Contextual Equivalence for a Probabilistic Language with Continuous Random Variables and Recursion

MITCHELL WAND, Northeastern University
RYAN CULPEPPER, Northeastern University
THEOPHILOS GIANNAKOPOULOS, BAE Systems, Burlington MA
ANDREW COBB, Northeastern University

We present a complete reasoning principle for contextual equivalence in an untyped probabilistic language. The language includes continuous (real-valued) random variables, conditionals, and scoring. It also includes recursion, since the standard call-by-value fixpoint combinator is expressible.

We demonstrate the usability of our characterization by proving several equivalence schemas, including familiar facts from lambda calculus as well as results specific to probabilistic programming. In particular, we use it to prove that reordering the random draws in a probabilistic program preserves contextual equivalence. This allows us to show, for example, that

$$(\texttt{let}\, x = e_1 \,\texttt{in let}\, y = e_2 \,\texttt{in}\, e_0) =_{\text{ctx}} (\texttt{let}\, y = e_2 \,\texttt{in let}\, x = e_1 \,\texttt{in}\, e_0)$$

(provided x does not occur free in e_2 and y does not occur free in e_1) despite the fact that e_1 and e_2 may have sampling and scoring effects.

CCS Concepts: • **Mathematics of computing → Probability and statistics**; • **Theory of computation** → *Operational semantics*;

Additional Key Words and Phrases: probabilistic programming, logical relations, contextual equivalence

ACM Reference format:
Mitchell Wand, Ryan Culpepper, Theophilos Giannakopoulos, and Andrew Cobb. 2017. Contextual Equivalence for a Probabilistic Language with Continuous Random Variables and Recursion. *Proc. ACM Program. Lang.* 1, 1, Article 1 (January 2017), 35 pages.
https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

A *probabilistic programming language* is a programming language enriched with two features—*sampling* and *scoring*—that enable it to represent probabilistic models. We introduce these two features with an example program that models linear regression.

The first feature, *sampling*, introduces probabilistic nondeterminism. It is used to represent random variables. For example, let $\texttt{normal}(m, s)$ be defined to nondeterministically produce a real number distributed according to a normal (Gaussian) distribution with mean m and scale s.

Here is a little model of linear regression that uses `normal` to randomly pick a slope and intercept for a line and then defines `f` as the resulting linear function:

```
A = normal(0, 10)
B = normal(0, 10)
f(x) = A*x + B
```

This material is based upon work sponsored by the Air Force Research Laboratory (AFRL) and the Defense Advanced Research Projects Agency (DARPA) under Contract No. FA8750-14-C-0002. The views expressed are those of the authors and do not reflect the official policy or position of the Department of Defense or the U.S. Government.
2017. 2475-1421/2017/1-ART1 $15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

This program defines a distribution on lines, centered on $y = 0x + 0$, with high variance. This distribution is called the *prior*, since it is specified prior to considering any evidence.

The second feature, *scoring*, adjusts the likelihood of the current execution's random choices. It is used to represent conditioning on observed data.

Suppose we have the following data points: $\{(2.0, 2.4), (3.0, 2.7), (4.0, 3.0)\}$. The smaller the error between the result of `f` and the observed data, the better the choice of `A` and `B`. We express these observations with the following addition to our program:

```
factor normalpdf(f(2.0)-2.4; 0, 1)
factor normalpdf(f(3.0)-2.7; 0, 1)
factor normalpdf(f(4.0)-3.0; 0, 1)
```

Here scoring is performed by the `factor` form, which takes a positive real number to multiply into the current execution's likelihood. We use `normalpdf(_;0,1)`—the density function of the standard normal distribution—to convert the difference between predicted and observed values into the score. This scoring function assigns high likelihood when the error is near 0, dropping off smoothly to low likelihood for larger errors.

After incorporating the observations, the program defines a distribution centered near $y = 0.3x + 1.8$, with low variance. This distribution is often called the *posterior distribution*, since it represents the distribution after the incorporation of evidence.

Computing the posterior distribution—or a workable approximation thereof—is the task of *probabilistic inference*. We say that this program has *inferred* (or sometimes *learned*) the parameters `A` and `B` from the data. Probabilistic inference encompasses an arsenal of techniques of varied applicability and efficiency. Some inference techniques may benefit if the program above is transformed to the following shape:

```
A = normal(0, 10)
factor Z(A)
B = normal(M(A), S(A))
```

The transformation relies on the conjugacy relationship between the normal prior for `B` and the normal scoring function of the observations. A useful equational theory for probabilistic programming must incorporate facts from mathematics in addition to standard concerns such as function inlining.

In this paper we build a foundation for such an equational theory for a probabilistic programming language. In particular, our language supports

- sampling continuous random variables,
- scoring (soft constraints), and
- conditionals, higher-order functions, and recursion.

Other such languages include Church [?], its descendants such as Venture [?] and Anglican [?], and other languages [???] and language models [???]. Our framework is able to justify the transformation above.

In Section 2 we present our model of a probabilistic language, including its syntax and semantics. We then define our logical relation (Section 3), our CIU relation (Section 4), and contextual ordering (Section 5); and we prove that all three relations coincide. In Section 6 we use this new machinery to define contextual equivalence and demonstrate a catalog of useful equivalence schemas, including β_v and `let`-associativity, as well as a method for importing first-order equivalences from mathematics. One unusual equivalence is `let`-commutativity:

$$(\texttt{let}\, x = e_1 \;\texttt{in}\; \texttt{let}\, y = e_2 \;\texttt{in}\; e_0) =_{\text{ctx}} (\texttt{let}\, y = e_2 \;\texttt{in}\; \texttt{let}\, x = e_1 \;\texttt{in}\; e_0)$$

(provided x does not occur free in e_2 and y does not occur free in e_1). This equivalence, while valid for a pure language, is certainly not valid for all effects (consider, for example, if there were an assignment statement in e_1 or e_2). We conclude with two related work sections: Section 7 demonstrates the correspondence between our language model and others, notably that of **?**, and Section 8 informally discusses other related work.

Throughout the paper, we limit proofs mostly to high-level sketches and representative cases. Additional details and cases for some proofs can be found in Appendix A.

Appendix B sketches the steps of the linear regression transformation above using the equivalences we prove in this paper.

2 PROBABILISTIC LANGUAGE MODEL

In this section we define our probabilistic language and its semantics. The semantics consists of three parts:

- A notion of *entropy* for modeling random behavior.
- An *evaluation* function that maps a program and entropy to a real-valued result and an importance weight. We define the evaluation function via an abstract machine. We then define a big-step semantics and prove it equivalent; the big-step formulation simplifies some proofs in Section 6.3 by making the structure of evaluation explicit.
- A mapping to *measures* over the real numbers, calculated by integrating the evaluation function with respect to the entropy space. A program with a finite, non-zero measure can be interpreted as an unnormalized probability distribution.

The structure of the semantics loosely corresponds to one inference technique for probabilistic programming languages: importance sampling. In an importance sampler, the entropy is approximated by a pseudo-random number generator (PRNG); the evaluation function is run many times with different initial PRNG states to produce a collection of weighted samples; and the weighted samples approximate the program's measure—either directly by conversion to a discrete distribution of results, or indirectly via computed statistical properties such as sample mean, variance, etc.

Our language is similar to that of **?**, but with the following differences:

- Our language requires `let`-binding of nontrivial intermediate expressions; this simplifies the semantics. This restriction is similar to but looser than A-normal form [**?**].
- Our model of entropy is a finite measure space made of *splittable* entropy points, rather than an infinite measure space containing *sequences* of real numbers.
- Our `sample` operation models a standard uniform random variable, rather than being parameterized over a distribution.

We revisit these differences in Section 7.

2.1 Syntax

The syntax of our language is given in Figure 1. For simplicity, we require sequencing to be made explicit using `let`. There is a constant $\mathtt{c}_r$ for each real number r, and there are various useful primitive operations.

The `sample` form draws from a uniform distribution on $[0, 1]$. Any other real-valued distribution of interest can be obtained by applying the appropriate *inverse cumulative distribution function*. For example, sampling from a normal distribution can be expressed as follows:

$$\mathtt{normal}(v_m, v_s) \triangleq (\mathtt{let}\, u = \mathtt{sample}\ \mathtt{in}\ \mathtt{normalinvcdf}(u; v_m, v_s))$$

Finally, the `factor` v weights (or "scores") the current execution by the value v.

$$
\begin{array}{lll}
v & ::= x \mid \lambda x.e \mid \mathtt{c}_r & \text{Syntactic Values} \\
e & ::= v \mid (v\ v) \mid \mathtt{let}\ x = e\ \mathtt{in}\ e & \text{Expressions} \\
 & \quad \mid op^n\ (v_1, \ldots, v_n) \mid \mathtt{if}\ v\ \mathtt{then}\ e\ \mathtt{else}\ e & \\
 & \quad \mid \mathtt{sample} \mid \mathtt{factor}\ v & \\
op^1 & ::= \mathtt{log} \mid \mathtt{exp} \mid \mathtt{real?} \mid | \mid \ldots & \text{Unary operations} \\
op^2 & ::= + \mid - \mid \times \mid \div \mid < \mid \leq \mid \ldots & \text{Binary operations} \\
op^3 & ::= \mathtt{normalinvcdf} \mid \mathtt{normalpdf} \mid \ldots & \text{Ternary operations} \\
K & ::= \mathtt{halt} \mid (x \to e)K & \text{Continuations}
\end{array}
$$

Fig. 1. Syntax of values, expressions, and continuations

$$
\frac{x \in \Gamma}{\Gamma \vdash x\ \mathsf{val}} \qquad \frac{\Gamma, x \vdash e\ \mathsf{exp}}{\Gamma \vdash \lambda x.e\ \mathsf{val}} \qquad \Gamma \vdash \mathtt{c}_r\ \mathsf{val}
$$

$$
\frac{\Gamma \vdash v\ \mathsf{val}}{\Gamma \vdash v\ \mathsf{exp}} \qquad \frac{\Gamma \vdash v_1\ \mathsf{val} \qquad \Gamma \vdash v_2\ \mathsf{val}}{\Gamma \vdash (v_1\ v_2)\ \mathsf{exp}} \qquad \frac{\Gamma \vdash e_1\ \mathsf{exp} \qquad \Gamma, x \vdash e_2\ \mathsf{exp}}{\Gamma \vdash \mathtt{let}\ x = e_1\ \mathtt{in}\ e_2\ \mathsf{exp}}
$$

$$
\frac{\Gamma \vdash v_i\ \mathsf{val} \quad (\forall i \in \{1, \ldots, n\})}{\Gamma \vdash op^n\ (v_1, \ldots, v_n)\ \mathsf{exp}} \qquad \frac{\Gamma \vdash v\ \mathsf{val} \qquad \Gamma \vdash e_1\ \mathsf{exp} \qquad \Gamma \vdash e_2\ \mathsf{exp}}{\Gamma \vdash \mathtt{if}\ v\ \mathtt{then}\ e_1\ \mathtt{else}\ e_2\ \mathsf{exp}}
$$

$$
\Gamma \vdash \mathtt{sample}\ \mathsf{exp} \qquad \frac{\Gamma \vdash v\ \mathsf{val}}{\Gamma \vdash \mathtt{factor}\ v\ \mathsf{exp}}
$$

$$
\vdash \mathtt{halt}\ \mathsf{cont} \qquad \frac{\{x\} \vdash e\ \mathsf{exp} \qquad \vdash K\ \mathsf{cont}}{\vdash (x \to e)K\ \mathsf{cont}}
$$

Fig. 2. Scoping rules for values, expressions, and continuations

The language is untyped, but we express the scoping relations by rules like typing rules. We write $\Gamma \vdash e$ exp for the assertion that e is a well-formed expression whose free variables are contained in the set Γ, and similarly for values and continuations. The scoping rules are given in Figure 2.

2.2 Modeling Entropy

The semantics uses an *entropy* component as the source of randomness. We assume an entropy space $\mathbb{S}$ along with its stock measure $\mu_{\mathbb{S}}$. We use σ and τ to range over values in $\mathbb{S}$. When we integrate over σ or τ, we implicitly use the stock measure; that is, we write $\int f(\sigma)\ d\sigma$ to mean $\int f(\sigma)\ \mu_{\mathbb{S}}(d\sigma)$. Following ?, we assume that $\mathbb{S}$ has the following properties:

Property 2.1 (Properties of Entropy).

(1) $\mu_{\mathbb{S}}(\mathbb{S}) = 1$

$$
\begin{array}{lcl}
\langle \sigma \mid \texttt{let}\, x = e_1 \texttt{ in } e_2 \mid K \mid \tau \mid w\rangle & \to & \langle \pi_L(\sigma) \mid e_1 \mid (x \to e_2)K \mid \pi_R(\sigma)::\tau \mid w\rangle \\
\langle \sigma \mid v \mid (x \to e_2)K \mid \sigma'::\tau \mid w\rangle & \to & \langle \sigma' \mid e_2[v/x] \mid K \mid \tau \mid w\rangle \\
\langle \sigma \mid ((\lambda x.e)\ v) \mid K \mid \tau \mid w\rangle & \to & \langle \sigma \mid e[v/x] \mid K \mid \tau \mid w\rangle \\
\langle \sigma \mid \texttt{sample} \mid K \mid \tau \mid w\rangle & \to & \langle \pi_R(\sigma) \mid \mathtt{c}_{\pi_U(\pi_L(\sigma))} \mid K \mid \tau \mid w\rangle \\
\langle \sigma \mid op^n\ (v_1, \ldots, v_n) \mid K \mid \tau \mid w\rangle & \to & \langle \sigma \mid \delta(op^n, v_1, \ldots, v_n) \mid K \mid \tau \mid w\rangle \text{ (if defined)} \\
\langle \sigma \mid \texttt{if } \mathtt{c}_r \texttt{ then } e_1 \texttt{ else } e_2 \mid K \mid \tau \mid w\rangle & \to & \langle \sigma \mid e_1 \mid K \mid \tau \mid w\rangle \text{ (if } r > 0 \text{)} \\
\langle \sigma \mid \texttt{if } \mathtt{c}_r \texttt{ then } e_1 \texttt{ else } e_2 \mid K \mid \tau \mid w\rangle & \to & \langle \sigma \mid e_2 \mid K \mid \tau \mid w\rangle \text{ (if } r \leq 0 \text{)} \\
\langle \sigma \mid \texttt{factor } \mathtt{c}_r \mid K \mid \tau \mid w\rangle & \to & \langle \sigma \mid \mathtt{c}_r \mid K \mid \tau \mid r \times w\rangle \text{ (provided } r > 0)
\end{array}
$$

Fig. 3. Small-step operational semantics

(2) *There is a function* $\pi_U : \mathbb{S} \to [0,1]$ *such that for all measurable* $f : [0,1] \to \mathbb{R}^+$,
$$\int f(\pi_U(\sigma))\ d\sigma = \int_0^1 f(x)\ \lambda(dx)$$
where λ *is the Lebesgue measure. That is,* π_U *represents a standard uniform sampler.*

(3) *There is a surjective pairing function '::' :* $\mathbb{S} \times \mathbb{S} \to \mathbb{S}$, *with projections* π_L *and* π_R, *all measurable.*

(4) *The projections are measure-preserving: for all measurable* $g : \mathbb{S} \times \mathbb{S} \to \mathbb{R}^+$,
$$\int g(\pi_L(\sigma), \pi_R(\sigma))\ d\sigma = \iint g(\sigma_1, \sigma_2)\ d\sigma_1\, d\sigma_2$$

Since $\mathbb{S} \cong \mathbb{S} \times \mathbb{S}$ and thus $\mathbb{S} \cong \mathbb{S}^n$ $(n \geq 1)$, we can also use entropy to encode non-empty *sequences* of entropy values.

We also use Tonelli's Theorem:

Lemma 2.2 (Tonelli). *Let* $f : \mathbb{S} \times \mathbb{S} \to \mathbb{R}^+$ *be measurable. Then*
$$\int \left(\int f(\sigma_1, \sigma_2)\ d\sigma_1\right) d\sigma_2 = \int \left(\int f(\sigma_1, \sigma_2)\ d\sigma_2\right) d\sigma_1$$

2.3 Operational Semantics

2.3.1 Small-Step Semantics. We define evaluation via an abstract machine with a small-step operational semantics. The semantics rewrites configurations $\langle \sigma \mid e \mid K \mid \tau \mid w\rangle$ consisting of:

- an entropy σ (representing the "current" value of the entropy),
- a closed expression e,
- a closed continuation K,
- an entropy τ (encoding a stack of entropies, one for each frame of K), and
- a positive real number w (representing the weight of the current run)

The rules for the semantics are given in Figure 3.

The semantics uses continuations for sequencing and substitutions for procedure calls. Since `let` $x = e_1$ `in` e_2 is the only sequencing construct, there is only one continuation-builder. The first rule recurs into the right-hand side of a `let`, using the left half of the entropy as its entropy, and saving the right half for use with e_2. The second rule ("return") substitutes the value of the expression into the body of the `let` and restores the top saved entropy value for use in the body. More precisely, we view the third component as an encoded pair of an entropy value and an encoded entropy stack, as mentioned in Section 2.2.[1] The return rule can be written using explicit projections as follows:

$$\langle \sigma \mid v \mid (x \to e_2)K \mid \tau \mid w\rangle \quad \to \quad \langle \pi_L(\tau) \mid e_2[v/x] \mid K \mid \pi_R(\tau) \mid w\rangle$$

[1] We defer the explanation of the initial entropy stack to Section 2.4.

Note that in the return rule the current entropy σ is dead. Except for the entropy and weight, these rules are standard for a continuation-passing interpreter for the lambda-calculus with `let`.

The δ partial function interprets primitive operations. We assume that all the primitive operations are measurable partial functions returning real values, and with the exception of `real?`, they are undefined if any of their arguments is a closure. A conditional expression evaluates to its first branch if the condition is a positive real constant, its second branch if nonpositive; if the condition is a closure, evaluation is stuck. Comparison operations and the real? predicate return 1 for truth and 0 for falsity.

The rule for `sample` uses π_U to extract from the entropy a real value in the interval $[0, 1]$. The entropy is split first, to make it clear that entropy is never reused, but the leftover entropy is dead per the return rule. The rule for `factor` v weights the current execution by v, provided v is a positive number; otherwise, evaluation is stuck.

When reduction of an initial configuration halts properly, there are two relevant pieces of information in the final configuration: the result value and the weight. Furthermore, we are only interested in real-valued final results. We define *evaluation* as taking an extra parameter A, a measurable set of reals. Evaluation produces a positive weight only if the result value is in the expected set.

$$\text{eval}(\sigma, e, K, \tau, w, A) = \begin{cases} w' & \text{if } \langle \sigma \mid e \mid K \mid \tau \mid w \rangle \to^* \langle \sigma' \mid r \mid \texttt{halt} \mid \tau' \mid w' \rangle, \\ & \text{where } r \in A \\ 0 & \text{otherwise} \end{cases}$$

We will also need approximants to eval:

$$\text{eval}^{(n)}(\sigma, e, K, \tau, w, A) = \begin{cases} w' & \text{if } \langle \sigma \mid e \mid K \mid \tau \mid w \rangle \to^* \langle \sigma' \mid r \mid \texttt{halt} \mid \tau' \mid w' \rangle \\ & \text{in } n \text{ or fewer steps, where } r \in A \\ 0 & \text{otherwise} \end{cases}$$

The following lemmas are clear from inspection of the small-step semantics.

Lemma 2.3. *If* $\langle \sigma \mid e \mid K \mid \tau \mid w \rangle \to \langle \sigma' \mid e' \mid K' \mid \tau' \mid w' \rangle$ *then*

(1) $\text{eval}^{(p+1)}(\sigma, e, K, \tau, w, A) = \text{eval}^{(p)}(\sigma', e', K', \tau', w', A)$
(2) $\text{eval}(\sigma, e, K, \tau, w, A) = \text{eval}(\sigma', e', K', \tau', w', A)$

Lemma 2.4 (weights are Linear).

(1) *Weights can be factored out of reduction sequences. That is,*

$$\langle \sigma \mid e \mid K \mid \tau \mid 1 \rangle \to^* \langle \sigma' \mid e' \mid K' \mid \tau' \mid w' \rangle,$$

if and only if for any $w > 0$

$$\langle \sigma \mid e \mid K \mid \tau \mid w \rangle \to^* \langle \sigma' \mid e' \mid K' \mid \tau' \mid w' \times w \rangle$$

(2) *Weights can be factored out of evaluation. That is, for all* $w > 0$,

$$\text{eval}(\sigma, e, K, \tau, w, A) = w \times \text{eval}(\sigma, e, K, \tau, 1, A),$$

and similarly for $\text{eval}^{(n)}$.

$$\sigma \vdash \lambda x.e \Downarrow \lambda x.e, 1 \qquad\qquad \sigma \vdash \mathtt{c}_r \Downarrow \mathtt{c}_r, 1$$

$$\frac{\sigma \vdash e[v/x] \Downarrow v', w}{\sigma \vdash ((\lambda x.e)\ v) \Downarrow v', w} \qquad \frac{\pi_L(\sigma) \vdash e_1 \Downarrow v_1, w_1 \qquad \pi_R(\sigma) \vdash e_2[v_1/x] \Downarrow v_2, w_2}{\sigma \vdash \mathtt{let}\, x = e_1\ \mathtt{in}\ e_2 \Downarrow v_2, w_2 \times w_1}$$

$$\frac{\delta(op^n, v_1, \ldots, v_n) = v}{\sigma \vdash op^n\ (v_1, \ldots, v_n) \Downarrow v, 1}$$

$$\frac{\sigma \vdash e_1 \Downarrow v, w \qquad r > 0}{\sigma \vdash \mathtt{if}\ \mathtt{c}_r\ \mathtt{then}\ e_1\ \mathtt{else}\ e_2 \Downarrow v, w} \qquad \frac{\sigma \vdash e_2 \Downarrow v, w \qquad r \leq 0}{\sigma \vdash \mathtt{if}\ \mathtt{c}_r\ \mathtt{then}\ e_1\ \mathtt{else}\ e_2 \Downarrow v, w}$$

$$\sigma \vdash \mathtt{sample} \Downarrow \mathtt{c}_{\pi_U(\pi_L(\sigma))}, 1 \qquad\qquad \frac{r > 0}{\sigma \vdash \mathtt{factor}\ \mathtt{c}_r \Downarrow \mathtt{c}_r, r}$$

Fig. 4. Big-step operational semantics

2.3.2 Big-Step Semantics. We regard the small-step semantics as normative, and we use it for our primary soundness and completeness results. However, for program transformations it is useful to have a big-step semantics as well. In this section, we define a big-step semantics and characterize its relation to the small-step semantics.

The big-step semantics is given in Figure 4. It has judgments of the form $\sigma \vdash e \Downarrow v, w$, where σ is a value of the entropy, e is a closed expression, v is a closed value, and w is a weight (a positive real number). Its intention is that when e is supplied with entropy σ, it returns v with weight w, consuming some portion (possibly empty) of the given entropy σ. The rules are those of a straightforward call-by-value λ-calculus, modified to keep track of the entropy and weight.

The translation from big-step to small-step semantics is straightforward:

Theorem 2.5 (Big-Step to Small-Step). *If $\sigma \vdash e \Downarrow v, w$, then for any K and τ, there exists a σ' such that*

$$\langle \sigma \mid e \mid K \mid \tau \mid 1 \rangle \rightarrow^* \langle \sigma' \mid v \mid K \mid \tau \mid w \rangle$$

Proof. By induction on the definition of $\Downarrow$. We will show selected cases.

Case $\sigma \vdash \lambda x.e \Downarrow \lambda x.e, 1$ **:** The required small-step reduction is empty. Similarly for $\mathtt{c}_r$.

Case $\sigma \vdash \mathtt{sample} \Downarrow \mathtt{c}_{\pi_U(\pi_L(\sigma))}, 1$ **:** The required reduction is the single step reduction

$$\langle \sigma \mid \mathtt{sample} \mid K \mid \tau \mid 1 \rangle \rightarrow \langle \pi_R(\sigma) \mid \mathtt{c}_{\pi_U(\pi_L(\sigma))} \mid K \mid \tau \mid 1 \rangle$$

Similarly for $\mathtt{factor}\ \mathtt{c}_r$ and the op^n rules.

Case $((\lambda x.e)\ v)$ **:** The rule is

$$\frac{\sigma \vdash e[v/x] \Downarrow v', w}{\sigma \vdash ((\lambda x.e)\ v) \Downarrow v', w}$$

By inversion, we have $\sigma \vdash e[v/x] \Downarrow v', w$. So the reduction sequence is:

$$\begin{array}{ll} \langle \sigma \mid ((\lambda x.e)\ v) \mid K \mid \tau \mid 1 \rangle & \\ \rightarrow\ \langle \sigma \mid e[v/x] \mid K \mid \tau \mid 1 \rangle & \\ \rightarrow^*\ \langle \sigma' \mid v' \mid K \mid \tau \mid w \rangle & \text{by the induction hypothesis} \end{array}$$

Similarly for the `if` rules.

Case $\texttt{let}\,x = e_1\ \texttt{in}\ e_2$ **:** The rule is

$$\frac{\pi_L(\sigma) \vdash e_1 \Downarrow v_1, w_1 \qquad \pi_R(\sigma) \vdash e_2[v_1/x] \Downarrow v_2, w_2}{\sigma \vdash \texttt{let}\,x = e_1\ \texttt{in}\ e_2 \Downarrow v_2, w_2 \times w_1}$$

By inversion, we have $\pi_L(\sigma) \vdash e_1 \Downarrow v_1, w_1$ and $\pi_R(\sigma) \vdash e_2[v_1/x] \Downarrow v_2, w_2$. So the required reduction sequence is:

$$\begin{array}{l} \langle \sigma \mid \texttt{let}\,x = e_1\ \texttt{in}\ e_2 \mid K \mid \tau \mid 1\rangle \\ \to\ \langle \pi_L(\sigma) \mid e_1 \mid (x \to e_2)K \mid \pi_R(\sigma)::\tau \mid w\rangle \\ \to^*\ \langle \sigma' \mid v_1 \mid (x \to e_2)K \mid \pi_R(\sigma)::\tau \mid w_1\rangle \\ \to\ \langle \pi_R(\sigma) \mid e_2[v_1/x] \mid K \mid \tau \mid w_1\rangle \\ \to^*\ \langle \sigma'' \mid v_2 \mid K \mid \tau \mid w_2 \times w_1\rangle \end{array}$$

where the third line follows from the induction hypothesis, and the last line follows from the other induction hypothesis and the linearity of weights (Lemma 2.4). □

Note that the weak quantifier ("there exists a σ'") corresponds to the fact that the entropy is dead in the return rule.

In order to prove a converse, we need some additional results about the small-step semantics.

Definition 2.6. Define $\geq$ to be the smallest relation defined by the following rules:

Rule 1:
$$(K, \tau) \geq (K, \tau)$$

Rule 2:
$$\frac{(K', \tau') \geq (K, \tau)}{((x \to e)K', \sigma::\tau') \geq (K, \tau)}$$

Lemma 2.7. *Let*

$$\langle \sigma_1 \mid e_1 \mid K_1 \mid \tau_1 \mid w_1\rangle \to \langle \sigma_2 \mid e_2 \mid K_2 \mid \tau_2 \mid w_2\rangle \to \dots$$

be a reduction sequence in the operational semantics. Then for each i in the sequence either

a. *there exists a smallest $j \leq i$ such that e_j is a value and $K_j = K_1$ and $\tau_j = \tau_1$, or*
b. $(K_i, \tau_i) \geq (K_1, \tau_1)$

Proof. See appendix. □

The next result is an interpolation theorem, which imposes structure on reduction sequences: any terminating computation starting with an expression e begins by evaluating e to a value v and then sending that value to the continuation K.

Theorem 2.8 (Interpolation Theorem). *If*

$$\langle \sigma \mid e \mid K \mid \tau \mid w\rangle \to^* \langle \sigma'' \mid v'' \mid \texttt{halt} \mid \tau'' \mid w''\rangle$$

then there exists a smallest n such that for some quantities σ', v, and w',

$$\langle \sigma \mid e \mid K \mid \tau \mid w\rangle \to^n \langle \sigma' \mid v \mid K \mid \tau \mid w' \times w\rangle \to^* \langle \sigma'' \mid v'' \mid \texttt{halt} \mid \tau'' \mid w''\rangle$$

Proof. If $K = \texttt{halt}$, then the result is trivial. Otherwise, apply the invariant of the preceding lemma, observing that $(\texttt{halt}, \tau') \not\geq (K, \tau)$ and that weights are multiplicative. □

Note that both Lemma 2.7 and Theorem 2.8 would be false if our language contained jumping control structures like `call/cc`.

Finally, we show that in the interpolation theorem, σ', v, and w' are independent of K.

Theorem 2.9 (Genericity Theorem). *Let* $w_1 > 0$ *and let* n *be the smallest integer such that for some quantities* σ', v, *and* w',

$$\langle \sigma \mid e \mid K_1 \mid \tau_1 \mid w_1 \rangle \to^n \langle \sigma' \mid v \mid K_1 \mid \tau_1 \mid w' \times w_1 \rangle$$

then for any K_2, τ_2, *and* w_2,

$$\langle \sigma \mid e \mid K_2 \mid \tau_2 \mid w_2 \rangle \to^n \langle \sigma' \mid v \mid K_2 \mid \tau_2 \mid w' \times w_2 \rangle$$

Proof. Let R be the smallest relation defined by the rules

$$((K_1, \tau_1), (K_2, \tau_2)) \in R \qquad \frac{((K, \tau), (K', \tau')) \in R}{(((x \to e)K, \sigma{::}\tau), ((x \to e)K', \sigma{::}\tau')) \in R}$$

Extend R to be a relation on configurations by requiring the weights to be related by a factor of w_2/w_1 and the remaining components of the configurations to be equal. It is easy to see, by inspection of the small-step rules, that R is a bisimulation over the first n steps of the given reduction sequence. □

We are now ready to state the converse of Theorem 2.5.

Definition 2.10. We say that a configuration $\langle \sigma \mid e \mid K \mid \tau \mid w \rangle$ *halts* iff

$$\langle \sigma \mid e \mid K \mid \tau \mid w \rangle \to^* \langle \sigma' \mid v \mid \mathtt{halt} \mid \tau' \mid w' \rangle$$

for some σ', v, τ' and w'.

Theorem 2.11 (Small-Step to Big-Step). *If*

$$\langle \sigma \mid e \mid K \mid \tau \mid w \rangle \to^* \langle \sigma'' \mid v'' \mid \mathtt{halt} \mid \tau'' \mid w'' \rangle,$$

then there exist σ', v' *and* w' *such that*

$$\sigma \vdash e \Downarrow v', w'$$

and

$$\langle \sigma'' \mid v' \mid K \mid \tau \mid w' \times w \rangle \to^* \langle \sigma' \mid v' \mid \mathtt{halt} \mid \tau'' \mid w'' \rangle$$

Proof. Given

$$\langle \sigma \mid e \mid K \mid \tau \mid w \rangle \to^* \langle \sigma'' \mid v'' \mid \mathtt{halt} \mid \tau'' \mid w'' \rangle$$

apply the Interpolation Theorem (Theorem 2.8) to get n, σ', v, and w' such that

$$\langle \sigma \mid e \mid K \mid \tau \mid w \rangle \to^n \langle \sigma' \mid v \mid K \mid \tau \mid w' \times w \rangle \to^* \langle \sigma'' \mid v'' \mid \mathtt{halt} \mid \tau'' \mid w'' \rangle$$

This gives us the second part of the conclusion. To get the first part, we proceed by (course-of-values) induction on n, and then by cases on e.

Case $\lambda x.e$ **:** For configurations of the form $\langle \sigma \mid \lambda x.e \mid K \mid \tau \mid w \rangle$, the expression is already a value, so n is 0. So set $v = \lambda x.e$ and $w' = 1$, and observe that $\sigma \vdash \lambda x.e \Downarrow \lambda x.e, 1$, as desired. The case of constants c_r is similar.

Case `sample` **:** We know

$$\langle \sigma \mid \mathtt{sample} \mid K \mid \tau \mid w \rangle \to \langle \pi_R(\sigma) \mid \mathsf{c}_{\pi_U(\pi_L(\sigma))} \mid K \mid \tau \mid w \rangle$$

so the value length is 1, and we also have $\sigma \vdash \mathtt{sample} \Downarrow \mathsf{c}_{\pi_U(\pi_L(\sigma))}, 1$, as desired. The cases of `factor` and of op^n are similar.

Case $((\lambda x.e)\ v)$ **:** Assume that the value length of $\langle \sigma \mid ((\lambda x.e)\ v) \mid K \mid \tau \mid w \rangle$ is $n+1$. So we have

$$\langle \sigma \mid ((\lambda x.e)\ v) \mid K \mid \tau \mid w \rangle \to \langle \sigma \mid e[v/x] \mid K \mid \tau \mid w \rangle \to^n \langle \sigma' \mid v' \mid K \mid \tau \mid w' \times w \rangle$$

By induction, we have $\sigma \vdash e[v/x] \Downarrow v', w'$. Hence, by the big-step rule for λ-expressions, we have $\sigma \vdash ((\lambda x.e)\ v) \Downarrow v', w'$, as desired. The cases for conditionals are similar.

Case $\texttt{let}\ x = e_1\ \texttt{in}\ e_2$ **:** Assume the value length of $\langle \sigma \mid \texttt{let}\ x = e_1\ \texttt{in}\ e_2 \mid K \mid \tau \mid w \rangle$ is n. Then the first n steps of its reduction sequence must be

$$\begin{array}{l} \langle \sigma \mid \texttt{let}\ x = e_1\ \texttt{in}\ e_2 \mid K \mid \tau \mid w \rangle \\ \to\ \ \langle \pi_L(\sigma) \mid e_1 \mid (x \to e_2)K \mid \pi_R(\sigma)::\tau \mid w \rangle \\ \to^m \langle \sigma' \mid v_1 \mid (x \to e_2)K \mid \pi_R(\sigma)::\tau \mid w_1 \times w \rangle \\ \to\ \ \langle \pi_R(\sigma)::\tau \mid e_2[v_1/x] \mid K \mid \tau \mid w_1 \times w \rangle \\ \to^p \langle \sigma'' \mid v \mid K \mid \tau \mid w_2 \times w_1 \times w \rangle \end{array}$$

where m and p are the value lengths of the configurations on the second and fourth lines, respectively. So $n = m + p + 2$, and we can apply the induction hypothesis to the two relevant configurations. Applying the induction hypothesis twice, we get

$$\pi_L(\sigma) \vdash e_1 \Downarrow v_1, w_1 \qquad \text{and} \qquad \pi_R(\sigma) \vdash e_2[v_1/x] \Downarrow v_2, w_2\ .$$

Hence, by the big-step rule for `let`, we conclude that

$$\sigma \vdash \texttt{let}\ x = e_1\ \texttt{in}\ e_2 \Downarrow v, w_2 \times w_1$$

as desired. □

2.4 From Evaluations to Measures

Up to now, we have considered only single runs of the machine, using particular entropy values. To obtain the overall meaning of the program we need to integrate over all possible values of the entropies σ and τ:

Definition 2.12. The *measure* of e and K is the measure on the reals defined by

$$\mu(e, K, A) = \iint \text{eval}(\sigma, e, K, \tau, 1, A)\ d\sigma\ d\tau$$

for each measurable set A of the reals.

This measure is similar to both Culpepper and Cobb's $\mu_e(A)$ and Borgström et al.'s $[\![e]\!]_S(A)$, but whereas they define measures on arbitrary syntactic values, our $\mu(e, K, -)$ is a measure on the reals. Furthermore, whereas their measures represent the meanings of intermediate expressions, our measure—due to the inclusion of the continuation argument K—represents the meanings of whole programs.

The simplicity of the definition above relies on the mathematical trick of encoding entropy stacks as entropy values; if we represented stacks directly the number of integrals would depend on the stack depth. Note that even for the base continuation ($K = \texttt{halt}$) we still integrate with respect to both σ and τ. Since $\mathbb{S} \not\cong \mathbb{S}^0$, there is no encoding for an empty stack as an entropy value; we cannot just choose a single arbitrary τ_{init} because $\mu_{\mathbb{S}}(\{\tau_{\text{init}}\}) = 0$. But since evaluation respects the stack discipline, it produces the correct result for any initial τ_{init}. So we integrate over *all* choices of τ_{init}, and since $\mu_{\mathbb{S}}(\mathbb{S}) = 1$ the empty stack "drops out" of the integral.

As before, we will also need the approximants:

$$\mu^{(n)}(e, K, A) = \iint \text{eval}^{(n)}(\sigma, e, K, \tau, 1, A)\ d\sigma\ d\tau$$

For these integrals to be well-defined, of course, we need to know that eval and its approximants are measurable.

Lemma 2.13 (eval is measurable). *For any e, K, $w \geq 0$, $A \in \Sigma_{\mathbb{R}}$, and n,* $\text{eval}(\sigma, e, K, \tau, w, A)$ *and* $\text{eval}^{(n)}(\sigma, e, K, \tau, w, A)$ *are measurable in σ and τ.*

Proof. See appendix. □

The next lemma establishes some properties of μ and the approximants $\mu^{(n)}$. In particular, it shows that μ is the limit of the approximants.

Lemma 2.14 (measures are monotonic). *In the following, e and K range over closed expressions and continuations, and let A range over measurable sets of reals.*

(1) $\mu(e, K, A) \geq 0$
(2) *for any m,* $\mu^{(m)}(e, K, A) \geq 0$
(3) *if $m \leq n$, then* $\mu^{(m)}(e, K, A) \leq \mu^{(n)}(e, K, A) \leq \mu(e, K, A)$
(4) $\mu(e, K, A) = \sup_n\{\mu^{(n)}(e, K, A)\}$

Finally, the next lemma's equations characterize how the approximant and limit measures, $\mu^{(n)}$ and μ, behave under the reductions of the small-step machine. Almost all the calculations in Section 3 depend only on these equations.

Lemma 2.15. *The following equations hold for approximant measures:*

$$\mu^{(p+1)}(\texttt{let}\, x = e_1 \texttt{ in } e_2, K, A) = \mu^{(p)}(e_1, (x \to e_2)K, A)$$
$$\mu^{(p+1)}(v, (x \to e)K, A) = \mu^{(p)}(e[v/x], K, A)$$
$$\mu^{(p+1)}((\lambda x.e\ v), K, A) = \mu^{(p)}(e[v/x], K, A)$$
$$\mu^{(p+1)}(op^n\ (v_1, \ldots, v_n), K, A) = \mu^{(p)}(\delta(op^n, v_1, \ldots, v_n), K, A) \quad \text{if defined}$$
$$\mu^{(p+1)}(\texttt{if c}_r \texttt{ then } e_1 \texttt{ else } e_2, K, A) = \mu^{(p)}(e_1, K, A) \quad \text{if } r > 0$$
$$\mu^{(p+1)}(\texttt{if c}_r \texttt{ then } e_1 \texttt{ else } e_2, K, A) = \mu^{(p)}(e_2, K, A) \quad \text{if } r \leq 0$$
$$\mu^{(p+1)}(\texttt{sample}, K, A) = \int_0^1 \mu^{(p)}(\texttt{c}_r, K, A)\, dr$$
$$\mu^{(p+1)}(\texttt{factor c}_r, K, A) = r \times \mu^{(p)}(\texttt{c}_r, K, A) \quad \text{if } r > 0$$

In addition, the analogous index-free equations hold for the unapproximated (limit) measure $\mu(-, -, -)$.

In general, the proofs of the equations of Lemma 2.15 involve unfolding the definition of the measure and applying Lemma 2.3 under the integral. The proof for `let` is representative:

Proof for `let`.

$$\begin{aligned}
&\mu^{(p+1)}(\texttt{let}\, x = e_1 \texttt{ in } e_2, K, A) \\
&= \iint \text{eval}^{(p+1)}(\sigma, \texttt{let}\, x = e_1 \texttt{ in } e_2, K, \tau, 1, A)\, d\sigma\, d\tau \\
&= \iint \text{eval}^{(p)}(\pi_L(\sigma), e_1, (x \to e_2)K, \pi_R(\sigma)::\tau, 1, A)\, d\sigma\, d\tau && \text{(Lemma 2.3)} \\
&= \iiint \text{eval}^{(p)}(\sigma', e_1, (x \to e_2)K, \sigma''::\tau, 1, A)\, d\sigma'\, d\sigma''\, d\tau && \text{(Property 2.1.4 on } \sigma) \\
&= \iint \text{eval}^{(p)}(\sigma', e_1, (x \to e_2)K, \pi_L(\tau')::\pi_R(\tau'), 1, A)\, d\sigma'\, d\tau' && \text{(Property 2.1.4 on } \tau') \\
&= \iint \text{eval}^{(p)}(\sigma', e_1, (x \to e_2)K, \tau', 1, A)\, d\sigma'\, d\tau' && (\pi_L(\tau')::\pi_R(\tau') = \tau') \\
&= \mu^{(p)}(e_1, (x \to e_2)K, A)
\end{aligned}$$

□

The proof for `factor` additionally uses linearity (Lemma 2.4), and the proof for `sample` additionally uses Property 2.1.2.

So far, our semantics speaks directly only about the meanings of whole programs. In the following sections, we develop a collection of relations for expressions and ultimately show that they respect the contextual ordering relation on expression induced by the semantics of whole programs.

3 THE LOGICAL RELATION

In this section, we will define a step-indexed logical relation on values, expressions, and continuations, and we prove the Fundamental Property for our relation.

We begin by defining step-indexed logical relations on *closed* values, *closed* expressions, and continuations (which are always closed) as follows:

$$
\begin{aligned}
(v_1, v_2) \in \mathbb{V}_n &\iff v_1 = v_2 = \mathsf{c}_r \;\text{ for some } r \\
&\quad\;\; \vee (v_1 = \lambda x.e \wedge v_2 = \lambda x.e' \\
&\qquad \wedge (\forall m < n)(\forall v, v')[(v, v') \in \mathbb{V}_m \implies (e[v/x], e'[v'/x]) \in \mathbb{E}_m]) \\
(e, e') \in \mathbb{E}_n &\iff (\forall m \leq n)(\forall K, K')(\forall A \in \Sigma_{\mathbb{R}}) \\
&\qquad [(K, K') \in \mathbb{K}_m \implies \mu^{(m)}(e, K, A) \leq \mu(e', K', A)] \\
(K, K') \in \mathbb{K}_n &\iff (\forall m \leq n)(\forall v, v')(\forall A \in \Sigma_{\mathbb{R}}) \\
&\qquad [(v, v') \in \mathbb{V}_m \implies \mu^{(m)}(v, K, A) \leq \mu(v', K', A)]
\end{aligned}
$$

The definitions are well-founded because $\mathbb{V}_\_$ refers to $\mathbb{E}_\_$ at strictly smaller indexes. Note that for all n, $\mathbb{V}_n \supseteq \mathbb{V}_{n+1} \supseteq \ldots$, and similarly for $\mathbb{E}$ and $\mathbb{K}$. That is, at higher indexes the relations make more distinctions and thus relate fewer things.

We use γ to range over substitutions of closed values for variables, and we define $\mathbb{G}_n$ by lifting $\mathbb{V}_n$ to substitutions as follows:

$$
\begin{aligned}
(\gamma, \gamma') \in \mathbb{G}_n^{\Gamma} &\iff \mathrm{dom}(\gamma) = \mathrm{dom}(\gamma') = \Gamma \\
&\qquad \wedge \forall x \in \Gamma, (\gamma(x), \gamma'(x)) \in \mathbb{V}_n
\end{aligned}
$$

Last, we define the logical relations on open terms. In each case, the relation is on terms of the specified sort that are well-formed with free variables in Γ:

$$
\begin{aligned}
(v, v') \in \mathbb{V}^{\Gamma} &\iff (\forall n)(\forall \gamma, \gamma')[(\gamma, \gamma') \in \mathbb{G}_n^{\Gamma} \implies (v\gamma, v'\gamma') \in \mathbb{V}_n] \\
(e, e') \in \mathbb{E}^{\Gamma} &\iff (\forall n)(\forall \gamma, \gamma')[(\gamma, \gamma') \in \mathbb{G}_n^{\Gamma} \implies (e\gamma, e'\gamma') \in \mathbb{E}_n] \\
(K, K') \in \mathbb{K} &\iff (\forall n)(K, K') \in \mathbb{K}_n
\end{aligned}
$$

The limit relation $\mathbb{K}$ is not indexed by Γ because we work only with closed continuations.

Our first goal is to show the so-called fundamental property of logical relations:

$$\Gamma \vdash e \;\mathsf{exp} \implies (e, e) \in \mathbb{E}^{\Gamma}$$

We begin with a series of compatibility lemmas. These show that the logical relations form a congruence under ("are compatible with") the scoping rules of values, expressions, and continuations. Note the correspondence between the scoping rules of Figure 2 and the compatibility rules of Figure 5.

Lemma 3.1 (Compatibility). *The implications summarized as inference rules in Figure 5 hold.*

Most of the lemmas follow by general nonsense about the lambda-calculus, the definitions of the logical relations, and calculations involving $\mu^{(n)}$ and μ using Lemma 2.15. The proof for application is representative:

$$\frac{x \in \Gamma}{(x, x) \in \mathbb{V}^\Gamma} \qquad \frac{(e, e') \in \mathbb{E}^{\Gamma, x}}{(\lambda x.e, \lambda x.e') \in \mathbb{V}^\Gamma} \qquad (\mathsf{c}_r, \mathsf{c}_r) \in \mathbb{V}^\Gamma \qquad \frac{(v, v') \in \mathbb{V}^\Gamma}{(v, v') \in \mathbb{E}^\Gamma}$$

$$\frac{(v_1, v_1') \in \mathbb{V}^\Gamma \qquad (v_2, v_2') \in \mathbb{V}^\Gamma}{((v_1\ v_2), (v_1'\ v_2')) \in \mathbb{E}^\Gamma} \qquad \frac{(e_1, e_1') \in \mathbb{E}^\Gamma \qquad (e_2, e_2) \in \mathbb{E}^{\Gamma, x}}{(\mathtt{let}\ x = e_1\ \mathtt{in}\ e_2, \mathtt{let}\ x = e_1'\ \mathtt{in}\ e_2') \in \mathbb{E}^\Gamma}$$

$$\frac{(v_i, v_i') \in \mathbb{V}^\Gamma \quad (i \in \{1, \dots, k\})}{(op^k\ (v_1, \dots, v_k), op^k\ (v_1', \dots, v_k')) \in \mathbb{E}^\Gamma}$$

$$\frac{(v, v') \in \mathbb{V}^\Gamma \qquad (e_1, e_1') \in \mathbb{E}^\Gamma \qquad (e_2, e_2) \in \mathbb{E}^\Gamma}{(\mathtt{if}\ v\ \mathtt{then}\ e_1\ \mathtt{else}\ e_2, \mathtt{if}\ v'\ \mathtt{then}\ e_1'\ \mathtt{else}\ e_2') \in \mathbb{E}^\Gamma}$$

$$(\mathtt{sample}, \mathtt{sample}) \in \mathbb{E} \qquad \frac{(v, v') \in \mathbb{V}^\Gamma}{(\mathtt{factor}\ v, \mathtt{factor}\ v') \in \mathbb{E}^\Gamma}$$

$$(\mathtt{halt}, \mathtt{halt}) \in \mathbb{K} \qquad \frac{(e_1, e_2) \in \mathbb{E}^{\{x\}} \qquad (K, K') \in \mathbb{K}}{((x \to e)K, (x \to e')K') \in \mathbb{K}}$$

Fig. 5. Compatibility rules for the logical relation

Proof for App. We must show that if $(v_1, v_1') \in \mathbb{V}^\Gamma$ and $(v_2, v_2') \in \mathbb{V}^\Gamma$, then $((v_1\ v_2), (v_1'\ v_2')) \in \mathbb{E}^\Gamma$.

Choose n, and assume $(\gamma, \gamma') \in \mathbb{G}_n^\Gamma$. Then $(v_1\gamma, v_1'\gamma') \in \mathbb{V}_n$ and $(v_2\gamma, v_2'\gamma') \in \mathbb{V}_n$. We must show $((v_1\gamma\ \ v_2\gamma), (v_1'\gamma'\ v_2'\gamma')) \in \mathbb{E}_n$.

If $v_1\gamma$ is of the form c_r, then $\mu^{(m)}(v_1\gamma, K, A) = 0$ for any m, K, and A, so the conclusion holds by Lemma 2.14.

Otherwise, assume $v_1\gamma$ is of the form $\lambda x.e$, and so $v_1'\gamma'$ is of the form $\lambda x.e'$. So choose $m \le n$ and A, and let $(K, K') \in \mathbb{K}_m$. We must show that

$$\mu^{(m)}((\lambda x.e\gamma\ v_2\gamma), K, A) \le \mu((\lambda x.e'\gamma'\ v_2'\gamma'), K', A).$$

If $m = 0$ the left-hand side is 0 and the inequality holds trivially. So consider $m \ge 1$. Since all the relevant terms are closed and the relations on closed terms are antimonotonic in the index, we have $(\lambda x.e\gamma, \lambda x.e'\gamma') \in \mathbb{V}_m$ and $(v_1\gamma, v_1'\gamma') \in \mathbb{V}_{m-1}$. Therefore $(e\gamma[v_2\gamma/x], e'\gamma'[v_2'\gamma'/x]) \in \mathbb{E}_{m-1}$.

Now, $\langle \sigma \mid (\lambda x.e\gamma\ \ v_2\gamma) \mid K \mid \tau \mid w \rangle \to \langle \sigma \mid e\gamma[v_2\gamma/x] \mid K \mid \tau \mid w \rangle$, and similarly for the primed side. So we have

$$\begin{aligned}
\mu^{(m)}((\lambda x.e\gamma\ \ v_2\gamma), K, A) &= \mu^{(m-1)}(e\gamma[v_2\gamma/x], K, A) && \text{(Lemma 2.15)} \\
&\le \mu(e'\gamma'[v_2'\gamma'/x], K', A) && \text{(by } (e\gamma[v_2\gamma/x], e'\gamma'[v_2'\gamma'/x]) \in \mathbb{E}_{m-1}) \\
&= \mu((\lambda x.e'\gamma'\ \ v_2'\gamma'), K', A)
\end{aligned}$$

□

More detailed proofs can be found in Appendix A.

Now we can prove the Fundamental Property:

Theorem 3.2 (Fundamental Property).

(1) $\Gamma \vdash e \text{ exp} \implies (e, e) \in \mathbb{E}^\Gamma$
(2) $\Gamma \vdash v \text{ val} \implies (v, v) \in \mathbb{V}^\Gamma$
(3) $\vdash K \text{ cont} \implies \forall n, (K, K) \in \mathbb{K}_n$

Proof. By induction on the derivation of $\Gamma \vdash e$ exp, etc, applying the corresponding compatibility rule from Lemma 3.1 at each point. □

The essential properties of the logical relation we wish to hold are soundness and completeness with respect to the contextual ordering. We address these properties in Section 5 after taking a detour to define another useful intermediate relation, $\mathbb{CIU}^\Gamma$, and establish its equivalence to $\mathbb{E}^\Gamma$.

4 CIU ORDERING

The CIU ("closed instantiation of uses") ordering of two terms asserts that they yield related observable behavior under a single substitution and a single continuation. We take "observable behavior" to be a program's measure over the reals, as we did for the logical relations.

Definition 4.1.

(1) If e and e' are closed expressions, then $(e, e') \in \mathbb{CIU}$ iff for all closed K and measurable A, $\mu(e, K, A) \leq \mu(e', K, A)$.
(2) If $\Gamma \vdash e$ exp and $\Gamma \vdash e'$ exp, then $(e, e') \in \mathbb{CIU}^\Gamma$ iff for all closing substitutions γ, $(e\gamma, e'\gamma) \in \mathbb{CIU}$.

Since it requires considering only a single substitution and a single continuation rather than related pairs, it is often easier to prove particular expressions related by $\mathbb{CIU}^\Gamma$. But in fact, this relation coincides with the logical relation, as we demonstrate now. One direction is an easy consequence of the Fundamental Property.

Lemma 4.2 ($\mathbb{E} \subseteq \mathbb{CIU}$). *If $(e, e') \in \mathbb{E}^\Gamma$ then $(e, e') \in \mathbb{CIU}^\Gamma$.*

Proof. Choose a closing substitution γ, a closed continuation K, and $A \in \Sigma_\mathbb{R}$. By the Fundamental Property, we have for all n, $(\gamma, \gamma) \in \mathbb{G}_n^\Gamma$ and $(K, K) \in \mathbb{K}_n$. Therefore, for all n, $\mu^{(n)}(e\gamma, K, A) \leq \mu(e'\gamma, K, A)$. So

$$\mu(e\gamma, K, A) = \sup_n \{\mu^{(n)}(e\gamma, K, A)\} \leq \mu(e'\gamma, K, A).$$

□

In the other direction:

Lemma 4.3 ($\mathbb{E}^\Gamma \circ \mathbb{CIU}^\Gamma \subseteq \mathbb{E}^\Gamma$). *If $(e_1, e_2) \in \mathbb{E}^\Gamma$ and $(e_2, e_3) \in \mathbb{CIU}^\Gamma$, then $(e_1, e_3) \in \mathbb{E}^\Gamma$.*

Proof. Choose n and $(\gamma, \gamma') \in \mathbb{G}_\Gamma^n$. We must show that $(e_1\gamma, e_3\gamma') \in \mathbb{E}_n^\Gamma$. So choose $m \leq n$, $(K, K') \in \mathbb{K}_m$, and $A \in \Sigma_\mathbb{R}$. Now we must show $\mu^{(m)}(e_1\gamma, K, A) \leq \mu(e_3\gamma', K', A)$.

We have $(e_1, e_2) \in \mathbb{E}^\Gamma$ and $(\gamma, \gamma') \in \mathbb{G}_\Gamma^n$, so $(e_1\gamma, e_2\gamma') \in \mathbb{E}_n$, and by $m \leq n$ we have $(e_1\gamma, e_2\gamma') \in \mathbb{E}_m$. So

$$\begin{aligned} \mu^{(n)}(e_1\gamma, K, A) &\leq \mu(e_2\gamma', K', A) && (\text{by } (e_1\gamma, e_2\gamma') \in \mathbb{E}_m) \\ &\leq \mu(e_3\gamma', K', A) && (\text{by } (e_2, e_3) \in \mathbb{CIU}) \end{aligned}$$

Therefore $(e_1, e_3) \in \mathbb{E}^\Gamma$. □

Lemma 4.4 ($\mathbb{CIU} \subseteq \mathbb{E}$). *If $(e, e') \in \mathbb{CIU}^\Gamma$ then $(e, e') \in \mathbb{E}^\Gamma$.*

Proof. Assume $(e, e') \in \mathbb{CIU}^\Gamma$. By the Fundamental Property, we know $(e, e) \in \mathbb{E}^\Gamma$. So we have $(e, e) \in \mathbb{E}^\Gamma$ and $(e, e') \in \mathbb{CIU}^\Gamma$. Hence, by Lemma 4.3, $(e, e') \in \mathbb{E}^\Gamma$. □

Theorem 4.5. *$(e, e') \in \mathbb{CIU}^\Gamma$ iff $(e, e') \in \mathbb{E}^\Gamma$.*

Proof. Immediate from Lemmas 4.2 and 4.4. □

5 CONTEXTUAL ORDERING

Finally, we arrive at the contextual order relation. We define the contextual ordering as the largest preorder that is both *adequate*—that is, it distinguishes terms that have different observable behavior by themselves—and *compatible*—that is, closed under context formation, and we show that the contextual ordering, the CIU ordering, and the logical relation all coincide. Thus in order to show two terms contextually ordered, it suffices to use the friendlier machinery of the CIU relation.

Definition 5.1 ($\mathbb{CTX}^\Gamma$). $\mathbb{CTX}$ is the largest family of relations R^Γ such that:

(1) R is adequate, that is, if $\Gamma = \emptyset$, then $(e, e') \in R^\Gamma$ implies that for all measurable subsets A of the reals, $\mu(e, \mathtt{halt}, A) \leq \mu(e', \mathtt{halt}, A)$.
(2) For each Γ, R^Γ is a preorder.
(3) The family of relations R is compatible, that is, it is closed under the type rules for expressions:
   (a) If $(e, e') \in R^{\Gamma,x}$, then $(\lambda x.e, \lambda x.e') \in R^\Gamma$.
   (b) If $(v_1, v_1') \in R^\Gamma$ and $(v_2, v_2') \in R^\Gamma$, then $((v_1\ v_2), (v_1'\ v_2')) \in R^\Gamma$.
   (c) If $(v, v') \in R^\Gamma$, then $(\mathtt{factor}\ v, \mathtt{factor}\ v') \in R^\Gamma$.
   (d) If $(e_1, e_1') \in R^\Gamma$ and $(e_2, e_2') \in R^{\Gamma,x}$,
   then $(\mathtt{let}\ x = e_1\ \mathtt{in}\ e_2, \mathtt{let}\ x = e_1'\ \mathtt{in}\ e_2') \in R^\Gamma$.
   (e) If $(v_1, v_1') \in R^\Gamma, \ldots, (v_n, v_n') \in R^\Gamma$,
   then $(op^n\ (v_1, \ldots, v_n), op^n\ (v_1', \ldots, v_n')) \in R^\Gamma$.
   (f) If $(v, v') \in R^\Gamma$, $(e_1, e_1') \in R^\Gamma$, and $(e_2, e_2') \in R^\Gamma$,
   then $(\mathtt{if}\ v\ \mathtt{then}\ e_1\ \mathtt{else}\ e_2, \mathtt{if}\ v'\ \mathtt{then}\ e_1'\ \mathtt{else}\ e_2') \in R^\Gamma$.

Note, as usual, that the union of any family of relations satisfying these conditions also satisfies these conditions, so the union of all of them is the largest such family of relations.

We prove that $\mathbb{E}^\Gamma$, $\mathbb{CIU}^\Gamma$, and $\mathbb{CTX}^\Gamma$ by first showing that $\mathbb{E}^\Gamma \subseteq \mathbb{CTX}^\Gamma$ and then that $\mathbb{CTX}^\Gamma \subseteq \mathbb{CIU}^\Gamma$. Then, having caught $\mathbb{CTX}^\Gamma$ between $\mathbb{E}^\Gamma$ and $\mathbb{CIU}^\Gamma$—two relations that we have already proven equivalent—we conclude that all of the relations coincide.

First, we must show that $\mathbb{E}^\Gamma \subseteq \mathbb{CTX}^\Gamma$. The heart of that proof is showing that $\mathbb{E}^\Gamma$ is compatible in the sense of Definition 5.1. That is *nearly* handled by the existing compatibility rules for $\mathbb{E}^\Gamma$ (Lemma 3.1), except for an occasional mismatch between expressions and values—that is, between $\mathbb{E}^\Gamma$ and $\mathbb{V}^\Gamma$ in the rules. So we need a lemma to address the mismatch (Lemma 5.3), which itself needs the following lemma due to ?.

Lemma 5.2. *If $(K, K') \in \mathbb{K}_n$ and $(v, v') \in \mathbb{V}_n$, then*

$$((z \to (z\ v))K, (z \to (z\ v'))K') \in \mathbb{K}_{n+2}$$

Proof. See appendix. □

Lemma 5.3. *For all closed values v, if $(v, v') \in \mathbb{E}$, then $(v, v') \in \mathbb{V}$.*

Proof. We will show that for all closed values v, v', if $(v, v') \in \mathbb{E}_{n+3}$, then $(v, v') \in \mathbb{V}_n$, from which the lemma follows.

If $v = \mathsf{c}_r$ and $v' = \mathsf{c}_{r'}$, then $r = r'$ and thus $(\mathsf{c}_r, \mathsf{c}_{r'}) \in \mathbb{V}$ because otherwise we would have $\mu(\mathsf{c}_r, \mathtt{halt}, \{r\}) = I_{\{r\}}(r) = 1$ and $\mu(\mathsf{c}_{r'}, \mathtt{halt}, \{r\}) = I_{\{r\}}(r') = 0$, violating the assumption $(\mathsf{c}_r, \mathsf{c}_{r'}) \in \mathbb{E}$.

If only one of v and v' is a constant, then $(v, v') \in \mathbb{E}_{n+3}$ is impossible, since constants and lambda-expressions are distinguishable by $\mathtt{real?}$ (which requires 3 steps to do so).

So assume $v = \lambda x.e$ and $v' = \lambda x.e'$. To establish $(v, v') \in \mathbb{V}_n$, choose $m < n$ and $(u, u') \in \mathbb{V}_m$. We must show that $(e[u/x], e'[u'/x]) \in \mathbb{E}_m$. To do that, choose $p \le m$, $(K, K') \in \mathbb{K}_p$, and $A \in \Sigma_{\mathbb{R}}$. We must show that

$$\mu^{(p)}(e[u/x], K, A) \le \mu(e'[u'/x], K', A)$$

Let $K_1 = (f \to (f\ u))K$ and $K_1' = (f \to (f\ u'))K'$. By monotonicity, $(u, u') \in \mathbb{V}_p$. By Lemma 5.2, $(K_1', K_1') \in \mathbb{K}_{p+2}$. Furthermore, $p \le m < n$, so $p + 2 \le n + 1$ and therefore $(\lambda x.e, \lambda x.e') \in \mathbb{E}_{p+2}$. And furthermore, we have

$$\langle \sigma \mid \lambda x.e \mid K_1 \mid \tau \mid w \rangle \to \langle \sigma \mid (\lambda x.e\ u) \mid K \mid \tau \mid w \rangle \to \langle \sigma \mid e[u/x] \mid K \mid \tau \mid w \rangle$$

and similarly on the primed side.

We can put the results together to get

$$\begin{aligned} \mu^{(p)}(e[u/x], K, A) &= \mu^{(p+2)}(\lambda x.e, K_1, A) \\ &\le \mu(\lambda x.e', K_1', A) \\ &= \mu(e'[u'/x], K', A) \end{aligned}$$

□

Theorem 5.4. *$\mathbb{E}^\Gamma \subseteq \mathbb{CTX}^\Gamma$.*

Proof. We will show that $\mathbb{E}$ forms a family of reflexive preorders that is adequate and compatible. Each $\mathbb{E}^\Gamma$ is reflexive by the Fundamental Property, and is a preorder because it is equal to $\mathbb{CIU}^\Gamma$, which is a preorder. To show that it is adequate, observe that $(\mathtt{halt}, \mathtt{halt}) \in \mathbb{K}$ by Lemma 3.1, hence for any measurable subset A of reals, $(e, e') \in \mathbb{E}^\Gamma$ implies $\mu(e, \mathtt{halt}, A) = \mu(e', \mathtt{halt}, A)$.

The $\mathbb{E}$-compatibility rules (Lemma 3.1) are almost exactly what is needed for $\mathbb{CTX}$-compatibility. The exceptions are in the application, operation, `if`, and `factor` rules where their hypotheses refer to $\mathbb{V}^\Gamma$ rather than $\mathbb{E}^\Gamma$. We fill the gap with Lemma 5.3. We show how this is done for `factor` v; the other cases are similar.

$$\begin{aligned} (v, v') \in \mathbb{E}^\Gamma &\implies (v, v') \in \mathbb{CIU}^\Gamma \\ &\implies (\forall\gamma)((v\gamma, v'\gamma) \in \mathbb{CIU}^\emptyset) \\ &\implies (\forall\gamma)((v\gamma, v'\gamma) \in \mathbb{E}^\emptyset) \\ &\implies (\forall\gamma)((v\gamma, v'\gamma) \in \mathbb{V}^\emptyset) && \text{(Lemma 5.3)} \\ &\implies (\forall\gamma)((\mathtt{factor}\ v\gamma, \mathtt{factor}\ v'\gamma) \in \mathbb{E}^\emptyset) && \text{(Lemma 3.1)} \\ &\implies (\forall\gamma)((\mathtt{factor}\ v\gamma, \mathtt{factor}\ v'\gamma) \in \mathbb{CIU}^\emptyset) \\ &\implies (\mathtt{factor}\ v, \mathtt{factor}\ v') \in \mathbb{CIU}^\Gamma \\ &\implies (\mathtt{factor}\ v, \mathtt{factor}\ v') \in \mathbb{E}^\Gamma \end{aligned}$$

□

Next, we must show that $\mathbb{CTX}^\Gamma \subseteq \mathbb{CIU}^\Gamma$ by induction on the closing substitution and then induction on the continuation. We use the following two lemmas to handle the closing substitution.

Lemma 5.5. *If $\Gamma, x \vdash e$ exp and $\Gamma \vdash v$ exp, then*

$$(e[v/x], (\lambda x.e\ v)) \in \mathbb{CIU}^\Gamma \text{ and } ((\lambda x.e\ v), e[v/x]) \in \mathbb{CIU}^\Gamma.$$

Proof. Let γ be a closing substitution for Γ. Then for any σ, closed K, and w, by Lemmas 2.15 and 2.14.4 we have

$$\langle \sigma \mid (\lambda x.e\gamma\ v\gamma) \mid K \mid \tau \mid w\rangle \rightarrow \langle \sigma \mid e\gamma[v\gamma/x] \mid K \mid \tau \mid w\rangle$$

Therefore for any $A \in \Sigma_{\mathbb{R}}$, $\mu((\lambda x.e\gamma\ v\gamma), K, A) = \mu(e\gamma[v\gamma/x], K, A)$. □

Lemma 5.6. *If* $(e, e') \in \mathbb{CTX}^{\Gamma,x}$, *and* $(v, v') \in \mathbb{CTX}^{\Gamma}$, *then* $(e[v/x], e'[v'/x]) \in \mathbb{CTX}^{\Gamma}$.

Proof. From the assumptions and the compatibility of $\mathbb{CTX}$, we have

$$((\lambda x.e\ v), (\lambda x.e'\ v')) \in \mathbb{CTX}^{\Gamma} \tag{1}$$

So now we have:

$$\begin{aligned}
&(e[v/x], (\lambda x.e\ v)) \in \mathbb{CIU}^{\Gamma} && \text{(Lemma 5.5)}\\
\implies\ &(e[v/x], (\lambda x.e\ v)) \in \mathbb{CTX}^{\Gamma} && (\mathbb{CIU}^{\Gamma} \subseteq \mathbb{CTX}^{\Gamma})\\
\implies\ &(e[v/x], (\lambda x.e'\ v')) \in \mathbb{CTX}^{\Gamma} && \text{(Equation (1) and transitivity of } \mathbb{CTX}^{\Gamma})\\
\implies\ &(e[v/x], e'[v'/x]) \in \mathbb{CTX}^{\Gamma} && \text{(Lemma 5.5 and transitivity of } \mathbb{CTX}^{\Gamma})
\end{aligned}$$

□

Now we are ready to complete the theorem. Here we need to use $\mathbb{CIU}$ rather than $\mathbb{E}$, so that we can deal with only one continuation rather than two.

Theorem 5.7 ($\mathbb{CTX}^{\Gamma} \subseteq \mathbb{CIU}^{\Gamma}$). *If* $(e, e') \in \mathbb{CTX}^{\Gamma}$, *then* $(e, e') \in \mathbb{CIU}^{\Gamma}$

Proof. By the preceding lemma, we have $(e\gamma, e'\gamma) \in \mathbb{CTX}$. So it suffices to show that for all $A \in \Sigma_{\mathbb{R}}$, if $(e, e') \in \mathbb{CTX}^{\emptyset}$ and $\vdash K$ cont, then $\mu(e, K, A) = \mu(e', K, A)$.

The proof proceeds by induction on K such that $\vdash K$ cont. The induction hypothesis on K is: for all closed e, e', if $(e, e') \in \mathbb{CTX}^{\emptyset}$, then $\mu(e, K, A) = \mu(e', K, A)$.

If $K = \mathtt{halt}$ and $(e, e') \in \mathbb{CTX}^{\emptyset}$, then $\mu(e, \mathtt{halt}, A) = \mu(e', \mathtt{halt}, A)$ by the adequacy of $\mathbb{CTX}^{\emptyset}$.

For the induction step, consider $(x \to e_1)K$, where $x \vdash e_1$ exp. Choose $(e, e') \in \mathbb{CTX}^{\emptyset}$. We must show $\mu(e, (x \to e_1)K, A) \leq \mu(e', (x \to e_1)K, A)$.

By the compatibility of $\mathbb{CTX}$, we have

$$(\mathtt{let}\, x = e\ \mathtt{in}\ e_1, \mathtt{let}\, x = e'\ \mathtt{in}\ e_1) \in \mathbb{CTX}^{\emptyset} \tag{2}$$

Then we have

$$\begin{aligned}
\mu(e, (x \to e_1)K, A) &= \mu(\mathtt{let}\, x = e\ \mathtt{in}\ e_1, K, A) && \text{(Lemma 2.15)}\\
&\leq \mu(\mathtt{let}\, x = e'\ \mathtt{in}\ e_1, K, A) && \text{(by IH at } K\text{, applied to (2))}\\
&= \mu(e', (x \to e_1)K, A) && \text{(Lemma 2.15)}
\end{aligned}$$

Thus completing the induction step. □

Summarizing the results:

Theorem 5.8. *For all* Γ, $\mathbb{CIU}^{\Gamma} = \mathbb{E}^{\Gamma} = \mathbb{CTX}^{\Gamma}$.

Proof. $\mathbb{CIU}^{\Gamma} = \mathbb{E}^{\Gamma} \subseteq \mathbb{CTX}^{\Gamma} \subseteq \mathbb{CIU}^{\Gamma}$ by Theorems 4.5, 5.4, and 5.7, respectively. □

$$
\begin{array}{rcll}
((\lambda x.e)\ v) & =_{\text{ctx}} & e[v/x] & (\beta_v) \\
\texttt{let}\, x = v\ \texttt{in}\ e & =_{\text{ctx}} & e[v/x] & (\text{let}_v) \\
\texttt{let}\, x = e\ \texttt{in}\ x & =_{\text{ctx}} & e & (\text{let}_{id}) \\
op\,(v_1, \cdots, v_n) & =_{\text{ctx}} & v \quad \text{where } \delta(op, v_1, \cdots, v_n) = v & (\delta) \\
\texttt{let}\, x_2 = (\texttt{let}\, x_1 = e_1\ \texttt{in}\ e_2)\ \texttt{in}\ e_3 & =_{\text{ctx}} & \texttt{let}\, x_1 = e_1\ \texttt{in}\ (\texttt{let}\, x_2 = e_2\ \texttt{in}\ e_3) & (\text{assoc}) \\
\texttt{let}\, x_1 = e_1\ \texttt{in}\ \texttt{let}\, x_2 = e_2\ \texttt{in}\ e_3 & =_{\text{ctx}} & \texttt{let}\, x_2 = e_2\ \texttt{in}\ \texttt{let}\, x_1 = e_1\ \texttt{in}\ e_3 & (\text{commut})
\end{array}
$$

In (assoc), $x_1 \notin FV(e_3)$. In (commut), $x_1 \notin FV(e_2)$ and $x_2 \notin FV(e_1)$.

Fig. 6. Catalog of equivalences

6 CONTEXTUAL EQUIVALENCE

Definition 6.1. If $\Gamma \vdash e$ exp and $\Gamma \vdash e'$ exp, we say e and e' are *contextually equivalent* ($e =_{\text{ctx}} e'$) if both $(e, e') \in \mathbb{CTX}^\Gamma$ and $(e', e) \in \mathbb{CTX}^\Gamma$.

In this section we use the machinery from the last few sections to prove the equivalence schemes listed in Figure 6. The equivalences fall into three categories:

(1) provable directly using CIU and Theorem 5.8
(2) dependent on "entropy-shuffling"
(3) mathematical properties of $\mathbb{R}$, probability distributions, etc

6.1 β_v, let$_v$, and δ

The proof for β_v demonstrates the general pattern of equivalence proofs using CIU: first we prove the equation holds for closed expressions, then we generalize to open terms by considering all closing substitutions.

Lemma 6.2. *If* $\vdash ((\lambda x.e)\ v)$ exp*, then* $((\lambda x.e)\ v) =_{ctx} e[v/x]$.

Proof. By Lemma 2.15, the definition of $\mathbb{CIU}$, and Theorem 5.8. □

Corollary 6.3 (β_v). *If* $\Gamma \vdash ((\lambda x.e)\ v)$ exp*, then* $((\lambda x.e)\ v) =_{ctx} e[v/x]$.

Proof. By Lemma 6.2, any closed instances of these expressions are contextually equivalent and thus CIU-equivalent. Hence the open expressions are CIU-equivalent and thus contextually equivalent. □

The proofs of let$_v$ and δ are similar.

6.2 Rearranging Entropy

The remaining equivalences from Figure 6 involve non-trivial changes to the entropy access patterns of their subexpressions. In this section we characterize a class of transformations on the entropy space that are measure-preserving. In the next section we use these functions to justify reordering and rearranging subexpression evaluation.

Definition 6.4 (measure-preserving). A function $\phi : \mathbb{S} \to \mathbb{S}$ is measure-preserving when for all measurable $g : \mathbb{S} \to \mathbb{R}^+$,

$$\int g(\phi(\sigma))\ d\sigma = \int g(\sigma)\ d\sigma$$

Note that this definition is implicitly specific to the stock entropy measure $\mu_{\mathbb{S}}$, which is sufficient for our needs.

More specifically, the kinds of functions we are interested in are ones that break apart the entropy into independent pieces using π_L and π_R and then reassemble the pieces of entropy using ::. Pieces may be discarded, but no piece may be used more than once.

For example, the following function is measure-preserving:

$$\phi_c(\sigma_1::(\sigma_2::\sigma_3)) = \sigma_2::(\sigma_1::\sigma_3)$$

Or equivalently, written using explicit projections:

$$\phi_c(\sigma) = \pi_L(\pi_R(\sigma))::(\pi_L(\sigma)::\pi_R(\pi_R(\sigma)))$$

We will use this function in Theorem 6.9 to justify `let`-reordering. Another example is

$$\phi_d(\sigma_1::\sigma_2) = \sigma_2$$

which could be used to drop dead `let` bindings.

To characterize such functions, we need some auxiliary definitions:

- A *path* $p = [d_1, \ldots, d_n]$ is a (possibly empty) list of directions (L or R). It represents a sequence of projections, and it can be viewed as a function from $\mathbb{S}$ to $\mathbb{S}$.
$$[d_1, \ldots, d_n](\sigma) = (\pi_{d_1} \circ \cdots \circ \pi_{d_n})(\sigma)$$
- A *finite shuffling function* (FSF) ϕ is either a path or $\phi_1::\phi_2$ where ϕ_1 and ϕ_2 are FSFs. It represents the disassembly and reassembly of entropy, and it can be viewed as a recursively defined function from $\mathbb{S}$ to $\mathbb{S}$.
$$\phi(\sigma) = \begin{cases} p(\sigma) & \text{if } \phi = p \\ \phi_1(\sigma)::\phi_2(\sigma) & \text{if } \phi = \phi_1::\phi_2 \end{cases}$$
- A sequence of paths is said to be *non-duplicating* if no path is the suffix of another path in the sequence.
- An FSF is said to be *non-duplicating* if the sequence of paths appearing in its definition is non-duplicating.

Lemma 6.5. *Let $p_1, \ldots, p_n$ be a non-duplicating sequence of paths and $g : \mathbb{S}^n \to \mathbb{R}^+$. Then*

$$\int g(p_1(\sigma), \ldots, p_n(\sigma))\, d\sigma = \int \ldots \int g(\sigma_1, \ldots, \sigma_n)\, d\sigma_1 \ldots\, d\sigma_n$$

Proof. By strong induction on the length of the longest path in the sequence, and by the definition of non-duplicating and Lemma 2.2 (Tonelli). □

Theorem 6.6. *If ϕ is a non-duplicating FSF then ϕ is measure preserving.*

Proof. We need to show that for any $g : \mathbb{S} \to \mathbb{R}^+$,

$$\int g(\phi(\sigma))\, d\sigma = \int g(\sigma'')\, d\sigma''$$

If ϕ has paths $p_1, \ldots, p_n$, then we can decompose ϕ using $s : \mathbb{S}^n \to \mathbb{S}$ such that

$$\phi(\sigma) = s(p_1(\sigma), \ldots, p_n(\sigma))$$

where the p_i are non-duplicating. Then by Lemma 6.5 it is enough to show that

$$\int \ldots \int g(s(\sigma_1, \ldots, \sigma_n))\, d\sigma_1 \ldots\, d\sigma_n = \int g(\sigma'')\, d\sigma''$$

We proceed by induction on ϕ.

- case $\phi = p$. This means that $n = 1$ and s is the identity function, so the equality holds trivially.

- case $\phi = \phi_1{::}\phi_2$. If m is the number of paths in ϕ_1, then there must be $s_1 : \mathbb{S}^m \to \mathbb{S}$ and $s_2 : \mathbb{S}^{n-m} \to \mathbb{S}$ such that

$$s(\sigma_1, \ldots, \sigma_m, \sigma_{m+1}, \ldots, \sigma_n) = s_1(\sigma_1, \ldots, \sigma_m){::}s_2(\sigma_{m+1}, \ldots, \sigma_n)$$

We can conclude that

$$\begin{aligned}
&\int \ldots \int g(s(\sigma_1, \ldots, \sigma_n))\, d\sigma_1 \ldots\, d\sigma_n \\
&= \int \ldots \int g(s_1(\sigma_1, \ldots, \sigma_m){::}s_2(\sigma_{m+1}, \ldots, \sigma_n))\, d\sigma_1 \ldots\, d\sigma_n \\
&= \iint g(\sigma{::}\sigma')\, d\sigma\, d\sigma' && \text{(IH twice)} \\
&= \int g(\sigma'')\, d\sigma'' && \text{(Property 2.1(4))}
\end{aligned}$$

□

6.3 Equivalences that depend on rearranging entropy

We first prove a general theorem relating value-preserving transformations on the entropy space:

Theorem 6.7. *Let e and e' be closed expressions, and let $\phi : \mathbb{S} \to \mathbb{S}$ be a measure-preserving transformation such that for all σ, K, τ, and A*

$$\text{eval}(\sigma, e, K, \tau, 1, A) \leq \text{eval}(\phi(\sigma), e', K, \tau, 1, A)$$

Then $(e, e') \in \mathbb{CTX}$.

Proof. Without loss of generality, assume e and e' are closed (otherwise apply a closing substitution). By Theorem 5.8, it is sufficient to show that for any K and A, $\mu(e, K, A) \leq \mu(e', K, A)$. We calculate:

$$\begin{aligned}
\mu(e, K, A) &= \iint \text{eval}(\sigma, e, K, \tau, 1, A)\, d\sigma\, d\tau \\
&\leq \iint \text{eval}(\phi(\sigma), e', K, \tau, 1, A)\, d\sigma\, d\tau \\
&= \iint \text{eval}(\sigma, e', K, \tau, 1, A)\, d\sigma\, d\tau && (\phi \text{ is measure-preserving}) \\
&= \mu(e', K, A)
\end{aligned}$$

□

Theorem 6.8. *Let e and e' be closed expressions, and let $\phi : \mathbb{S} \to \mathbb{S}$ be a measure-preserving transformation such that for all v and w,*

$$\sigma \vdash e \Downarrow v, w \implies \phi(\sigma) \vdash e' \Downarrow v, w.$$

Then $(e, e') \in \mathbb{CTX}$.

Proof. We will use Theorem 6.7. Assume $\text{eval}(\sigma, e, K, \tau, 1, A) = r > 0$. Hence by Theorem 2.11, there exist quantities v', w', σ', and τ' such that

$$\langle \sigma \mid e \mid K \mid \tau \mid 1 \rangle \to^* \langle \sigma' \mid v' \mid \texttt{halt} \mid \tau' \mid r \rangle$$

with $v' \in A$. By Theorem 2.11 there exist v'', σ'', and w'' such that

$$\sigma \vdash e \Downarrow v'', w'' \text{ and } \langle \sigma'' \mid v'' \mid K \mid \tau \mid w'' \rangle \to^* \langle \sigma' \mid v' \mid \texttt{halt} \mid \tau' \mid r \rangle$$

By the assumption of the theorem, we have $\phi(\sigma) \vdash e' \Downarrow v'', w''$.
Therefore, by Theorem 2.5, there is a σ''' such that

$$\langle \sigma' \mid e \mid K \mid t \mid 1 \rangle \to^* \langle \sigma''' \mid v'' \mid K \mid \tau \mid w'' \rangle$$

We claim that $\mathrm{eval}(\phi(s), e', K, \tau, 1, A) = r$. We proceed by cases on K. We have $\langle \sigma'' \mid v'' \mid K \mid \tau \mid w'' \rangle \to^* \langle \phi(s) \mid v' \mid \mathtt{halt} \mid \tau'' \mid r \rangle$. If $K = \mathtt{halt}$, this reduction must have length 0. Therefore $v'' = v' \in A$ and $w'' = r$, so $\mathrm{eval}(\phi(s), e', K, \tau, 1, A) = r$.

Otherwise assume $K = (x \to e_3)K'$. Then both $\langle \sigma'' \mid v" \mid K \mid t \mid w'' \rangle$ and $\langle \sigma''' \mid v" \mid K \mid t \mid w'' \rangle$ take a step to $\langle \pi_L(\tau) \mid e_3[v''/x] \mid K' \mid \pi_R(\tau) \mid w'' \rangle$, whence $\mathrm{eval}(\phi(s), e', K, \tau, 1, A) = \mathrm{eval}(\sigma, e, K, \tau, 1, A) = r$, as desired, thus establishing the requirement of Theorem 6.7. □

Now we can finally prove the commutativity theorem promised at the beginning.

Theorem 6.9. *Let e_1 and e_2 be closed expressions, and $\{x_1, x_2\} \vdash e_0$ $\mathtt{exp}$. Then the expressions*

$$\mathtt{let}\, x_1 = e_1 \;\mathtt{in}\; \mathtt{let}\, x_2 = e_2 \;\mathtt{in}\; e_0$$

and

$$\mathtt{let}\, x_2 = e_2 \;\mathtt{in}\; \mathtt{let}\, x_1 = e_1 \;\mathtt{in}\; e_0$$

are contextually equivalent.

Proof using big-step semantics. Let e and e' denote the two expressions of the theorem. We will use Theorem 6.8 with the function $\phi_c(\sigma_1{::}(\sigma_2{::}\sigma_3)) = \sigma_2{::}(\sigma_1{::}\sigma_3)$, which preserves entropy as shown in the preceding section. We will show that if $\sigma \vdash e \Downarrow v, w$, then $\phi(\sigma) \vdash e \Downarrow v, w$.

Inverting $\sigma \vdash e \Downarrow v, w$, we know there must be a derivation

$$\dfrac{\pi_L(\sigma) \vdash e_1 \Downarrow v_1, w_1 \qquad \dfrac{\pi_L(\pi_R(\sigma)) \vdash e_2 \Downarrow v_2, w_{21} \qquad \pi_R(\pi_R(\sigma)) \vdash e_0[v_1/x_1][v_2/x_2] \Downarrow v, w_{22}}{\pi_R(\sigma) \vdash \mathtt{let}\, x_2 = e_2 \;\mathtt{in}\; e_0 \Downarrow v, w_2}}{\sigma \vdash \mathtt{let}\, x_1 = e_1 \;\mathtt{in}\; \mathtt{let}\, x_2 = e_2 \;\mathtt{in}\; e_0 \Downarrow v, w}$$

where $w = w_1 \times w_2 = w_1 \times (w_{21} \times w_{22})$.

Since e_1 and e_2 are closed, they evaluate to closed v_1 and v_2, and so the substitutions $[v_1/x_1]$ and $[v_2/x_2]$ commute. Using that and the associativity and commutativity of multiplication, we can rearrange the pieces to get

$$\dfrac{\pi_L(\pi_R(\sigma)) \vdash e_2 \Downarrow v_2, w_{21} \qquad \dfrac{\pi_L(\sigma) \vdash e_1 \Downarrow v_1, w_1 \qquad \pi_R(\pi_R(\sigma)) \vdash e_0[v_2/x_2][v_1/x_1] \Downarrow v, w_{22}}{\pi_L(\sigma){::}\pi_R(\pi_R(\sigma)) \vdash \mathtt{let}\, x_1 = e_1 \;\mathtt{in}\; e_0 \Downarrow v, w_1 \times w_{22}}}{\pi_L(\pi_R(\sigma)){::}\pi_L(\sigma){::}\pi_R(\pi_R(\sigma)) \vdash \mathtt{let}\, x_2 = e_2 \;\mathtt{in}\; \mathtt{let}\, x_1 = e_1 \;\mathtt{in}\; e_0 \Downarrow v, w}$$

The entropy in the last line is precisely $\phi(\sigma)$, so the requirement of Theorem 6.8 is established. □

Theorem 6.9 can also be proven directly from the small-step semantics using the interpolation and genericity theorems (2.8 and 2.9) to recover the structure that the big-step semantics makes explicit. The proof may be found in Appendix A.6.

Corollary 6.10 (Commutativity). *Let e_1 and e_2 be expressions such that x_1 is not free in e_2 and x_2 is not free in e_1. Then*

$$(\mathtt{let}\, x_1 = e_1 \;\mathtt{in}\; \mathtt{let}\, x_2 = e_2 \;\mathtt{in}\; e_0) =_{ctx} (\mathtt{let}\, x_2 = e_2 \;\mathtt{in}\; \mathtt{let}\, x_1 = e_1 \;\mathtt{in}\; e_0)$$

Proof. Same as Corollary 6.3: since all of the closed instances are equivalent by Theorem 6.9, the open expressions are equivalent. □

The proofs of `let`-associativity and let_{id} follow the same structure, except that associativity uses $\phi_a((\sigma_1{::}\sigma_2){::}\sigma_3) = \sigma_1{::}(\sigma_2{::}\sigma_3)$ and let_{id} uses $\phi_i(\sigma_1{::}\sigma_2) = \sigma_1$.

6.4 Quasi-denotational Reasoning

In this section we give a powerful "quasi-denotational" reasoning tool that shows that if two expressions are denote the same measure, they are contextually equivalent. This allows us to import mathematical facts about real arithmetic and probability distributions.

To support this kind of reasoning, we need a notion of measure for a (closed) expression independent of a program continuation. We define $\hat{\mu}(e, -)$ as a measure over arbitrary *syntactic values*—not just real numbers as with $\mu(e, K, -)$. This measure corresponds directly to the μ_e of **?** and $[\![e]\!]_S$ of **?**. The definition of $\hat{\mu}$ uses a generalization of eval from measurable sets of reals (A) to measurable sets of syntactic values (V). This requires a measurable space for syntactic values; we take the construction of **?**, Figure 5 mutatis mutandis.

Definition 6.11.

$$\hat{\mu}(e, V) = \iint \text{eval}(\sigma, e, \texttt{halt}, \tau, 1, V)\ d\sigma\, d\tau$$

$$\text{eval}(\sigma, e, K, \tau, w, V) = \begin{cases} w' & \text{if } \langle \sigma \mid e \mid K \mid \tau \mid w \rangle \to^* \langle \sigma' \mid v \mid \texttt{halt} \mid \tau' \mid w' \rangle, \\ & \text{where } v \in V \\ 0 & \text{otherwise} \end{cases}$$

Our goal is to relate an expression's measure $\hat{\mu}(e, -)$ with the measure of that expression with a program continuation ($\mu(e, K, -)$). Then if two expressions have the same measures, we can use CIU to show them contextually equivalent.

First we need a lemma about decomposing evaluations. It is easiest to state if we define the value and weight projections of evaluation:

$$\text{ev}(\sigma, e, K, \tau) = \begin{cases} v & \text{when } \langle \sigma \mid e \mid K \mid \tau \mid 1 \rangle \to^* \langle \sigma' \mid v \mid \texttt{halt} \mid \tau' \mid w \rangle \\ \bot & \text{otherwise} \end{cases}$$

$$\text{ew}(\sigma, e, K, \tau) = \begin{cases} w & \text{when } \langle \sigma \mid e \mid K \mid \tau \mid 1 \rangle \to^* \langle \sigma' \mid v \mid \texttt{halt} \mid \tau' \mid w \rangle \\ 0 & \text{otherwise} \end{cases}$$

Note that

$$\text{eval}(\sigma, e, K, \tau, 1, V) = I_V(\text{ev}(\sigma, e, K, \tau)) \times \text{ew}(\sigma, e, K, \tau)$$

$$\hat{\mu}(e, A) = \mu(e, \texttt{halt}, A) \qquad \text{for } A \in \Sigma_{\mathbb{R}}$$

Lemma 6.12.

$$\text{ev}(\sigma, e, K, \tau) = \text{ev}(\sigma', \text{ev}(\sigma, e, \texttt{halt}, \tau'), K, \tau)$$

$$\text{ew}(\sigma, e, K, \tau) = \text{ew}(\sigma', \text{ev}(\sigma, e, \texttt{halt}, \tau'), K, \tau) \times \text{ew}(\sigma, e, \texttt{halt}, \tau')$$

Proof. By reduction-sequence surgery using Theorem 2.9. Note that the primed variables are dead: σ' because ev returns a value and τ' because `halt` does not use its entropy stack. □

Next we need a lemma from measure theory:

Lemma 6.13. *If μ and ν are measures and $\nu(A) = \int I_A(f(x)) \times w(x)\ \mu(dx)$, then*

$$\int g(y)\ \nu(dy) = \int g(f(x)) \times w(x)\ \mu(dx)$$

Proof. By the pushforward and Radon-Nikodym lemmas from measure theory. □

Now we are ready for the main theorem, which says that $\mu(e, K, -)$ can be expressed as an integral over $\hat{\mu}(e, -)$ where K appears only in the integrand and e appears only in the measure of integration.

Theorem 6.14.

$$\mu(e,K,A) = \iiint \mathrm{eval}(\sigma,v,K,\tau,1,A)\ \hat{\mu}(e,dv)\,d\sigma\,d\tau$$

Proof. By integral calculations and Lemma 6.12:

$$\begin{aligned}
\mu(e,K,A) &= \iint \mathrm{eval}(\sigma,e,K,\tau,1,A)\ d\sigma\,d\tau \\
&= \iint I_A(\mathrm{ev}(\sigma,e,K,\tau)) \times \mathrm{ew}(\sigma,e,K,\tau)\ d\sigma\,d\tau \\
&= \iiiint I_A(\mathrm{ev}(\sigma,e,K,\tau)) \times \mathrm{ew}(\sigma,e,K,\tau)\ d\sigma'\,d\tau'\,d\sigma\,d\tau && (\mu_{\mathbb{S}}(\mathbb{S})=1) \\
&= \iiiint\ I_A(\mathrm{ev}(\sigma',\mathrm{ev}(\sigma,e,\mathtt{halt},\tau'),K,\tau)) \times \mathrm{ew}(\sigma',\mathrm{ev}(\sigma,e,\mathtt{halt},\tau'),K,\tau) \\
&\qquad\qquad \times \mathrm{ew}(\sigma,e,\mathtt{halt},\tau')\ d\sigma'\,d\tau'\,d\sigma\,d\tau \\
& && \text{(Lemma 6.12)} \\
&= \iiint I_A(\mathrm{ev}(\sigma',v,K,\tau)) \times \mathrm{ew}(\sigma',v,K,\tau)\ \hat{\mu}(e,dv)\,d\sigma'\,d\tau && \text{(Lemma 6.13)} \\
&= \iiint \mathrm{eval}(\sigma',v,K,\tau,1,A)\ \hat{\mu}(e,dv)\,d\sigma'\,d\tau
\end{aligned}$$

□

As a consequence, two real-valued expressions are contextually equivalent if their expression measures agree:

Theorem 6.15 ($\hat{\mu}$ is quasi-denotational). *If e and e' are closed expressions such that*

- *e and e' are almost always real-valued—that is, $\hat{\mu}(e, Values - \mathbb{R}) = 0$ and likewise for e'—and*
- *for all $A \in \Sigma_{\mathbb{R}}$, $\hat{\mu}(e,A) = \hat{\mu}(e',A)$*

then $e =_{ctx} e'$.

Proof. The two conditions together imply that $\hat{\mu}(e,-) = \hat{\mu}(e',-)$.

We use Theorem 5.8; we must show $(e,e') \in \mathbb{CIU}$ and $(e',e) \in \mathbb{CIU}$. Choose a continuation K and a measurable set $A \in \Sigma_{\mathbb{R}}$. Then

$$\begin{aligned}
\mu(e,K,A) &= \iiint \mathrm{eval}(\sigma,v,K,\tau,1,A)\ \hat{\mu}(e,dv)\,d\sigma\,d\tau && \text{(by Lemma 6.14)} \\
&= \iiint \mathrm{eval}(\sigma,v,K,\tau,1,A)\ \hat{\mu}(e',dv)\,d\sigma\,d\tau && (\hat{\mu}(e,-)=\hat{\mu}(e',-)) \\
&= \mu(e',K,A) && \text{(by Lemma 6.14 again)}
\end{aligned}$$

The proof of $(e',e) \in \mathbb{CIU}$ is symmetric. □

Theorem 6.15 allows us to import many useful facts from mathematics about real numbers, real operations, and real-valued probability distributions. For example, here are a few equations useful in the transformation of the linear regression example from Section 1:

- $x + y = y + x$
- $(y + x) - z = x - (z - y)$
- $\mathtt{normalpdf}(x - y; 0, s) = \mathtt{normalpdf}(y; x, s)$
- The closed-form posterior and normalizer for a normal observation with normal conjugate prior [?]:

$$\begin{aligned}
&\mathtt{let}\ m = \mathtt{normal}(m_0, s_0)\ \mathtt{in} \\
&\mathtt{let}\ _ = \mathtt{factor}\ \mathtt{normalpdf}(d; m, s)\ \mathtt{in} \\
&m
\end{aligned}$$

$$= \begin{aligned}
&\mathtt{let}\ m = \mathtt{normal}(\ \left(\tfrac{1}{s_0^2}+\tfrac{1}{s^2}\right)^{-1}\left(\tfrac{m_0}{s_0^2}+\tfrac{d}{s^2}\right), \left(\tfrac{1}{s_0^2}+\tfrac{1}{s^2}\right)^{-1/2}\ )\ \mathtt{in} \\
&\mathtt{let}\ _ = \mathtt{factor}\ \mathtt{normalpdf}(d; m_0, (s_0^2+s^2)^{1/2})\ \mathtt{in} \\
&m
\end{aligned}$$

Note that we must keep the normalizer (the marginal likelihood of d); it is needed to score the hyper-parameters m_0 and s_0.

Section 7.2 contains an additional application of Theorem 6.15.

7 FORMALLY RELATED WORK

Our language model differs from other models of probabilistic languages, such as that of ?, in the following ways. Our language

- uses *splitting* rather than *sequenced* entropy,
- requires `let`-binding of nontrivial intermediate expressions, and
- directly models only the standard uniform distribution.

These differences, while they make our proofs easier, do *not* amount to fundamental differences in the meaning of probabilistic programs. In this section, we show how our semantics corresponds to other formulations.

7.1 Splitting versus Sequenced Entropy

Let the sequenced entropy space $\mathbb{T}$ be the space of finite sequences ("traces") of real numbers [?, Section 3.3]:

$$\mathbb{T} = \bigcup_{n \geq 0} \mathbb{R}^n$$

Its stock measure $\mu_{\mathbb{T}}$ is the sum of the standard Lebesgue measures on $\mathbb{R}^n$ (but restricted to the Borel algebras on $\mathbb{R}^n$ rather than their completions with negligible sets). Note that $\mu_{\mathbb{T}}$ is infinite.

We write ϵ for the empty sequence and $r{::}t$ for the sequence consisting of r followed by the elements of t. Integration with respect to $\mu_{\mathbb{T}}$ has the following property:

$$\int f(t)\ \mu_{\mathbb{T}}(dt) = f(\epsilon) + \iint f(r{::}t)\ \mu_{\mathbb{T}}(dt)\ \lambda(dr)$$

We define $\ddot{\rightarrow}$, $\ddot{\mathrm{eval}}(t, e, K, w, A)$, and $\ddot{\mu}(e, K, A)$[2] as the sequenced-entropy analogues of $\rightarrow$, eval, and μ. Here are some representative rules of $\ddot{\rightarrow}$:

$$\begin{aligned}
\langle t \mid \mathtt{let}\, x = e_1\ \mathtt{in}\ e_2 \mid K \mid w\rangle &\ \ddot{\rightarrow}\ \langle t \mid e_1 \mid (x \rightarrow e_2)K \mid w\rangle \\
\langle t \mid v \mid (x \rightarrow e_2)K \mid w\rangle &\ \ddot{\rightarrow}\ \langle t \mid e_2[v/x] \mid K \mid w\rangle \\
\langle r{::}t \mid \mathtt{sample} \mid K \mid w\rangle &\ \ddot{\rightarrow}\ \langle t \mid \mathtt{c}_r \mid K \mid w\rangle \quad (\text{when } 0 \leq r \leq 1) \\
\langle t \mid \mathtt{factor}\ \mathtt{c}_r \mid K \mid w\rangle &\ \ddot{\rightarrow}\ \langle t \mid \mathtt{c}_r \mid K \mid w \times r\rangle \quad (\text{when } r > 0)
\end{aligned}$$

and here are the definitions of $\ddot{\mathrm{eval}}$ and $\ddot{\mu}$:

$$\ddot{\mathrm{eval}}(t, e, K, w, A) = \begin{cases} w' & \text{if } \langle t \mid e \mid K \mid w\rangle \ddot{\rightarrow}^* \langle \epsilon \mid r \mid \mathtt{halt} \mid w'\rangle, \text{ where } r \in A \\ 0 & \text{otherwise} \end{cases}$$

$$\ddot{\mu}(e, K, A) = \int \ddot{\mathrm{eval}}(t, e, K, 1, A)\ \mu_{\mathbb{T}}(dt)$$

Note that an evaluation counts only if it completely exhausts its entropy sequence t. The approximants $\ddot{\mathrm{eval}}^{(n)}$ and $\ddot{\mu}^{(n)}$ are defined as before; in particular, they are indexed by number of steps, not by random numbers consumed.

In general, the entropy access pattern is so different between split and sequenced entropy models that there is no correspondence between individual evaluations, and yet the resulting measures are equivalent.

Lemma 7.1. *If* $\langle t \mid e \mid K \mid w\rangle \ddot{\rightarrow} \langle t' \mid e' \mid K' \mid w'\rangle$, *then* $\ddot{\mathrm{eval}}^{(p+1)}(t, e, K, w, A) = \ddot{\mathrm{eval}}^{(p)}(t', e', K', w', A)$.

[2] The dots are intended as a mnemonic for sequencing.

Proof. By the definition of $\ddot{\text{eval}}^{(p+1)}$. □

Lemma 7.2. *The equations of Lemma 2.15 also hold for $\ddot{\mu}^{(n)}$ and $\ddot{\mu}$.*

Proof. By definition of $\ddot{\mu}$ and Lemma 7.1. In fact, in contrast to Lemma 2.15, most of the cases are utterly straightforward, because no entropy shuffling is necessary. The `sample` case is different, because it relies on the structure of the entropy space:

$$\begin{aligned}\ddot{\mu}(e, K, A) &= \int \ddot{\text{eval}}(t, \texttt{sample}, K, 1, A)\ d(t) \\ &= \ddot{\text{eval}}(\epsilon, \texttt{sample}, K, 1, A) + \iint \ddot{\text{eval}}(r{::}t, \texttt{sample}, K, 1, A)\ \mu_{\mathbb{T}}(dt)\ \lambda(dr) \\ &= 0 + \iint I_{[0,1]}(r) \times \ddot{\text{eval}}(t, \texttt{c}_r, K, 1, A)\ \mu_{\mathbb{T}}(dt)\ \lambda(dr) \\ &= \int_0^1 \ddot{\mu}(\texttt{c}_r, K, A)\ \lambda(dr)\end{aligned}$$

□

Theorem 7.3 ($\ddot{\mu} = \mu$). *For all e, K, and $A \in \Sigma_{\mathbb{R}}$, $\ddot{\mu}(e, K, A) = \mu(e, K, A)$.*

Proof. We first show $\ddot{\mu}^{(n)} = \mu^{(n)}$ by induction on n. The base case is $\ddot{\mu}^{(0)}(e, K, A) = \mu^{(0)}(e, K, A)$. There are two subcases: if $e = r$ and $K = \texttt{halt}$, then both results are $I_A(r)$. Otherwise, both measures are 0. Lemma 7.2 handles the inductive case. Finally, since the approximants are pointwise equivalent, their limits are equivalent. □

7.2 Distributions

The language of **?** supports multiple real-valued distributions with real parameters; sampling from a distribution, in addition to consuming a random number, multiplies the current execution weight by the *density* of the distribution at that point. In this section we show that `sample` is equally expressive, given the inverse-CDF operations.

For each real-valued distribution of interest with n real-valued parameters, we add the following to the language: a sampling form $\texttt{D}(v_1, \ldots, v_n)$ and operations `Dpdf`, `Dcdf`, and `Dinvcdf` representing the distribution's density function, cumulative distribution function, and inverse cumulative distribution function, respectively. The operations take $n + 1$ arguments; by convention we write a semicolon before the parameters. For example, $\texttt{gammapdf}(x; k, s)$ represents the density at x of the gamma distribution with shape k and scale s.

We define the semantics of `D` using the sequenced-entropy framework by extending $\ddot{\rightarrow}$ with the following rule schema:

$$\langle r{::}t \mid \texttt{D}(r_1, \ldots, r_n) \mid K \mid w\rangle \ddot{\rightarrow} \langle t \mid r \mid K \mid w \times w'\rangle \quad \text{where } w' = \texttt{Dpdf}(r; r_1, \ldots, r_n) > 0$$

Theorem 7.4. *$\texttt{D}(v_1, \ldots, v_n)$ and $\texttt{Dinvcdf}(\texttt{sample}; v_1, \ldots, v_n)$ are CIU-equivalent (and thus contextually equivalent).*

Proof. By Theorem 6.15. Both expressions are real-valued. We must show that their real measures are equal. We abbreviate the parameters as $\vec{v}$. The result follows from the relationship between the density function and the cumulative density function.

$$\hat{\mu}(\texttt{Dinvcdf}(\texttt{sample}; \vec{v}), A) = \int_0^1 I_A(\texttt{Dinvcdf}(x; \vec{v}))\ dx$$

We change the variable of integration with $x = \texttt{Dcdf}(t; \vec{v})$ and $\frac{dx}{dt} = \texttt{Dpdf}(t; \vec{v})$:

$$\begin{aligned} &= \int_{-\infty}^{\infty} I_A(\texttt{Dinvcdf}(\texttt{Dcdf}(t;\vec{v});\vec{v})) \times \texttt{Dpdf}(t;\vec{v})\ dt \\ &= \int_{-\infty}^{\infty} I_A(t) \times \texttt{Dpdf}(t;\vec{v})\ dt \\ &= \hat{\mu}(\texttt{D}(\vec{v}), A) \end{aligned}$$

□

7.3 From let-style to direct-style

Let us call the language of Section 2.1 $\mathcal{L}$ (for “let”) and the direct-style analogue $\mathcal{D}$ (for “direct”). Once again following **?**, we give the semantics of $\mathcal{D}$ using a CS-style abstract machine, in contrast to the CSK-style machines we have used until now [**?**].

Here are the definitions of expressions and evaluation contexts for $\mathcal{D}$:

$$\begin{aligned} e &::= v \mid \texttt{let}\ x = e\ \texttt{in}\ e \mid \texttt{(}e\ e\texttt{)} \mid op\,\texttt{(}e, \ldots, e\texttt{)} \mid \texttt{if}\ e\ \texttt{then}\ e\ \texttt{else}\ e \\ E &::= [\,] \mid \texttt{let}\ x = E\ \texttt{in}\ e \mid \texttt{(}E\ e\texttt{)} \mid \texttt{(}v\ E\texttt{)} \mid op\texttt{(}v, \ldots, E, e, \ldots\texttt{)} \mid \texttt{if}\ E\ \texttt{then}\ e\ \texttt{else}\ e \end{aligned}$$

Here are some representative rules for its abstract machine:

$$\begin{aligned} \langle t \mid E[\texttt{((}\lambda x.e\texttt{)}\ v\texttt{)}] \mid w\rangle &\to_{\mathcal{D}} \langle t \mid E[e[v/x]] \mid w\rangle \\ \langle r{::}t \mid E[\texttt{sample}] \mid w\rangle &\to_{\mathcal{D}} \langle t \mid E[\texttt{c}_r] \mid w\rangle \end{aligned}$$

And here are the corresponding definitions of evaluation and measure:

$$\begin{aligned} \text{eval}_{\mathcal{D}}(t, e, w, A) &= \begin{cases} w' & \text{if } \langle t \mid e \mid w\rangle \to_{\mathcal{D}}^{*} \langle \epsilon \mid r \mid w'\rangle, \text{ where } r \in A \\ 0 & \text{otherwise} \end{cases} \\ \mu_{\mathcal{D}}(e, A) &= \int \text{eval}_{\mathcal{D}}(t, e, 1, A)\ \mu_{\mathbb{T}}(dt) \end{aligned}$$

To show that our $\mathcal{L}$ corresponds with $\mathcal{D}$, we define a translation $\text{tr}[\![-]\!] : \mathcal{D} \to \mathcal{L}$. More precisely, $\text{tr}[\![-]\!]$ translates $\mathcal{D}$-expressions to $\mathcal{L}$-expressions, such as

$$\begin{aligned} \text{tr}[\![r]\!] &= r \\ \text{tr}[\![\lambda x.e]\!] &= \lambda x.\text{tr}[\![e]\!] \\ \text{tr}[\![\texttt{(}e_1\ e_2\texttt{)}]\!] &= \texttt{let}\ x_1 = \text{tr}[\![e_1]\!]\ \texttt{in}\ \texttt{let}\ x_2 = \text{tr}[\![e_2]\!]\ \texttt{in}\ \texttt{(}x_1\ x_2\texttt{)} \\ \text{tr}[\![\texttt{let}\ x = e_1\ \texttt{in}\ e_2]\!] &= \texttt{let}\ x = \text{tr}[\![e_1]\!]\ \texttt{in}\ \text{tr}[\![e_2]\!] \end{aligned}$$

and it translates $\mathcal{D}$-evaluation contexts to $\mathcal{L}$-continuations, such as

$$\begin{aligned} \text{tr}[\![[\,]]\!] &= \texttt{halt} \\ \text{tr}[\![E[\texttt{(}[\,]\ e_2\texttt{)}]]\!] &= (x_1 \to \texttt{let}\ x_2 = e_2\ \texttt{in}\ \texttt{(}x_1\ x_2\texttt{)})\text{tr}[\![E]\!] \\ \text{tr}[\![E[\texttt{(}v_1\ [\,]\texttt{)}]]\!] &= (x_2 \to \texttt{(}v_1\ x_2\texttt{)})\text{tr}[\![E]\!] \\ \text{tr}[\![E[\texttt{let}\ x = [\,]\ \texttt{in}\ e]]\!] &= (x \to e)\text{tr}[\![E]\!] \end{aligned}$$

Now we demonstrate the correspondence of evaluation and then lift it to measures.

Lemma 7.5 (Simulation). $\text{eval}_{\mathcal{D}}(t, E[e], w, A) = \ddot{\text{eval}}(t, \text{tr}[\![e]\!], \text{tr}[\![E]\!], w, A)$

Proof. From the CS-style machine above we can derive a corresponding CSK machine (call it $\to_{\mathcal{D}\text{CSK}}$); the technique is standard [**?**]. Then it is straightforward to show that

$$\langle t \mid e \mid K \mid w\rangle \to_{\mathcal{D}\text{CSK}} \langle t' \mid e' \mid K' \mid w\rangle \implies \langle t \mid \text{tr}[\![e]\!] \mid \text{tr}[\![K]\!] \mid w\rangle \ddot{\to}^{*} \langle t' \mid \text{tr}[\![e']\!] \mid \text{tr}[\![K']\!] \mid w'\rangle$$

and thus the evaluators agree. □

THEOREM 7.6. $\mu_{\mathcal{D}}(E[e], A) = \mu(\text{tr}[\![e]\!], \text{tr}[\![E]\!], A)$

PROOF. By definition of $\mu_{\mathcal{D}}$ and Lemmas 7.3 and 7.5. □

The equational theory for $\mathcal{D}$ is the *pullback* of the $\mathcal{L}$ equational theory over $\text{tr}[\![-]\!]$. Compare with ?, which explores the pullback of $\lambda\beta\eta$ and related calculi over the call-by-value CPS transformation. For our language $\mathcal{D}$, associativity and commutativity combine to yield a generalization of their β_{flat} and β'_{Ω} equations to "single-evaluation" contexts S:

$$S ::= [\,] \mid (S\ e) \mid (e\ S) \mid \texttt{let}\ x = S\ \texttt{in}\ e \mid \texttt{let}\ x = e\ \texttt{in}\ S$$
$$\mid op\,(e, \ldots, S, e, \ldots) \mid \texttt{if}\ S\ \texttt{then}\ e\ \texttt{else}\ e$$

$$\texttt{let}\ x = e\ \texttt{in}\ S[x] \ =_{\text{ctx}}\ S[e] \qquad \text{when } x \notin FV(S) \cup FV(e) \qquad (\text{let}_S)$$

8 INFORMALLY RELATED WORK

The construction of our logical relation follows the tutorial of ? on the construction of biorthogonal, step-indexed [?] logical relations. Instead of termination, we use the program measure as the observable behavior, following ?. But unlike that work, where the meaning of an expression is a measure over arbitrary syntactic values, we define the meaning of an expression and continuation together (representing a whole program) as a measure over the reals. This allows us to avoid the complication of defining a relation on measurable sets of syntactic values [?, the $\mathcal{A}$ relation].

There has been previous work on contextual equivalence for probabilistic languages with only *discrete* random variables. In particular, ? define a step-indexed, biorthogonal logical relation whose structure is similar to ours, except that they sum where we integrate, and they use the probability of termination as the basic observation whereas we compare measures. Others have applied bisimulation techniques [??] to languages with discrete choice; ? have constructed fully abstract models for PCF with discrete probabilistic choice using probabilistic coherence spaces.

? gives a denotational semantics for a higher-order, typed language with continuous random variables, scoring, and normalization but without recursion. Using a variant of that denotational semantics, ? proves the soundness of the `let`-reordering transformation for a first-order language.

A ADDITIONAL PROOFS

A.1 From Section 2.3

PROOF OF 2.7. By induction on i. At $i = 1$, property (b) is true. So assume the proposition at i, and check it at $i + 1$. If property (a) holds for some $j \leq i$, then (a) holds at $i + 1$. Otherwise, assume that property (b) holds at i, that is, assume that $(K_i, \sigma_i) \geq (K_1, \sigma_1)$.

If e_i is not a value, then either $(K_{i+1}, \tau_{i+1}) = (K_i, \tau_i)$, or else $K_{i+1} = (x \to e)K_i$ and $\tau_{i+1} = \sigma' :: \tau_i$ for some x, e and σ', in which case $(K_{i+1}, \tau_{i+1}) \geq (K_1, \tau_1)$ by Rule 2 above. So the property holds at $i + 1$.

If e_i is a value, then consider the last step in the derivation of $(K_i, \tau_i) \geq (K_1, \tau_1)$. If the last step was Rule 1, then property (a) holds at i and therefore it holds at $i + 1$. If the last step was Rule 2, then $(K_i, \tau_i) = ((x \to e)K', \sigma' :: \tau')$ for some σ', where $(K', \tau') \geq (K_1, \tau_1)$. So the configuration at step i is a return from a `let`, and $(K_{i+1}, \tau_{i+1}) = (K', \tau') \geq (K_1, \sigma_1)$ by inversion on Rule 2, so again the property holds at $i + 1$. □

A.2 From Section 2.4

PROOF OF 2.13 (MEASURABILITY). The argument goes as follows. Following ?, Figure 5, turn the set of expressions and continuations into a metric space by setting $d(\mathsf{c}_r, \mathsf{c}_{r'}) = |r - r'|$; $d((e_1\ e_2), (e'_1\ e'_2) = d(e_1, e'_1) + d(e_2, e'_2)$, etc., setting $d(e, e') = \infty$ if e and e' are not the same up to constants. Extend this to become a measurable space on configurations by constructing the product space, using the Borel sets for the weights and the natural measurable space on the entropy components. Note that in this space, singletons are measurable sets.

The next-configuration function *next-state* : Config → Config is measurable; the proof follows the pattern of **?**, Lemmas 72–84. It follows that the n-fold composition of *next-state*, $\textit{next-state}^{(n)}$ is measurable, as is $\textit{finish} \circ \textit{next-state}^{(n)}$, where *finish* extracts the weight of halted configurations.

Now we consider the measurability of eval. Let B be a Borel set in the reals and set

$$\begin{aligned} C &= \{(\sigma, e, K, \tau, w) \mid \text{eval}(\sigma, e, K, \tau, w, A) \in B\} \\ &= \bigcup_n ((\textit{finish} \circ \textit{next-state}^{(n)})^{-1}(B)) \end{aligned}$$

Since C is equal the countable union of measurable sets, it is measurable, and thus eval is measurable with respect to the product space of all of its arguments. To show $\text{eval}(\sigma, e, K, \tau, w, A)$ is measurable with respect to (σ, τ) for any fixed e, K, w, and A, we note that

$$(\sigma, \tau \mapsto \text{eval}(\sigma, e, K, \tau, w, A)) = \text{eval} \circ (\sigma, \tau \mapsto (\sigma, e, K, \tau, w, A))$$

The function $(\sigma, \tau \mapsto (\sigma, e, K, \tau, w, A))$ is measurable, as it is just a product of constant and identity functions. Thus the composition is measurable. ◻

Proof for 2.15 (`sample`).

$$\begin{aligned} &\mu^{(p+1)}(\texttt{sample}, K, A) \\ &= \iint \text{eval}^{(p+1)}(\sigma, \texttt{sample}, K, \tau, 1, A)\ d\sigma\ d\tau \\ &= \iint \text{eval}^{(p)}(\pi_R\sigma, \texttt{c}_{\pi_U(\pi_L(\sigma))}, K, \tau, 1, A)\ d\sigma\ d\tau && \text{(Lemma 2.3)} \\ &= \iiint \text{eval}^{(p)}(\sigma_2, \texttt{c}_{\pi_U(\sigma_1)}, K, \tau, 1, A)\ d\sigma_1\ d\sigma_2\ d\tau && \text{(Property 2.1.4)} \\ &= \iiint \text{eval}^{(p)}(\sigma_2, \texttt{c}_{\pi_U(\sigma_1)}, K, \tau, 1, A)\ d\sigma_2\ d\tau\ d\sigma_1 && \text{(Lemma 2.2 twice)} \\ &= \int \mu^{(p)}(\texttt{c}_{\pi_U(\sigma_1)}, K, A)\ d\sigma_1 \\ &= \int_0^1 \mu^{(p)}(\texttt{c}_r, K, A)\ dr && \text{(Property 2.1.2)} \end{aligned}$$

◻

A.3 From Section 3

Proof of 3.1 (Variables). We must show that for all n and $(\gamma, \gamma') \in \mathbb{G}_n^\Gamma$, $(\gamma(x), \gamma'(x)) \in \mathbb{V}_n$. But that is true by the definition of $\mathbb{G}_n^\Gamma$. ◻

Proof of 3.1 (λ). Without loss of generality, assume $x \notin \Gamma$, and hence for any γ, $(\lambda x.e)\gamma = \lambda x.e\gamma$. We must show, for all n, if $(\gamma, \gamma') \in \mathbb{G}_n^\Gamma$, then $(\lambda x.e\gamma, \lambda x.e'\gamma') \in \mathbb{V}_n$.

Following the definition of $\mathbb{V}_n$, choose $m < n$ and $(v, v') \in \mathbb{V}_m$. We must show that $(e\gamma[v/x], e'\gamma'[v'/x]) \in \mathbb{E}_n$ Since $m < n$, we have $(\gamma, \gamma') \in \mathbb{G}_m^\Gamma$ and $(v, v') \in \mathbb{V}_m$. Therefore $(\gamma[v/x], \gamma'[v'/x]) \in \mathbb{G}_m^{\Gamma, x}$, so $(e\gamma[v/x], e'\gamma'[v'/x]) \in \mathbb{E}_m$ by the definition of $\mathbb{E}_m$. ◻

Proof of 3.1 (Value-compatibility implies expression-compatibility). Choose n and $(\gamma, \gamma') \in \mathbb{G}_n^\Gamma$, so we have $(v\gamma, v'\gamma') \in \mathbb{V}_n$. We must show that $(v\gamma, v'\gamma') \in \mathbb{E}_n$.

Following the definition of $\mathbb{E}_n$, choose $m \le n$, $(K, K') \in \mathbb{K}_m$, and A. Since $m \le n$, we have $(v\gamma, v'\gamma') \in \mathbb{V}_m$, so $\mu^{(m)}(v, K, A) \le \mu(v', K', A)$. Since we have this for all $m \le n$, we conclude that $(v\gamma, v'\gamma') \in \mathbb{E}_n$. ◻

Proof of 3.1 (application). Choose n, and assume $(\gamma, \gamma') \in \mathbb{G}_n^\Gamma$. Then $(v_1\gamma, v_1'\gamma') \in \mathbb{V}_n$ and $(v_2\gamma, v_2'\gamma') \in \mathbb{V}_n$. We must show $((v_1\gamma\ \ v_2\gamma), (v_1'\gamma'\ v_2'\gamma')) \in \mathbb{E}_n$

If $v_1\gamma$ is of the form c_r, then $\mu^{(m)}(v_1\gamma, K, A) = 0$ for any m, K, and A, so by Lemma 2.14 the conclusion holds.

Otherwise, assume $v_1\gamma$ is of the form $\lambda x.e$, and so $v_1'\gamma'$ is of the form $\lambda x.e'$. So choose $m \leq n$ and A, and let $(K, K') \in \mathbb{K}_m$. We must show that

$$\mu^{(m)}((\lambda x.e\gamma\ v_2\gamma), K, A) \leq \mu((\lambda x.e'\gamma'\ v_2'\gamma'), K', A).$$

If $m = 0$ the left-hand side is 0. So assume $m \geq 1$. Since all the relevant terms are closed and the relations on closed terms are antimonotonic in the index, we have $(\lambda x.e\gamma, \lambda x.e'\gamma') \in \mathbb{V}_m$ and $(v_1\gamma, v_1'\gamma') \in \mathbb{V}_{m-1}$. Therefore $(e\gamma[v_2\gamma/x], e'\gamma'[v_2'\gamma'/x]) \in \mathbb{E}_{m-1}$.

Now, $\langle \sigma \mid (\lambda x.e\gamma\ \ v_2\gamma) \mid K \mid \tau \mid w\rangle \to \langle \sigma \mid e\gamma[v_2\gamma/x] \mid K \mid \tau \mid w\rangle$, and similarly for the primed side. So we have

$$\begin{aligned}
\mu^{(m)}((\lambda x.e\gamma\ \ v_2\gamma), K, A) &= \mu^{(m-1)}(e\gamma[v_2\gamma/x], K, A) && \text{(Lemma 2.15)}\\
&\leq \mu(e'\gamma'[v_2'\gamma'/x], K', A) && \text{(by } (e\gamma[v_2\gamma/x], e'\gamma'[v_2'\gamma'/x]) \in \mathbb{E}_{m-1})\\
&= \mu((\lambda x.e'\gamma'\ \ v_2'\gamma'), K', A)
\end{aligned}$$

□

Proof of 3.1 (operations). Choose n, $(\gamma, \gamma') \in \mathbb{G}_n^\Gamma$, $m \leq n$, $(K, K') \in \mathbb{K}_m$, and A.

Since the arguments are related, for each i either $v_i\gamma$ and $v_i'\gamma'$ are the same real number c_{r_i} or both are closures. So either

$$\delta(op^k, v_1\gamma, \ldots, v_k\gamma) = \delta(op^k, v_1'\gamma', \ldots, v_k'\gamma') = \mathsf{c}_r$$

or both are undefined, in which case both measures are 0. Assuming the result is defined and $m > 0$,

$$\begin{aligned}
&\mu^{(m)}(op^k\ (v_1\gamma, \ldots, v_k\gamma), K, A)\\
&= \mu^{(m-1)}(\mathsf{c}_r, K, A) && \text{(Lemma 2.15)}\\
&\leq \mu(\mathsf{c}_r, K', A) && \text{(by definition of } \mathbb{K}_m)\\
&= \mu(op^k\ (v_1'\gamma', \ldots, v_1'\gamma'), K', A)
\end{aligned}$$

□

Proof of 3.1 (`halt`). We must show that for any m and any $(v, v') \in \mathbb{V}_n$, $\mu^{(m)}(v, \mathtt{halt}, A) \leq \mu(v', \mathtt{halt}, A)$. But $(v, v') \in \mathbb{V}_n$ implies either $v = v' = \mathsf{c}_r$ or both v and v' are λ-expressions, in which case both sides of the inequality are 0. □

Proof of 3.1 (continuations). Choose n, $m \leq n$, $(v, v') \in \mathbb{V}_m$, and $A \in \Sigma_\mathbb{R}$. We need to show $\mu^{(m)}(v, (x \to e)K, A) \leq \mu(v', (x \to e')K', A)$. Assume $m > 0$, otherwise trivial.

By Lemma 2.15, the left-hand side is $\mu^{(m-1)}(e[v/x], K, A)$ and the right-hand side is $\mu(e'[v'/x], K, A)$. The inequality follows from $(e, e') \in \mathbb{E}^{\{x\}}$. □

To establish compatibility for `let`, we need some finer information:

Lemma A.1. *Given n, and e, e' with a single free variable x, with the property that*

$$(\forall p \leq n)(\forall v, v')[(v, v') \in \mathbb{V}_p \implies (e[v/x], e'[v'/x]) \in \mathbb{E}_p]$$

Then for all $m \leq n$,

$$(K, K') \in \mathbb{K}_m \implies ((x \to e)K, (x \to e')K') \in \mathbb{K}_m$$

Proof. Choose $m \leq n$ and $(K, K') \in \mathbb{K}_m$. To show $((x \to e)K, (x \to e')K') \in \mathbb{K}_m$, choose $p \leq m$, $(v, v') \in \mathbb{V}_p$, and A.

We must show $\mu^{(p)}(v, (x \to e)K, A) \leq \mu(v', (x \to e')K', A)$.

If $p = 0$, the result is trivial. So assume $p > 0$ and calculate:

$$\begin{aligned} & \mu^{(p)}(v, (x \to e)K, A) \\ & = \mu^{(p-1)}(e[v/x], K, A) && \text{(Lemma 2.15)} \\ & \leq \mu(e'[v'/x], K', A) \\ & = \mu(v', (x \to e')K', A) \end{aligned}$$

where the inequality follows from $(v, v') \in \mathbb{V}_p \subseteq \mathbb{V}_{p-1}$ and $(K, K') \in \mathbb{K}_m \subseteq \mathbb{K}_p \subseteq \mathbb{K}_{p-1}$. □

Now we can prove compatibility under `let`.

Proof of 3.1 (`let`). Choose n and $(\gamma, \gamma') \in \mathbb{G}_n^\Gamma$. So we have $(e_1\gamma, e_1'\gamma') \in \mathbb{E}_m$ for all $m \leq n$.

Furthermore, if $m \leq n$ and $(v, v') \in \mathbb{V}_m$, then $(\gamma[x := v], \gamma'[x := v']) \in \mathbb{G}^m_{\Gamma,x}$. Therefore $(e_2\gamma[x := v], e_2'\gamma'[x := v']) \in \mathbb{E}_m$. So $(e_2\gamma, e_2'\gamma')$ satisfies the hypothesis of Lemma A.1.

So choose $m \leq n$ and $(K, K') \in \mathbb{K}_m$. Without loss of generality, assume $m > 0$. Then by Lemma A.1 we have

$$((x \to e\gamma)K, (x \to e'\gamma')K') \in \mathbb{K}_m \tag{3}$$

Choose A. Now we can calculate:

$$\begin{aligned} \mu^{(m)}(\mathtt{let}\, x = e_1\gamma \ \mathtt{in}\ e_2\gamma, K, A) &= \mu^{(m-1)}(e_1\gamma, (x \to e_2\gamma)K, A) && \text{(Lemma 2.15)} \\ &\leq \mu(e_1'\gamma', (x \to e_2'\gamma')K', A) \\ &= \mu(\mathtt{let}\, x = e_1'\gamma' \ \mathtt{in}\ e_2\gamma', K, A) \end{aligned}$$

where the inequality follows from $(e_1\gamma, e_1'\gamma') \in \mathbb{E}_m$ and (3). □

Proof of 3.1 (`if`). Choose n, $(\gamma, \gamma') \in \mathbb{G}_n^\Gamma$, $m \leq n$, $(K, K') \in \mathbb{K}_m$, and $A \in \Sigma_\mathbb{R}$. Assume that $m > 0$, otherwise the result follows trivially.

Suppose $v\gamma = v'\gamma' = \mathsf{c}_r$, and if $r > 0$. Then

$$\begin{aligned} \mu^{(m)}(\mathtt{if}\ v\gamma\ \mathtt{then}\ e_1\gamma\ \mathtt{else}\ e_2\gamma, K, A) &= \mu^{(m-1)}(e_1\gamma, K, A) && \text{(Lemma 2.15)} \\ &\leq \mu(e_1'\gamma, K', A) \\ &= \mu(\mathtt{if}\ v'\gamma'\ \mathtt{then}\ e_1'\gamma'\ \mathtt{else}\ e_2'\gamma', K', A) \end{aligned}$$

Likewise for $r \leq 0$ and e_2, e_2'.

Otherwise, neither $v\gamma$ nor $v'\gamma'$ is a real constant, and both expressions are stuck and have measure 0. □

Everything so far is just an adaptation of the deterministic case. Now we consider our two effects.

Proof of 3.1 (`factor`). Choose n and $(\gamma, \gamma') \in \mathbb{G}^n_\Gamma$. We must show $(\mathtt{factor}\ v\gamma, \mathtt{factor}\ v'\gamma') \in \mathbb{E}_n$. Since $(v, v') \in \mathbb{V}^\Gamma$, it must be that $(v\gamma, v'\gamma') \in \mathbb{V}_n$. So either $v\gamma = v'\gamma' = \mathsf{c}_r$ for some r, or $v\gamma$ is a lambda-expression.

Assume $v\gamma = v'\gamma' = \mathsf{c}_r$ for some $r > 0$. Choose $1 \le m \le n$, $(K, K') \in \mathbb{K}_m$, and $A \in \Sigma_{\mathbb{R}}$. Then we have

$$\begin{aligned} \mu^{(m)}(\mathtt{factor}\ \mathsf{c}_r, K, A) &= r \times \mu^{(m-1)}(\mathsf{c}_r, K, A) && \text{(Lemma 2.15)} \\ &\le r \times \mu(\mathsf{c}_r, K', A) \\ &= \mu(\mathtt{factor}\ \mathsf{c}_r, K', A) \end{aligned}$$

So $(\mathtt{factor}\ \mathsf{c}_r, \mathtt{factor}\ \mathsf{c}_r) \in \mathbb{E}_n$ as desired.

If $v\gamma$ is c_r for $r \le 0$ or a lambda-expression, then $\mathtt{factor}\ v\gamma$ is stuck, so $\mu^{(m)}(\mathtt{factor}\ \mathsf{c}_r, K, A) = 0$ and the desired result holds again. □

Proof of 3.1 (sample). It will suffice to show that for all m, $(K, K') \in \mathbb{K}_m$, and $A \in \Sigma_{\mathbb{R}}$,

$$\mu^{(m)}(\mathtt{sample}, K, A) \le \mu(\mathtt{sample}, K', A)$$

At $m = 0$, the left-hand side is 0 and the result is trivial. If $m > 0$, then

$$\begin{aligned} \mu^{(m)}(\mathtt{sample}, K, A) &= \int \mu^{(m-1)}(\mathsf{c}_{\pi_U(\sigma)}, K, A)\ d\sigma && \text{(Lemma 2.15)} \\ &\le \int \mu(\mathsf{c}_{\pi_U(\sigma)}, K', A)\ d\sigma \\ &= \mu(\mathtt{sample}, K', A) \end{aligned}$$

□

A.4 From Section 5

Proof of 5.2. Let K_1 denote $(z \to (z\ v))K$, and let K_1' denote $(z \to (z\ v'))K'$. To show $(K_1, K_1') \in \mathbb{K}_{n+2}$, choose $2 \le m \le n+2$, $(u, u') \in \mathbb{V}_m$, and $A \in \Sigma_{\mathbb{R}}$. We must show

$$\mu^{(m)}(u, K_1, A) \le \mu(u', K_1', A)$$

There are two possibilities for $(u, u') \in \mathbb{V}_m$:

1. $u = u' = \mathsf{c}_r$. Then $\langle \sigma \mid \mathsf{c}_r \mid K_1 \mid \tau \mid w \rangle \to \langle \sigma \mid (\mathsf{c}_r\ v) \mid K \mid \tau \mid w \rangle$, which is stuck, so $\mu^{(m)}(u, K_1, A) = 0 \le \mu(u', K_1', A)$.

2. $u = \lambda x.e$ and $u' = \lambda x.e'$ where for all $p < m$ and all $(u_1, u_1') \in \mathbb{V}_p$, $(e[u_1/x], e'[u_1'/x]) \in \mathbb{E}_p$.

Now, for any σ and w, we have

$$\begin{aligned} &\langle \sigma \mid \lambda x.e \mid K_1 \mid \tau \mid w \rangle \\ &\to \langle \sigma \mid (\lambda x.e\ v) \mid K \mid \tau \mid w \rangle \\ &\to \langle \sigma \mid e[v/x] \mid K \mid \tau \mid w \rangle \end{aligned}$$

so $\mu^{(m)}(\lambda x.e, K_1, A) = \mu^{(m-2)}(e[v/x], K, A)$, and similarly on the primed side (but with $\mu(-,-,-)$ in place of $\mu^{(m)}(-,-,-)$ and with equality in place of $\le$).

Next, observe $m - 2 \le n$, so $(v, v') \in \mathbb{V}_{m-2}$ and hence $(e[v/x], e'[v'/x]) \in \mathbb{E}_{m-2}$ by the property of e and e' above. Therefore, $\mu^{(m-2)}(e[v/x], K, A) \le \mu(e'[v'/x], K', A)$.

Putting the pieces together, we have

$$\begin{aligned} &\mu^{(m)}(\lambda x.e, K_1, A) \\ &= \mu^{(m-2)}(e[v/x], K, A) \\ &\le \mu(e'[v'/x], K', A) \\ &= \mu(\lambda x.e', K_1', A) \end{aligned}$$

□

A.5 From Section 6.2

Proof of 6.5. By Tonelli's Theorem (Lemma 2.2) and the fact that g is arbitrary we can freely rearrange the parameters to g without loss of generality. In particular, all paths ending with L are assumed to come before any paths ending with R.

Let l be the length of the longest path or 0 if $n = 0$. We proceed by strong induction on l.

- case $l = 0$. For every i, we know that p_i must be the empty list. Since we also know that the paths are non-duplicating it follows that $n \leq 1$. If $n = 1$ then the equality holds trivially, and if $n = 0$ then by Property 2.1(1) we get
$$\int g()\ d\sigma = g()$$
- case $l > 0$. Since at least one path is not the empty list, it follows from non-duplication that no path is the empty list. For each $i \in \{1, \ldots, n\}$, let q_i be p_i with the last direction removed. Assume without loss of generality that $p_1 \ldots p_k$ end with L and $p_{k+1} \ldots p_n$ end with R, so $p_i(\sigma) = q_i(\pi_L(\sigma))$ for $1 \leq i \leq k$ and and $p_i(\sigma) = q_i(\pi_R(\sigma))$ for $k + 1 \leq i \leq n$. Then we can conclude:
$$\begin{aligned}
&\int g(p_1(\sigma), \ldots, p_n(\sigma))\ d\sigma \\
&= \int g(q_1(\pi_L(\sigma)), \ldots, q_k(\pi_L(\sigma)), q_{k+1}(\pi_R(\sigma)), \ldots, q_n(\pi_R(\sigma)))\ d\sigma \\
&= \iint g(q_1(\sigma'), \ldots, q_k(\sigma'), q_{k+1}(\sigma''), \ldots, q_n(\sigma''))\ d\sigma'\ d\sigma'' && \text{(Property 2.1(4))}
\end{aligned}$$
Since every q_i is strictly shorter than p_i, we can apply the induction hypothesis, first at σ' and then at σ''.
$$\begin{aligned}
&= \int \left(\int \ldots \int g(\sigma_1, \ldots, \sigma_k, q_{k+1}(\sigma''), \ldots, q_n(\sigma''))\ d\sigma_1 \ldots\ d\sigma_k\right) d\sigma'' && \text{(IH)} \\
&= \int \ldots \int g(\sigma_1, \ldots, \sigma_n)\ d\sigma_1 \ldots\ d\sigma_n && \text{(IH)}
\end{aligned}$$

□

A.6 From Section 6.3

Proof of 6.9 using Small-Step Semantics. Let e and e' denote the two expressions of the theorem. We will use Theorem 6.7. To do so, we will consider the evaluations of e and e'. For each evaluation, we will use the Interpolation Theorem to define waypoints in the evaluation. We then use the Genericity Theorem to establish that the ending configurations are the same.

We begin by watching the first expression evaluate in an arbitrary continuation K, saved entropy τ, and initial weight w:

$$\begin{aligned}
& \langle \sigma \mid \texttt{let}\, x_1 = e_1\ \texttt{in}\ \texttt{let}\, x_2 = e_2\ \texttt{in}\ e_0 \mid K \mid \tau \mid w \rangle \\
\to\ & \langle \pi_L(\sigma) \mid e_1 \mid (x_1 \to \texttt{let}\, x_2 = e_2\ \texttt{in}\ e_0)K \mid \pi_R(\sigma)::\tau \mid w \rangle \\
\to^*\ & \langle \sigma_1 \mid v_1 \mid (x_1 \to \texttt{let}\, x_2 = e_2\ \texttt{in}\ e_0)K \mid \pi_R(\sigma)::\tau \mid w_1 \times w \rangle && \text{(Interpolation)} \\
\to\ & \langle \pi_R(\sigma) \mid \texttt{let}\, x_2 = e_2\ \texttt{in}\ e_0[v_1/x_1] \mid K \mid \tau \mid w_1 \times w \rangle \\
\to\ & \langle \pi_L(\pi_R(\sigma)) \mid e_2 \mid (x_2 \to e_0[v_1/x_1])K \mid \pi_R(\pi_R(\sigma))::\tau \mid w_1 \times w \rangle \\
\to^*\ & \langle \sigma_2 \mid v_2 \mid (x_2 \to e_0[v_1/x_1])K \mid \pi_R(\pi_R(\sigma))::\tau \mid w_2 \times w_1 \times w \rangle && \text{(Interpolation)} \\
\to\ & \langle \pi_R(\pi_R(\sigma)) \mid e_0[v_1/x_1][v_2/x_2] \mid K \mid \tau \mid w_2 \times w_1 \times w \rangle
\end{aligned}$$

Next, we outline the analogous computation with the second expression e', starting in a different entropy σ', but with the same continuation K, saved entropy τ, and weight w. We proceed under the assumption that e' reduces to a value; we will validate this assumption later.

$$\begin{array}{lll} & \langle \sigma' \mid \texttt{let}\, x_2 = e_2 \texttt{ in let}\, x_1 = e_1 \texttt{ in } e_0 \mid K \mid \tau \mid w \rangle & \\ \to & \langle \pi_L(\sigma') \mid e_2 \mid (x_2 \to \texttt{let}\, x_1 = e_1 \texttt{ in } e_0)K \mid \pi_R(\sigma')::\tau \mid w \rangle & \\ \to^* & \langle \sigma_2' \mid v_2' \mid (x_2 \to \texttt{let}\, x_1 = e_1 \texttt{ in } e_0)K \mid \pi_R(\sigma')::\tau \mid w_2' \times w \rangle & \text{(Interpolation)} \\ \to & \langle \pi_R(\sigma') \mid \texttt{let}\, x_1 = e_1 \texttt{ in } e_0[v_2'/x_2] \mid K \mid \tau \mid w_2' \times w \rangle & \\ \to & \langle \pi_L(\pi_R(\sigma')) \mid e_1 \mid (x_1 \to e_0[v_2'/x_2])K \mid \pi_R(\pi_R(\sigma'))::\tau \mid w_2' \times w \rangle & \\ \to^* & \langle \sigma_1' \mid v_1' \mid (x_1 \to e_0[v_2'/x_2])K \mid \pi_R(\pi_R(\sigma'))::\tau \mid w_1' \times w_2' \times w \rangle & \text{(Interpolation)} \\ \to & \langle \pi_R(\pi_R(\sigma')) \mid e_0[v_2'/x_2][v_1'/x_1] \mid K \mid \tau \mid w_1' \times w_2' \times w \rangle & \end{array}$$

To get these computations to agree, we choose σ' so that the entropies for e_1, e_2 and the substitution instances of e_0 are the same in both calculations. So we choose σ' such that

$$\begin{array}{lll} \pi_L(\pi_R(\sigma')) = \pi_L(\sigma) & & \text{(entropy for } e_1) \\ \pi_L(\sigma') = \pi_L(\pi_R(\sigma)) & & \text{(entropy for } e_2) \\ \pi_R(\pi_R(\sigma')) = \pi_R(\pi_R(\sigma)) & & \text{(entropy for } e_0) \end{array}$$

This can be accomplished by using ϕ_c from Section 6.2; we set

$$\sigma' = \phi_c(\sigma) = \pi_L(\pi_R(\sigma))::(\pi_L(\sigma)::\pi_R(\pi_R(\sigma)))$$

Applying the genericity theorem at e_1 we conclude that that e_1 reduces to a value at entropy $\pi_L(\pi_R\sigma')) = \pi_L(\sigma)$ and continuation $(x_1 \to e_0[v_2'/x_2])K$, that $v_1 = v_1'$, and that $w_1 = w_1'$. Similarly applying the genericity theorem at e_2 we conclude that e_2 reduces to a value $v_2' = v_2$ and $w_2 = w_2'$. So the two calculations culminate in identical configurations.

So we conclude that

$$\text{eval}(\sigma, e, K, \tau, 1, A) = \text{eval}(\phi_c(\sigma), e', K, \tau, 1, A)$$

ϕ_c is a non-duplicating FSF, so it is measure-preserving by Theorem 6.6. Then by Theorem 6.7, $(e, e') \in \mathbb{CTX}$. The converse holds by symmetry. □

B EXAMPLE TRANSFORMATION SKETCH

This section gives a sketch of the transformation of the linear regression example from Section 1.

Here is the initial program:

```
A = normal(0, 10)
B = normal(0, 10)
f(x) = A*x + B
factor normalpdf(f(2) - 2.4; 0, 1)
factor normalpdf(f(3) - 2.7; 0, 1)
factor normalpdf(f(4) - 3.0; 0, 1)
```

Our first goal is to reshape the program to expose the conjugacy relationship between B's prior and the observations. Specifically, we need the observations to be expressed with B as the mean. The following steps perform the reshaping.

Inline f using let_v:

```
A = normal(0, 10)
B = normal(0, 10)
factor normalpdf((A*2 + B) - 2.4; 0, 1)
factor normalpdf((A*3 + B) - 2.7; 0, 1)
factor normalpdf((A*4 + B) - 3.0; 0, 1)
```

Rewrite using $((y+x)-z) = (x-(z-y))$ three times. (This combines associativity and commutativity of + as well as other facts relating + and −.)

```
A = normal(0, 10)
B = normal(0, 10)
factor normalpdf(B - (2.4 - A*2); 0, 1)
factor normalpdf(B - (2.7 - A*3); 0, 1)
factor normalpdf(B - (3.0 - A*4); 0, 1)
```

Rewrite using $\texttt{normalpdf}(x - y; 0, s) = \texttt{normalpdf}(y; x, s)$ three times.

```
A = normal(0, 10)
B = normal(0, 10)
factor normalpdf(2.4 - A*2; B, 1)
factor normalpdf(2.7 - A*3; B, 1)
factor normalpdf(3.0 - A*4; B, 1)
```

Now the conjugacy relationship is explicit. The equation from Section 6.4 gives a closed-form solution to the posterior with respect to a single observation, and it applies to a specific shape of expression. Our new goal is to reshape the program so the conjugacy transformation can be applied to the first observation.

First, apply let_v in reverse, *after* the first occurrence of B:

```
A = normal(0, 10)
B = normal(0, 10)
factor normalpdf(2.4 - A*2; B, 1)
B1 = B
factor normalpdf(2.7 - A*3; B1, 1)
factor normalpdf(3.0 - A*4; B1, 1)
```

Use let_S to pull out the argument to the first normalpdf:

```
A = normal(0, 10)
err1 = 2.4 - A*2
B = normal(0, 10)
factor normalpdf(err1; B, 1)
B1 = B
factor normalpdf(2.7 - A*3; B1, 1)
factor normalpdf(3.0 - A*4; B1, 1)
```

Apply let-associativity twice (viewing the factor statement as an implicit let with an unused variable).

```
A = normal(0, 10)
err1 = 2.4 - A*2
B1 = { B = normal(0, 10)
       factor normalpdf(err1; B, 1)
       B }
factor normalpdf(2.7 - A*3; B1, 1)
```

```
factor normalpdf(3.0 - A*4; B1, 1)
```

Now the right-hand side of the B1 binding is in the right shape. We apply the conjugacy rule from Section 6.4:

```
A = normal(0, 10)
err1 = 2.4 - A*2
B1 = { B = normal( ( 1/10^2 + 1/1^2 )^(−1) ( 0/10^2 + err1/1^2 ) , ( 1/10^2 + 1/1^2 )^(−1/2) )
       factor normalpdf(err1; 0, (10^2 + 1^2)^(1/2) )
       B }
factor normalpdf(2.7 - A*3; B1, 1)
factor normalpdf(3.0 - A*4; B1, 1)
```

We have processed one observation. Now we must clean up and reset the program so the remaining observations can be processed. (One of the properties of conjugacy is that the posterior has the same form as the prior, so observations can be absorbed incrementally.)

We use let_S to give names to the new mean and scale parameters for B, let-associativity to ungroup the results, and let_v to eliminate the B1 binding. Finally, we use commutativity to move the new `factor` expression up and out of the way.

```
A = normal(0, 10)
err1 = 2.4 - A*2
factor normalpdf(err1; 0, (10^2 + 1^2)^(1/2) )
m1 = ( 1/10^2 + 1/1^2 )^(−1) ( 0/10^2 + err1/1^2 )
s1 = ( 1/10^2 + 1/1^2 )^(−1/2)
B = normal(m1, s1)
factor normalpdf(2.7 - A*3; B, 1)
factor normalpdf(3.0 - A*4; B, 1)
```

That completes the processing of the first observation. In its place we have B drawn from the posterior distribution (with respect to that observation) and a `factor` expression to score A independent of the choice of B. We can now repeat the process for the remaining observations.

Alternatively, we could have imported a transformation that used the closed-form formula for the posterior and normalizer given multiple observations. That would have led to different shaping steps.