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

Mitchell Wand[1], Theophilos Giannakopoulos[2], Andrew Cobb[1], and Ryan Culpepper[1]

[1] Northeastern University
[2] BAE Systems

Abstract. We present a complete reasoning principle for contextual equivalence in an untyped probabilistic programming language. The language includes continuous random variables, conditionals, and scoring. The language also includes recursion, since in an untyped language the standard call-by-value fixpoint combinator is expressible. The language is similar to that of Borgström et al. [5].
To demonstrate the usability of our characterization, we use it to prove that reordering the draws in a probabilistic program preserves contextual equivalence. This allows us to show, for example, that

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

(provided x is not among the free variables of e_2 and y is not among the free variables of e_1) despite the fact that e_1 and e_2 may have the effect of drawing from the source of entropy.

1 Introduction

Our goal is to give a useful characterization of contextual equivalence for a non-trivial probabilistic programming language. By "probabilistic programming language", we mean a language intended to represent probability distributions, or measures in general. By non-trivial, we mean a language that supports

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

Such languages include Church [9], its descendants such as Venture [11] and Anglican [20], and others [10, 12, 13].

* 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.

By contextual equivalence, we mean the usual notion: two expressions are to be contextually equivalent if there is no context that gives them different observable behavior. Contextual equivalence is important because it is the gold standard of equivalence: if a compiler transforms a term into a contextually equivalent term, then it is always safe to do the transformation.

By useful, we mean that the characterization can be used to prove the correctness of some non-trivial equivalence. Our motivating example is

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

(provided x is not among the free variables of e_2 and y is not among the free variables of e_1). This equivalence, while valid for a pure language, is certainly not valid for all effects (imagine, for example, that there is an assignment statement in e_1 or e_2).

While there has been much work on the operational (and sometimes denotational) semantics of individual terms in probabilistic programming languages, there has been less work on contextual equivalence. Bizjak and Birkedal [2] characterize contextual equivalence for a language with a powerful type system, but with only discrete choice. Culpepper and Cobb [7] characterize contextual equivalence for a language with continuous random variables and scoring, but with only simple types (no recursion). Staton [17] shows the equivalence above using denotational semantics, but only for a non-recursive first-order language.

We begin in section 2 by defining an *untyped* language, similar to that of Borgström et al. [5]. Because it is an untyped language, it can express the usual call-by-value fixpoint operator. Section 3 describes our axiomatic model of entropy; this is different from the model in Borgström et al. [5].

In section 4, we give this language a conventional small-step semantics that describes the behavior of a program with a given value for the entropy, and prove a variety of easy properties. We then prove an Interpolation Theorem (Theorem 7) and a Genericity Theorem (Theorem 8). The Interpolation Theorem says that 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, and the Genericity Theorem says that the value v is independent of K. The proof given here is novel (at least to us), and may be useful in other contexts. These theorems essentially allow us to use big-step reasoning in our small-step semantics (see also Wand and Clinger [19, Theorem 17]).

Section 5 describes the observable behavior of programs in terms of *measures* obtained by integrating the behavior of a program over all choices of the entropy.

In section 6, we define a step-indexed logical relation on values, expressions, and continuations, and we prove the Fundamental Property for our relation. In sections 7 and 8, we show that the logical relation and its close cousin the CIU relation are sound and complete for contextual equivalence. This reaches our first goal, that of a useful characterization of contextual equivalence, in the form of the CIU theorem (Theorem 45). The presentation here closely follows that of Pitts [14], modified to take account of our notion of observable behavior.

We then apply these results to the claimed commutativity result. In section 9, we characterize a useful set of transformations on the entropy space that

are guaranteed to be measure-preserving, and in section 10 we use these transformations, along with the Interpolation and Genericity Theorems, to prove the commutativity result.

We conclude with related work and open problems.

2 Syntax

The syntax of our language is given in figure 1.

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

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

We have a constant $\mathtt{c}_r$ for each real number r. `sample` draws from a uniform distribution on $[0, 1]$, and `factor` v weights (or "scores") the current execution by the value v. For simplicity, we require sequencing to be made explicit using `let`.

The language is untyped, but we express the scoping relations by rules like typing rules. We write $\Gamma \vdash e\ \mathtt{exp}$ 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.

3 Modeling Entropy

The semantics uses an *entropy* component as the source of randomness. Following Culpepper and Cobb [7] 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 $\mu_{\mathbb{S}}$.

We further assume that $\mathbb{S}$ has the following properties:

Property 1 (Properties of Entropy).

1. $\mu_{\mathbb{S}}(\mathbb{S}) = 1$
2. It comes with a uniform sampler, that 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.

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

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

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

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

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

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

3. It comes with 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 (Tonelli). *Let $f : \mathbb{S} \times \mathbb{S} \to \mathbb{R}^+$ be measurable. Then*

$$\int \left(\int f(\sigma_1, \sigma_2)\, d\sigma_1 \right) d\sigma_2 = \int \left(\int f(\sigma_1, \sigma_2)\, d\sigma_2 \right) d\sigma_1$$

4 Operational Semantics

We give this language 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)

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

Fig. 3. Small-step operational semantics

The rules for the semantics are given in figure 3.

The semantics uses continuations for sequencing and substitutions for procedure calls. Since $\texttt{let}\, x = e_1\ \texttt{in}\ e_2$ 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 3.[3] The return rule can be written using explicit projections as follows:

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

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. We define *evaluation* as taking an extra parameter, a measurable set of reals. Evaluation produces a positive weight only if the result value is in the expected set.

[3] We defer the explanation of the initial entropy stack to section 5.

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

We will also need approximants to eval:

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

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

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

Lemma 4 (weights are Linear).

1. $$\langle \sigma \mid e \mid K \mid \tau \mid 1 \rangle \to^* \langle \sigma' \mid e' \mid K' \mid \tau' \mid w' \rangle,$$
   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. *For all* $w > 0$,
   $$\mathrm{eval}(\sigma, e, K, \tau, w, A) = w \times \mathrm{eval}(\sigma, e, K, \tau, 1, A),$$
   and similarly for $\mathrm{eval}^{(n)}$.

The next result is an interpolation theorem, which says that 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.

Definition 5. *Define* $\succeq$ *to be the smallest relation defined by the following rules:*

$$\textsc{Rule 1:}\quad (K, \tau) \succeq (K, \tau) \qquad \textsc{Rule 2:}\quad \frac{(K', \tau') \succeq (K, \tau)}{((x \to e)K', \sigma{::}\tau') \succeq (K, \tau)}$$

Lemma 6. *Let*

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

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. $(K_i, \tau_i) \succeq (K_1, \tau_1)$

Proof (Sketch): By induction on i and inspection of the reduction rules—in particular, whether they extend, contract, or preserve the continuation and entropy stack, and whether they produce a value.

Theorem 7 (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',

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

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

Notice that both the lemma and the theorem would be false if our language contained jumping control structures like `call/cc`.

Theorem 8 (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.

This theorem tells us that in the interpolation theorem, σ', v, and w' are independent of K.

5 From Evaluations to Measures

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

Definition 9. *The* measure *of e and K at A, $\mu(e, K, A)$ is defined as*

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

More precisely, for any e and K, the mapping $A \mapsto \mu(e, K, A)$ is a measure. It is similar to both Culpepper and Cobb's $\mu_e(A)$ and Borgström et al.'s $[\![e]\!]_{\mathbb{S}}(A)$, but whereas they define measures on arbitrary syntactic values, our $\mu(e, K, -)$ is a measure on the reals.

Encoding entropy stacks as entropy values enables the simple definition above; 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 $\mathbb{S} \ncong \mathbb{S}^0$, 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:

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

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

Lemma 10 (eval is measurable). *For any e, K, $w \geq 0$, $A \in \Sigma_{\mathbb{R}}$, and n,* $\mathrm{eval}(\sigma, e, K, \tau, w, A)$ *and* $\mathrm{eval}^{(n)}(\sigma, e, K, \tau, w, A)$ *are measurable in σ and τ.*

Proof (Sketch): The argument goes as follows. Following Borgström et al. [5, Figure 5], turn the set of expressions and continuations into a metric space by setting $d(\mathtt{c}_r, \mathtt{c}_{r'}) = |r - r'|$; $d(\mathtt{(}e_1\ e_2\mathtt{)}, \mathtt{(}e_1'\ e_2'\mathtt{)}) = d(e_1, e_1') + d(e_2, e_2')$, 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 $\to$ Config is measurable; the proof follows the pattern of Borgström et al. [4, Lemmas 72–84]. It follows that the n-fold composition of *next-state*, *next-state*$^{(n)}$ is measurable, as is *finish* $\circ$ *next-state*$^{(n)}$, where *finish* extracts the weight of halted configurations.

Let B be a Borel set in the reals and set

$$C = \{(\sigma, e, K, \tau, w) \mid \mathrm{eval}(\sigma, e, K, \tau, w, A) \in B\}$$

C is measurable, since it is equal to the countable union

$$\bigcup_n ((\mathit{finish} \circ \mathit{next\text{-}state}^{(n)})^{-1}(B))$$

Thus eval is measurable with respect to the product space of all of its arguments.

To show $\mathrm{eval}(\sigma, e, K, \tau, w, A)$ is measurable with respect to (σ, τ) for any fixed e, K, w, and A, we note that

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

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.

Lemma 11 (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. $\mu(e, K, A) = \sup_n \{\mu^{(n)}(e, K, A)\}$

The next set of lemmas show how the measure $\mu(-, -, -)$ behaves under the reductions of the small-step machine. Almost all the calculations in Section 6 depend only on these lemmas.

Lemma 12. $\mu^{(p+1)}(\texttt{let}\, x = e_1 \;\texttt{in}\; e_2, K, A) = \mu^{(p)}(e_1, (x \to e_2)K, A)$.

Proof.

$$\begin{aligned}
&\mu^{(p+1)}(\texttt{let}\, x = e_1 \;\texttt{in}\; e_2, K, A) \\
&= \iint \mathrm{eval}^{(p+1)}(\sigma, \texttt{let}\, x = e_1 \;\texttt{in}\; e_2, K, \tau, 1, A)\, d\sigma\, d\tau \\
&= \iint \mathrm{eval}^{(p)}(\pi_L(\sigma), e_1, (x \to e_2)K, \pi_R(\sigma){::}\tau, 1, A)\, d\sigma\, d\tau \\
&= \iiint \mathrm{eval}^{(p)}(\sigma', e_1, (x \to e_2)K, \sigma''{::}\tau, 1, A)\, d\sigma'\, d\sigma''\, d\tau \qquad \text{(Lemma 1.4 on } \sigma) \\
&= \iint \mathrm{eval}^{(p)}(\sigma', e_1, (x \to e_2)K, \pi_L(\tau'){::}\pi_R(\tau'), 1, A)\, d\sigma'\, d\tau' \\
&\qquad\qquad \text{(Lemma 1.4 on } \tau') \\
&= \iint \mathrm{eval}^{(p)}(\sigma', e_1, (x \to e_2)K, \tau', 1, A)\, d\sigma'\, d\tau' \qquad (\pi_L(\tau'){::}\pi_R(\tau') = \tau') \\
&= \mu^{(p)}(e_1, (x \to e_2)K, A)
\end{aligned}$$

Lemma 13. $\mu^{(p+1)}(v, (x \to e)K, A) = \mu^{(p)}(e[v/x], K, A)$

Proof. Similar.

Lemma 14. $\mu^{(p+1)}(\,(\lambda x.e\ v), K, A) = \mu^{(p)}(e[v/x], K, A)$

Lemma 15. $\mu^{(p+1)}(op^n(v_1, \ldots, v_n), K, A) = \mu^{(p)}(\delta(op^n, v_1, \ldots, v_n), K, A)$, *when defined.*

Lemma 16. $\mu^{(p+1)}(\texttt{if}\ \texttt{c}_r\ \texttt{then}\ e_1\ \texttt{else}\ e_2, K, A) = \mu^{(p)}(e_1, K, A)$ *if* $r > 0$, *and* $\mu^{(p+1)}(\texttt{if}\ \texttt{c}_r\ \texttt{then}\ e_1\ \texttt{else}\ e_2, K, A) = \mu^{(p)}(e_1, K, A)$ *if* $r \leq 0$.

Proof (14–16). Immediate from the definitions.

The property for `sample` follows a slightly different pattern:

Lemma 17. $\mu^{(p+1)}(\texttt{sample}, K, A) = \int \mu^{(p)}(\texttt{c}_{\pi_U(\sigma)}, K, A)\, d\sigma$

Proof.

$$\begin{aligned}
&\mu^{(p+1)}(\texttt{sample}, K, A) \\
&= \iint \mathrm{eval}^{(p+1)}(\sigma, \texttt{sample}, K, \tau, 1, A)\, d\sigma\, d\tau \\
&= \iint \mathrm{eval}^{(p)}(\pi_R\sigma, \texttt{c}_{\pi_U(\pi_L(\sigma))}, K, \tau, 1, A)\, d\sigma\, d\tau \\
&= \iiint \mathrm{eval}^{(p)}(\sigma_2, \texttt{c}_{\pi_U(\sigma_1)}, K, \tau, 1, A)\, d\sigma_1\, d\sigma_2\, d\tau && \text{(Lemma 1.4)} \\
&= \iiint \mathrm{eval}^{(p)}(\sigma_2, \texttt{c}_{\pi_U(\sigma_1)}, K, \tau, 1, A)\, d\sigma_2\, d\tau\, d\sigma_1 && \text{(Lemma 2 twice)} \\
&= \int \mu^{(p)}(\texttt{c}_{\pi_U(\sigma_1)}, K, A)\, d\sigma_1
\end{aligned}$$

Lemma 18. $\mu^{(p+1)}(\texttt{factor c}_r, K, A) = r \times \mu^{(p)}(\texttt{c}_r, K, A)$*, provided* $r > 0$*.*

Proof. Immediate from the definitions and the Linearity Lemma (Lemma 4).

We note that all the relations in these lemmas similarly hold for the unapproximated measure $\mu(-,-,-)$.

6 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, expressions, and continuations.

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

Note that for all n, $\mathbb{V}_n \supseteq \mathbb{V}_{n+1} \supseteq \ldots$, and similarly for $\mathbb{E}$ and $\mathbb{K}$.

We use γ to range over substitutions of closed values for variables, and extend these relations to substitutions by

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

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:

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

We do not need a version of $\mathbb{K}$ indexed by Γ because we only work with closed continuations.

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

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

Most of the theorem follows by general nonsense about the lambda-calculus and logical relations, relying mostly on Lemmas 12–18.

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.

Lemma 19 (variables are value-compatible). *If $x \in \Gamma$, then $(x, x) \in \mathbb{V}^\Gamma$.*

Proof. 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 $\mathbb{G}_n^\Gamma$.

Lemma 20 (λ-expressions are value-compatible). *If $(e, e') \in \mathbb{E}^{\Gamma, x}$, then $(\lambda x.e, \lambda x.e') \in \mathbb{V}^\Gamma$.*

Proof. 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 $(e\gamma[v/x], e'\gamma'[v'/x]) \in \mathbb{E}_m$. 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 $\mathbb{E}_m$.

Lemma 21 (value-compatibility implies expression-compatibility). *If $(v, v') \in \mathbb{V}^\Gamma$, then $(v, v') \in \mathbb{E}^\Gamma$.*

Proof. 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 $m \leq n$, $(K, K') \in \mathbb{K}_m$, and A. Since $m \leq n$, we have $(v\gamma, v'\gamma') \in \mathbb{V}_m$, so $\mu^{(m)}(v, K, A) \leq \mu(v', K', A)$. Since we have this for all $m \leq n$, we conclude that $(v\gamma, v'\gamma') \in \mathbb{E}_n$.

Lemma 22 (compatibility under application). *If $(v_1, v_1') \in \mathbb{V}^\Gamma$ and $(v_2, v_2') \in \mathbb{V}^\Gamma$, then $(\,(v_1\ v_2),\, (v_1'\ v_2')\,) \in \mathbb{E}^\Gamma$.*

Proof. 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 $(\texttt{(}v_1\gamma\ v_2\gamma\texttt{)}, \texttt{(}v_1'\gamma'\ v_2'\gamma'\texttt{)}) \in \mathbb{E}_n$

If $v_1\gamma$ is of the form $\texttt{c}_r$, then $\mu^{(m)}(v_1\gamma, K, A) = 0$ for any m, K, and A, so by Lemma 11 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

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

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

$$\begin{aligned}
&\mu^{(m)}(\texttt{(}\lambda x.e\gamma\ v_2\gamma\texttt{)}, K, A) \\
&= \mu^{(m-1)}(e\gamma[v_2\gamma/x], K, A) && \text{(Lemma 14)} \\
&\leq \mu(e'\gamma'[v_2'\gamma'/x], K', A) && \text{(by } (e\gamma[v_2\gamma/x], e'\gamma'[v_2'\gamma'/x]) \in \mathbb{E}_{m-1}\text{)} \\
&= \mu(\texttt{(}\lambda x.e'\gamma'\ v_2'\gamma'\texttt{)}, K', A)
\end{aligned}$$

Lemma 23 (compatibility under operations). *If $(v_i, v_i') \in \mathbb{V}^\Gamma$ for all $i \in \{1, \ldots, k\}$, then $(op^k(v_1, \ldots, v_k), op^k(v_1', \ldots, v_k')) \in \mathbb{E}^\Gamma$.*

Proof. Similar.

Lemma 24 (`halt` related to itself). $(\texttt{halt}, \texttt{halt}) \in \mathbb{K}$.

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

Lemma 25. *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:

$$\begin{aligned}
&\mu^{(p)}(v, (x \to e)K, A) \\
&= \mu^{(p-1)}(e[v/x], K, A) && \text{(Lemma 13)} \\
&\leq \mu(e'[v'/x], K', A) \\
&= \mu(v', (x \to e')K', A)
\end{aligned}$$

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

Lemma 26 (compatibility under `let`). *If* $(e_1, e_1') \in \mathbb{E}^\Gamma$ *and* $(e_2, e_2) \in \mathbb{E}^{\Gamma,x}$, *then* $(\texttt{let}\, x = e_1\ \texttt{in}\ e_2, \texttt{let}\, x = e_1'\ \texttt{in}\ e_2') \in \mathbb{E}^\Gamma$.

Proof. 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 $(\gamma[x := v], \gamma'[x := v']) \in \mathbb{G}_{\Gamma,x}^m$. 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 25.

So choose $m \leq n$ and $(K, K') \in \mathbb{K}_m$. Without loss of generality, assume $m > 0$. Then by lemma 25 we have

$$((x \to e\gamma)K, (x \to e'\gamma')K') \in \mathbb{K}_m \tag{1}$$

Choose A. Now we can calculate:

$$\begin{aligned}
&\mu^{(m)}(\texttt{let}\, x = e_1\gamma\ \texttt{in}\ e_2\gamma, K, A) \\
&= \mu^{(m-1)}(e_1\gamma, (x \to e_2\gamma)K, A) && \text{(Lemma 12)} \\
&\leq \mu(e_1'\gamma', (x \to e_2'\gamma')K', A) \\
&= \mu(\texttt{let}\, x = e_1'\gamma'\ \texttt{in}\ e_2\gamma', K, A)
\end{aligned}$$

where the inequality follows from $(e_1\gamma, e_1'\gamma') \in \mathbb{E}_m$ and (1).

Lemma 27 (compatibility under `if`). *If* $(v, v') \in \mathbb{V}^\Gamma$ *and* $(e_1, e_1') \in \mathbb{E}^\Gamma$ *and* $(e_2, e_2) \in \mathbb{E}^\Gamma$, *then* $(\texttt{if}\ v\ \texttt{then}\ e_1\ \texttt{else}\ e_2, \texttt{if}\ v'\ \texttt{then}\ e_1'\ \texttt{else}\ e_2') \in \mathbb{E}^\Gamma$.

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

$$\begin{aligned}
&\mu^{(m)}(\texttt{if}\ v\gamma\ \texttt{then}\ e_1\gamma\ \texttt{else}\ e_2\gamma, K, A) \\
&= \mu^{(m-1)}(e_1\gamma, K, A) && \text{(Lemma 16)} \\
&\leq \mu(e_1'\gamma, K', A) \\
&= \mu(\texttt{if}\ v'\gamma'\ \texttt{then}\ e_1'\gamma'\ \texttt{else}\ e_2'\gamma', K', A)
\end{aligned}$$

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.

Lemma 28 (compatibility under `factor`). *If* $(v, v') \in \mathbb{V}^\Gamma$, *then* $(\texttt{factor}\ v, \texttt{factor}\ v') \in \mathbb{E}^\Gamma$.

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

Assume $v\gamma = v'\gamma' = \mathtt{c}_r$ 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

$$\begin{aligned}
&\mu^{(m)}(\texttt{factor}\ \mathtt{c}_r, K, A) \\
&= r \times \mu^{(m-1)}(\mathtt{c}_r, K, A) && \text{(Lemma 18)} \\
&\le r \times \mu(\mathtt{c}_r, K', A) \\
&= \mu(\texttt{factor}\ \mathtt{c}_r, K', A)
\end{aligned}$$

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

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

Lemma 29 (compatibility of `sample`).

$$(\texttt{sample}, \texttt{sample}) \in \mathbb{E}$$

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

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

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

$$\begin{aligned}
&\mu^{(m)}(\texttt{sample}, K, A) \\
&= \textstyle\int \mu^{(m-1)}(\mathtt{c}_{\pi_U(\sigma)}, K, A)\, d\sigma && \text{(Lemma 17)} \\
&\le \textstyle\int \mu(\mathtt{c}_{\pi_U(\sigma)}, K', A)\, d\sigma \\
&= \mu(\texttt{sample}, K', A)
\end{aligned}$$

Now we can prove the Fundamental Property:

Theorem 30 (Fundamental Property).

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

Proof. By induction on the derivation of $\Gamma \vdash e\ \mathsf{exp}$, etc. The base cases are lemmas 19 and 24. The lemmas above deal with each of the other scoping rules in turn.

7 CIU Ordering

The CIU ("closed instantiation of uses") ordering of two terms asserts that they yield related observations under a single substitution and a single continuation. For our purposes, we take the observables to be the behaviors on measurable subsets of the reals, as we did for the logical relations. In this section, we show that this relation coincides with the logical relation.

In one direction, this is an easy consequence of the Fundamental Property.

Definition 31.

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, $(e\gamma, e'\gamma) \in \mathbb{CIU}$.*

Lemma 32 ($\mathbb{E} \subseteq \mathbb{CIU}$). *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

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

Lemma 33 ($\mathbb{E}^\Gamma \circ \mathbb{CIU}^\Gamma \subseteq \mathbb{E}^\Gamma$). *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 34 ($\mathbb{CIU} \subseteq \mathbb{E}$). *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 33, $(e, e') \in \mathbb{E}^\Gamma$.

Theorem 35. *$(e, e') \in \mathbb{CIU}^\Gamma$ iff $(e, e') \in \mathbb{E}^\Gamma$.*

Proof. Immediate from Lemmas 32 and 34.

8 Contextual Ordering

We begin by defining the contextual ordering as the largest preorder that is adequate (that is, it distinguishes terms that have different observable behavior by themselves) and is compatible (closed under context formation). For our purposes, we take the observables to be the behavior on measurable subsets of the reals. We will show that the contextual ordering, the CIU ordering, and the logical relation all coincide.

Definition 36 ($\mathbb{CTX}^\Gamma$)**.** $\mathbb{CTX}$ *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 $(v_1, v_1') \in R^\Gamma$, ..., $(v_n, v_n') \in R^\Gamma$,*
   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.

Definition 37. *If $\Gamma \vdash e$ $\mathtt{exp}$ and $\Gamma \vdash e'$ $\mathtt{exp}$, we say e and e' are* contextually equivalent *($e =_{ctx} e'$) if both $(e, e') \in \mathbb{CTX}^\Gamma$ and $(e', e) \in \mathbb{CTX}^\Gamma$.*

We begin with a lemma due to Pitts [14].

Lemma 38. *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. 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 $2 \leq m \leq n+2$, $(u, u') \in \mathbb{V}_m$, and $A \in \Sigma_{\mathbb{R}}$. We must show

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

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

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

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 $\leq$).

Next, observe $m-2 \leq n$, 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, $\mu^{(m-2)}(e[v/x], K, A) \leq \mu(e'[v'/x], K', A)$.

Putting the pieces together, we have

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

Lemma 39. *If $(\mathtt{c}_r, \mathtt{c}_{r'}) \in \mathbb{E}$, then $r = r'$, and in particular, $(\mathtt{c}_r, \mathtt{c}_{r'}) \in \mathbb{V}$.*

Proof. We prove the contrapositive. Let $r \neq r'$, and choose A to be a measurable subset of the reals containing r but not r'. Then we have

$$\begin{aligned}
\mu(\mathtt{c}_r, \mathtt{halt}, A) &= \iint \mathrm{eval}(\sigma_1, \mathtt{c}_r, \mathtt{halt}, \sigma_2, 1, A)\, d\sigma_1\, d\sigma_2 = I_A(r) = 1 \\
\mu(\mathtt{c}_{r'}, \mathtt{halt}, A) &= \iint \mathrm{eval}(\sigma_1, \mathtt{c}_{r'}, \mathtt{halt}, \sigma_2, 1, A)\, d\sigma_1\, d\sigma_2 = I_A(r') = 0
\end{aligned}$$

So $(\mathtt{c}_r, \mathtt{c}_{r'}) \notin \mathbb{E}$.

Lemma 40. *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 $v = \mathtt{c}_r$ and $v' = \mathtt{c}_{r'}$, then by the preceding lemma $r = r'$, and the result is trivial. 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 `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 $p \leq m$, $(K, K') \in \mathbb{K}_p$, and $A \in \Sigma_{\mathbb{R}}$. We must show that

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

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 38, $(K_1', K_1') \in \mathbb{K}_{p+2}$. Furthermore, $p \leq m < n$, so $p+2 \leq n+1$ 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

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

Theorem 41. $\mathbb{E}^\Gamma \subseteq \mathbb{CTX}^\Gamma$.

Proof. We will show that $\mathbb{E}$ forms a family of equivalence relations 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 $(\texttt{halt}, \texttt{halt}) \in \mathbb{K}$ by lemma 24, hence for any measurable subset A of reals, $(e, e') \in \mathbb{E}^\Gamma$ implies $\mu(e, \texttt{halt}, A) = \mu(e', \texttt{halt}, A)$.

The $\mathbb{E}$-compatibility lemmas (Lemmas 21–28) are almost exactly what is needed for $\mathbb{CTX}$-compatibility. The exceptions are Lemmas 22 and 28, because their hypotheses refer to $\mathbb{V}^\Gamma$, rather than $\mathbb{E}^\Gamma$. We fill the gap with Lemma 40. We show how this is done for $\texttt{factor}\ v$; the cases for application, primitives, and conditions are similar.

$$
\begin{aligned}
&(v, v') \in \mathbb{E}^\Gamma \\
&\quad\Longrightarrow\ (v, v') \in \mathbb{CIU}^\Gamma \\
&\quad\Longrightarrow\ (\forall\gamma)((v\gamma, v'\gamma) \in \mathbb{CIU}^\emptyset) \\
&\quad\Longrightarrow\ (\forall\gamma)((v\gamma, v'\gamma) \in \mathbb{E}^\emptyset) \\
&\quad\Longrightarrow\ (\forall\gamma)((v\gamma, v'\gamma) \in \mathbb{V}^\emptyset) && \text{(Lemma 40)} \\
&\quad\Longrightarrow\ (\forall\gamma)((\texttt{factor}\ v\gamma, \texttt{factor}\ v'\gamma) \in \mathbb{E}^\emptyset) \\
&\quad\Longrightarrow\ (\forall\gamma)((\texttt{factor}\ v\gamma, \texttt{factor}\ v'\gamma) \in \mathbb{CIU}^\emptyset) \\
&\quad\Longrightarrow\ (\texttt{factor}\ v, \texttt{factor}\ v') \in \mathbb{CIU}^\Gamma \\
&\quad\Longrightarrow\ (\texttt{factor}\ v, \texttt{factor}\ v') \in \mathbb{E}^\Gamma
\end{aligned}
$$

We conclude by showing that the contextual ordering is contained in the the CIU ordering. We need two more lemmas before proceeding to the theorem.

Lemma 42. *If* $\Gamma, x \vdash e\ \mathsf{exp}$ *and* $\Gamma \vdash v\ \mathsf{exp}$*, then*

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

Proof. Let γ be a closing substitution for Γ. Then for any σ, closed K, and w, by lemmas 14 and 11.4 we have

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

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 43. *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

$$(\ (\lambda x.e\ v),\ (\lambda x.e'\ v')\ ) \in \mathbb{CTX}^\Gamma \tag{2}$$

So now we have:

$$
\begin{aligned}
&(e[v/x], (\lambda x.e\ v)) \in \mathbb{CIU}^\Gamma && \text{(Lemma 42)}\\
&\implies (e[v/x], (\lambda x.e\ v)) \in \mathbb{CTX}^\Gamma && (\mathbb{CIU}^\Gamma \subseteq \mathbb{CTX}^\Gamma)\\
&\implies (e[v/x], (\lambda x.e'\ v')) \in \mathbb{CTX}^\Gamma && \text{(Equation (2) and transitivity of } \mathbb{CTX}^\Gamma)\\
&\implies (e[v/x], e'[v'/x]) \in \mathbb{CTX}^\Gamma && \text{(Lemma 42 and transitivity of } \mathbb{CTX}^\Gamma)
\end{aligned}
$$

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 44 ($\mathbb{CTX}^\Gamma \subseteq \mathbb{CIU}^\Gamma$)**.** *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 $K = \texttt{halt}$ and $(e, e') \in \mathbb{CTX}^\emptyset$, then $\mu(e, \texttt{halt}, A) = \mu(e', \texttt{halt}, A)$ 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

$$(\texttt{let}\, x = e\ \texttt{in}\ e_1, \texttt{let}\, x = e'\ \texttt{in}\ e_1) \in \mathbb{CTX}^\emptyset \tag{3}$$

Then we have

$$
\begin{aligned}
&\mu(e, (x \to e_1)K, A)\\
&= \mu(\texttt{let}\, x = e\ \texttt{in}\ e_1, K, A) && \text{(Lemma 12)}\\
&\leq \mu(\texttt{let}\, x = e'\ \texttt{in}\ e_1, K, A) && \text{(by IH at } K\text{, applied to (3))}\\
&= \mu(e', (x \to e_1)K, A) && \text{(Lemma 12)}
\end{aligned}
$$

Thus completing the induction step.

Summarizing the results:

Theorem 45. *For all* Γ*,* $\mathbb{E}^\Gamma = \mathbb{CIU}^\Gamma = \mathbb{CTX}^\Gamma$*.*

Proof. From Theorems 35, 41, and 44.

9 Reordering Samples

In this section we characterize a class of transformations on the entropy space that are measure-preserving. We will use one such function in section 10 to justify reordering `let` bindings; other such functions may be useful in justifying other equivalences.

Definition 46 (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:

$$\phi_1(\sigma_1 \mathbin{::} (\sigma_2 \mathbin{::} \sigma_3)) = \sigma_2 \mathbin{::} (\sigma_1 \mathbin{::} \sigma_3)$$

Or equivalently, written using explicit projections:

$$\phi_1(\sigma) = \pi_L(\pi_R(\sigma)) \mathbin{::} (\pi_L(\sigma) \mathbin{::} \pi_R(\pi_R(\sigma)))$$

We will use this function in theorem 50 to justify `let`-reordering. Another example is

$$\phi_2(\sigma_1 \mathbin{::} \sigma_2) = \sigma_2$$

which could be used to justify dropping a dead `let` binding.

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 $\phi_1 \mathbin{::} \phi_2$ 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 $\mathbb{S}$.
$$\phi(\sigma) = \begin{cases} p(\sigma) & \text{if } \phi = p \\ \phi_1(\sigma) \mathbin{::} \phi_2(\sigma) & \text{if } \phi = \phi_1 \mathbin{::} \phi_2 \end{cases}$$
- 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 47. *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 (Sketch): By strong induction on the length of the longest path in the sequence, and by the definition of non-duplicating and Lemma 2 (Tonelli).

Theorem 48. *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 47 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

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

10 Commutativity

We are now prepared to prove the commutativity theorem promised at the beginning. We first prove a general theorem relating value-preserving transformations on the entropy space:

Theorem 49. *Let e and e' be expressions, and let $\phi : \mathbb{S} \to \mathbb{S}$ be a measure-preserving transformation such that for all K and τ,*

$$(\text{eval}(\sigma, e, K, \tau, 1, A) = r > 0) \implies (\text{eval}(\phi(\sigma), e', K, \tau, 1, A) = r)$$

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

Proof. Without loss of generality, assume e and e' are closed (otherwise apply a closing substitution). By Theorem 45, it is sufficient to show that for any K and A, $\mu(e, K, A) \leq \mu(e', K, A)$. We calculate:

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

Now we can finally prove the desired equivalence.

Theorem 50. *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. Let e and e' denote the two expressions of the theorem. We will use Theorem 49. 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:

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

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.

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

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

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

This can be accomplished by using ϕ_1 from section 9; we set

$$\sigma' = \phi_1(\sigma) = \pi_L(\pi_R(\sigma))\mathbf{::}(\pi_L(\sigma)\mathbf{::}\pi_R(\pi_R(\sigma)))$$

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

$$\mathrm{eval}(\sigma, e, K, \tau, 1, A) = \mathrm{eval}(\phi_1(\sigma), e', K, \tau, 1, A)$$

ϕ_1 is a non-duplicating FSF, so it is measure-preserving by Theorem 48. Then by Theorem 49, $(e, e') \in \mathbb{CTX}$. The converse holds by symmetry.

Corollary 51. *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 the expressions*

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

and

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

are contextually equivalent.

Proof. By the preceding theorem, any closed instances of these expressions are contextually equivalent, and hence CIU-equivalent. Hence the open expressions are CIU-equivalent, and hence contextually equivalent.

11 Formalization

We have formalized the development through Section 9 using Coq 8.6. For the most part, the proofs directly follow the proofs in the paper. We have not formalized the proof of the measurability of eval.

We use the Autosubst library [16] to manage binders in the syntax and the Coquelicot library [3] for representing and reasoning about real numbers. In addition to the axioms provided by the Coquelicot library, we also have axiomatized the entropy source and basic properties of Lebesgue integration, such as linearity and monotonicity, Lebesgue's monotone convergence theorem, and Tonelli's theorem, for integration with respect to the measure on the entropy source. We have also taken functional extensionality and restricted forms of the law of the excluded middle as axioms.

12 A Variation

Because our proofs were formalized, it was relatively easy to experiment with variations on our definitions.

In particular, the proofs as presented depend on the existence of an operation `real?` to distinguish between real constants and lambda-abstractions. As a result, our definition of $\mathbb{V}$ is familiar. However, if we change the definition of $\mathbb{V}$, we can obtain all our results without relying on the existence of the reflective operator `real?`.

The necessary change is to allow $\mathbb{V}$ to relate an lambda-abstraction and constant when the application of the lambda-abstraction would result in an error (either a stuck configuration or non-termination). For example, $\lambda x.(\mathtt{c}_r\ x)$ should be related to $\mathtt{c}_r$. Relating these terms is sensible because (in the absence of `real?`):

- Continuations that treat the terms as constants cannot distinguish the terms. Misusing the lambda-abstraction will result in a 0 on the left-hand side of the inequality in the definition of $\mathbb{E}$.
- Continuations that treat the terms as functions cannot distinguish the terms. Misusing the constant as an abstraction will result in a 0 on the right-hand side of the inequality of the definition of $\mathbb{E}$, but correctly applying the lambda-abstraction will either not terminate with the given fuel or will result in the same error, both of which result in a 0 on the left-hand side of the inequality.

Since all erroneous terms are indistinguishable (all evaluating to 0), the concrete change is to add the following disjunct to the definition of $\mathbb{V}$:

$$\begin{array}{l} v_1 = \lambda x.e \wedge v_2 = \mathtt{c}_r \\ \wedge\ (\forall m < n)(\forall v)((e[v/x], \mathit{loop}) \in \mathbb{E}_m) \end{array}$$

where *loop* is defined as $(\!(\lambda x.(\!(x\ x)\!))\ (\!(\lambda x.(\!(x\ x)\!))\!)$, or any other non-terminating expression. If the cause of the error (i.e., from applying a constant as if it were an

abstraction) were somehow observable through μ, *loop* would have to be replaced with a term that generated that specific error.

With this change, our Coq development goes through with only local changes.

13 Related Work

Our language and semantics are similar to those of Borgström et al. [5] and Culpepper and Cobb [7], both of which include continuous random variables and scoring (or "soft observations"). Like Borgström et al. [5], our language is untyped and thus includes recursion and non-termination, and we define its meaning in terms of a small-step reduction semantics. However, we adapt the semantics to use the "splitting" entropy measure space from Culpepper and Cobb [7], so that the entropy can be divided into independent parts based on the syntactic structure of the expression being evaluated. That makes the construction of the measure-preserving entropy-shuffling functions for `let`-reordering tractable. This construction would not work in the semantics of Borgström et al. [5], where the entropy is accumulated in a sequence.

The construction of our logical relation follows the tutorial of Pitts [14] on the construction of biorthogonal, step-indexed [1] logical relations. Instead of termination, we use the program measure as the observable behavior, following Culpepper and Cobb [7]. 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 [7, the $\mathcal{A}$ relation].

There has been previous work on contextual equivalence for probabilistic languages with only *discrete* random variables. In particular, Bizjak and Birkedal [2] 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 [6, 15] to languages with discrete choice; Ehrhard et al. [8] have constructed fully abstract models for PCF with discrete probabilistic choice using probabilistic coherence spaces.

Staton et al. [18] 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, Staton [17] proves the soundness of the `let`-reordering transformation for a first-order language.

14 Open Problems and Future Work

We now have several different definitions of contextual equivalence for probabilistic programming languages. One set of open problems is to relate these different languages and definitions:

– Borgström et al. [5] has a language similar to ours, but their entropy space ("traces") consists of finite sequences of real numbers, and their semantics threads the sequence through evaluation, in contrast to our "entropy splitting" semantics. Their measure on traces also differs from ours. They have an operational semantics, so they implicitly have a contextual preorder. It would be enlightening to know if it coincided with ours.
– A limitation of our semantics is that it does not distinguish among raising an error (like trying to apply a constant) and failing to terminate: all contribute 0 to the measure. It might be useful to refine the semantics to distinguish these cases.

Bibliography

[1] Ahmed, A.J.: Step-indexed syntactic logical relations for recursive and quantified types. In: *Proc. 15th European Symposium on Programming* (ESOP '06). pp. 69–83 (2006)

[2] Bizjak, A., Birkedal, L.: Step-indexed logical relations for probability. In: Proc. 18th International Conference on Foundations of Software Science and Computation Structures. pp. 279–294. FoSSaCS '15 (2015)

[3] Boldo, S., Lelay, C., Melquiond, G.: Coquelicot: A user-friendly library of real analysis for coq. Mathematics in Computer Science 9(1), 41–62 (2015)

[4] Borgström, J., Dal Lago, U., Gordon, A.D., Szymczak, M.: A lambda-calculus foundation for universal probabilistic programming (long version) (Jan 2017), https://arxiv.org/abs/1512.08990

[5] Borgström, J., Lago, U.D., Gordon, A.D., Szymczak, M.: A lambda-calculus foundation for universal probabilistic programming. In: *Proc. 21st ACM SIGPLAN International Conference on Functional Programming*. pp. 33–46. ICFP '16 (2016)

[6] Crubillé, R., Lago, U.D.: On probabilistic applicative bisimulation and call-by-value λ-calculi. In: *Proc. 23rd European Symposium on Programming*. pp. 209–228. ESOP '14 (2014)

[7] Culpepper, R., Cobb, A.: Contextual equivalence for probabilistic programs with continuous random variables and scoring. In: *Proc. 26th European Symposium on Programming*. pp. 368–392. ESOP '17 (2017)

[8] Ehrhard, T., Tasson, C., Pagani, M.: Probabilistic coherence spaces are fully abstract for probabilistic PCF. In: *Proc. 41st ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages* (POPL '14). pp. 309–320 (2014)

[9] Goodman, N.D., Mansinghka, V.K., Roy, D.M., Bonawitz, K., Tenenbaum, J.B.: Church: a language for generative models. In: Proc. 24th Conference in Uncertainty in Artificial Intelligence. pp. 220–229. UAI '08 (2008)

[10] Kiselyov, O., Shan, C.: Embedded probabilistic programming. In: Proc. IFIP TC 2 Working Conference on Domain-Specific Languages. pp. 360–384. DSL '09 (2009)

[11] Mansinghka, V., Selsam, D., Perov, Y.: Venture: a higher-order probabilistic programming platform with programmable inference (Mar 2014), http://arxiv.org/abs/1404.0099

[12] Narayanan, P., Carette, J., Romano, W., Shan, C., Zinkov, R.: Probabilistic inference by program transformation in hakaru (system description). In: Proc. 13th International Symposium on Functional and Logic Programming. pp. 62–79. FLOPS '16 (2016)

[13] Paige, B., Wood, F.: A compilation target for probabilistic programming languages. In: Proc. 31th International Conference on Machine Learning. pp. 1935–1943. ICML '14 (2014)

[14] Pitts, A.M.: Step-indexed biorthogonality: a tutorial example. In: Ahmed, A., Benton, N., Birkedal, L., Hofmann, M. (eds.) Modelling, Controlling and Reasoning About State. Dagstuhl Seminar Proceedings (2010), http://drops.dagstuhl.de/opus/volltexte/2010/2806/

[15] Sangiorgi, D., Vignudelli, V.: Environmental bisimulations for probabilistic higher-order languages. In: *Proc. 43rd ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*. pp. 595–607. POPL '16 (2016)

[16] Schäfer, S., Tebbi, T., Smolka, G.: Autosubst: Reasoning with de bruijn terms and parallel substitutions. In: Proc. 6th International Conference on Interactive Theorem Proving. pp. 359–374. ITP '15 (2015)

[17] Staton, S.: Commutative semantics for probabilistic programming. In: *Proc. 26th European Symposium on Programming*. pp. 855–879. ESOP '17 (2017)

[18] Staton, S., Yang, H., Wood, F., Heunen, C., Kammar, O.: Semantics for probabilistic programming: higher-order functions, continuous distributions, and soft constraints. In: *Proc. 31st IEEE Symposium on Logic in Computer Science*. pp. 525–534. LICS '16 (2016)

[19] Wand, M., Clinger, W.D.: Set constraints for destructive array update optimization. J. Funct. Program. 11(3), 319–346 (2001)

[20] Wood, F., van de Meent, J., Mansinghka, V.: A new approach to probabilistic programming inference. In: Proc. 17th International Conference on Artificial Intelligence and Statistics. pp. 1024–1032. AISTATS '14 (2014)