Coalgebraic Tools for Randomness-Conserving Protocols

Dexter Kozen and Matvey Soloviev

Cornell University

Abstract. We propose a coalgebraic model for constructing and reasoning about state-based protocols that implement efficient reductions among random processes. We provide basic tools that allow efficient protocols to be constructed in a compositional way and analyzed in terms of the tradeoff between latency and loss of entropy. We show how to use these tools to construct various entropy-conserving reductions between processes.

1 Introduction

In low-level performance-critical computations—for instance, data-forwarding devices in packet-switched networks—it is often desirable to minimize local state in order to achieve high throughput. But if the situation requires access to a source of randomness, say to implement randomized routing or load-balancing protocols, it may be necessary to convert the output of the source to a form usable by the protocol. As randomness is a scarce resource to be conserved like any other, these conversions should be performed as efficiently as possible and with a minimum of machinery.

In this paper we propose a coalgebraic model for constructing and reasoning about state-based protocols that implement efficient reductions among random processes. Efficiency is measured by the ratio of entropy produced to entropy consumed. The efficiency cannot exceed the information-theoretic bound of unity, but it should be as close to unity as can be achieved with simple state-based devices. We provide basic tools that allow efficient protocols to be constructed in a compositional way and analyzed in terms of the tradeoff between latency and loss of entropy.

We use these tools to construct the following reductions between processes, where k is the latency parameter:

- d-uniform to c-uniform with loss $\Theta(k^{-1})$
- d-uniform to arbitrary rational with loss $\Theta(k^{-1})$
- d-uniform to arbitrary with loss $\Theta(k^{-1})$
- arbitrary to c-uniform with loss $\Theta(\log k/k)$
- $(1/r, (r-1)/r)$ to c-uniform with loss $\Theta(k^{-1})$

Omitted proofs can be found in the full version of the paper [11].

1.1 Related Work

Since von Neumann's classic result showing how to simulate a fair coin with a coin of unknown bias, many authors have studied variants of this problem. Our work is heavily inspired by the work of Elias [8], who studies entropy-optimal generation of uniform distributions from known sources. The definition of conservation of entropy and a concept related to latency are defined there. Mossel, Peres, and Hillar [17] characterize the set of functions $f : (0,1) \to (0,1)$ for which it is possible to simulate an $f(p)$-biased coin with a p-biased coin when p is unknown. Peres [16] shows how to iterate von Neumann's procedure for producing a fair coin from a biased coin to approximate the entropy bound. Blum [2] shows how to extract a fair coin from a Markov chain. Pae and Loui [13, 14] present several simulations for optimal conversions between discrete distributions, known and unknown. The main innovation in this paper is the coalgebraic model that allows compositional reasoning about such reductions.

There is also a large body of related work on extracting randomness from weak random sources (e.g. [7,12,18,19]). These models typically work with imperfect knowledge of the input source and provide only approximate guarantees on the quality of the output. Here we assume that the statistical properties of the input and output are known completely, and simulations must be exact.

2 Definitions

Informally, a **reduction** from a stochastic process X to another stochastic process Y is a deterministic protocol that consumes a finite or infinite stream of letters from an alphabet Σ and produces a finite or infinite stream of letters from another alphabet Γ. If the letters of the input stream are distributed as X, then the letters of the output stream should be distributed as Y. Of particular interest are reductions between **Bernoulli processes**, in which the letters of the input and output streams are independent and identically distributed according to distributions μ on Σ and ν on Γ, respectively. In this case, we say that procedure is a reduction from μ to ν.

To say that the protocol is **deterministic** means that the only source of randomness is the input stream. It makes sense to talk about the expected number of input letters read before halting or the probability that the first letter emitted is a, but any such statistical measurements are taken with respect to the distribution of the input stream.

There are several ways to formalize the notion of a reduction. One approach, following [16], is to model a reduction as a map $f : \Sigma^* \to \Gamma^*$ that is monotone with respect to the prefix relation on strings; that is, if $x, y \in \Sigma^*$ and x is a prefix of y, then $f(x)$ is a prefix of $f(y)$. Monotonicity implies that f can be extended uniquely by continuity to domain $\Sigma^* \cup \Sigma^\omega$ and range $\Gamma^* \cup \Gamma^\omega$. The map f would then constitute a reduction from the stochastic process $X = X_0X_1X_2\cdots$ to $f(X_0X_1X_2\cdots)$. To be a reduction from μ to ν, it must be that if the X_i are

independent and identically distributed as μ, and if Y_i is the value of the ith letter of $f(X_0X_1X_2\cdots)$, then the Y_i are independent and identically distributed as ν.

In this paper we propose an alternative state-based approach in which protocols are modeled as coalgebras $\delta : S \times \Sigma \to S \times \Gamma^*$, where S is a (possibly infinite) set of **states**. This approach allows a more streamlined treatment of common programming constructions such as composition, which is perhaps more appealing from a programming perspective.

2.1 Protocols and Reductions

Let Σ, Γ be finite alphabets. Let Σ^* denote the set of finite words and Σ^ω the set of ω-words (streams) over Σ. We use $x, y, \ldots$ for elements of Σ^* and $\alpha, \beta, \ldots$ for elements of Σ^ω. The symbols $\preceq$ and $\prec$ denote the prefix and proper prefix relations, respectively.

If μ is a probability measure on Σ, we endow Σ^ω with the product measure in which each symbol is distributed as μ. The notation $\Pr(A)$ for an event A refers to this measure. The measurable sets of Σ^ω are the Borel sets of the Cantor space topology whose basic open sets are the **intervals** $\{\alpha \in \Sigma^\omega \mid x \prec \alpha\}$ for $x \in \Sigma^*$, and $\mu(\{\alpha \in \Sigma^\omega \mid x \prec \alpha\}) = \mu(x)$, where $\mu(a_1a_2\cdots a_n) = \mu(a_1)\mu(a_2)\cdots\mu(a_n)$.

A **protocol** is a coalgebra (S, δ) where $\delta : S \times \Sigma \to S \times \Gamma^*$. We can immediately extend δ to domain $S \times \Sigma^*$ by coinduction:

$$\begin{aligned}\delta(s, \varepsilon) &= (s, \varepsilon)\\ \delta(s, ax) &= \text{let } (t, y) = \delta(s, a) \text{ in let } (u, z) = \delta(t, x) \text{ in } (u, yz).\end{aligned}$$

Since the two functions agree on $S \times \Sigma$, we use the same name. It follows that

$$\delta(s, xy) = \text{let } (t, z) = \delta(s, x) \text{ in let } (u, w) = \delta(t, y) \text{ in } (u, zw).$$

By a slight abuse, we define the **length** of the output as the length of its second component as a string in Γ^* and write $|\delta(s, x)|$ for $|z|$, where $\delta(s, x) = (t, z)$.

A protocol δ also induces a partial map $\delta^\omega : S \times \Sigma^\omega \rightharpoonup \Gamma^\omega$ by coinduction:

$$\delta^\omega(s, a\alpha) = \text{let } (t, z) = \delta(s, a) \text{ in } z \cdot \delta^\omega(t, \alpha).$$

It follows that

$$\delta^\omega(s, x\alpha) = \text{let } (t, z) = \delta(s, x) \text{ in } z \cdot \delta^\omega(t, \alpha).$$

Given $\alpha \in \Sigma^\omega$, this defines a unique infinite string in $\delta^\omega(s, \alpha) \in \Gamma^\omega$ except in the degenerate case in which only finitely many output letters are ever produced. A protocol is said to be **productive** (with respect to a given probability measure on input streams) if, starting in any state, an output symbol is produced within finite expected time. It follows that infinitely many output letters are produced with probability 1.

Now let ν be a probability measure on Γ. Endow Γ^ω with the product measure in which each symbol is distributed as ν. As with μ, define $\nu(a_1 a_2 \cdots a_n) = \nu(a_1)\nu(a_2)\cdots\nu(a_n)$ for $a_i \in \Gamma$. We say that a protocol (S, δ, s) with start state $s \in S$ is a **reduction from** μ **to** ν if for all $y \in \Gamma^*$,

$$\Pr(y \preceq \delta^\omega(s, \alpha)) = \nu(y), \tag{1}$$

where the probability Pr is with respect to the product measure μ on Σ^ω. This implies that the symbols of $\delta^\omega(s, \alpha)$ are independent and identically distributed as ν.

2.2 Restart Protocols

A **prefix code** is a subset $A \subseteq \Sigma^*$ such that every element of Σ^ω has at most one prefix in A. Thus the elements of a prefix code are $\preceq$-incomparable. A prefix code is **exhaustive** (with respect to a given probability measure on input streams) if $\Pr(\{\alpha \in \Sigma^\omega \text{ has a prefix in } A\}) = 1$. By König's lemma, if every $\alpha \in \Sigma^\omega$ has a prefix in A, then A is finite.

A **restart protocol** is protocol (S, δ, s) of a special form determined by a function $f : A \rightharpoonup \Gamma^*$, where A is an exhaustive prefix code. Here s is a designated start state. Intuitively, starting in s, we read symbols of Σ from the input stream until encountering a string $x \in A$, output $f(x)$, then return to s and repeat. Note that we are not assuming A to be finite.

Formally, we can take the state space to be

$$S = \{u \in \Sigma^* \mid x \not\preceq u \text{ for any } x \in A\}$$

and define $\delta : S \times \Sigma \to S \times \Gamma^*$ by

$$\delta(u, a) = \begin{cases} (ua, \varepsilon), & ua \notin A, \\ (\varepsilon, z), & ua \in A \text{ and } f(ua) = z \end{cases}$$

with start state ε. Then for all $x \in A$, $\delta(\varepsilon, x) = (\varepsilon, f(x))$.

As with the more general protocols, we can extend to a function on streams, but here the definition takes a simpler form: for $x \in A$,

$$\delta^\omega(\varepsilon, x\alpha) = f(x) \cdot \delta^\omega(\varepsilon, \alpha), \quad x \in A,\ \alpha \in \Sigma^\omega.$$

A restart protocol is **positive recurrent** (with respect to a given probability measure on input streams) if, starting in the start state s, the probability of eventually returning to s is 1, and moreover the expected time before the next visit to s is finite. All finite-state restart protocols are positive recurrent, but infinite-state ones need not be.

2.3 Convergence

We will have the occasion to discuss the convergence of random variables. There are several notions of convergence in the literature, but for our purposes the most useful is **convergence in probability**. Let X and X_n, $n \geq 0$ be bounded nonnegative random variables. We say that the sequence X_n **converges to** X **in probability** and write $X_n \xrightarrow{\Pr} X$ if for all fixed $\delta > 0$,

$$\Pr(|X_n - X| > \delta) = o(1).$$

Let $\mathbb{E}(X)$ denote the expected value of X and $\mathbb{V}(X)$ its variance.

Lemma 1.

(i) *If* $X_n \xrightarrow{\Pr} X$ *and* $X_n \xrightarrow{\Pr} Y$, *then* $X = Y$ *with probability 1.*
(ii) *If* $X_n \xrightarrow{\Pr} X$ *and* $Y_n \xrightarrow{\Pr} Y$, *then* $X_n + Y_n \xrightarrow{\Pr} X + Y$ *and* $X_n Y_n \xrightarrow{\Pr} XY$.
(iii) *If* $X_n \xrightarrow{\Pr} X$ *and* X *is bounded away from 0, then* $1/X_n \xrightarrow{\Pr} 1/X$.
(iv) *If* $\mathbb{V}(X_n) = o(1)$ *and* $\mathbb{E}(X_n) = e$ *for all* n, *then* $X_n \xrightarrow{\Pr} e$.

Proof. Clause (iv) follows from the Chebyshev bound. Please see [11] for details. □

See [4, 9] for a more thorough introduction.

2.4 Efficiency

The **efficiency** of a protocol is the long-term ratio of entropy production to entropy consumption. Formally, for a fixed protocol $\delta : S \times \Sigma \to S \times \Gamma^*$, $s \in S$, and $\alpha \in \Sigma^\omega$, define the random variable

$$E_n(\alpha) = \frac{|\delta(s, \alpha_n)|}{n} \cdot \frac{H(\nu)}{H(\mu)}, \tag{2}$$

where H is the Shannon entropy $H(p_1, \ldots, p_n) = -\sum_{i=1}^n p_i \log p_i$ (logarithms are base 2 if not otherwise annotated), μ and ν are the input and output distributions, respectively, and α_n is the prefix of α of length n. Intuitively, the Shannon entropy measures the number of fair coin flips the distribution is worth, and the random variable E_n measures the ratio of entropy production to consumption after n steps of δ starting in state s. Here $|\delta(s, \alpha_n)|\, H(\nu)$ (respectively, $nH(\mu)$) is the contribution along α to the production (respectively, consumption) of entropy in the first n steps. We write $E_n^{\delta,s}$ when we need to distinguish the E_n associated with different protocols and start states.

In most cases of interest, E_n converges in probability to a unique constant value independent of start state and history. When this occurs, we call this constant value the **efficiency** of the protocol δ and denote it by Eff_δ. Notationally, $E_n \xrightarrow{\Pr} \mathsf{Eff}_\delta$. One must be careful when analyzing infinite-state protocols: The

efficiency is well-defined for finite-state protocols, but may not exist in general. For restart protocols, it is enough to measure the ratio for one iteration of the protocol.

In §3.2 we will give sufficient conditions for the existence of Eff_δ that is satisfied by all protocols considered in §4.

2.5 Latency

The **latency** of a protocol from a given state s is the expected consumption before producing at least one output symbol, starting from state s. This is proportional to the expected number of input letters consumed before emitting at least one symbol. The latency of a protocol is finite if and only if the protocol is productive. All positive recurrent restart protocols that emit at least one symbol are productive. We will often observe a tradeoff between latency and efficiency.

Suppose we iterate a positive recurrent restart protocol only until at least one output symbol is produced. That is, we start in the start state s and choose one string x in the prefix code randomly according to μ. If at least one output symbol is produced, we stop. Otherwise, we repeat the process. The sequence of iterations to produce at least one output symbol is called an **epoch**. The latency is the expected consumption during an epoch. If p is the probability of producing an output symbol in one iteration, then the sequence of iterations in a epoch forms a Bernoulli process with success probability p. The latency is thus $1/p$, the expected stopping time of the Bernoulli process, times the expected consumption in one iteration, which is finite due to the assumption that the protocol is positive recurrent.

3 Basic Results

Let $\delta : S \times \Sigma \to S \times \Gamma^*$ be a protocol. We can associate with each $y \in \Gamma^*$ and state $s \in S$ a prefix code in Σ^*, namely

$$\mathsf{pc}_\delta(s, y) = \{\text{minimal-length strings } x \in \Sigma^* \text{ such that } y \preceq \delta(s, x)\}.$$

The string y is generated as a prefix of the output if and only if exactly one $x \in \mathsf{pc}_\delta(s, y)$ is consumed as a prefix of the input. These events must occur with the same probability, so

$$\nu(y) = \Pr(y \prec \delta^\omega(s, \alpha)) = \mu(\mathsf{pc}_\delta(s, y)). \tag{3}$$

Note that $\mathsf{pc}_\delta(s, y)$ need not be finite.

Lemma 2. *If $A \subseteq \Gamma^*$ is a prefix code, then so is $\bigcup_{y \in A} \mathsf{pc}_\delta(s, y) \subseteq \Sigma^*$, and*

$$\nu(A) = \mu(\bigcup_{y \in A} \mathsf{pc}_\delta(s, y)).$$

If $A \subseteq \Gamma^*$ is exhaustive, then so is $\bigcup_{y \in A} \mathsf{pc}_\delta(s, y) \subseteq \Sigma^*$.

Proof. Please see [11]. □

Lemma 3.

(i) *The partial function* $\delta^\omega(s,-) : \Sigma^\omega \rightharpoonup \Gamma^\omega$ *is continuous, thus Borel measurable.*
(ii) $\delta^\omega(s,\alpha)$ *is almost surely infinite; that is,* $\mu(\mathrm{dom}\,\delta^\omega(s,-)) = 1$.
(iii) *The measure* ν *on* Γ^ω *is the push-forward measure* $\nu = \mu \circ \delta^\omega(s,-)^{-1}$.

Proof. Please see [11]. □

Lemma 4. *If* δ *is a reduction from* μ *to* ν*, then the random variables* E_n *defined in* (2) *are continuous and uniformly bounded by an absolute constant* $R > 0$ *depending only on* μ *and* ν.

Proof. Please see [11]. □

3.1 Composition

Protocols can be composed sequentially as follows. If

$$\delta_1 : S \times \Sigma \to S \times \Gamma^* \qquad\qquad \delta_2 : T \times \Gamma \to T \times \Delta^*,$$

then

$$(\delta_1 \,;\, \delta_2) : S \times T \times \Sigma \to S \times T \times \Delta^*$$
$$(\delta_1 \,;\, \delta_2)((s,t),a) = \text{let } (u,y) = \delta_1(s,a) \text{ in let } (v,z) = \delta_2(t,y) \text{ in } ((u,v),z).$$

Intuitively, we run δ_1 for one step and then run δ_2 on the output of δ_1. The following theorem shows that the map on infinite strings induced by the sequential composition of protocols is almost everywhere equal to the functional composition of the induced maps of the component protocols.

Theorem 1. *The partial maps* $(\delta_1 \,;\, \delta_2)^\omega((s,t),-)$ *and* $\delta_2^\omega(t, \delta_1^\omega(s,-))$ *of type* $\Sigma^\omega \rightharpoonup \Delta^\omega$ *are defined and agree on all but a* μ*-nullset.*

Proof. We restrict inputs to the subset of Σ^ω on which δ_1^ω is defined and produces a string in Γ^ω on which δ_2^ω is defined. These sets are of measure 1. To show that $(\delta_1 \,;\, \delta_2)^\omega((s,t),\alpha) = \delta_2^\omega(t, \delta_1^\omega(s,\alpha))$, we show that the binary relation

$$\beta R \gamma \;\Leftrightarrow\; \exists \alpha \in \Sigma^\omega \; \exists s \in S \; \exists t \in T \;\; \beta = (\delta_1 \,;\, \delta_2)^\omega((s,t),\alpha) \wedge \gamma = \delta_2^\omega(t, \delta_1^\omega(s,\alpha))$$

on Δ^ω is a bisimulation. Please see [11] for details. □

Corollary 1. *If* $\delta_1(s,-)$ *is a reduction from* μ *to* ν *and* $\delta_2(t,-)$ *is a reduction from* ν *to* o*, then* $(\delta_1 \,;\, \delta_2)((s,t),-)$ *is a reduction from* μ *to* o.

Proof. Please see [11]. □

Theorem 2. *If* $\delta_1(s,-)$ *is a reduction from* μ *to* ν *and* $\delta_2(t,-)$ *is a reduction from* ν *to* o*, and if* Eff_{δ_1} *and* Eff_{δ_2} *exist, then* $\mathsf{Eff}_{\delta_1;\delta_2}$ *exists and* $\mathsf{Eff}_{\delta_1;\delta_2} = \mathsf{Eff}_{\delta_1} \cdot \mathsf{Eff}_{\delta_2}$.

Proof. Please see [11]. □

In the worst case, the latency of compositions of protocols is also the product of their latencies: if the first protocol only outputs one character at a time, then the second protocol may have to wait the full latency of the first protocol for each of the characters it needs to read in order to emit a single one.

3.2 Serial Protocols

Consider a sequence $(S_0, \delta_0, s_0), (S_1, \delta_1, s_1), \ldots$ of positive recurrent restart protocols defined in terms of maps $f_k : A_k \to \Gamma^*$, where the A_k are exhaustive prefix codes, as described in §2.2. These protocols can be combined into a single **serial protocol** δ that executes one iteration of each δ_k, then goes on to the next. Formally, the states of δ are the disjoint union of the S_k, and δ is defined so that $\delta(s_k, x) = (s_{k+1}, f_k(x))$ for $x \in A_k$, and within S_k behaves like δ_k.

Let C_k and P_k be the number of input symbols consumed and produced, respectively, in one iteration of the component protocol δ_k starting from s_k. Let $e(n)$ be the index of the component protocol $\delta_{e(n)}$ in which the n-th step of the combined protocol occurs. These are random variables whose values depend on the input sequence $\alpha \in \Sigma^\omega$. Let $c_k = \mathbb{E}(C_k)$ and $p_k = \mathbb{E}(P_k)$.

To derive the efficiency of serial protocols, we need a form of the law of large numbers (see [4, 9]). Unfortunately, the law of large numbers as usually formulated does not apply verbatim, as the random variables in question are bounded but not independent, or (under a different formulation) independent but not bounded. Our main result, Theorem 3 below, can be regarded as a specialized version of this result adapted to our needs.

Our version requires that the variances of certain random variables vanish in the limit. This holds under a mild condition (4) on the growth rate of m_n, the maximum consumption in the nth component protocol, and is true for all serial protocols considered in this paper. The condition (4) is satisfied by all finite serial protocols in which m_n is bounded, or $m_n = O(n)$ and c_n is unbounded.

Lemma 5. *Let $\mathbb{V}(X)$ denote the variance of X. Let $m_n = \max_{x \in A_k} |x| \cdot H(\mu)$. If*

$$m_n = o(\sum_{i=0}^{n-1} c_i), \tag{4}$$

then

$$\mathbb{V}(\frac{\sum_{i=0}^{n} C_i}{\sum_{i=0}^{n} c_i}) = o(1) \qquad \mathbb{V}(\frac{C_n}{\sum_{i=0}^{n-1} c_i}) = o(1). \tag{5}$$

If in addition $p_n = \Theta(c_n)$, then

$$\mathbb{V}(\frac{\sum_{i=0}^{n} P_i}{\sum_{i=0}^{n} p_i}) = o(1) \qquad \mathbb{V}(\frac{P_n}{\sum_{i=0}^{n-1} p_i}) = o(1). \tag{6}$$

Proof. Please see [11]. □

The following is our main theorem.

Theorem 3. *Let δ be a serial protocol with finite-state components $\delta_0, \delta_1, \ldots$ satisfying* (4). *If the limit $\ell = \lim_n \frac{\sum_{i=0}^{n} p_i}{\sum_{i=0}^{n} c_i}$ exists, then the efficiency of the serial protocol exists and is equal to ℓ.*

Proof. Please see [11]. □

4 Reductions

In this section we present a series of reductions between distributions of certain forms. Each example defines a sequence of positive recurrent restart protocols (§2.2) indexed by a latency parameter k with efficiency tending to 1. By Theorem 3, these can be combined in a serial protocol (§3.2) with asymptotically optimal efficiency, albeit at the cost of unbounded latency.

4.1 Uniform ⇒ Uniform

Let $c, d \geq 2$. In this section we construct a family of restart protocols with latency k mapping d-uniform streams to c-uniform streams with efficiency $1 - \Theta(k^{-1})$. The Shannon entropy of the input and output distributions are $\log d$ and $\log c$, respectively.

Let $m = \lfloor k \log_c d \rfloor$. Then $c^m \leq d^k < c^{m+1}$. It follows that

$$\frac{c^m}{d^k} = \Theta(1) \qquad\qquad 1 - \Theta(1) < \frac{m \log c}{k \log d} \leq 1. \tag{7}$$

Let the c-ary expansion of d^k be

$$d^k = \sum_{i=0}^{m} a_i c^i, \tag{8}$$

where $0 \leq a_i \leq c - 1$, $a_m \neq 0$.

The protocol P_k is defined as follows. Do k calls on the d-uniform distribution. For each $0 \leq i \leq m$, for $a_i c^i$ of the possible outcomes, emit a c-ary string of length i, every possible such string occurring exactly a_i times. For a_0 outcomes, nothing is emitted, and this is lost entropy, but this occurs with probability $a_0 d^{-k}$. After that, restart the protocol.

By elementary combinatorics,

$$\sum_{i=0}^{m-1} (m-i) a_i c^i \leq \sum_{i=0}^{m-1} (m-i)(c-1)c^i = \frac{c(c^m - 1)}{c-1} - m. \tag{9}$$

In each run of P_k, the expected number of c-ary digits produced is

$$\begin{aligned}
\sum_{i=0}^{m} i a_i c^i d^{-k} &= d^{-k} \Big(\sum_{i=0}^{m} m a_i c^i - \sum_{i=0}^{m} (m-i) a_i c^i \Big) \\
&\geq m - d^{-k} \Big(\frac{c(c^m - 1)}{c-1} - m \Big) && \text{by (8) and (9)} \\
&= m - \Theta(1) && \text{by (7),}
\end{aligned}$$

thus the entropy production is at least $m \log c - \Theta(1)$. The number of d-ary digits consumed is k, thus the entropy consumption is $k \log d$. The efficiency is

$$\frac{m \log c - \Theta(1)}{k \log d} \geq 1 - \Theta(k^{-1}).$$

The output is uniformly distributed, as there are $\sum_{i=\ell}^{m} a_i c^i$ equal-probability outcomes that produce a string of length ℓ or greater, and each output letter a appears as the ℓth output letter in equally many strings of the same length, thus is output with equal probability.

4.2 Uniform ⇒ Rational

Let $c, d \geq 2$. In this section, we will present a family of restart protocols D_k mapping d-uniform streams over Σ to streams over a c-symbol alphabet $\Gamma = \{1, \ldots, c\}$ with rational symbol probabilities with a common denominator d, e.g. $p_1 = a_1/d, \ldots, p_c = a_c/d$. Unlike the protocols in the previous section, here we emit a fixed number of symbols in each round while consuming a variable number of input symbols according to a particular prefix code $S \subseteq \Sigma^*$. The protocol D_k has latency at most $kH(p_1, \ldots, p_c)/\log d + 2$ and efficiency $1 - \Theta(k^{-1})$, exhibiting a similar tradeoff to the previous family.

To define D_k, we will construct an exhaustive prefix code S over the source alphabet, which will be partitioned into pairwise disjoint sets $S_y \subseteq \Sigma^*$ associated with each k-symbol output word $y \in \Gamma^k$. All input strings in the set S_y will map to the output string y.

By analogy with $p_1, \ldots, p_c$, let p_y denote the probability of the word $y = s_1 \cdots s_k$ in the output process. Since the symbols of y are chosen independently, p_y is the product of the probabilities of the individual symbols. It is therefore of the form $p_y = a_y d^{-k}$, where $a_y = a_{s_1} \cdots a_{s_k}$ is an integer.

Let $m_y = \lfloor \log_d a_y \rfloor$ and let $a_y = \sum_{j=0}^{m_y} a_{yj} d^j$ be the d-ary expansion of a_y. We will choose a set of $\sum_{y \in \Gamma^k} \sum_{j=0}^{m_y} a_{yj}$ prefix-incomparable codewords and assign them to the S_y so that each S_y contains a_{yj} codewords of length $k - j$ for each $0 \leq j \leq m_y$. This is possible by the Kraft inequality (see [5, Theorem 5.2.1] or [1, Theorem 1.6]); we need only verify that $\sum_{y \in \Gamma^k} \sum_{j=0}^{m_y} a_{yj} d^{-(k-j)} \leq 1$. In fact, equality holds:

$$\sum_{j=0}^{m_y} a_{yj} d^{-(k-j)} = a_y d^{-k} = p_y, \qquad \text{so} \sum_{y \in \Gamma^k} \sum_{j=0}^{m_y} a_{yj} d^{-(k-j)} = \sum_{y \in \Gamma^k} p_y = 1. \tag{10}$$

Since the d symbols of the input process are distributed uniformly, the probability that the input stream begins with a given string of length n is d^{-n}. So

$$\Pr(y \preceq \delta_k^\omega(*, x)) = \Pr(\exists x \in S_y : x \preceq x) = \sum_{x \in S_y} d^{-|x|} = \sum_{j=0}^{m_y} a_{yj} d^{-(k-j)}$$

is p_y as required, and D_k is indeed a reduction. Moreover, by (10), the probability that a prefix is in some S_y is 1, so the code is exhaustive.

To analyze the efficiency of the simulation, we will use the following lemma.

Lemma 6. *Let the d-ary expansion of a be $\sum_{i=0}^{m} a_i d^i$, where $m = \lfloor \log_d a \rfloor$. Then*

$$\left(\log_d a - \frac{2d-1}{d-1}\right) a < \left(m - \frac{d}{d-1}\right) a < \sum_{i=0}^{m} i a_i d^i \leq ma.$$

Proof. Please see [11]. □

Observe now that

$$\begin{aligned}
\sum_{x \in S_y} d^{-|x|} \cdot |x| &= \sum_{j=0}^{m_y} a_{yj} d^{j-k}(k-j) = kp_y - \sum_{j=0}^{m_y} j a_{yj} d^{j-k} = kp_y - d^{-k} \sum_{j=0}^{m_y} j a_{yj} d^j \\
&\overset{6}{<} kp_y - d^{-k}\left(\log_d a_y - \frac{2d-1}{d-1}\right) a_y = kp_y - \left(\log_d (p_y d^k) - \frac{2d-1}{d-1}\right) a_y d^{-k} \\
&= kp_y - \left(\log_d p_y + k - \frac{2d-1}{d-1}\right) p_y = \frac{2d-1}{d-1} p_y - p_y \log_d p_y.
\end{aligned}$$

Thus the expected number of input symbols consumed is

$$\sum_{y \in \Gamma^k} \sum_{x \in S_y} d^{-|x|} \cdot |x| < \sum_{y \in \Gamma^k} \left(\frac{2d-1}{d-1} p_y - p_y \log_d p_y\right) = \frac{2d-1}{d-1} + \frac{kH(p_1, \ldots, p_c)}{\log d}.$$

and as $H(U_d) = \log d$, the expected consumption of entropy is at most

$$H(U_d) \cdot \left(\frac{2d-1}{d-1} + \frac{kH(p_1, \ldots, p_c)}{\log d}\right) = \log d \cdot \frac{2d-1}{d-1} + kH(p_1, \ldots, p_c).$$

The number of output symbols is k, so the production of entropy is $kH(p_1, \ldots, p_c)$. Thus the efficiency is at least

$$\frac{kH(p_1, \ldots, p_c)}{kH(p_1, \ldots, p_c) + \log d \cdot \frac{2d-1}{d-1}} = \frac{1}{1 + \Theta(k^{-1})} = 1 - \Theta(k^{-1}).$$

4.3 Uniform ⇒ Arbitrary

Now suppose the target distribution is over an alphabet $\Gamma = \{1, \ldots, c\}$ with arbitrary real probabilities $p_1^*, \ldots, p_c^*$. We exhibit a family of restart protocols D_k that map the uniform distribution over a d-symbol alphabet Σ to the distribution $\{p_i^*\}$ with efficiency $1 - \Theta(k^{-1})$. Moreover, if the p_i^* and basic arithmetic operations are computable, then so is D_k. We assume that $d > 1/\min_i p_i^*$, so by the pigeonhole principle, $d > c$. If we want to convert a uniform distribution over fewer symbols to $\{p_i^*\}$, we can treat groups of k subsequent symbols as a single symbol from a d^k-sized alphabet.

Unlike the other protocols we have seen so far, these protocols require infinitely many states in general. This follows from a cardinality argument: there

are only countably many reductions specified by finite-state protocols, but uncountably many probability distributions on c symbols.

We will use the set of real probability distributions $\{p_y\}_{y\in\Gamma^k}$ on k-symbol output strings as our state space. As the initial state, we use the extension of the target distribution onto k-symbol strings $\{p^*_y\}$ with $p^*_{s_1\cdots s_k} = p^*_{s_1}\cdots p^*_{s_k}$.

The construction of the protocol D_k at each state $\{p_y\}_{y\in\Gamma^k}$ closely follows the one for rational target distributions presented in §4.2. Since we can no longer assume that the probabilities p_y are of the form $a_y d^{-k}$ for some integer a_y, we will instead use the greatest a_y such that $q_y := a_y d^{-k} \le p_y$, namely $a_y = \lfloor p_y d^k \rfloor$. We have $p_y - q_y > d^{-k}$. Of course, these may no longer sum to 1, and so we also define a **residual probability** $rd^{-k} = 1 - \sum_{y\in\Gamma^k} q_y < (c/d)^k$.

As in §4.2, we construct sets of prefix-incomparable codewords S_y for each k-symbol output word y based on the d-ary expansion of a_y, with the aim that the probability of encountering a codeword in S_y is exactly $a_y d^{-k}$. If the protocol encounters a codeword in S_y, it outputs y and restarts.

We also construct a set S_r based on the d-ary expansion of the residual r. If a codeword in S_r is encountered, then we output nothing, and instead transition to a different state to run the protocol again with the **residual distribution** $\{p'_y\}_{y\in\Gamma^k}$, where p'_y is the probability we lost when rounding down earlier:

$$p'_y = \frac{p_y - q_y}{\sum_{y\in\Gamma^k}(p_y - q_y)} = \frac{p_y - q_y}{rd^{-k}}.$$

The correctness of the protocol follows because each additional generation of states acts contractively on the distribution of output symbols, with a unique fixpoint at the true distribution. Roughly speaking, suppose the protocol we execute at the state $\{p'_y\}$ has an error within ε, i.e. the probability that it will output the string y is bounded by $p'_y \pm \varepsilon$. As in §4.2, at state $\{p_y\}$, we encounter a string in S_y and output y with probability q_y. With probability rd^{-k}, we encounter a string in S_r and pass to the state $\{p'_y\}$, where we output y with probability bounded by $p'_y \pm \varepsilon$. Hence the total probability of emitting y is bounded by

$$q_y + rd^{-k}\left(\frac{p_y - q_y}{rd^{-k}} \pm \varepsilon\right) = p_y \pm rd^{-k}\varepsilon.$$

In particular, the error at $\{p_y\}$ is at most $(c/d)^k\varepsilon$.

Lemma 7. *If $\{q_y\}_{y\in\Gamma^k}$ is such that $0 < p^*_y - q_y < 1/d^k$ for all y, then*

$$\left| -\sum_{y\in\Gamma^k} q_y \log q_y - kH(p^*_1, \ldots, p^*_c) \right| = O(kC^{-k})$$

for some constant $C > 1$.

Proof. Please see [11]. □

Following the analysis of §4.2, the expected number of input symbols consumed in the initial state $\{p_y^*\}$ is

$$\begin{aligned}
&\sum_{y\in\Gamma^k}\sum_{x\in S_y} d^{-|x|}\cdot|x| + \sum_{x\in S_r} d^{-|x|}\cdot|x| \\
&< \frac{2d-1}{d-1} + \frac{-\sum_{y\in\Gamma^k} q_y \log q_y - rd^{-k}\log(rd^{-k})}{\log d} \\
&\overset{\text{Lemma 7}}{\leq} \frac{2d-1}{d-1} + \frac{kH(p_1^*,\ldots,p_c^*) + O(kC^{-k})}{\log d} + O(1). \qquad (11)
\end{aligned}$$

At any state, we emit nothing and pass to a residual distribution with probability $rd^{-k} < c^k/d^k$. Since we know nothing about the structure of the residual distribution in relation to the original distribution $\{p_y^*\}$, the bound (11) does not apply for the expected number of input symbols consumed at these other states. However, we have a naive bound of

$$\begin{aligned}
&\sum_{y\in\Gamma^k}\sum_{x\in S_y} d^{-|x|}\cdot|x| + \sum_{x\in S_r} d^{-|x|}\cdot|x| \\
&< \frac{2d-1}{d-1} + \frac{-\sum_{y\in\Gamma^k} q_y \log q_y - rd^{-k}\log(rd^{-k})}{\log d} \\
&\leq \frac{2d-1}{d-1} + \frac{kH(U_c)}{\log d} + O(1) = \Theta(k)
\end{aligned}$$

since the uniform distribution maximizes entropy over all possible distributions $\{q_y\}$, and this is sufficient for our purposes: by the geometric sum formula, the expected number of additional states we will traverse without emitting anything is just $c^k/(d^k - c^k) = \Theta(D^{-k})$, where $D = d/c > 1$ by assumption. Hence, in the initial state, we expect to only consume $\Theta(kD^{-k})$ symbols for some constant D while dealing with residual distributions. We conclude that the total expected number of input symbols consumed to produce k output symbols, hence the latency, is at most

$$\frac{2d-1}{d-1} + \frac{kH(p_1^*,\ldots,p_c^*)}{\log d} + \Theta(kE^{-k})$$

for a constant $E > 1$, so as in §4.2, the efficiency is at least $1 - \Theta(k^{-1})$.

There is still one issue to resolve if we wish to construct a serial protocol with kth component D_k. The observant reader will have noticed that, as D_k is not finite-state, its consumption is not uniformly bounded by some m_k, as required by Lemma 5. However, the computation of one epoch of D_k consists of a series of stages, and the consumption at each stage is uniformly bounded by $m_k = k\log c/\log d + 2$. In each stage, if digits are produced, the epoch halts, otherwise the computation proceeds to the next stage. Each stage, when started in its start state, consumes at most m_k digits and produces exactly k digits with probability at least $1 - (c/d)^k$ and produces no digits with probability at most

$(c/d)^k$. The next lemma shows that this is enough to derive the conclusion of Lemma 5.

Lemma 8. *Let m_k be a uniform bound on the consumption in each stage of one epoch of D_k, as defined in the preceding paragraph. If the m_k satisfy condition* (4)*, then the variances* (5) *vanish in the limit.*

Proof. Please see [11]. □

4.4 Arbitrary ⇒ Uniform with $\Theta(\log k/k)$ Loss

In this section describe a family of restart protocols B_k for transforming an arbitrary d-ary distribution with real probabilities $p_1, \ldots, p_d$ to a c-ary uniform distribution with efficiency $H(p_1, \ldots, p_d)/\log c - \Theta(\log k/k)$. The remainder of this section is omitted due to space constraints; please see [11].

4.5 $(\frac{1}{r}, \frac{r-1}{r}) \Rightarrow (r-1)$-Uniform with $\Theta(k^{-1})$ Loss

Let $r \in \mathbb{N}$, $r \geq 2$. In this section we show that a coin with bias $1/r$ can generate an $(r-1)$-ary uniform distribution with $\Theta(k^{-1})$ loss of efficiency. This improves the result of the previous section in this special case. The remainder of this section is omitted due to space constraints; please see [11].

5 Conclusion

We have introduced a coalgebraic model for constructing and reasoning about state-based protocols that implement entropy-conserving reductions between random processes. We have provided provide basic tools that allow efficient protocols to be constructed in a compositional way and analyzed in terms of the tradeoff between latency and loss of entropy. We have illustrated the use of the model in various reductions.

An intriguing open problem is to improve the loss of the protocol of §4.4 to $\Theta(1/k)$. Partial progress has been made in §4.5, but we were not able to generalize this approach.

5.1 Discussion: The Case for Coalgebra

What are the benefits of a coalgebraic view? Many constructions in the information theory literature are expressed in terms of trees; e.g. [3, 10]. Here we have defined protocols as coalgebras (S, δ), where $\delta : S \times \Sigma \to S \times \Gamma^*$, a form of Mealy automata. These are not trees in general. However, the class admits a final coalgebra $D : (\Gamma^*)^{\Sigma^+} \times \Sigma \to (\Gamma^*)^{\Sigma^+} \times \Gamma^*$, where

$$D(f, a) = (f@a, f(a)) \qquad f@a(x) = f(ax),\ a \in \Sigma,\ x \in \Sigma^+.$$

Here the extension to streams $D^\omega : (\Gamma^*)^{\Sigma^+} \times \Sigma^\omega \rightharpoonup \Gamma^\omega$ takes the simpler form

$$D^\omega(f, a\alpha) = f(a) \cdot D^\omega(f@a, \alpha).$$

A state $f : \Sigma^+ \to \Gamma^*$ can be viewed as a labeled tree with nodes Σ^* and edge labels Γ^*. The nodes xa are the children of x for $x \in \Sigma^*$ and $a \in \Sigma$. The label on the edge (x, xa) is $f(xa)$. The tree $f@x$ is the subtree rooted at $x \in \Sigma^*$, where $f@x(y) = f(xy)$. For any coalgebra (S, δ), there is a unique coalgebra morphism $h : (S, \delta) \to ((\Gamma^*)^{\Sigma^+}, D)$ defined coinductively by

$$(h(s)@a, h(s)(a)) = \text{let } (t, z) = \delta(s, a) \text{ in } (h(t), z),$$

where $s \in S$ and $a \in \Sigma$. The coalgebraic view allows arbitrary protocols to inherit structure from the final coalgebra under h^{-1}, thereby providing a mechanism for transferring results on trees, such as entropy rate, to results on state transition systems.

There are other advantages as well. In this paper we have considered only **homogeneous** measures on Σ^ω and Γ^ω, those induced by Bernoulli processes in which the probabilistic choices are independent and identically distributed, for finite Σ and Γ. However, the coalgebraic definitions of protocol and reduction make sense even if Σ and Γ are countably infinite and even if the measures are non-homogeneous.

We have observed that a fixed measure μ on Σ induces a unique homogeneous measure, also called μ, on Σ^ω. But in the final coalgebra, we can go the other direction: For an arbitrary probability measure μ on Σ^ω and state $f : \Sigma^+ \to \Gamma^*$, there is a unique assignment of transition probabilities on Σ^+ compatible with μ, namely the conditional probability

$$f(xa) = \frac{\mu(\{\alpha \mid xa \prec \alpha\})}{\mu(\{\alpha \mid x \prec \alpha\})},$$

or 0 if the denominator is 0. This determines the probabilistic behavior of the final coalgebra as a protocol starting in state f when the input stream is distributed as μ. This behavior would also be reflected in any protocol (S, δ) starting in any state $s \in h^{-1}(f)$ under the same measure on input streams, thus providing a semantics for (S, δ) even under non-homogeneous conditions.

In addition, as in Lemma 3(iii), any measure μ on Σ^ω induces a push-forward measure $\mu \circ (D^\omega)^{-1}$ on Γ^ω. This gives a notion of reduction even in the non-homogeneous case. Thus we can lift the entire theory to Mealy automata that operate probabilistically relative to an arbitrary measure μ on Σ^ω. These are essentially discrete Markov transition systems with observations in Γ^*.

Even more generally, one can envision a continuous-space setting in which the state set S and alphabets Σ and Γ need not be discrete. The appropriate generalization would give reductions between discrete-time, continuous-space Markov transition systems as defined for example in [6, 15].

As should be apparent, in this paper we have only scratched the surface of this theory, and there is much left to be done.

Acknowledgments

Thanks to Joel Ouaknine and Aaron Wagner for valuable discussions. Thanks to the Bellairs Research Institute of McGill University for providing a wonderful research environment. Research was supported by NSF grants CCF1637532, IIS-1703846, and IIS-1718108, AFOSR grant FA9550-12-1-0040, ARO grant W911NF-17-1-0592, and a grant from the Open Philanthropy project.

References

1. Jiri Adamek. *Foundations of Coding*. Wiley, 1991.
2. Manuel Blum. Independent unbiased coin flips from a correlated biased source: a finite state Markov chain. *Combinatorica*, 6(2):97–108, 1986.
3. Georg Böcherer and Rana Ali Amjad. Informational divergence and entropy rate on rooted trees with probabilities. In *Proc. IEEE Int. Symp. Information Theory*, June 2014.
4. K. L. Chung. *A Course in Probability Theory*. Academic Press, 2nd edition, 1974.
5. Thomas M. Cover and Joy A. Thomas. *Elements of Information Theory*. Wiley-Interscience, August 1991.
6. Ernst-Erich Doberkat. *Stochastic Relations: Foundations for Markov Transition Systems*. Studies in Informatics. Chapman Hall, 2007.
7. Yevgeniy Dodis, Ariel Elbaz, Roberto Oliveira, and Ran Raz. Improved randomness extraction from two independent sources. In K. Jansen et al., editor, *Approx and Random 2004*, volume 3122 of *LNCS*, pages 334–344. Springer, 2004.
8. Peter Elias. The efficient construction of an unbiased random sequence. *Ann. Math. Stat.*, 43(3):865–870, 1992.
9. William Feller. *An Introduction to Probability Theory and Its Applications*, volume 1. Wiley, 2nd edition, 1971.
10. Thomas Hirschler and Wolfgang Woess. Comparing entropy rates on finite and infinite rooted trees with length functions. *IEEE Trans. Information Theory*, December 2017.
11. Dexter Kozen and Matvey Soloviev. Coalgebraic tools for randomness-conserving protocols. Technical report, Cornell University, July 2018.
12. Noam Nisan and Amnon Ta-shma. Extracting randomness: A survey and new constructions. *Journal of Computer and System Sciences*, 58:148–173, 1999.
13. S. Pae and M. C. Loui. Optimal random number generation from a biased coin. In *Proc. 16th ACM-SIAM Symposium on Discrete Algorithms*, pages 1079–1088, Vancouver, Canada, January 2005.
14. S. Pae and M. C. Loui. Randomizing functions: Simulation of discrete probability distribution using a source of unknown distribution. *Trans. Information Theory*, 52(11):4965–4976, 2006.
15. Prakash Panangaden. *Labelled Markov Processes*. Imperial College Press, 2009.
16. Yuval Peres. Iterating von Neumann's procedure for extracting random bits. *Ann. Stat.*, 20(1):590–597, 1992.
17. Yuval Peres, Elchanan Mossel, and Christopher Hillar. New coins from old: Computing with unknown bias. *Combinatorica*, 25(6):707–724, 2005.
18. Aravind Srinivasan and David Zuckerman. Computing with very weak random sources. *SIAM J. Computing*, 28:264–275, 1999.
19. Amnon Ta-shma. On extracting randomness from weak random sources. In *Proc. 28th ACM Symp. Theory of Computing*, pages 276–285, 1996.