three key ideas. First, we augment the standard, boolean assertions of the program logic PRHL with *premetrics*, i.e. mappings from pairs of memories to non-negative extended reals. We use premetrics to model *expected Lipschitz* properties, a probabilistic analogue of *sensitivity* (also known as *Lipschitz*) properties—the distance between outputs should not be much larger than the distance between inputs. Concretely, if the premetrics in the pre- and post-conditions are $\mathfrak{d}$ and $\mathfrak{d}'$, judgments state that the expected value of $\mathfrak{d}'$ over the coupling of the output distributions is at most $f \circ \mathfrak{d}$ on the input memories, where we call the function $f : \mathbb{R} \to \mathbb{R}$ a *d-transformer*.

Our second technical ingredient handles composition. As we saw, one of the most useful features of couplings is that they can be easily composed. We observe that one can define a sequential composition theorem for premetrics (similar, for instance to the sequential composition theorem of differential privacy) if the distance transformer is an *affine* function, i.e. $f(x) = ax+b$ with $a, b \geq 0$; this relies on a basic property of expected values called linearity of expectation. As a result, we can reason about the sequential composition of two programs by combining the d-transformers via function composition, giving clean proof rules for sequential composition and loops. Overall, premetrics give an orthogonal, quantitative method of reasoning about couplings, while composing harmoniously alongside couplings.

Finally, we develop different reasoning principles which capture different ways in which couplings and premetrics interact. These principles are particularly useful to capture existing methods for proving rapid mixing of Markov chains.

Applications. We demonstrate our techniques on three challenging case studies of relational expectation properties. The first two examples were only recently considered within their respective communities.

Stability of stochastic gradient method. Stability [12, 18] is used in machine learning for measuring how changes in the training set influence the quality of an algorithm's prediction after the training phase. In particular, stability yields good generalization bounds for a large class of algorithms based on empirical risk minimization (ERM). Recently, Hardt et al. [20] show stability of the Stochastic Gradient Method (SGM), a widely used algorithm in machine learning. We verify stability claim using our logic.

Population dynamics. Evolutionary algorithms [22] are a useful modeling tool for biological or social phenomena. They can be used to analyze population dynamics, both in the infinite population setting, where evolution is *deterministic*, and in the finite population setting, where evolution can be *stochastic*. We formally analyze a variant of the so-called *RSM* (replication-selection-mutate) model, which captures the evolution of an unstructured, asexual haploid population (see, e.g., [21]). Recently, a series of papers prove rapid mixing of the RSM model under some mild conditions [16, 30, 34]. We formally verify rapid mixing in a simplified setting, where the evolution function is strictly contractive.

Path coupling. Path coupling [13] is an elaboration of the coupling method (see, e.g., [27, 28, 32, 33]) that provides simple proofs of convergence for a class of Markov chains. Path coupling can be seen as an engineering tool for couplings: if we can give a coupling for one step of the Markov chain started in neighboring states, then path coupling combines the pieces to give a coupling started from any two (possibly distant) states. Technically, if the underlying space is equipped with a *path metric*, i.e., where the measure between two elements is the length of the shortest Φ-path between them, and for every two states *related by* Φ the expected distance between the two distributions obtained by applying a single iteration is upper bounded by β, then taking T steps of the Markov chain from two initial states at distance k yields two distributions that have expected distance at most $\beta^T \cdot k$. Path coupling has numerous applications in statistical mechanics, molecular evolution, security. A canonical example of path coupling is graph coloring, to show the convergence of the Glauber dynamics for drawing approximately uniform samplings from the set of colorings of a graph [13]. We prove rapid mixing of the Glauber dynamics using our logic.

Outline. After giving a motivating example (§ 2) and reviewing some mathematical preliminaries (§ 3), we present the following contributions.

- A core probabilistic relational program logic $\mathbb{E}$PRHL based on probabilistic couplings, with premetrics to reason about relational expectation properties, and a proof of soundness for the logic (§ 4).
- A formal proof of uniform stability for two versions of SGM [20], relying on proof rules to perform probabilistic case analysis (§ 5).
- A formal proof of rapid mixing for Markov chain modeling dynamics of a general model of population evolution, relying on a proof rule representing the *optimal* coupling of two distributions (§ 6).
- A formal proof of rapid mixing for the Glauber dynamics from statistical physics, relying on a proof rule internaliz-

ing the path coupling principle [13], a method for combining couplings where the expected distance for each piece can be bounded (§ 7).

We conclude by surveying related work (§ 8) and presenting some future directions (§ 9).

2. Stability of Stochastic Gradient Method

To give a taste of our approach, let's consider a motivating example from machine learning. In a typical learning setting, there is a set of possible *training examples* Z, a *parameter space* $\mathbb{R}^d$, and a *loss function* $\ell : Z \to \mathbb{R}^d \to [0, 1]$. An algorithm A takes a finite set $S \in Z^n$ of *training examples*—assumed to be drawn independently from some unknown distribution $\mathcal{D}$—and produces a parameter $w \in \mathbb{R}^d$ such that the expected loss of $\ell(-, w)$ on a fresh sample from $\mathcal{D}$ should be as small as possible. When the algorithm is randomized, we think of A as a function $Z^n \to \mathbb{D}(\mathbb{R}^d)$.

In order to minimize the loss, a natural idea is to find some parameter w that minimizes the average error on the training set and hope that this output also has low error on the true distribution; such parameters are said to *generalize*. When the loss function ℓ is well-behaved this optimization problem, known as *empirical risk minimization* in the literature, can be solved efficiently. However, even if w has low loss on the training set, it may have high loss on fresh samples from the true distribution. Intuitively, the algorithm may select parameters that are too specific to the training sample. This problem—also known as *overfitting*—is a serious problem in machine learning.

To avoid overfitting, Bousquet and Elisseeff [12] considered algorithmic notions of stability. Roughly, the algorithm should produce similar outputs when executed on two training sets that differ in a single example, so that the output does not depend too much on any single training example. More formally:

Definition (Bousquet and Elisseeff [12], Hardt et al. [20])**.** *Let $A : Z^n \to \mathbb{D}(\mathbb{R}^d)$ be an algorithm for some loss function $\ell : Z \to \mathbb{R}^d \to [0, 1]$. The algorithm A is said to be ϵ-*uniformly stable *if for all input sets $S, S' \in Z^n$ that differ in a single element,*[1] *we have*

$$\mathbb{E}_{w \sim A(S)}[\ell(z, w)] - \mathbb{E}_{w \sim A(S')}[\ell(z, w)] \leq \epsilon$$

for all $z \in Z$, where $\mathbb{E}_{x \sim \mu}[f(x)]$ denotes the expected value of $f(x)$ when x is drawn from distribution μ.

From our point of view, ϵ-stability is a relational expectation property. By the following observation, we can reduce the two sources of randomness into one source by finding a single distribution that models both runs.

Fact. *For every pair of training sets $S, S' \in Z^n$ that differ in a single element, suppose that we can find a joint distribution $\mu(S, S') \in \mathbb{D}(\mathbb{R}^d \times \mathbb{R}^d)$ such that $\pi_1(\mu) = A(S)$ and $\pi_2(\mu) = A(S')$. If*

$$\mathbb{E}_{(w,w') \sim \mu(S,S')}[\ell(z, w) - \ell(z, w')] \leq \epsilon$$

for every $z \in Z$, then A is ϵ-uniformly stable.

The joint distribution $\mu(S, S')$ is known as a *coupling*. Using our logic EpRHL, we can construct couplings and reason about the expected distance on the coupled distributions. To demonstrate our approach, we verify ϵ-stability of several versions of the *Stochastic Gradient Method* (SGM), a simple and classical algorithm for machine learning. For the most basic version, the parameter space is $\mathbb{R}^d$ (i.e., that the algorithm is trying to learn d real parameters). SGM maintains a parameter w and iteratively updates this parameter to reduce the loss. Each iteration, SGM selects a uniformly random example z from the input training set S and computes the *gradient*[2] g of the function $\ell(z, -) : \mathbb{R}^d \to [0, 1]$ evaluated at the current parameter w—this indicates the direction to move w in order to reduce $\ell(z, w)$. Then, SGM updates w to step along g. After running T iterations, the algorithm returns w as the final parameter choice. We can implement SGM in an imperative language with the following code.

```
$w \leftarrow w_0$;
$t \leftarrow 0$;
while $t < T$ do
  $i \xleftarrow{\$} [n]$;
  $g \leftarrow \nabla \ell(S[i], -)(w)$;
  $w \leftarrow w - \alpha_t \cdot g$;
  $t \leftarrow t + 1$;
return $w$
```

To introduce some notation, the first step in the loop samples a uniformly random element index i from $[n] = \{0, 1, \ldots, n-1\}$, while the second step computes the gradient g. We will model the gradient operator ∇ as a higher-order function with type $(\mathbb{R}^d \to [0, 1]) \to (\mathbb{R}^d \to \mathbb{R}^d)$.[3] The third step in the loop updates w; the step size α_t is a real number that depends on the iteration t.

Our goal is to verify that this program is ϵ-uniformly stable, assuming natural conditions on the loss function ℓ. At a high level, suppose we have two training sets $S_\triangleleft, S_\triangleright$ differing in a single example. Viewing the sets as lists, this means that the two lists have the same length, and $S[i]_\triangleleft = S[i]_\triangleright$ for all indices i except for a particular index $i = j$. Then, we construct a coupling between the two distributions on outputs and bound the expected distance between the outputs $w_\triangleleft$ and $w_\triangleright$. If $\ell(z, -)$ is a Lipschitz function, i.e. $|\ell(z, w) - \ell(z, w')| \leq L\|w - w'\|$ for all

[1] In other words, S and S' have the same cardinality and their symmetric difference contains exactly two elements.

[2] Roughly speaking, the gradient acts a multidimensional analogue of the usual derivative.

[3] This operation is only well-defined if the input function is differentiable; this holds for most of the loss functions considered in the machine learning literature.

$w, w' \in \mathbb{R}^d$, bounding the expected distance between the parameters also bounds the expected losses, implying uniform stability.

We can carry out the verification in $\mathbb{E}$PRHL, a relational program logic with judgments of the form

$$\vdash \{\Phi; \mathfrak{d}\}\ s_1 \sim_f s_2\ \{\Psi; \mathfrak{d}'\}.$$

Here, s_1, s_2 are two imperative programs, the formulas Φ and Ψ are assertions over pairs of memories $\mathcal{M} \times \mathcal{M}$, the premetrics $\mathfrak{d}, \mathfrak{d}'$ are maps $\mathcal{M} \times \mathcal{M} \to \mathbb{R}^+$, and $f : \mathbb{R}^+ \to \mathbb{R}^+$ is a positive affine function (i.e., of the form $x \mapsto ax + b$ for $a, b \in \mathbb{R}^+$). The rough idea is that for two initial memories (m_1, m_2) satisfying the precondition Φ, there is a coupling of the output distributions such that the expected value of $\mathfrak{d}'$ on the coupling is at most $f(\mathfrak{d}(m_1, m_2))$ and all pairs of output memories with positive probability satisfy Ψ.

To sketch the verification, let us focus on the loop. Let s_a be the sampling command, and let s_b be the remainder of the loop body. First, we have

$$\vdash \{\Phi; \|w_\triangleleft - w_\triangleright\|\}\ s_a \sim_{\mathrm{id}} s_a\ \{i_\triangleleft = i_\triangleright; \|w_\triangleleft - w_\triangleright\|\}. \quad (1)$$

We abbreviate trivial invariants like $t_\triangleleft = t_\triangleright$ and $Adj(S_\triangleleft, S_\triangleright)$ as Φ. The postcondition $i_\triangleleft = i_\triangleright$ indicates that the coupling assumes both executions sample the same index i.

Now, we know that the training sets $S_\triangleleft$ and $S_\triangleright$ differ in a single example, say at index j. There are two cases: either we have sampled $i_\triangleleft = i_\triangleright = j$, or we have sampled $i_\triangleleft = i_\triangleright \neq j$. In the first case, we can apply properties of the loss function ℓ and gradient operator ∇ to prove:

$$\vdash \{S[i]_\triangleleft \neq S[i]_\triangleright; \|w_\triangleleft - w_\triangleright\|\}\ s_b \sim_{+\gamma} s_b\ \{\Phi'; \|w_\triangleleft - w_\triangleright\|\} \quad (2)$$

where $+\gamma$ is the d-transformer $x \mapsto x + \gamma$ for a constant γ. In the second case we know that the example $S[i]$ is the same in both executions, so we can prove:

$$\vdash \{S[i]_\triangleleft = S[i]_\triangleright; \|w_\triangleleft - w_\triangleright\|\}\ s_b \sim_{\mathrm{id}} s_b\ \{\Phi'; \|w_\triangleleft - w_\triangleright\|\} \quad (3)$$

That is, the expected distance does not increase. To combine these two cases, note that the first case happens with probability $1/n$—this is the probability of sampling the differing index j—while the second case happens with probability $1 - 1/n$. We can scale the bounds accordingly, yielding

$$\vdash \{\Phi; \|w_\triangleleft - w_\triangleright\|\}\ s_a; s_b \sim_f s_a; s_b\ \{\Phi; \|w_\triangleleft - w_\triangleright\|\}, \quad (4)$$

where $f(x) = (1/n) \cdot (x + \gamma) + (1 - 1/n) \cdot x = x + \gamma/n$.

Now that we have a bound on how the distance grows in the body, we can apply the loop rule. This rule simply takes the T-fold composition of the bounding function f:

$$\vdash \{\Phi; \|w_\triangleleft - w_\triangleright\|\}\ \mathsf{sgm} \sim_{f^T} \mathsf{sgm}\ \{\Phi; \|w_\triangleleft - w_\triangleright\|\}. \quad (5)$$

Here, f^T is the d-transformer $+T\gamma/n$. Assuming that the loss function $\ell(-, z)$ is Lipschitz, we know that $|\ell(w, z) - \ell(w', z)| \leq L \cdot \|w - w'\|$ for some constant L and so

$$\vdash \{\Phi; \|w_\triangleleft - w_\triangleright\|\}\ \mathsf{sgm} \sim_{L \cdot f^T} \mathsf{sgm}\ \{\Phi; |\ell(w_\triangleleft, z) - \ell(w_\triangleright, z)|\} \quad (6)$$

for every example $z \in Z$, where $(L \cdot f^T)(x) = L \cdot f^T(x)$. Since $w_\triangleleft$ and $w_\triangleright$ are both initially w_0, we have constructed a coupling μ of the output distributions such that

$$\mathbb{E}_\mu[|\ell(w_\triangleleft, z) - \ell(w_\triangleright, z)|] \leq \|w_0 - w_0\| + LT\gamma/n = LT\gamma/n.$$

Since the left side is larger than $\mathbb{E}_\mu[\ell(w_\triangleleft, z) - \ell(w_\triangleright, z)]$, we conclude that SGM is $LT\gamma/n$-uniform stable.

3. Preliminaries

Before we can present our logic, we first review basic definitions and notations from probability theory, especially those related to expected values and probabilistic couplings.

3.1 Probability theory and notations

We let $\mathbb{B}$ be the set of booleans, $\mathbb{R}$ be the set of real numbers and $\overline{\mathbb{R}} = \mathbb{R} \cup \{-\infty, +\infty\}$ be the set of extended real numbers. Moreover, we let $\mathbb{R}^+$ and $\overline{\mathbb{R}}^+$ denote the set of positive real numbers and positive extended real numbers respectively. Throughout the paper, we adopt the following convention: if $a > 0$ and $b \geq 0$ then $a \times +\infty + b = +\infty$.

To define d-transformers, we let $\mathcal{F}$ be the set of positive affine functions, mapping $x \mapsto ax + b$ where $a, b \in \mathbb{R}^+$; $\mathcal{L} \subseteq \mathcal{F}$ be the set of positive linear functions, mapping $x \mapsto ax$; and $\mathcal{C} \subseteq \mathcal{F}$ be the set of positive constant functions, mapping $x \mapsto b$. We will use the metavariables f for $\mathcal{F}$ and bolded letters (e.g., $\boldsymbol{\gamma}$) for $\mathcal{C}$. Functions can be added: $(f + f')(x) = f(x) + f(x')$; scaled by constant functions: $(\boldsymbol{\gamma} \cdot f)(x) = \gamma \cdot f(x)$; and iterated k times: $f^k(x) = f \circ \cdots \circ f$. Finally, we will use some shorthand for common functions. For scaling, we write $\times\gamma(x) = \gamma x$. For translation, we write $+\gamma(x) = x + \gamma$. The identity function will be simply id (equivalently, $\times 1$ or $+0$).

Our programs in our language are interpreted in terms of sub-distributions, whose definition we recall here.

Definition 3.1. *A (discrete)* sub-distribution *over a set A is a mapping $\mu : A \to \mathbb{R}^+$ such that its support*

$$\mathrm{supp}(\mu) \triangleq \{a \in A \mid \mu(a) \neq 0\}$$

is discrete and $|\mu| \triangleq \sum_{a \in \mathrm{supp}(\mu)} \mu(a)$ is well-defined and satisfies $|\mu| \leq 1$.

We let $\mathbb{D}(A)$ denote the set of discrete sub-distributions over A. Note that $\mathbb{D}(A)$ is partially ordered using the pointwise inequality inherited from reals. Similarly, equality of distributions is defined extensionally: two distributions are equal if they assign the same value (i.e., probability) to each element in their domain.

Events are mappings $E : A \to \mathbb{B}$. The probability of an event E w.r.t. a sub-distribution μ, written as $\Pr_\mu[E]$, is defined as $\sum_{x|E(x)} \mu(x)$. Likewise, the expectation of a function $f : A \to \mathbb{R}^+$ w.r.t. a sub-distribution $\mu \in \mathbb{D}(A)$, written $\mathbb{E}_{x\sim\mu}[f(x)]$ or $\mathbb{E}_\mu[f]$, is defined as $\sum_x \mu(x)f(x)$ when this sum exists, and $+\infty$ otherwise.

A key feature of our logic is that it supports two kinds of reasoning: assertions can model qualitative properties, while *premetrics* can reason about quantitative aspects.

Definition 3.2. *A* premetric *is a mapping $\mathfrak{d} : A \times A \to \overline{\mathbb{R}}^+$ such that $\mathfrak{d}(x,x) = 0$ for every $x \in A$. Premetrics are partially ordered using the pointwise order inherited from the extended reals. A* hemimetric *is a premetric $\mathfrak{d}'$ that satisfies the triangular inequality:*

$$\forall x, y, z.\, \mathfrak{d}'(x,z) \le \mathfrak{d}'(x,y) + \mathfrak{d}'(y,z)$$

We write $\mathfrak{d}' \in \mathsf{HemiMet}$ when $\mathfrak{d}'$ is a hemimetric.

Every binary relation Φ induces a premetric pd_Φ, called the *path distance*:

$$\mathrm{pd}_\Phi(a,a') = \begin{cases} \min_n\{(a,a') \in \Phi^n\} & \text{if } (a,a') \in \Phi^* \\ +\infty & \text{otherwise.} \end{cases}$$

Here, Φ^* denote the reflexive transitive closure of Φ and Φ^n denotes its n-fold composition for every $n \in \mathbb{N}$. This mapping is order-preserving.

3.2 Expectation couplings

A *probabilistic coupling* is a distribution over the product space of two distributions, such that its first and second marginals coincide with the first and second distributions. Formally, two sub-distributions $\mu_a \in \mathbb{D}(A)$ and $\mu_b \in \mathbb{D}(B)$ are coupled by $\mu \in \mathbb{D}(A \times B)$, written $\mu \blacktriangleleft \langle \mu_a \;\&\; \mu_b\rangle$, iff $\pi_1(\mu) = \mu_a$ and $\pi_2(\mu) = \mu_b$.

Expectation couplings are a quantitative extension of probabilistic couplings. Informally, a δ-expectation coupling w.r.t. a premetric $\mathfrak{d}'$ is a coupling μ such that the expected value of $\mathfrak{d}'$ under μ is upper bounded by δ.

Definition 3.3 (Expectation liftings)**.** *Let $\mathfrak{d}' : A \times B \to \overline{\mathbb{R}}^+$ and $\delta \in \overline{\mathbb{R}}^+$. Moreover, let $\mu_a \in \mathbb{D}(A)$, $\mu_b \in \mathbb{D}(B)$ and $\mu \in \mathbb{D}(A \times B)$. Then μ is an δ-expectation coupling for μ_a and μ_b w.r.t. $\mathfrak{d}'$, written $\mu \blacktriangleleft_{\mathfrak{d}',\delta} \langle \mu_a \;\&\; \mu_b\rangle$, if $\mu \blacktriangleleft \langle \mu_a \;\&\; \mu_b\rangle$ and $\mathbb{E}_\mu[\mathfrak{d}'] \le \delta$.*

We also recall the notion of probabilistic couplings w.r.t. a relation $\Psi : A \times B \to \mathbb{B}$, which is used in relational program logics such as pRHL for interpreting judgments. We say that μ is a coupling for μ_a and μ_b w.r.t. Ψ, written $\mu \blacktriangleleft^\Psi \langle \mu_a \;\&\; \mu_b\rangle$ if $\mu \blacktriangleleft \langle \mu_a \;\&\; \mu_b\rangle$ and $\mathrm{supp}(\mu) \subseteq \Psi$; this definition strengthens the definition of coupling by requiring that the coupling μ verifies $\mathrm{supp}(\mu) \subseteq \Psi$. Our logic will combine the two notions of couplings in order to support both qualitative and quantitative reasoning. Specifically, we say that μ is a δ-coupling w.r.t. Ψ and $\mathfrak{d}'$, written $\mu \blacktriangleleft^\Psi_{\mathfrak{d}',\delta} \langle \mu_a \;\&\; \mu_b\rangle$, if both $\mu \blacktriangleleft_{\mathfrak{d}',\delta} \langle \mu_a \;\&\; \mu_b\rangle$ and $\mu \blacktriangleleft^\Psi \langle \mu_a \;\&\; \mu_b\rangle$.

Couplings enjoy several closure properties. In particular, they are closed under sequential composition.

Proposition 3.4. *Let $\Phi : A \times B \to \mathbb{B}$, $\mathfrak{d} : A \times B \to \overline{\mathbb{R}}^+$, $\Psi : A \times B \to \mathbb{B}$, $\mathfrak{d}' : A \times B \to \overline{\mathbb{R}}^+$, $\delta \in \overline{\mathbb{R}}^+$, and $f \in \mathcal{F}$. Let $\mu_a \in \mathbb{D}(A)$, $M_a : A \to \mathbb{D}(A')$, and set $\mu'_a = \mathbb{E}_{\mu_a}[M_a]$. Let $\mu_b \in \mathbb{D}(b)$, $M_b : B \to \mathbb{D}(B')$, and set $\mu'_b = \mathbb{E}_{\mu_b}[M_b]$. Let $\mu \in \mathbb{D}(A \times B)$, $M : (A \times B) \to \mathbb{D}(A' \times B')$, and set $\mu' = \mathbb{E}_\mu[M]$. Assume:*

- *$\mu \blacktriangleleft^\Phi_{\mathfrak{d},\delta} \langle \mu_a \;\&\; \mu_b\rangle$;*
- *$M(a,b) \blacktriangleleft^\Psi_{\mathfrak{d}',f(\mathfrak{d}(a,b))} \langle M_a(a) \;\&\; M_b(b)\rangle$ for every a, b such that $\Phi\ a\ b$.*

Then $\mu' \blacktriangleleft^\Psi_{\mathfrak{d}',f(\delta)} \langle \mu'_a \;\&\; \mu'_b\rangle$, where $\mu' = \mathbb{E}_\mu[M]$.

4. Program logic

With the preliminaries out of the way, we are ready to present our program logic.

4.1 Programming language

We base our development on pWHILE, a core language with deterministic assignments, probabilistic assignments, conditionals, and loops. We implicitly assume that programs are well-typed w.r.t. a standard typing discipline. The syntax of statements is defined by the grammar:

$$\begin{aligned} s ::= {} & x \leftarrow e \mid x \xleftarrow{\$} g \mid s; s \mid \mathsf{skip} \\ & \mid \mathsf{if}\ e\ \mathsf{then}\ s\ \mathsf{else}\ s \mid \mathsf{while}\ e\ \mathsf{do}\ s \end{aligned}$$

where x, e, and g range over variables in $\mathcal{V}$, expressions in $\mathcal{E}$ and distribution expressions in $\mathcal{D}$ respectively. $\mathcal{E}$ is defined inductively from $\mathcal{V}$ and operators, while $\mathcal{D}$ consists of parameterized distributions—for instance, the uniform distribution $[n]$ over the set $\{0, \dots, n-1\}$ or the Bernoulli distribution $\mathbf{Bern}(e)$ with parameter e.

Following the seminal work of Kozen [25], the denotational semantics of programs is given as sub-distribution transformers. One difference is that we only consider discrete sub-distributions and as a consequence avoid issues of measurability.

We first define memories as type-preserving mappings from variables to values—formally, we define an interpretation for each type and require that a variable of type T is mapped to an element of the interpretation of T. We let $\mathcal{M}$ denote the set of memories. Then, the semantics $[\![e]\!]_m$ of a (well-typed) expression e is defined in the usual way as an element of the interpretation of the type of e, and parameterized by a memory m. The interpretation of distribution expressions is defined and denoted likewise. Finally, we define the semantics of statements.

Definition 4.1 (Semantics of statements)**.**

- *The semantics $[\![s]\!]_m$ of a statement s w.r.t. to some initial memory m is a sub-distribution over states, and is defined by the clauses of Fig. 1.*
- *The (lifted) semantics $[\![s]\!]_\mu$ of a statement s w.r.t. to some initial sub-distribution μ over memories is a*

sub-distribution over states, and is defined as $[\![s]\!]_\mu \triangleq \mathbb{E}_{m\sim\mu}[[\![s]\!]_m]$.

The semantics of programs given in Figure 1 is standard. The most interesting case is for loops, where the interpretation of a while loop is the limit of the interpretations of its finite unrollings. Formally, the n^{th} *truncated iterate* of the loop while b do s is defined as

$$\overbrace{\text{if } b \text{ then } s; \ldots; \text{if } b \text{ then } s}^{n \text{ times}}; \text{if } b \text{ then abort}$$

which we represent using the shorthand $(\text{if } b \text{ then } s)^n_{|\neg b}$. For any initial sub-distribution μ, applying the truncated iterates yields an increasing and bounded sequence of sub-distributions. We take the limit of this sequence to give a semantics to the while loop.

4.2 Proof system

$\mathbb{E}$pRHL judgments are of the form

$$\{\Phi; \eth\}\ s_1 \sim_f s_2\ \{\Psi; \eth'\}$$

for programs s_1, s_2, assertions $\Phi, \Psi : \mathcal{M} \times \mathcal{M} \to \mathbb{B}$, premetrics $\eth, \eth' : \mathcal{M} \times \mathcal{M} \to \overline{\mathbb{R}}^+$, and a function $f \in \mathcal{F}$. We will refer to f as a *distance transformer*.

Definition 4.2. *A judgment* $\{\Phi; \eth\}\ s_1 \sim_f s_2\ \{\Psi; \eth'\}$ *is valid iff for every memories* m_1, m_2 *s.t.* $(m_1, m_2) \models \Phi$, *there exists* μ *such that*

$$\mu \blacktriangleleft^{\Psi}_{\eth', f(\eth(m_1,m_2))} \langle [\![s_1]\!]_{m_1} \ \&\ [\![s_2]\!]_{m_2} \rangle$$

Figure 2 summarizes the main rules of the logic. We comment on some key rules below, but first introduce some notation and terminology used in the rules. First, note that each boolean expression e naturally yields two assertions $e_{\triangleleft}$ and $e_{\triangleright}$, resp. called its left and right injections:

$$m_1 \models e \iff m_1, m_2 \models e_{\triangleleft}$$
$$m_2 \models e \iff m_1, m_2 \models e_{\triangleright}$$

The notation naturally extends to mappings from memories to booleans. Second, several rules use substitutions. Given a memory m, variable x and expression e such that the types of x and e agree, we let $m[x \leftarrow e]$ denote the unique memory m' such that $m(y) = m'(y)$ if $y \neq x$ and $m'(x) = [\![e]\!]_m$. Then, given a variable x (resp. x'), an expression e (resp. e'), and an assertion Φ, we define the assertion $\Phi[x_{\triangleleft}, x'_{\triangleright} \leftarrow e_{\triangleleft}, e'_{\triangleright}]$ by the clause:

$$\Phi[x_{\triangleleft}, x'_{\triangleright} \leftarrow e_{\triangleleft}, e'_{\triangleright}](m_1, m_2) \triangleq \Phi(m_1[x \leftarrow e], m'_2[x' \leftarrow e'])$$

Substitution of premetrics is defined similarly. One can also define one-sided substitutions, for instance $\Phi[x_{\triangleleft} \leftarrow e_{\triangleleft}]$.

We now turn to the rules of the proof system. We only consider structural rules, i.e. rules that do not look at the structure of the program, and two-sided rules, i.e. rules in which the left and right programs have the same structure. Later we prove that many one-sided rules are derivable.

The [CONSEQ] rule captures the fact that validity is preserved by weakening the post-conditions, strengthening the pre-conditions, and increasing the distance transformer.

The [STRUCT] rule captures the fact that validity is preserved by replacing programs by equivalent ones. The rules for proving program equivalence are given in Fig. 4, and manipulate judgments of the form $\Phi \vdash s \equiv s'$, where Φ is a relational assertion. We keep the notion of structural equivalence as simple as possible.

The [ASSG] rule is similar to the usual rule for assignments, and substitutes into the pre-condition and premetric the expressions used in the assignments.

The [RAND] rule is similar to the pRHL rule for random assignments assignments; again, one substitutes into the pre-condition and premetric. Informally the rule requires to exhibit the existence of a coupling, given as a bijection between their support, between the two distributions used for sampling in the left and right program.

The [SEQCASE] rule combines sequential composition with a case analysis on properties satisfied by intermediate memories, i.e. after executing the programs s_1 and s_2. Informally, the rule considers events $e_1 \ldots e_n$ such that Ψ entails $\bigvee_i e_{i_{\triangleleft}}$. Provided one can relate for every i the programs s'_1 and s'_2 with distance transformer f_i, pre-condition $\Psi \wedge e_{i_{\triangleleft}}; \eth'$ and postcondition $\Psi'; \eth''$, one can conclude that $s_1; s'_1$ and $s_2; s'_2$ are related under distance transformer f, where f upper bounds the functions f_i weighted by the probability of each case.

The [WHILE] rule for while loops considers two loops that execute synchronously, and whose loop bodies satisfy the invariant $\Psi; \eth'$. The rule additionally requires that both loops perform exactly n steps, and that there exists a variant i initially set to n and decreasing by 1 at each iteration. Assuming that f_k denotes the distance transformer corresponding to the $(n-k)$th iteration, i.e. the iteration where the variant i is equal to k, the distance transformer for the while loops is the function f, defined as $f_1 \circ \cdots \circ f_n$.

The [TRANS] rule is a structural rule that internalizes the main theorem of path coupling when the post-premetric is a hemimetric the pre-premetric is a path-distance for some assertion Ψ'. There are three main premises, corresponding to the case where the initial memories are at distance 0, 1, or $+\infty$. Two additional conditions are required: the post-condition must be transitive (which we enforce by requiring it to be of the form Ψ^*), and for every Φ-path of minimal length between two memories m and m' related by Φ, there exists a (possibly different) $\Phi \wedge \Phi'$-path of the same length between m and m'. These technical conditions are required

$$\begin{aligned} [\![\mathsf{skip}]\!]_m &= \delta_m & [\![x \xleftarrow{\$} g]\!]_m &= \mathbb{E}_{v \sim [\![g]\!]_m}[\delta_{m[x:=v]}] \\ [\![x \leftarrow e]\!]_m &= \delta_{m[x \leftarrow [\![e]\!]_m]} & [\![\mathsf{if}\ e\ \mathsf{then}\ s_1\ \mathsf{else}\ s_2]\!]_m &= \mathsf{if}\ [\![e]\!]_m\ \mathsf{then}\ [\![s_1]\!]_m\ \mathsf{else}\ [\![s_2]\!]_m \\ [\![s_1; s_2]\!]_m &= \mathbb{E}_{\xi \sim [\![s_1]\!]_m}[[\![s_2]\!]_\xi] & [\![\mathsf{while}\ b\ \mathsf{do}\ s]\!]_m &= \lim_{n \to \infty} [\![(\mathsf{if}\ b\ \mathsf{then}\ s)^n_{|\neg b}]\!]_m \end{aligned}$$

Figure 1. Denotational semantics of programs

for soundness. Finally, the rule requires that the distance transformer $f \in \mathcal{L}$ is a linear function.[4]

The [FRAME-D] rule is a structural rule, analogous to a typical frame rule, that allows to modify the premetric in a judgment. Assuming that the premetric $\mathfrak{d}''$ is not modified by the statements of the judgments and f is a linear function such that $x \leq f(x)$ for all x, validity is preserved when adding $\mathfrak{d}''$ to the pre-premetric and post-premetric of the judgment.

Proposition 4.3 (Soundness)**.** *For every derivable judgment $\vdash \{\Phi; \mathfrak{d}\}\ s_1 \sim_f s_2\ \{\Psi; \mathfrak{d}'\}$ and initial memories m_1 and m_2 such that $(m_1, m_2) \models \Phi$, there exists μ such that*

$$\mu \blacktriangleleft^{\Psi}_{\mathfrak{d}', f(\mathfrak{d}(m_1, m_2))} \langle [\![s_1]\!]_{m_1} \ \&\ [\![s_2]\!]_{m_2} \rangle.$$

Proof. By induction on the derivation. We defer the details to the appendix. □

4.3 Derived rules and weakest precondition

Fig. 3 presents some derived rules of our logic. This includes rules for constructs such as sequential composition and conditionals and one-sided rules, i.e. rules that only operate on a single statement (left or right). As in [11], there are many other one-sided rules that can be derived similarly from the core rules.

The [SEQ] rule for sequential composition simply composes the two product programs in sequence. This rule reflects the compositional property of couplings. It can be derived from the rule [SEQCASE] by taking e_1 to be true.

The [COND] rule for conditional statements requires that the two guards of the left and right programs are equivalent under the precondition, and then that each branch is related.

The [CASE] rule allows proving a judgment by case analysis; specifically, the validity of a judgment can be established from the validity of two judgments, one where the boolean-valued pre-condition is strengthened with e and the other where the pre-condition is strengthened with $\neg e$.

The [ASSG-L] is the left one-sided rule for assignment. It can be derived from the assignment rule using structural equivalence. There exists similar one-sided rules for other constructs, notably random assignments and conditionals. Using one sided-rules, one can define a relational weakest precondition calculus wp, taking as inputs two loop-free and deterministic programs, a post-condition, and a premetric, and returning a pre-condition and a premetric.

[4] This assumption can be weakened to an affine function, at the cost of requiring that $\mathfrak{d}$ is bounded by some finite constant, and adjusting the distance transformer in the conclusion of the rule.

Proposition 4.4. *Let $(\Phi', \mathfrak{d}') = \mathsf{wp}(s_1, s_2, \Psi, \mathfrak{d}')$. Assume $\Phi \implies \Phi'$ and $\mathfrak{d}(m_1, m_2) \leq \mathfrak{d}'(m_1, m_2)$ for every $(m_1, m_2) \models \Phi$. Then $\vdash \{\Phi; \mathfrak{d}\}\ s_1 \sim_{\mathrm{id}} s_2\ \{\Psi; \mathfrak{d}'\}$.*

4.4 Embedding pRHL

$\mathbb{E}$pRHL is an extension of pRHL in the following sense.

Proposition 4.5. *If $\vdash_{\mathrm{pRHL}} \{\Phi\}\ s_1 \sim s_2\ \{\Psi\}$ then for every real-valued assertion $\mathfrak{d}'$ and distance transformer $f \in \mathcal{F}$, $\vdash \{\Phi; \lambda\_. +\infty\}\ s_1 \sim_f s_2\ \{\Psi; \mathfrak{d}'\}$*

5. Revisiting uniform stability

Now that we have described the logic, let's return to the Stochastic Gradient Method we first saw in § 2. Recall that the loss function has type $\ell : Z \to \mathbb{R}^d \to [0, 1]$. We consider two versions: one where the loss function $\ell(z, -)$ is convex, and one where $\ell(z, -)$ may be non-convex. The algorithm is the same in both cases, but the stability properties require different proofs. For convenience, we reproduce the code.

```
$w \leftarrow w_0$;
$t \leftarrow 0$;
while $t < T$ do
  $i \xleftarrow{\$} [n]$;
  $g \leftarrow \nabla\ell(S[i], -)(w)$;
  $w \leftarrow w - \alpha_t \cdot g$;
  $t \leftarrow t + 1$;
return $w$
```

We will assume that $\ell(z, -)$ is *L-Lipschitz* for all z: for all $w, w' \in \mathbb{R}^d$, we can bound $|\ell(z, w) - \ell(z, w')| \leq L\|w - w'\|$ where $\|\cdot\|$ is the usual Euclidean norm on $\mathbb{R}^d$. Furthermore, we will assume that the loss function is *β-smooth*: the gradient $\nabla\ell(z, -) : \mathbb{R}^d \to \mathbb{R}^d$ must be β-Lipschitz.

5.1 SGM with convex loss

Suppose that the function $\ell(z, -)$ is a *convex* function every z, i.e., the following dot product is non-negative: $\langle(\nabla\ell(z, -))(w) - (\nabla\ell(z, -))(w'), w - w'\rangle \geq 0$. We can prove uniform stability of SGM in this case by following the sketch in § 2. We refer back to the judgments there, briefly describing how to apply the rules (for lack of space, we defer details to the appendix). Let s_a be the sampling command,

$$[\textsc{Conseq}]\ \dfrac{\begin{array}{c}\vdash \{\Phi;\mathfrak{d}\}\ s_1 \sim_f s_2\ \{\Psi;\mathfrak{d}'\} \qquad \Phi' \implies \Phi \qquad \Psi \implies \Psi' \\ \forall m_1, m_2 \models \Phi'.\ f(\mathfrak{d}(m_1,m_2)) \le f'(\mathfrak{d}''(m_1,m_2)) \qquad \forall m_1, m_2 \models \Psi.\ \mathfrak{d}'''(m_1,m_2) \le \mathfrak{d}'(m_1,m_2)\end{array}}{\vdash \{\Phi';\mathfrak{d}''\}\ s_1 \sim_{f'} s_2\ \{\Psi';\mathfrak{d}'''\}}$$

$$[\textsc{Struct}]\ \dfrac{\vdash \{\Phi;\mathfrak{d}\}\ s_1 \sim_f s_2\ \{\Psi;\mathfrak{d}'\} \qquad \Phi_1 \vdash s_1 \equiv s_1' \qquad \Phi_2 \vdash s_2 \equiv s_2' \qquad \forall (m_1,m_2) \models \Phi.\ \Phi_1(m_1) \wedge \Phi_2(m_2)}{\vdash \{\Phi;\mathfrak{d}\}\ s_1' \sim_f s_2'\ \{\Psi;\mathfrak{d}'\}}$$

$$[\textsc{Assg}]\ \dfrac{}{\vdash \{\Psi[x_{1_\triangleleft} \leftarrow e_1, x_{2_\triangleright} \leftarrow e_2]; \mathfrak{d}[x_{1_\triangleleft} \leftarrow e_1, x_{2_\triangleright} \leftarrow e_2]\}\ x_1 \leftarrow e_1 \sim_{\mathrm{id}} x_2 \leftarrow e_2\ \{\Psi;\mathfrak{d}'\}}$$

$$[\textsc{Rand}]\ \dfrac{h : \mathrm{supp}(g_1) \xrightarrow{\text{1-1}} \mathrm{supp}(g_2) \qquad \forall v \in \mathrm{supp}(g_1).\ g_1(v) = g_2(h(v))}{\vdash \{\forall v \in \mathrm{supp}(g_1).\ \Psi[x_{1_\triangleleft} \leftarrow v, x_{2_\triangleright} \leftarrow h(v)]; \mathbb{E}_{v\sim g_1}[\mathfrak{d}'[x_{1_\triangleleft} \leftarrow v, x_{2_\triangleright} \leftarrow h(v)]]\}\ x_1 \xleftarrow{\$} g_1 \sim_{\mathrm{id}} x_2 \xleftarrow{\$} g_2\ \{\Psi;\mathfrak{d}'\}}$$

$$[\textsc{SeqCase}]\ \dfrac{\begin{array}{c}\vdash \{\Phi;\mathfrak{d}\}\ s_1 \sim_{f_0} s_2\ \{\Psi;\mathfrak{d}'\} \qquad \forall i \in I.\ \vdash \{\Psi \wedge e_{i_\triangleleft};\mathfrak{d}'\}\ s_1' \sim_{f_i} s_2'\ \{\Psi';\mathfrak{d}''\} \\ \forall m_1, m_2 \models \Phi.\ (\sum_{i\in I} \Pr_{[\![s_1]\!]_{m_1}}[e_i] \cdot f_i) \circ f_0 \le f \qquad \Psi \implies \bigvee_{i\in I} e_{i_\triangleleft}\end{array}}{\vdash \{\Phi;\mathfrak{d}\}\ s_1; s_1' \sim_f s_2; s_2'\ \{\Psi';\mathfrak{d}''\}}$$

$$[\textsc{While}]\ \dfrac{\begin{array}{c}\forall 0 < k \le n.\ \vdash \{\Psi \wedge e_{1_\triangleleft} \wedge i_\triangleleft = k; \mathfrak{d}_k\}\ s_1 \sim_{f_k} s_2\ \{\Psi \wedge i_\triangleleft = k-1; \mathfrak{d}_{k-1}\} \\ \Psi \implies e_\triangleleft = e_\triangleright \wedge (i_\triangleleft \le 0 \iff \neg e_\triangleleft)\end{array}}{\vdash \{\Psi \wedge i_\triangleleft = n; \mathfrak{d}_n\}\ \mathsf{while}\ e_1\ \mathsf{do}\ s_1 \sim_{f_1\circ\cdots\circ f_n} \mathsf{while}\ e_2\ \mathsf{do}\ s_2\ \{\Psi \wedge i_\triangleleft = 0; \mathfrak{d}_0\}}$$

$$[\textsc{Trans}]\ \dfrac{\begin{array}{c}f \in \mathcal{L} \qquad \mathfrak{d}' \in \mathsf{HemiMet} \qquad \Phi \implies d_{\Phi\wedge\Phi'} = d_{\Phi'} \\ \vdash \{\Phi \wedge =; -\}\ s \sim_0 s\ \{\Psi^*;\mathfrak{d}'\} \qquad \vdash \{\Phi \wedge \Phi'; d_{\Phi\wedge\Phi'}\}\ s \sim_f s\ \{\Psi^*;\mathfrak{d}'\} \qquad \vdash \{\Phi \wedge \neg(\Phi'^*); -\}\ s \sim_f s\ \{\Psi;\mathfrak{d}'\}\end{array}}{\vdash \{\Phi; d_{\Phi'}\}\ s \sim_f s\ \{\Psi^*;\mathfrak{d}'\}}$$

$$[\textsc{Frame-D}]\ \dfrac{\begin{array}{c}f \in \mathcal{L} \qquad \mathfrak{d}'' \# \mathrm{MV}(s_1, s_2) \qquad \forall m_1, m_2 \models \Phi.\ \mathfrak{d}''(m_1,m_2) \le f(\mathfrak{d}''(m_1,m_2)) \\ \vdash \{\Phi;\mathfrak{d}\}\ s_1 \sim_f s_2\ \{\Psi;\mathfrak{d}'\}\end{array}}{\vdash \{\Phi;\mathfrak{d} + \mathfrak{d}''\}\ s_1 \sim_f s_2\ \{\Psi;\mathfrak{d}' + \mathfrak{d}''\}}$$

Figure 2. Selected proof rules

$$[\textsc{Seq}]\ \dfrac{\vdash \{\Phi;\mathfrak{d}\}\ s_1 \sim_f s_2\ \{\Xi;\mathfrak{d}'\} \qquad \vdash \{\Xi;\mathfrak{d}'\}\ s_1' \sim_{f'} s_2'\ \{\Psi;\mathfrak{d}''\}}{\vdash \{\Phi;\mathfrak{d}\}\ s_1; s_1' \sim_{f'\circ f} s_2; s_2'\ \{\Psi;\mathfrak{d}''\}}$$

$$[\textsc{Case}]\ \dfrac{\vdash \{\Phi \wedge e_\triangleleft;\mathfrak{d}\}\ s_1 \sim_f s_2\ \{\Psi;\mathfrak{d}'\} \qquad \vdash \{\Phi \wedge \neg e_\triangleleft;\mathfrak{d}\}\ s_1 \sim_f s_2\ \{\Psi;\mathfrak{d}'\}}{\vdash \{\Phi;\mathfrak{d}\}\ s_1 \sim_f s_2\ \{\Psi;\mathfrak{d}'\}}$$

$$[\textsc{Cond}]\ \dfrac{\Phi \implies e_{1_\triangleleft} = e_{2_\triangleright} \qquad \vdash \{\Phi \wedge e_{1_\triangleleft};\mathfrak{d}\}\ s_1 \sim_f s_2\ \{\Psi;\mathfrak{d}'\} \qquad \vdash \{\Phi \wedge \neg e_{1_\triangleleft};\mathfrak{d}\}\ s_1' \sim_f s_2'\ \{\Psi;\mathfrak{d}'\}}{\vdash \{\Phi;\mathfrak{d}\}\ \mathsf{if}\ e_1\ \mathsf{then}\ s_1\ \mathsf{else}\ s_1' \sim_f \mathsf{if}\ e_2\ \mathsf{then}\ s_2\ \mathsf{else}\ s_2'\ \{\Psi;\mathfrak{d}'\}}$$

$$[\textsc{Assg-L}]\ \dfrac{}{\vdash \{\Psi[x_{1_\triangleleft} \leftarrow e_1]; \mathfrak{d}[x_{1_\triangleleft} \leftarrow e_1]\}\ x_1 \leftarrow e_1 \sim_{\mathrm{id}} \mathsf{skip}\ \{\Psi;\mathfrak{d}'\}}$$

Figure 3. Selected derived rules

$$\frac{}{\Phi \vdash s \equiv s} \qquad \frac{\Phi \vdash s_1 \equiv s_2}{\Phi \vdash s_2 \equiv s_1} \qquad \frac{}{\Phi \vdash x \xleftarrow{\$} \delta_x \equiv \mathsf{skip}}$$

$$\frac{\Phi \implies x = e}{\Phi \vdash x \leftarrow e \equiv \mathsf{skip}} \qquad \frac{}{\Phi \vdash s; \mathsf{skip} \equiv s} \qquad \frac{}{\Phi \vdash \mathsf{skip}; s \equiv s}$$

$$\frac{\Phi \vdash s_1 \equiv s_1'}{\Phi \vdash s_1; s_2 \equiv s_1'; s_2} \qquad \frac{\top \vdash s_2 \equiv s_2'}{\Phi \vdash s_1; s_2 \equiv s_1; s_2'}$$

$$\frac{\Phi \implies e}{\Phi \vdash \mathsf{if}\ e\ \mathsf{then}\ s\ \mathsf{else}\ s' \equiv s} \qquad \frac{\Phi \implies \neg e}{\Phi \vdash \mathsf{if}\ e\ \mathsf{then}\ s\ \mathsf{else}\ s' \equiv s'}$$

$$\frac{\Phi \wedge e \vdash s_1 \equiv s_2 \qquad \Phi \wedge \neg e \vdash s_1' \equiv s_2'}{\Phi \vdash \mathsf{if}\ e\ \mathsf{then}\ s_1\ \mathsf{else}\ s_1' \equiv \mathsf{if}\ e\ \mathsf{then}\ s_2\ \mathsf{else}\ s_2'}$$

Figure 4. Equivalence rules

and s_b be the rest of the loop body. If the step sizes satisfy $\alpha_t \leq 2/\beta$, we will prove the following judgment:

$$\vdash \{\Phi; \|w_\triangleleft - w_\triangleright\|\}\ \mathsf{sgm} \sim_{+\gamma} \mathsf{sgm}\ \{\Phi; |\ell(w_\triangleleft, z) - \ell(w_\triangleright, z)|\},$$

where $\Phi \triangleq Adj(S_\triangleleft, S_\triangleright) \wedge (w_0)_\triangleleft = (w_0)_\triangleright \wedge t_\triangleleft = t_\triangleright$ and

$$\gamma \triangleq \frac{2L^2}{n} \sum_{t=0}^{T-1} \alpha_t.$$

By soundness, this implies that SGM is γ-uniformly stable.

As before, we will first prove a simpler judgment:

$$\vdash \{\Phi; \|w_\triangleleft - w_\triangleright\|\}\ \mathsf{sgm} \sim_{+\gamma/L} \mathsf{sgm}\ \{\Phi; \|w_\triangleleft - w_\triangleright\|\}.$$

Let j be the differing index in the list of examples S (i.e., $S[j]_\triangleleft \neq S[j]_\triangleright$). First, we couple the samplings in s_a with the identity coupling, using rule [RAND] with $h = \mathrm{id}$ (Eq. (1)). Next, we perform a case analysis on whether we sample the differing vertex or not. We can define guards $e_= \triangleq i = j$ and $e_\neq \triangleq i \neq j$, and then apply the probabilistic case rule [SEQCASE]. In the case $e_=$, we can use the Lipschitz property of $\ell(z, -)$ and some properties of the norm $\|\cdot\|$ to prove

$$\vdash \{\Phi \wedge e_=; \|w_\triangleleft - w_\triangleright\|\}\ s_b \sim_{+2\alpha_t L} s_b\ \{\Phi; \|w_\triangleleft - w_\triangleright\|\};$$

this corresponds to Eq. (2). In the case $e_\neq$, we know that the examples are the same in both runs. So, can use the Lipschitz, smoothness, and convexity of $\ell(z, -)$ to prove:

$$\vdash \{\Phi \wedge e_\neq; \|w_\triangleleft - w_\triangleright\|\}\ s_b \sim_{\mathrm{id}} s_b\ \{\Phi; \|w_\triangleleft - w_\triangleright\|\};$$

this corresponds to Eq. (3). Applying [SEQCASE], noting that the probability of $e_\neq$ is $1 - 1/n$ and the probability of $e_=$ is $1/n$, we can bound the expected distance for the loop body (Eq. (4)). Applying the rule [WHILE], we can bound the distance for the whole loop (Eq. (5)). Finally, we can use the Lipschitz property of $\ell(z, -)$ and the rule [CONSEQ] to prove the desired judgment.

5.2 SGM with non-convex loss

When the loss function is non-convex, the previous proof no longer goes through. However, we can still verify the uniform stability bound by Hardt et al. [20]. Roughly, their proof proceeds by showing that with sufficiently high probability, SGM does not select the differing example until many iterations have already passed. If the step size α_t is taken to be rapidly decreasing, SGM will be contracting when it visits the differing example.

Technically, Hardt et al. [20] prove uniform stability by dividing the proof into two pieces. First they show that with sufficiently high probability, the algorithm does not select the differing example before a fixed cutoff time t_0. In particular, with high probability the parameters $w_\triangleleft$ and $w_\triangleright$ are equal up to iteration t_0. Then, they prove a uniform stability bound for SGM started at iteration t_0, assuming $w_\triangleleft = w_\triangleright$.

This proof can also be carried out in $\mathbb{E}$PRHL, with some extensions. First, we split the SGM program into two loops: iterations before t_0, and iterations after t_0. The probability of $w_\triangleleft \neq w_\triangleright$ is is precisely the expected value of the indicator function $\mathbb{1}_{\{w_\triangleleft \neq w_\triangleright\}}$, which is 1 if the parameters are not equal and 0 otherwise. Thus, we can bound the probability for the first loop by bounding this expected value in $\mathbb{E}$PRHL. For the second loop, we can proceed much like we did for standard SGM: assume that the parameters are initially equal, and then bound the expected distance on parameters.

The most difficult part is gluing these two pieces together. Roughly, we want to perform case analysis on $w_\triangleleft = w_\triangleright$ but this event depends on both sides—the existing probabilistic case rule [SEQCASE] does not apply. However, we can give an advanced probabilistic case rule, [SEQCASE-A], that does the trick. We defer the details to the appendix.

6. Population dynamics

Our second example comes from the field of evolutionary biology. Consider an infinite population separated into $m \in \mathbb{N}$ classes of organisms. The population at time t is described by a probability vector $\vec{x}_t = (x_1, \ldots, x_m)$, where x_i represents the fraction of the population belonging to the class i. In the RSM model, the evolution is described by a function f—called the *step function*—which updates the probability vectors. More precisely, the population at time $t+1$ is given as the average of $N \in \mathbb{N}$ samples according to the distribution $f(\vec{x}_t)$. A central question is whether this process mixes rapidly: starting from two possibly different population distributions, how fast do the populations converge?

We will verify a probabilistic property that is the main result needed to show rapid mixing: there is a coupling of the population distributions such that the expected distance between the two populations decreases exponentially quickly. Concretely, let $m \in \mathbb{N}$ be the number of different classes. We will work with real vectors $\vec{x} = (x_1, \ldots, x_n) \in \mathbb{R}^m$, along with the associated norm: $\|\vec{x}\|_1 = \sum_{i=1}^m |x_i|$. Let the simplex

Δ_m be the set of non-negative vectors with norm 1:

$$\Delta_m = \{\vec{x} \in \mathbb{R}^m \mid x_i \geq 0, \|\vec{x}\|_1 = 1\}$$

Elements of Δ_m can be viewed as probability distributions over the classes $\{1, \dots, m\}$; this is how we will encode the distribution of species in the population.

In the RSM model, we maintain two vectors: the true class frequencies, and the current empirical frequencies. For each of T timesteps, we first apply a function $step : \Delta_m \to \Delta_m$ to the empirical frequencies, yielding the updated true frequencies. We will assume that this function is L-Lipschitz:

$$\|step(\vec{x}) - step(\vec{y})\|_1 \leq L \cdot \|\vec{x} - \vec{y}\|_1$$

for $L < 1$.

Then, we draw N samples from the distribution given by the true frequency, to give the updated empirical frequencies. We can model this evolutionary process as a simple probabilistic program $\mathsf{popdyn}(T)$ which repeats T iterations of the evolutionary step.

```
$\vec{x} \leftarrow x_0; t \leftarrow 0;$
while $t < T$ do
  $\vec{p} \leftarrow step(\vec{x});$
  $\vec{x} \leftarrow \vec{0}; j \leftarrow 0;$
  while $j < N$ do
    $\vec{z} \xleftarrow{\$} \mathbf{Mult}(\vec{p});$
    $\vec{x} \leftarrow \vec{x} + (1/N) \cdot \vec{z};$
    $j \leftarrow j + 1;$
  $t \leftarrow t + 1$
```

Here, $\mathbf{Mult}(\vec{p})$ is the multinomial distribution with parameters $\vec{p}$. We model this distribution as producing vectors in Δ_m: with probability p_i, we produce the vector with all entries 0 except the ith entry, which is set to 1.

We run this process starting from two arbitrary initial distribution $(x_0)_\triangleleft$ and $(x_0)_\triangleright$. To show rapid mixing, the crux of the proof is to construct a coupling where the expected distance between the true distributions $x_\triangleleft$ and $x_\triangleright$ is decreasing exponentially fast as the number of steps T increases. This follows from the following judgement:

$$\vdash \{\top; \mathfrak{d}\}\ \mathsf{popdyn}(T) \sim_{\times L^T} \mathsf{popdyn}(T)\ \{\Psi; \mathfrak{d}\},$$

where $\Psi \triangleq \|\vec{x}_\triangleleft - \vec{x}_\triangleright\|_1 < 1/N \to \vec{x}_\triangleleft = \vec{x}_\triangleright$.

For some notion, let s_{out} and s_{in} be the outer and inner loops, and let w_{out} and w_{in} be the respective loop bodies. We can break the verification task into two steps. First, we focus on the inner loop. We want to prove:

$$\vdash \{\vec{x}_\triangleleft = \vec{x}_\triangleright; \|\vec{p}_\triangleleft - \vec{p}_\triangleright\|_1\}\ s_{in} \sim_{\mathrm{id}} s_{in}\ \{\Psi; \|\vec{x}_\triangleleft - \vec{x}_\triangleright\|_1\}$$

hiding trivial invariants asserting j, t are equal in both runs. By the loop rule [WHILE], it suffices to prove:

$$\vdash \{\Psi \wedge e_\triangleleft = k; \mathfrak{d}_k\}\ w_{in} \sim_{\mathrm{id}} w_{in}\ \{\Psi \wedge e_\triangleleft = k-1; \mathfrak{d}_{k-1}\}$$

for each $0 < k \leq N$, where $\mathfrak{d}_k = \|x_\triangleleft - x_\triangleright\|_1 + (k/N) \cdot \|p_\triangleleft - p_\triangleright\|_1$ and the decreasing variant is $e \triangleq N - j$.

Let the sampling command be w'_{in}, and the remainder of the loop body be w''_{in}. To prove the desired judgment, we first couple the multinomial samples with the rule [MULT-OPT] in Fig. 5. In the appendix, we show that this rule is sound by considering the *optimal coupling*—a standard coupling construction that minimizes the probability of returning different samples—of two multinomial distributions. Using the rule of consequence to scale the premetrics by $1/N$, we have:

$$\vdash \{\top; (1/N)\|\vec{p}_\triangleleft - \vec{p}_\triangleright\|_1\}\ w'_{in} \sim_{\mathrm{id}} w'_{in}\ \{\top; (1/N)\|\vec{x}_\triangleleft - \vec{x}_\triangleright\|_1\}$$

Noting that the sampling command does not modify the vectors $\vec{x}, \vec{p}$, we can add the premetric

$$\mathfrak{d}' = \|\vec{x}_\triangleleft - \vec{x}_\triangleright\|_1 + (k-1)/N \cdot \|\vec{p}_\triangleleft - \vec{p}_\triangleright\|_1$$

to the premetric in the precondition and the postcondition by the frame rule [FRAME-D].

For the deterministic commands w''_{in}, the assignment rule [ASSG] gives:

$$\vdash \{\top; \mathfrak{d}''[\vec{x} \leftarrow (\vec{x} + (1/N)\vec{z})]\}\ w''_{in} \sim_{\mathrm{id}} w''_{in}\ \{\top; \mathfrak{d}''\}$$

where the substitution is made on the respective sides, and

$$\mathfrak{d}'' = \|x_\triangleleft - x_\triangleright\|_1 + (k-1)/N \cdot \|p_\triangleleft - p_\triangleright\|_1.$$

Applying the rule of consequence with triangle inequality on the precondition, we can combine this judgment with the judgment for w'_{in} to verify the inner loop.

Turning to the outer loop, it suffices to prove

$$\vdash \{\Psi; \|\vec{x}_\triangleleft - \vec{x}_\triangleright\|_1\}\ w_{out} \sim_{\times L} w_{out}\ \{\Psi; \|\vec{x}_\triangleleft - \vec{x}_\triangleright\|_1\}.$$

This follows from the assignment rule and the judgment for the inner loop, so we can apply the loop rule [WHILE] to conclude.

7. Graph coloring

For our final example, we will consider a randomized algorithm for approximately sampling uniformly from the valid colorings of a finite graph. This algorithm, known as the *Glauber dynamics*, is a prime example of algorithm whose rapid mixing can be established using path coupling [13].

First, we recall some basic definitions and notations. Consider a graph G with a finite set of vertices V and a symmetric relation $E \subseteq V \times V$ representing the edges, and let C be a finite set of *colors*. A *coloring* of G is a map $w : V \to C$; a coloring is *valid* if neighboring vertices receive different colors: if $(a, b) \in E$, then $w(a) \neq w(b)$. We will write $w(V')$ for the set of colors at vertices V'.

For a graph G and a fixed set of colors C, there may be multiple (or perhaps zero) valid colorings. Jerrum [23]

$$[\textsc{Mult-Opt}]\ \frac{}{\vdash \{\top; \|\vec{p}_{\triangleleft} - \vec{p}_{\triangleright}\|_1\}\ \vec{x}_{\triangleleft} \xleftarrow{\$} \mathbf{Mult}(\vec{p}_{\triangleleft}) \sim_{\mathrm{id}} \vec{x}_{\triangleright} \xleftarrow{\$} \mathbf{Mult}(\vec{p}_{\triangleright})\ \{\vec{x}_{\triangleleft}, \vec{x}_{\triangleright} \in \{0,1\}^m; \|\vec{x}_{\triangleleft} - \vec{x}_{\triangleright}\|_1\}}$$

Figure 5. Optimal coupling rule for multinomial

proposed a simple Markov chain for sampling colorings, called the Glauber dynamics. The process is quite simple. Beginning at any valid coloring w, we draw a uniform vertex v and a uniform color c, and then change the color of v to c in w if this gives a valid coloring. The Glauber dynamics repeats this process for some finite number of steps T and returns the final coloring. We can model this process with the following program glauber(T):

```
$i \leftarrow 0$;
while $i < T$ do
  $v \xleftarrow{\$} V$;
  $c \xleftarrow{\$} C$;
  if $\mathcal{V}_G(w, (v, c))$ then $w \leftarrow w[v \mapsto c]$;
  $i \leftarrow i + 1$;
return $w$
```

The guard $\mathcal{V}_G(w, (v, c))$ is true when the vertex v in coloring w can be colored c. Jerrum [23] proved that the distribution on outputs for this process converges rapidly (as we take more and more steps) to the uniform distribution on valid colorings of G. While the original proof was quite technical, Bubley and Dyer [13] gave a much simpler proof of the convergence by applying their *path coupling* technique. Roughly, the path coupling technique allows to study the distance between the distributions obtained by executing the transition function of the Markov process (i.e. the loop body of the program above) on two colorings that differ on exactly one vertex and find a coupling where the expected distance (measuring in how many vertices the colorings differ) is at most $\beta < 1$. The path coupling machinery then gives a coupling of the processes started at two colorings at any distance, after executing for T iterations, and allows to conclude that after T steps the expected distance between two executions started with colorings at distance k is upper bounded by $\beta^T \cdot k$.

In $\mathbb{E}$pRHL, this property corresponds to the following judgment:

$$\vdash \{\Phi; \mathrm{pd}_{\mathrm{Adj}}\}\ \mathsf{glauber}(T) \sim_{\times(|V| \cdot \beta^T)} \mathsf{glauber}(T)\ \{\top; \mathrm{pd}_{\mathrm{Adj}}\}$$

In the judgment above, Adj is an assertion that holds between two states iff the colorings (held in the variable w) differ in the color of a single vertex, and $\mathfrak{d}' \triangleq \mathrm{pd}_{\mathrm{Adj}}$ and hence $\mathfrak{d}'$ counts the number of vertices with $w_{\triangleleft}(v) \neq w_{\triangleright}(v)$. In addition, Φ captures some properties of the graph; in particular, Φ states that Δ is the maximal degree of vertices in G, i.e., each vertex in G as at most Δ neighbors. Finally, β is a constant, strictly less than 1 under suitable conditions on Δ, $|V|$, and $|C|$.

Proceeding from the conclusion, the outline of the proof is as follows. One first applies the [While] rule, then the [Trans] rule on the loop bodies. We apply the rule with $\Phi, \Psi \triangleq i_{\triangleleft} = i_{\triangleright}$ and $\Phi' \triangleq \mathrm{Adj}$, and $f(x) = \beta \cdot x$. To check the side premises, first note that f is a linear function. Since Φ and Φ' refer to disjoint variables, we also have $\Phi \implies d_{\Phi \wedge \Phi'} = d_{\Phi}$ and $\Psi \iff \Psi^*$. Moreover, since the distance between two colorings is bounded by the number of vertices, we only have to consider the case where the two colorings are at distance 0 and 1. We consider the latter case, so we must prove

$$\vdash \{\Phi \wedge \mathrm{Adj}; d_{\mathrm{Adj}}\}\ s \sim_f s\ \{\Phi; d_{\mathrm{Adj}}\},$$

where s denotes the loop body. In this case, we apply the [SeqCase] rule, in such a way that the first judgment considers s_{samp}, consisting of the two random samplings in the loop body, and the second judgment considers the deterministic statement s_{rest}, consisting of the conditional statement and the updates. In the application of the rule, we consider three mutually exclusive cases; the intermediate assertion used for the rule is derived from the choice of the couplings made for the random samplings—we will return to the case analysis for [SeqCase] below.

Let v_δ be the vertex that is colored differently (a and b respectively) in the two input states. We will first couple the vertex samplings with the identity coupling so that $v_{\triangleleft} = v_{\triangleright}$, using the rule [Rand] with $h = \mathrm{id}$. This gives:

$$\vdash \{\Phi \wedge \mathrm{Adj}; d_{\mathrm{Adj}}\}\ v \xleftarrow{\$} V \sim_{\mathrm{id}} v \xleftarrow{\$} V\ \{\Phi \wedge v_{\triangleleft} = v_{\triangleright}; d_{\mathrm{Adj}}\}.$$

Next, we can perform a case analysis on $v_{\triangleleft}$ using the rule [Case]. If $v_{\triangleleft}$ is not a neighbor of v_δ, then we couple samplings so that $c_{\triangleleft} = c_{\triangleright}$ with [Rand] with $h = \mathrm{id}$. Otherwise, we couple $c_{\triangleleft} = \pi^{ab}(c_{\triangleright})$, where π^{ab} swaps a and b and leaves all other colors unchanged. Combined:

$$\vdash \{\Phi \wedge \mathrm{Adj}; d_{\mathrm{Adj}}\}\ s_{samp} \sim_{\mathrm{id}} s_{samp}\ \{\Theta; d_{\mathrm{Adj}}\},$$

where s_{samp} are the two sampling commands, and

$$\begin{aligned} \Theta \triangleq \Phi &\wedge v_{\triangleleft} = v_{\triangleright} \\ &\wedge v_{\triangleleft} = v_\delta \to c_{\triangleleft} = \pi^{ab}(c_{\triangleright}) \\ &\wedge v_{\triangleleft} \neq v_\delta \to c_{\triangleleft} = c_{\triangleright}. \end{aligned}$$

Now, we combine the sampling commands s_{samp} with the remaining commands s_{rest} using the rule [SeqCase]. We distinguish three mutually-exclusive cases, depending on how the distance changes under the coupling. Let q_b, q_g, q_n be the probability of the three cases.

- In the *bad case*, the distance grows to 2. We use the guard $e_b \triangleq v \in \mathcal{N}_G(v_\delta) \wedge c = b \wedge b \notin w(\mathcal{N}_G(v_\delta))$, and the assignment and consequence rules gives:

$$\vdash \{\Theta \wedge e_{b_{\triangleleft}}; d_{\mathrm{Adj}}\}\ s_{rest} \sim_{\times 2} s_{rest}\ \{i_{\triangleleft} = i_{\triangleright}; d_{\mathrm{Adj}}\}.$$

Note that $q_b \leq \Delta/|V||C|$ since for e_b to hold, we must select a neighbor of v_δ and the color b in the first side.

- In the *good case*, the distance shrinks to zero. We use the guard $e_g \triangleq v = v_\delta \wedge c \notin w(\mathcal{N}_G(v))$. By applying the assignment and consequence rules, we can prove:

  $$\vdash \{\Theta \wedge e_{g_\triangleleft}; d_{\mathrm{Adj}}\}\ s_{rest} \sim_{\times 0} s_{rest}\ \{i_\triangleleft = i_\triangleright; d_{\mathrm{Adj}}\}.$$

  It will be useful to note a *lower* bound on the probability of this case: since we must choose the differing vertex and a color different from its neighbors, and there are at most Δ neighbors, $q_g \geq (|C| - \Delta)/|C||V|$.
- In the *neutral case*, the distance stays unchanged. We use the guard $e_n \triangleq v \in \mathcal{N}_G(v_\delta) \cup v_\delta \to c \in w(\mathcal{N}_G(v))$, and the assignment rule gives:

  $$\vdash \{\Theta \wedge e_{n_\triangleleft}; d_{\mathrm{Adj}}\}\ s_{rest} \sim_{\mathrm{id}} s_{rest}\ \{i_\triangleleft = i_\triangleright; d_{\mathrm{Adj}}\}.$$

To put everything together, we need to bound the average change in distance. Since the cases are mutually exclusive and at least one case holds, we know $q_n = 1 - q_b - q_n$. Combining the three cases, we need to bound the function $x \mapsto (q_n + 2 \cdot q_b) \cdot x = (1 - q_g + q_b) \cdot x$. By the upper bound on q_b and the lower bound on q_g, we can conclude

$$\vdash \{\Phi \wedge \mathrm{Adj}; d_{\mathrm{Adj}}\}\ s_{samp}; s_{rest} \sim_f s_{samp}; s_{rest}\ \{\Phi; d_{\mathrm{Adj}}\}$$

for $f(x) = \beta \cdot x$, with

$$\beta \triangleq 1 - \frac{1}{|V|} + \frac{2\Delta}{|C||V|}.$$

When the number of colors $|C|$ is strictly larger than 2Δ, this constant β is strictly less than 1 and the Glauber dynamics is rapidly mixing.

8. Related work

While there are many natural examples of relational expectation properties, to date these properties have received little attention from the formal verification community. However, some existing systems that can prove specific examples of relational expectation properties. For instance, the standard target property in *masking* implementations in cryptography is a variant of probabilistic non-interference, known as *probing security*. Recent work introduces quantitative masking strength or QMS [17], a quantitative generalization that measures average leakage of the programs. Similarly, the bounded moment model [6] is a qualitative, expectation-based non-interference property for capturing security of parallel implementations against differential power analyses. Current verification technology for the bounded moment model is based on a meta-theorem which reduces security in the bounded moment model to probing security, and a custom program logic inspired from pRHL for proving probing security. It would be interesting to develop a custom program logic based on $\mathbb{E}$pRHL to verify a broader class of parallel implementations.

For another example, there are formal verification techniques for verifying *incentive properties* in mechanism design. These properties are relational, and when the underlying mechanism is randomized (or when the inputs are randomized), incentive properties describe the expected payoff of an agent in two executions. Barthe et al. [5, 8] show how to use a relational type system to verify these properties. While their approach is also based on couplings, they reason about expectations only at the top level, as a consequence of a particular coupling. In particular, it is not possible to compose reasoning about expected values.

Also related to our work, there have been prior efforts to verify rapid mixing for Markov chains via the path coupling method. In particular, Barthe et al. [11] use ×pRHL, a proof-relevant variant of pRHL, to extract a product program that can then be verified (using an external system) in order to prove rapid mixing. This approach can be used for proving formally rapid convergence of Markov chains, and in particular it has been applied to prove convergence of the Glauber dynamics. Our system improves upon this work in two respects. First, we can internalize the path coupling principle as a rule in our logic. Second, the amount of probabilistic reasoning is lower in our system, and confined to the [SeqCase] rule.

A bit further afield, there are many formal verification efforts in verifying probabilistic relational properties. One line is the growing body of work on verifying *differential privacy*, a notion of database privacy that can be seen as probabilistic notion of sensitivity (see, e.g., [10] for a survey of language-based techniques). Probabilistic relational properties are also central to verifying cryptographic protocols (e.g., [2, 3]). Finally, researchers have considered probabilistic relational properties in *approximate computing*, where a program may be subject to random faults or instruction skips in order to order to improve power usage (e.g., [14]); typically, key properties relate the approximate version of a program with its exact counterpart.

9. Conclusion

We have developed a program logic to reason about relational expectation properties, and demonstrated the applicability of the logic on challenging examples from machine learning, evolutionary biology, and statistical physics. Next, we intend to formally verify the soundness of the logic in a proof assistant and to mechanize the proofs of the examples considered in this paper and of other examples from the literature. One appealing direction for future work is to develop a verified library of machine learning algorithms; for instance, it would be extremely interesting to formally verify a recent result by [31], which establishes convergence of a practical variant of Stochastic Gradient. Another potential direction is to verify more general results about population dynamics, including the general case from [30].

References

[1] G. Barthe, B. Grégoire, and S. Zanella-Béguelin. Formal certification of code-based cryptographic proofs. In *ACM SIGPLAN–SIGACT Symposium on Principles of Programming Languages (POPL), Savannah, Georgia*, pages 90–101, New York, 2009.

[2] G. Barthe, F. Dupressoir, B. Grégoire, C. Kunz, B. Schmidt, and P. Strub. Easycrypt: A tutorial. In *Foundations of Security Analysis and Design VII (FOSAD)*, volume 8604 of *Lecture Notes in Computer Science*, pages 146–166. Springer-Verlag, 2013. Tutorial Lectures.

[3] G. Barthe, C. Fournet, B. Grégoire, P. Strub, N. Swamy, and S. Z. Béguelin. Probabilistic relational verification for cryptographic implementations. In *ACM SIGPLAN–SIGACT Symposium on Principles of Programming Languages (POPL), San Diego, California*, pages 193–206, 2014.

[4] G. Barthe, T. Espitau, B. Grégoire, J. Hsu, L. Stefanesco, and P.-Y. Strub. Relational reasoning via probabilistic coupling. In *International Conference on Logic for Programming, Artificial Intelligence and Reasoning (LPAR), Suva, Fiji*, volume 9450 of *Lecture Notes in Computer Science*, pages 387–401. Springer-Verlag, 2015.

[5] G. Barthe, M. Gaboardi, E. J. G. Arias, J. Hsu, A. Roth, and P. Strub. Higher-order approximate relational refinement types for mechanism design and differential privacy. In *ACM SIGPLAN–SIGACT Symposium on Principles of Programming Languages (POPL), Mumbai, India*, pages 55–68, 2015.

[6] G. Barthe, F. Dupressoir, S. Faust, B. Grégoire, F. Standaert, and P. Strub. Parallel implementations of masking schemes and the bounded moment leakage model. *IACR Cryptology ePrint Archive*, 2016:912, 2016.

[7] G. Barthe, N. Fong, M. Gaboardi, B. Grégoire, J. Hsu, and P. Strub. Advanced probabilistic couplings for differential privacy. In *ACM SIGSAC Conference on Computer and Communications Security (CCS), Vienna, Austria*, pages 55–67, 2016.

[8] G. Barthe, M. Gaboardi, E. J. G. Arias, J. Hsu, A. Roth, and P. Strub. Computer-aided verification in mechanism design. In *Conference on Web and Internet Economics (WINE), Montréal, Québec*, 2016.

[9] G. Barthe, M. Gaboardi, B. Grégoire, J. Hsu, and P. Strub. Proving differential privacy via probabilistic couplings. In *IEEE Symposium on Logic in Computer Science (LICS), New York, New York*, pages 749–758, 2016.

[10] G. Barthe, M. Gaboardi, J. Hsu, and B. C. Pierce. Programming language techniques for differential privacy. *ACM SIGLOG News*, 3(1):34–53, Jan. 2016.

[11] G. Barthe, B. Grégoire, J. Hsu, and P. Strub. Coupling proofs are probabilistic product programs. In *ACM SIGPLAN–SIGACT Symposium on Principles of Programming Languages (POPL), Paris, France*, 2017.

[12] O. Bousquet and A. Elisseeff. Stability and generalization. *Journal of Machine Learning Research*, 2:499–526, 2002.

[13] R. Bubley and M. Dyer. Path coupling: A technique for proving rapid mixing in Markov chains. In *IEEE Symposium on Foundations of Computer Science (FOCS), Miami Beach, Florida*, pages 223–231, 1997.

[14] M. Carbin, D. Kim, S. Misailovic, and M. C. Rinard. Proving acceptability properties of relaxed nondeterministic approximate programs. In *ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI), Beijing, China*, pages 169–180, 2012.

[15] S. Chaudhuri, S. Gulwani, and R. Lublinerman. Continuity analysis of programs. In *ACM SIGPLAN–SIGACT Symposium on Principles of Programming Languages (POPL), Madrid, Spain*, pages 57–70, 2010.

[16] N. M. Dixit, P. Srivastava, and N. K. Vishnoi. A finite population model of molecular evolution: Theory and computation. *Journal of Computational Biology*, 19(10):1176–1202, 2012.

[17] H. Eldib, C. Wang, M. M. I. Taha, and P. Schaumont. Quantitative masking strength: Quantifying the power side-channel resistance of software code. *IEEE Transansactions on CAD of Integrated Circuits and Systems*, 34(10):1558–1568, 2015.

[18] A. Elisseeff, T. Evgeniou, and M. Pontil. Stability of randomized learning algorithms. *Journal of Machine Learning Research*, 6:55–79, 2005.

[19] N. Foster, D. Kozen, K. Mamouras, M. Reitblatt, and A. Silva. Probabilistic NetKAT. In *European Symposium on Programming (ESOP), Eindhoven, The Netherlands*, volume 9632 of *Lecture Notes in Computer Science*, pages 282–309. Springer-Verlag, 2016.

[20] M. Hardt, B. Recht, and Y. Singer. Train faster, generalize better: Stability of stochastic gradient descent. In *International Conference on Machine Learning (ICML), New York, NY*, volume 48 of *Journal of Machine Learning Research*, pages 1225–1234. JMLR.org, 2016.

[21] D. L. Hartl and A. G. Clark. Sinauer Associates, 2006.

[22] T. Jansen. *Analyzing evolutionary algorithms: The computer science perspective*. Springer Science & Business Media, 2013.

[23] M. Jerrum. A very simple algorithm for estimating the number of k-colorings of a low-degree graph. *Random Structures and Algorithms*, 7(2):157–166, 1995.

[24] B. L. Kaminski, J. Katoen, C. Matheja, and F. Olmedo. Weakest precondition reasoning for expected run-times of probabilistic programs. In *European Symposium on Programming (ESOP), Eindhoven, The Netherlands*, volume 9632 of *Lecture Notes in Computer Science*, pages 364–389. Springer-Verlag, 2016.

[25] D. Kozen. Semantics of probabilistic programs. In *IEEE Symposium on Foundations of Computer Science (FOCS), San Juan, Puerto Rico*, pages 101–114, 1979.

[26] D. Kozen. A probabilistic PDL. *Journal of Computer and System Sciences*, 30(2):162–178, 1985.

[27] D. A. Levin, Y. Peres, and E. L. Wilmer. *Markov chains and mixing times*. American Mathematical Society, 2009.

[28] T. Lindvall. *Lectures on the coupling method*. Courier Corporation, 2002.

[29] C. Morgan, A. McIver, and K. Seidel. Probabilistic predicate transformers. *ACM Transactions on Programming Languages and Systems*, 18(3):325–353, 1996.

[30] I. Panageas, P. Srivastava, and N. K. Vishnoi. Evolutionary dynamics in finite populations mix rapidly. In *ACM–SIAM Symposium on Discrete Algorithms (SODA), Arlington, Virginia*, pages 480–497, 2016.

[31] O. Shamir. Without-replacement sampling for stochastic gradient methods: Convergence results and application to distributed optimization. *CoRR*, abs/1603.00570, 2016.

[32] H. Thorisson. *Coupling, Stationarity, and Regeneration*. Springer-Verlag, 2000.

[33] C. Villani. *Optimal transport: Old and new*. Springer-Verlag, 2008.

[34] N. K. Vishnoi. The speed of evolution. In *ACM–SIAM Symposium on Discrete Algorithms (SODA), San Diego, California*, pages 1590–1601, 2015.