Step-Indexed Logical Relations for Probability

Aleš Bizjak and Lars Birkedal

Aarhus University
{abizjak,birkedal}@cs.au.dk

Abstract. It is well-known that constructing models of higher-order probabilistic programming languages is challenging. We show how to construct step-indexed logical relations for a probabilistic extension of a higher-order programming language with impredicative polymorphism and recursive types. We show that the resulting logical relation is sound and complete with respect to the contextual preorder and, moreover, that it is convenient for reasoning about concrete program equivalences. Finally, we extend the language with dynamically allocated first-order references and show how to extend the logical relation to this language. We show that the resulting relation remains useful for reasoning about examples involving both state and probabilistic choice.

1 Introduction

It is well known that it is challenging to develop techniques for reasoning about programs written in probabilistic higher-order programming languages. A probabilistic program evaluates to a distribution of values, as opposed to a set of values in the case of nondeterminism or a single value in the case of deterministic computation. Probability distributions form a monad. This observation has been used as a basis for several denotational domain-theoretic models of probabilistic languages and also as a guide for designing probabilistic languages with monadic types [15,21,20]. Game semantics has also been used to give models of probabilistic programming languages [9,12] and a fully abstract model using coherence spaces for PCF with probabilistic choice was recently presented [13].

The majority of models of probabilistic programming languages have been developed using denotational semantics. However, Johann et.al. [14] developed operationally-based logical relations for a polymorphic programming language with effects. Two of the effects they considered were probabilistic choice and *global* ground store. However, as pointed out by the authors [14], extending their construction to local store and, in particular, higher-order local store, is likely to be problematic. Recently, operationally-based bisimulation techniques have been extended to probabilistic extensions of PCF [7,8]. The operational semantics of probabilistic higher-order programming languages has been investigated in [16].

Step-indexed logical relations [2,3] have proved to be a successful method for proving contextual approximation and equivalence for programming languages with a wide range of features, including computational effects.

In this paper we show how to extend the method of step-indexed logical relations to reason about contextual approximation and equivalence of probabilistic

A. Pitts (Ed.): FOSSACS 2015, LNCS 9034, pp. 279–294, 2015.
DOI: 10.1007/978-3-662-46678-0_18

higher-order programs. To define the logical relation we employ biorthogonality [17,19] and step-indexing. Biorthogonality is used to ensure completeness of the logical relation with respect to contextual equivalence, but it also makes it possible to keep the value relations simple, see Fig. 1. Moreover, the definition using biorthogonality makes it possible to "externalize" the reasoning in many cases when proving example equivalences. By this we mean that the reasoning reduces to algebraic manipulations of probabilities. This way, the quantitative aspects do not complicate the reasoning much, compared to the usual reasoning with step-indexed logical relations. To define the biorthogonal lifting we use two notions of observation; the termination probability and its stratified version approximating it. We define these and prove the required properties in Section 3.

We develop our step-indexed logical relations for the call-by-value language $\mathbf{F}^{\mu,\oplus}$. This is system **F** with recursive types, extended with a single probabilistic choice primitive `rand`. The primitive `rand` takes a natural number n and reduces with uniform probability to one of $1, 2, \ldots, n$. Thus `rand` n represents the uniform probability distribution on the set $\{1, 2, \ldots, n\}$. We choose to add `rand` instead of just a single coin flip primitive to make the examples easier to write.

To show that the model is useful we use it to prove some example equivalences in Section 5. We show two examples based on parametricity. In the first example, we characterize elements of the universal type $\forall\alpha.\alpha \to \alpha$. In a deterministic language, and even in a language with nondeterministic choice, the only interesting element of this type is the identity function. However, since in a probabilistic language we not only observe the end result, but also the likelihood with which it is returned, it turns out that there are many more elements. Concretely, we show that the elements of the type $\forall\alpha.\alpha \to \alpha$ that are of the form $\Lambda\alpha.\lambda x.e$, correspond precisely to *left-computable* real numbers in the interval $[0, 1]$. In the second example we show a free theorem involving functions on lists. We show additional equivalences in the Appendix, including the correctness of von Neumann's procedure for generating a fair sequence of coin tosses from an unfair coin, and equivalences from the recent papers using bisimulations [7,8].

We add dynamically allocated references to the language and extend the logical relation to the new language in Section 6. For simplicity we only sketch how to extend the construction with first-order state. This already suggests that an extension with general references can be done in the usual way for step-indexed logical relations. We conclude the section by proving a representation independence result involving both state and probabilistic choice.

All the references to the Appendix in this paper refer to appendix in the online long version [6].

2 The Language $\mathbf{F}^{\mu,\oplus}$

The language is a standard pure functional language with recursive, universal and existential types with an additional choice primitive `rand`. The base types include the type of natural numbers `nat` with some primitive operations.

The grammar of terms e is

$$\begin{aligned} e ::= {} & x \mid \langle\rangle \mid \mathtt{rand}\, e \mid \underline{n} \mid \mathtt{if}_1\ e\ \mathtt{then}\ e_1\ \mathtt{else}\ e_2 \mid \mathtt{P}\, e \mid \mathtt{S}\, e \mid \langle e_1, e_2\rangle \mid \mathtt{proj}_i\, e \\ & \mid \lambda x.e \mid e_1\, e_2 \mid \mathtt{inl}\ e \mid \mathtt{inr}\ e \mid \mathtt{match}\,(e, x_1.e_1, x_2.e_2) \mid \Lambda.e \mid e[] \\ & \mid \mathtt{pack}\, e \mid \mathtt{unpack}\ e_1\ \mathtt{as}\ x\ \mathtt{in}\ e_2 \mid \mathtt{fold}\, e \mid \mathtt{unfold}\, e \end{aligned}$$

We write $\underline{n}$ for the numeral representing the natural number n and $\mathtt{S}$ and $\mathtt{P}$ are the successor and predecessor functions, respectively. For convenience, numerals start at $\underline{1}$. Given a numeral $\underline{n}$, the term $\mathtt{rand}\,\underline{n}$ evaluates to one of the numerals $\underline{1}, \ldots, \underline{n}$ with uniform probability. There are no types in the syntax of terms, e.g., instead of $\Lambda\alpha.e$ and $e\,\tau$ we have $\Lambda.e$ and $e[]$. This is for convenience only.

We write $\alpha, \beta, \ldots$ for *type variables* and $x, y, \ldots$ for *term variables*. The notation $\tau[\vec{\tau}/\vec{\alpha}]$ denotes the simultaneous capture-avoiding substitution of types $\vec{\tau}$ for the free type variables $\vec{\alpha}$ in the type τ; $e[\vec{v}/\vec{x}]$ denotes simultaneous capture-avoiding substitution of values $\vec{v}$ for the free term variables $\vec{x}$ in the term e.

We write **Stk** for the set of evaluation contexts given by the call-by-value reduction strategy. Given two evaluation contexts E, E' we define their composition $E \circ E'$ by induction on E in the natural way. Given an evaluation context E and expression e we write $E[e]$ for the term obtained by plugging e into E. For any two evaluation contexts E and E' and a term e we have $E[E'[e]] = (E \circ E')[e]$.

For a type variable context Δ, the judgment $\Delta \vdash \tau$ expresses that the free type variables in τ are included in Δ. The typing judgments are entirely standard with the addition of the typing of $\mathtt{rand}$ which is given by the rule

$$\frac{\Delta \mid \Gamma \vdash e : \mathtt{nat}}{\Delta \mid \Gamma \vdash \mathtt{rand}\ e : \mathtt{nat}}.$$

The complete set of typing rules are in the Appendix. We write $\mathfrak{T}(\Delta)$ for the set of types well-formed in context Δ, and $\mathfrak{T}$ for the set of *closed* types τ. We write $\mathbf{Val}\,(\tau)$ and $\mathbf{Tm}\,(\tau)$ for the sets of *closed* values and terms of type τ, respectively. We write **Val** and **Tm** for the set of *all*[1] *closed* values and closed terms, respectively. $\mathbf{Stk}\,(\tau)$ denotes the set of τ-accepting evaluation contexts, i.e., evaluation contexts E, such that given any closed term e of type τ, $E[e]$ is a typeable term. **Stk** denotes the set of all evaluation contexts.

For a typing context $\Gamma = x_1{:}\tau_1, \ldots, x_n{:}\tau_n$ with $\tau_1, \ldots, \tau_n \in \mathfrak{T}$, let $\mathbf{Subst}(\Gamma)$ denote the set of type-respecting value substitutions, i.e. for all i, $\gamma(x_i) \in \mathbf{Val}\,(\tau_i)$. In particular, if $\Delta \mid \Gamma \vdash e : \tau$ then $\varnothing \mid \varnothing \vdash e\gamma : \tau\delta$ for any $\delta \in \mathfrak{T}^{\Delta}$ and $\gamma \in \mathbf{Subst}(\Gamma\delta)$, and the type system satisfies standard properties of progress and preservation and a canonical forms lemma.

The operational semantics of the language is a standard call-by-value semantics but weighted with $p \in [0, 1]$ which denotes the likelihood of that reduction. We write $\overset{p}{\rightsquigarrow}$ for the one-step reduction relation. All the usual β reductions have weight equal to 1 and the reduction from $\mathtt{rand}\,\underline{n}$ is

$$\mathtt{rand}\,\underline{n} \overset{\frac{1}{n}}{\rightsquigarrow} \underline{k} \qquad \text{for } k \in \{1, 2, \ldots, n\}.$$

[1] In particular, we do not require them to be typeable.

The rest of the rules are given in Fig. 5 in the Appendix. The operational semantics thus gives rise to a Markov chain with closed terms as states. In particular for each term e we have $\sum_{e' \mid e \overset{p}{\rightsquigarrow} e'} p \leq 1$.

3 Observations and Biorthogonality

We will use biorthogonality to define the logical relation. This section provides the necessary observation predicates used in the definition of the biorthogonal lifting of value relations to expression relations. Because of the use of biorthogonality the value relations (see Fig. 1) remain as simple as for a language without probabilistic choice. The new quantitative aspects only appear in the definition of the biorthogonal lifting ($\top\top$-closure) defined in Section 4. Two kinds of observations are used. The probability of termination, $\mathfrak{P}^{\Downarrow}(e)$, which is the actual probability that e terminates, and its approximation, the *stratified* termination probability $\mathfrak{P}^{\Downarrow}_k(e)$, where $k \in \mathbb{N}$ denotes, intuitively, the number of computation steps. The stratified termination probability provides the link between steps in the operational semantics and the indexing in the definition of the interpretation of types.

The probability of termination, $\mathfrak{P}^{\Downarrow}(\cdot)$, is a function of type $\mathbf{Tm} \to \mathcal{I}$ where $\mathcal{I}$ is the unit interval $[0,1]$. Since $\mathcal{I}$ is a pointed ω-cpo for the usual order, so is the space of all functions $\mathbf{Tm} \to \mathcal{I}$ with pointwise ordering. We define $\mathfrak{P}^{\Downarrow}(\cdot)$ as a fixed point of the continuous function Φ on this ω-cpo: Let $\mathcal{F} = \mathbf{Tm} \to \mathcal{I}$ and define $\Phi : \mathcal{F} \to \mathcal{F}$ as

$$\Phi(f)(e) = \begin{cases} 1 & \text{if } e \in \mathbf{Val} \\ \displaystyle\sum_{e \overset{p}{\rightsquigarrow} e'} p \cdot f(e') & \text{otherwise} \end{cases}$$

Note that if e is stuck then $\Phi(f)(e) = 0$ since the empty sum is 0.

The function Φ is monotone and preserves suprema of ω-chains. The proof is straightforward and can be found in the Appendix. Thus Φ has a least fixed point in $\mathcal{F}$ and we denote this fixed point by $\mathfrak{P}^{\Downarrow}(\cdot)$, i.e., $\mathfrak{P}^{\Downarrow}(e) = \sup_{n \in \omega} \Phi^n(\bot)(e)$.

To define the stratified observations we need the notion of a path. Given terms e and e' a path π from e to e', written $\pi : e \rightsquigarrow^* e'$, is a sequence $e \overset{p_1}{\rightsquigarrow} e_1 \overset{p_2}{\rightsquigarrow} e_2 \overset{p_3}{\rightsquigarrow} \cdots \overset{p_n}{\rightsquigarrow} e'$. The *weight* $\mathfrak{W}(\pi)$ of a path π is the product of the weights of reductions in π. We write $\mathfrak{R}$ for the set of all paths and $\cdot$ for their concatenation (when defined). For a non-empty path $\pi \in \mathfrak{R}$ we write $\ell(\pi)$ for its last expression.

We call reductions of the form $\mathtt{unfold}(\mathtt{fold}\, v) \overset{1}{\rightsquigarrow} v$ *unfold-fold* reductions and reductions of the form $\mathtt{rand}\, \underline{n} \overset{\frac{1}{n}}{\rightsquigarrow} \underline{k}$ *choice* reductions. If *none* of the reductions in a path π is a choice reduction we call π *choice-free* and similarly if none of the reductions in π is an unfold-fold reductions we call π *unfold-fold free*.

We define the following types of multi-step reductions which we use in the definition of the logical relation.

- $e \overset{\text{cf}}{\Longrightarrow} e'$ if there is a *choice-free* path from e to e'

– $e \overset{\text{uff}}{\Longrightarrow} e'$ if there is an *unfold-fold* free path from e to e'.
– $e \overset{\text{cuff}}{\Longrightarrow} e'$ if $e \overset{\text{cf}}{\Longrightarrow} e'$ and $e \overset{\text{uff}}{\Longrightarrow} e'$.

The following useful lemma states that all but choice reductions preserve the probability of termination. As a consequence, we will see that all but choice reductions preserve equivalence.

Lemma 3.1. *Let* $e, e' \in \mathbf{Tm}$ *and* $e \overset{cf}{\Longrightarrow} e'$. *Then* $\mathfrak{P}^{\Downarrow}(e) = \mathfrak{P}^{\Downarrow}(e')$.

The proof proceeds on the length of the reduction path with the strengthened induction hypothesis stating that the probabilities of termination of all elements on the path are the same. To define the stratified probability of termination that approximates $\mathfrak{P}^{\Downarrow}(\cdot)$ we need an auxiliary notion.

Definition 3.2. *For a closed expression* $e \in \mathbf{Tm}$ *we define* $\mathbf{Red}(e)$ *as the (unique) set of paths containing* exactly one *unfold-fold or choice reduction and* ending *with such a reduction. More precisely, we define the function* $\mathbf{Red} : \mathbf{Tm} \to \mathcal{P}(\mathfrak{R})$ *as the least function satisfying*

$$\mathbf{Red}(e) = \begin{cases} \{e \overset{1}{\rightsquigarrow} e'\} & \text{if } e = E[\mathtt{unfold}(\mathtt{fold}\, v)] \\ \{e \overset{p}{\rightsquigarrow} E[\underline{k}] \mid p = \frac{1}{n}, k \in \{1, 2, \ldots, n\}\} & \text{if } e = E[\mathtt{rand}\, \underline{n}] \\ \left\{(e \overset{1}{\rightsquigarrow} e') \cdot \pi \mid \pi \in \mathbf{Red}(e')\right\} & \text{if } e \overset{1}{\rightsquigarrow} e' \text{ and } e \overset{cuff}{\Longrightarrow} e' \\ \emptyset & \text{otherwise} \end{cases}$$

where we order the power set $\mathcal{P}(\mathfrak{R})$ *by subset inclusion.*

Using $\mathbf{Red}(\cdot)$ we define a monotone map $\Psi : \mathcal{F} \to \mathcal{F}$ that preserves ω-chains.

$$\Psi(f)(e) = \begin{cases} 1 & \text{if } \exists v \in \mathbf{Val}, e \overset{\text{cuff}}{\Longrightarrow} v \\ \sum_{\pi \in \mathbf{Red}(e)} \mathfrak{W}(\pi) \cdot f(\ell(\pi)) & \text{otherwise} \end{cases}$$

and then define $\mathfrak{P}_k^{\Downarrow}(e) = \Psi^k(\bot)(e)$. The intended meaning of $\mathfrak{P}_k^{\Downarrow}(e)$ is the probability that e terminates within k unfold-fold and choice reductions. Since Ψ is monotone we have that $\mathfrak{P}_k^{\Downarrow}(e) \leq \mathfrak{P}_{k+1}^{\Downarrow}(e)$ for any k and e.

The following lemma is the reason for counting only certain reductions, cf.[10]. It allows us to stay at the same step-index even when taking steps in the operational semantics. As a consequence we will get a more extensional logical relation. The proof is by case analysis and can be found in the Appendix.

Lemma 3.3. *Let* $e, e' \in \mathbf{Tm}$. *If* $e \overset{cuff}{\Longrightarrow} e'$ *then for all* k, $\mathfrak{P}_k^{\Downarrow}(e) = \mathfrak{P}_k^{\Downarrow}(e')$.

The following is immediate from the definition of the chain $\left\{\mathfrak{P}_k^{\Downarrow}(e)\right\}_{k=0}^{\infty}$ and the fact that $\mathtt{rand}\, \underline{n}$ reduces with uniform probability.

Lemma 3.4. *Let* e *be a closed term. If* $e \overset{1}{\rightsquigarrow} e'$ *and the reduction is an unfold-fold reduction then* $\mathfrak{P}_{k+1}^{\Downarrow}(e) = \mathfrak{P}_k^{\Downarrow}(e')$. *If the reduction from* e *is a choice reduction, then* $\mathfrak{P}_{k+1}^{\Downarrow}(e) = \frac{1}{|\mathbf{Red}(e)|} \sum_{\pi \in \mathbf{Red}(e)} \mathfrak{P}_k^{\Downarrow}(\ell(\pi))$.

The following proposition is needed to prove adequacy of the logical relation with respect to contextual equivalence. It is analogous to the property used to prove adequacy of step-indexed logical relations for deterministic and nondeterministic languages. Consider the case of may-equivalence. To prove adequacy in this case (cf. [4, Theorem 4.8]) we use the fact that if e may-terminates, then there is a natural number n such that e terminates in n steps. This property does not hold in the probabilistic case, but the property analogous to it that is sufficient to prove adequacy still holds.

Proposition 3.5. *For each* $e \in \mathbf{Tm}$ *we have* $\mathfrak{P}^{\Downarrow}(e) \leq \sup_{k\in\omega}\left(\mathfrak{P}_k^{\Downarrow}(e)\right)$.

Proof. We only give a sketch; the full proof can be found in the Appendix. We use Scott induction on the set $\mathcal{S} = \left\{ f \in \mathcal{F} \mid \forall e, f(e) \leq \sup_{k\in\omega}\left(\mathfrak{P}_k^{\Downarrow}(e)\right)\right\}$. It is easy to see that $\mathcal{S}$ is closed under limits of ω-chains and that $\bot \in \mathcal{S}$ so we only need to show that $\mathcal{S}$ is closed under Φ. We can do this by considering the kinds of reductions from e when considering $\Phi(f)(e)$ for $f \in \mathcal{S}$.

4 Logical, CIU and Contextual Approximation Relations

The contextual and CIU (**c**losed **i**nstantiations of **u**ses [18]) approximations are defined in a way analogous to the one for deterministic programming languages. We require some auxiliary notions. A *type-indexed relation* $\mathcal{R}$ is a set of tuples $(\Delta, \Gamma, e, e', \tau)$ such that $\Delta \vdash \Gamma$ and $\Delta \vdash \tau$ and $\Delta \mid \Gamma \vdash e : \tau$ and $\Delta \mid \Gamma \vdash e' : \tau$. We write $\Delta \mid \Gamma \vdash e \mathrel{\mathcal{R}} e' : \tau$ for $(\Delta, \Gamma, e, e', \tau) \in \mathcal{R}$.

Definition 4.1 (Precongruence). *A type-indexed relation* $\mathcal{R}$ *is* reflexive *if* $\Delta \mid \Gamma \vdash e : \tau$ *implies* $\Delta \mid \Gamma \vdash e \mathrel{\mathcal{R}} e : \tau$. *It is* transitive *if* $\Delta \mid \Gamma \vdash e \mathrel{\mathcal{R}} e' : \tau$ *and* $\Delta \mid \Gamma \vdash e' \mathrel{\mathcal{R}} e'' : \tau$ *implies* $\Delta \mid \Gamma \vdash e \mathrel{\mathcal{R}} e'' : \tau$. *It is* compatible *if it is closed under the term forming rules, e.g.,*[2]

$$\frac{\Delta \mid \Gamma, x{:}\tau_1 \vdash e \mathrel{\mathcal{R}} e' : \tau_2}{\Delta \mid \Gamma \vdash \lambda x.e \mathrel{\mathcal{R}} \lambda x.e' : \tau_1 \to \tau_2} \qquad \frac{\Delta \mid \Gamma \vdash e \mathrel{\mathcal{R}} e' : \mathtt{nat}}{\Delta \mid \Gamma \vdash \mathtt{rand}\, e \mathrel{\mathcal{R}} \mathtt{rand}\, e' : \mathtt{nat}}$$

A precongruence *is a reflexive, transitive and compatible type-indexed relation.*

The compatibility rules guarantee that a compatible relation is sufficiently big, i.e., at least reflexive. In contrast, the notion of adequacy, which relates the operational semantics with the relation, guarantees that it is not too big. In the deterministic case, a relation $\mathcal{R}$ is adequate if when $e \mathrel{\mathcal{R}} e'$ are two related closed terms, then if e terminates so does e'. Here we need to compare probabilities of termination instead, since these are our observations.

Definition 4.2. *A type-indexed relation* $\mathcal{R}$ *is* adequate *if for all* e, e' *such that* $\varnothing \mid \varnothing \vdash e \mathrel{\mathcal{R}} e' : \tau$ *we have* $\mathfrak{P}^{\Downarrow}(e) \leq \mathfrak{P}^{\Downarrow}(e')$.

[2] We only show a few rules, the rest are analogous and can be found in the Appendix.

The *contextual approximation relation*, written $\Delta \mid \Gamma \vdash e \lesssim^{ctx} e' : \tau$, is defined as the *largest adequate precongruence* and the *CIU approximation relation*, written $\Delta \mid \Gamma \vdash e \lesssim^{CIU} e' : \tau$, is defined using evaluation contexts in the usual way, e.g. [18], using $\mathfrak{P}^{\Downarrow}(\cdot)$ for observations. The fact that the largest adequate precongruence exists is proved as in [18].

Logical Relation. We now define the step-indexed logical relation. We present the construction in the elementary way with explicit indexing instead of using a logic with guarded recursion as in [10] to remain self-contained.

Interpretations of types will be defined as decreasing sequences of relations on *typeable* values. For *closed types* τ and σ we define the sets $\mathbf{VRel}(\tau,\sigma)$, $\mathbf{SRel}(\tau,\sigma)$ and $\mathbf{TRel}(\tau,\sigma)$ to be the sets of decreasing sequences of relations on typeable values, evaluation contexts and expressions respectively. The types τ and σ denote the types of the left-hand side and the right-hand side respectively, i.e. if $(v,u) \in \varphi(n)$ for $\varphi \in \mathbf{VRel}(\tau,\sigma)$ then v has type τ and u has type σ. The order relation $\leq$ on these sets is defined pointwise, e.g. for $\varphi, \psi \in \mathbf{VRel}(\tau,\sigma)$ we write $\varphi \leq \psi$ if $\forall n \in \mathbb{N}, \varphi(n) \subseteq \psi(n)$. We implicitly use the inclusion from $\mathbf{VRel}(\tau,\sigma)$ to $\mathbf{TRel}(\tau,\sigma)$. The reason for having relations on values and terms of different types on the left and right-hand sides is so we are able to prove parametricity properties in Section 5.

We define maps $\cdot^{\top}_{\tau,\sigma} : \mathbf{VRel}(\tau,\sigma) \to \mathbf{SRel}(\tau,\sigma)$ and $\cdot^{\bot}_{\tau,\sigma} : \mathbf{SRel}(\tau,\sigma) \to \mathbf{TRel}(\tau,\sigma)$. We usually omit the type indices when they can be inferred from the context. The maps are defined as follows

$$r^{\top}_{\tau,\sigma}(n) = \left\{ (E,E') \;\middle|\; \forall k \leq n, \forall (v,v') \in r(k), \mathfrak{P}^{\Downarrow}_k(E[v]) \leq \mathfrak{P}^{\Downarrow}(E'[v']) \right\}$$

and $r^{\bot}_{\tau,\sigma}(n) = \left\{ (e,e') \;\middle|\; \forall k \leq n, \forall (E,E') \in r(k), \mathfrak{P}^{\Downarrow}_k(E[e]) \leq \mathfrak{P}^{\Downarrow}(E'[e']) \right\}$. Note that we only count steps evaluating the left term in defining $r^{\top}$ and $r^{\bot}$. We write $r^{\top\top} = r^{\top\bot}$ for their composition from $\mathbf{VRel}(\tau,\sigma)$ to $\mathbf{TRel}(\tau,\sigma)$. The function $\cdot^{\top}$ is order-reversing and $\cdot^{\top\top}$ is order-preserving and inflationary.

Lemma 4.3. *Let τ,σ be closed types and $r,s \in \mathbf{VRel}(\tau,\sigma)$. Then $r \leq r^{\top\top}$ and if $r \leq s$ then $s^{\top} \leq r^{\top}$ and $r^{\top\top} \leq s^{\top\top}$.*

For a type-variable context Δ we define $\mathbf{VRel}(\Delta)$ using $\mathbf{VRel}(\cdot,\cdot)$ as

$$\mathbf{VRel}(\Delta) = \left\{ (\varphi_1,\varphi_2,\varphi_r) \;\middle|\; \varphi_1,\varphi_2 \in \mathfrak{T}^{\Delta}, \forall \alpha \in \Delta, \varphi_r(\alpha) \in \mathbf{VRel}(\varphi_1(\alpha),\varphi_2(\alpha)) \right\}$$

where the first two components give syntactic types for the left and right hand sides of the relation and the third component is a relation between those types.

The interpretation of types, $[\![\cdot \vdash \cdot]\!]$ is by induction on the judgement $\Delta \vdash \tau$. For a judgment $\Delta \vdash \tau$ and $\varphi \in \mathbf{VRel}(\Delta)$ we have $[\![\Delta \vdash \tau]\!](\varphi) \in \mathbf{VRel}(\varphi_1(\tau),\varphi_2(\tau))$ where the φ_1 and φ_2 are the first two components of φ and $\varphi_1(\tau)$ denotes substitution. Moreover $[\![\cdot]\!]$ is *non-expansive* in the sense that $[\![\Delta \vdash \tau]\!](\varphi)(n)$ can depend only on the values of $\varphi_r(\alpha)(k)$ for $k \leq n$, see [5] for this metric view of step-indexing. The interpretation of types is defined in Fig. 1. Observe that the value relations are as simple as for a language without probabilistic choice. The crucial difference is hidden in the $\top\top$-closure of value relations.

$$
\begin{aligned}
[\![\Delta \vdash \mathtt{nat}]\!](\varphi)(n) &= \{(\underline{k}, \underline{k}) \mid k \in \mathbb{N}, k > 0\} \\
[\![\Delta \vdash \tau \to \sigma]\!](\varphi)(n) &= \{(\lambda x.e, \lambda y.e') \mid \forall j \leq n, \forall (v,v') \in [\![\Delta \vdash \tau]\!](\varphi)(j), \\
&\qquad ((\lambda x.e)\, v, (\lambda y.e')\, v') \in [\![\Delta \vdash \sigma]\!](\varphi)^{\top\top}(j)\} \\
[\![\Delta \vdash \forall \alpha.\tau]\!](\varphi)(n) &= \{(\Lambda.e, \Lambda.e') \mid \forall \sigma, \sigma' \in \mathfrak{T}, \forall r \in \mathbf{VRel}(\sigma, \sigma'), \\
&\qquad (e, e') \in [\![\Delta, \alpha \vdash \tau]\!](\varphi[\alpha \mapsto r])^{\top\top}(n)\} \\
[\![\Delta \vdash \exists \alpha.\tau]\!](\varphi)(n) &= \{(\mathtt{pack}\, v, \mathtt{pack}\, v') \mid \exists \sigma, \sigma' \in \mathfrak{T}, \exists r \in \mathbf{VRel}(\sigma, \sigma'), \\
&\qquad (v, v') \in [\![\Delta, \alpha \vdash \tau]\!](\varphi[\alpha \mapsto r])(n)\} \\
[\![\Delta \vdash \mu \alpha.\tau]\!](\varphi)(0) &= \mathbf{Val}(\varphi_1(\mu\alpha.\tau)) \times \mathbf{Val}(\varphi_2(\mu\alpha.\tau)) \\
[\![\Delta \vdash \mu \alpha.\tau]\!](\varphi)(n+1) &= \{(\mathtt{fold}\, v, \mathtt{fold}\, v') \mid \\
&\qquad (v, v') \in [\![\Delta, \alpha \vdash \tau]\!](\varphi[\alpha \mapsto [\![\Delta \vdash \mu\alpha.\tau]\!](\varphi)])(n)\}
\end{aligned}
$$

Fig. 1. Interpretation of types. The cases for sum and product types are in Appendix.

Context extension lemmas. To prove soundness and completeness we need lemmas stating how extending evaluation contexts preserves relatedness. We only show the case for `rand`. The rest are similarly simple.

Lemma 4.4. *Let $n \in \mathbb{N}$. If $(E, E') \in [\![\Delta \vdash \mathtt{nat}]\!](\varphi)^{\top}(n)$ are related evaluation contexts then $(E \circ (\mathtt{rand}\, []), E' \circ (\mathtt{rand}\, [])) \in [\![\Delta \vdash \mathtt{nat}]\!](\varphi)^{\top}(n)$.*

Proof. Let $n \in \mathbb{N}$ and $(v, v') \in [\![\Delta \vdash \tau]\!](\varphi)(n)$. By construction we have $v = v' = \underline{m}$ for some $m \in \mathbb{N}$, $m \geq 1$. Let $k \leq n$. If $k = 0$ the result is immediate, so assume $k = \ell + 1$. Using Lemma 3.4 we have $\mathfrak{P}_k^{\Downarrow}(E[\mathtt{rand}\, \underline{m}]) = \frac{1}{m} \sum_{i=1}^{m} \mathfrak{P}_\ell^{\Downarrow}(E[\underline{i}])$ and using the assumption $(E, E') \in [\![\Delta \vdash \mathtt{nat}]\!](\varphi)^{\top}(n)$, the fact that $k \leq n$ and monotonicity in the step-index the latter term is less than $\frac{1}{m} \sum_{i=1}^{m} \mathfrak{P}^{\Downarrow}(E'[\underline{i}])$ which by definition of $\mathfrak{P}^{\Downarrow}(\cdot)$ is equal to $\mathfrak{P}^{\Downarrow}(E'[\mathtt{rand}\, \underline{m}])$.

We define the logical approximation relation for open terms given the interpretations of types in Fig. 1. We define $\Delta \mid \Gamma \vdash e \lesssim^{log} e' : \tau$ to mean

$$\forall n \in \mathbb{N}, \forall \varphi \in \mathbf{VRel}(\Delta), \forall (\gamma, \gamma') \in [\![\Delta \vdash \Gamma]\!](\varphi)(n), (e\gamma, e'\gamma) \in [\![\Delta \vdash \tau]\!]\varphi^{\top\top}(n)$$

Here $[\![\Delta \vdash \Gamma]\!]$ is the obvious extension of interpretation of types to interpretation of contexts which relates substitutions, mapping variables to values. We have

Proposition 4.5 (Fundamental Property). *The logical approximation relation $\lesssim^{log}$ is compatible. In particular it is reflexive.*

Proof. The proof is a simple consequence of the context extension lemmas. We show the case for `rand`. We have to show that $\Delta \mid \Gamma \vdash e \lesssim^{log} e' : \mathtt{nat}$ implies $\Delta \mid \Gamma \vdash \mathtt{rand}\, e \lesssim^{log} \mathtt{rand}\, e' : \mathtt{nat}$. Let $n \in \mathbb{N}$, $\varphi \in \mathbf{VRel}(\Delta)$ and $(\gamma, \gamma') \in [\![\Delta \vdash \Gamma]\!](\varphi)(n)$. Let $f = e\gamma$ and $f' = e'\gamma'$. Then our assumption gives us $(f, f') \in [\![\Delta \vdash \mathtt{nat}]\!](\varphi)^{\top\top}(n)$ and we are to show $(\mathtt{rand}\, f, \mathtt{rand}\, f') \in [\![\Delta \vdash \mathtt{nat}]\!](\varphi)^{\top\top}(n)$. Let $j \leq n$ and $(E, E') \in [\![\Delta \vdash \mathtt{nat}]\!](\varphi)^{\top}(j)$. Then from Lemma 4.4 we have $(E \circ (\mathtt{rand}\, []), E' \circ (\mathtt{rand}\, [])) \in [\![\Delta \vdash \mathtt{nat}]\!](\varphi)^{\top}(j)$ which suffices by the definition of the orthogonality relation and the assumption $(f, f') \in [\![\Delta \vdash \mathtt{nat}]\!](\varphi)^{\top\top}(n)$.

We now want to relate logical, CIU and contextual approximation relations.

Corollary 4.6. *Logical approximation relation* $\lesssim^{log}$ *is adequate.*

Proof. Assume $\varnothing \mid \varnothing \vdash e \lesssim^{log} e' : \tau$. We are to show that $\mathfrak{P}^{\Downarrow}(e) \leq \mathfrak{P}^{\Downarrow}(e')$. Straight from the definition we have $\forall n \in \mathbb{N}, (e, e') \in [\![\varnothing \vdash \tau]\!]^{\top\top}(n)$. The empty evaluation context is always related to itself (at any type). This implies $\forall n \in \mathbb{N}, \mathfrak{P}_n^{\Downarrow}(e) \leq \mathfrak{P}^{\Downarrow}(e')$ which further implies (since the right-hand side is independent of n) that $\sup_{n\in\omega}\left(\mathfrak{P}_n^{\Downarrow}(e)\right) \leq \mathfrak{P}^{\Downarrow}(e')$. Using Proposition 3.5 we thus have $\mathfrak{P}^{\Downarrow}(e) \leq \sup_{n\in\omega}\left(\mathfrak{P}_n^{\Downarrow}(e)\right) \leq \mathfrak{P}^{\Downarrow}(e')$ concluding the proof.

We now have that the logical relation is adequate and compatible. This does not immediately imply that it is contained in the contextual approximation relation, since we do not know that it is transitive. However we have the following lemma where by transitive closure we mean that for each Δ, Γ and τ we take the transitive closure of the relation $\{(e, e') \mid \Delta \mid \Gamma \vdash e \lesssim^{log} e' : \tau\}$. This is another type-indexed relation.

Lemma 4.7. *The transitive closure of* $\lesssim^{log}$ *is compatible and adequate.*

Proof. Transitive closure of an adequate relation is adequate. Similarly the transitive closure of a compatible and *reflexive* relation (in the sense of Definition 4.1) is again compatible (and reflexive).

Theorem 4.8 (CIU Theorem). *The relations* $\lesssim^{log}$, $\lesssim^{CIU}$ *and* $\lesssim^{ctx}$ *coincide.*

Proof. It is standard (e.g. [18]) that $\lesssim^{ctx}$ is included in $\lesssim^{CIU}$. We show that the logical approximation relation is contained in the CIU approximation relation in the standard way for biorthogonal step-indexed logical relations. To see that $\lesssim^{log}$ is included in $\lesssim^{ctx}$ we have by Lemma 4.7 that the transitive closure of $\lesssim^{log}$ is an adequate precongruence, thus included in $\lesssim^{ctx}$. And $\lesssim^{log}$ is included in the transitive closure of $\lesssim^{log}$. Corollary A.13 in the appendix completes the cycle of inclusions.

Using the logical relation and Theorem 4.8 we can prove some extensionality properties. The proofs are standard and can be found in the Appendix.

Lemma 4.9 (Functional Extensionality for Values). *Suppose* $\tau, \sigma \in \mathfrak{T}(\Delta)$ *and let* f *and* f' *be two* values *of type* $\tau \to \sigma$ *in context* $\Delta \mid \Gamma$. *If for all* $u \in \mathbf{Val}(\tau)$ *we have* $\Delta \mid \Gamma \vdash f\, u \lesssim^{ctx} f'\, u : \sigma$ *then* $\Delta \mid \Gamma \vdash f \lesssim^{ctx} f' : \tau \to \sigma$.

The extensionality for *expressions*, as opposed to only *values*, of function type does not hold in general due to the presence of choice reductions. See Remark 5.2 for an example. We also have extensionality for *values* of universal types.

Lemma 4.10 (Extensionality for the Universal Type). *Let* $\tau \in \mathfrak{T}(\Delta, \alpha)$ *be a type. Let* f, f' *be two* values *of type* $\forall\alpha.\tau$ *in context* $\Delta \mid \Gamma$. *If for all closed types* σ *we have* $\Delta \mid \Gamma \vdash f[] \lesssim^{ctx} f'[] : \tau[\sigma/\alpha]$ *then* $\Delta \mid \Gamma \vdash f \lesssim^{ctx} f' : \forall\alpha.\tau$.

5 Examples

We now use our logical relation to prove some example equivalences. We show two examples involving polymorphism. In the Appendix we show additional examples. In particular we show the correctness of von Neumann's procedure for generating a fair sequence of coin tosses from an unfair coin. That example in particular shows how the use of biorthogonality allows us to "externalize" the reasoning to arithmetic manipulations.

We first define $\mathtt{fix} \;:\; \forall\alpha,\beta.((\alpha\to\beta)\to(\alpha\to\beta)) \;\to\; (\alpha\to\beta)$ be the term $\Lambda.\Lambda.\lambda f.\lambda z.\delta_f(\mathtt{fold}\,\delta_f)\,z$ where δ_f is the term $\lambda y.\mathtt{let}\,y' = \mathtt{unfold}\,y\,\mathtt{in}\,f\,(\lambda x.y'\,y\,x)$. This is a call-by-value fixed-point combinator. We also write $e_1 \oplus e_2$ for the term $\mathtt{if}_1\ \mathtt{rand}\,\underline{2}\ \mathtt{then}\ e_1\ \mathtt{else}\ e_2$. Note that the choice is made before evaluating e_i's.

We characterize inhabitants of a polymorphic type and show a free theorem. For the former, we need to know which real numbers can be probabilities of termination of programs. Recall that a real number r is *left-computable* if there exists a *computable* increasing (not necessarily strictly) sequence $\{q_n\}_{n\in\omega}$ of *rational numbers* such that $r = \sup_{n\in\omega} q_n$. In Appendix B we prove

Proposition 5.1. *For any expression e, $\mathfrak{P}^{\Downarrow}(e)$ is a left-computable real number and for any left-computable real number r in the interval $[0,1]$ there is a closed term e_r of type $\mathbf{1}\to\mathbf{1}$ such that $\mathfrak{P}^{\Downarrow}(e_r\,\langle\rangle) = r$.*

Inhabitants of the Type $\forall\alpha.\alpha\to\alpha$. In this section we use further syntactic sugar for sequencing. When $e, e' \in \mathbf{Tm}$ are closed terms we write $e; e'$ for $(\lambda\_.e')\,e$, i.e. first run e, ignore the result and then run e'. We will need the property that for all terms $e, e' \in \mathbf{Tm}$, $\mathfrak{P}^{\Downarrow}(e; e') = \mathfrak{P}^{\Downarrow}(e)\cdot\mathfrak{P}^{\Downarrow}(e')$. The proof is by Scott induction and can be found in the Appendix.

Using Proposition 5.1 we have for each left-computable real r in the interval $[0,1]$ an inhabitant t_r of the type $\forall\alpha.\alpha\to\alpha$ given by $\Lambda.\lambda x.e_r\,\langle\rangle; x$.

We now show that these are the only inhabitants of $\forall\alpha.\alpha\to\alpha$ of the form $\Lambda.\lambda x.e$. Given such an inhabitant let $r = \mathfrak{P}^{\Downarrow}(e[\langle\rangle/x])$. We know from Proposition 5.1 that r is left-computable.

Given a value v of type τ and $n\in\mathbb{N}$ we define relations $R(n) = \{(\langle\rangle, v)\}$ and $S(n) = \{(v,\langle\rangle)\}$. Note that the relations are independent of n, i.e. R and S are constant relations. By reflexivity of the logical relation and the relational actions of types we have

$$\forall n, (e[\langle\rangle/x], e[v/x]) \in R^{\top\top}(n) \qquad \text{and} \qquad \forall n, (e[v/x], e[\langle\rangle/x]) \in S^{\top\top}(n) \quad (1)$$

from which we conclude that $\mathfrak{P}^{\Downarrow}(e[\langle\rangle/x]) = \mathfrak{P}^{\Downarrow}(e[v/x])$. We now show that v and $e[v/x]$ are CIU-equivalent. Let $E \in \mathbf{Stk}(\tau)$ be an evaluation context. Let $q = \mathfrak{P}^{\Downarrow}(E[v])$. Define the evaluation context $E' = -; e_q\,\langle\rangle$. Then $(E, E') \in S^{\top}(n)$ for all n which then means, using (1) and Proposition 3.5, that $\mathfrak{P}^{\Downarrow}(E[e[v/x]]) \le \mathfrak{P}^{\Downarrow}(E'[e[\langle\rangle/x]])$. We then have

$$\mathfrak{P}^{\Downarrow}(E'[e[\langle\rangle/x]]) = \mathfrak{P}^{\Downarrow}(e[\langle\rangle/x])\cdot\mathfrak{P}^{\Downarrow}(e_q\,\langle\rangle) = r\cdot\mathfrak{P}^{\Downarrow}(E[v])$$

and so $\mathfrak{P}^{\Downarrow}(E[e[v/x]]) \le r\cdot\mathfrak{P}^{\Downarrow}(E[v])$.

Similarly we have $(E', E) \in R^{\top}(n)$ for all n which implies $\mathfrak{P}^{\Downarrow}(E[e[v/x]]) \geq \mathfrak{P}^{\Downarrow}(E'[e[\langle\rangle/x]])$. We also have $\mathfrak{P}^{\Downarrow}(E'[e[\langle\rangle/x]]) = r \cdot \mathfrak{P}^{\Downarrow}(E[v])$.

So we have proved $\mathfrak{P}^{\Downarrow}(E[e[v/x]]) = r \cdot \mathfrak{P}^{\Downarrow}(E[v]) = \mathfrak{P}^{\Downarrow}(e[v/x]) \cdot \mathfrak{P}^{\Downarrow}(E[v])$. It is easy to show by Scott induction, that $\mathfrak{P}^{\Downarrow}(E[t_r[]\, v]) = \mathfrak{P}^{\Downarrow}(e_r\, \langle\rangle) \cdot \mathfrak{P}^{\Downarrow}(E[v])$. We have thus shown that for any value v, the terms $e[v/x]$ and $\mathfrak{P}^{\Downarrow}(t_r[]\, v)$ are CIU-equivalent. Using Theorem 4.8 and Lemmas 4.10 and 4.9 we conclude that the terms $\forall\alpha.\lambda x.e$ and t_r are contextually equivalent.

Remark 5.2. Unfortunately we cannot so easily characterize general values of the type $\forall\alpha.\alpha \to \alpha$, that is, those not of the form $\Lambda.v$ for a value v. Consider the term $\Lambda.t_{\frac{1}{2}} \oplus t_{\frac{1}{3}}$. It is a straightforward calculation that for any evaluation context E and value v, $\mathfrak{P}^{\Downarrow}\left(E\left[\left(t_{\frac{1}{2}} \oplus t_{\frac{1}{3}}\right)\, v\right]\right) = \frac{5}{12}\mathfrak{P}^{\Downarrow}(E[v]) = \mathfrak{P}^{\Downarrow}\left(E\left[t_{\frac{5}{12}}\, v\right]\right)$ thus if $\Lambda.t_{\frac{1}{2}} \oplus t_{\frac{1}{3}}$ is equivalent to any $\Lambda.t_r$ it must be $\Lambda.t_{\frac{5}{12}}$.

Let E be the evaluation context $E = \mathtt{let}\ f\ = -[]\ \mathtt{in}\ \mathtt{let}\ x\ = f\,\langle\rangle\ \mathtt{in}\ f\,\langle\rangle$. We compute $\mathfrak{P}^{\Downarrow}\left(E\left[\Lambda.t_{\frac{1}{2}} \oplus t_{\frac{1}{3}}\right]\right) = \frac{13}{72}$ and $\mathfrak{P}^{\Downarrow}\left(E\left[\Lambda.t_{\frac{5}{12}}\right]\right) = \frac{25}{144}$ showing that $\Lambda.t_{\frac{1}{2}} \oplus t_{\frac{1}{3}}$ is *not* equivalent to $\Lambda.t_{\frac{5}{12}}$.

This example also shows that extensionality for *expressions*, as opposed to *values*, of function type does not hold. The reason is that probabilistic choice is a computational effect and so it matters how many times we evaluate the term and this is what the constructed evaluation context uses to distinguish the terms.

A Free Theorem for Lists. Let τ be a type and α not free in τ. We write $[\tau]$ for the type of lists $\mu\alpha.(\mathbf{1}+\tau\times\alpha)$, $\mathtt{nil}$ for the empty list and $\mathtt{cons} : \forall\alpha.\alpha \to [\alpha] \to [\alpha]$ for the other constructor $\mathtt{cons} = \Lambda.\lambda x.\lambda xs.\mathtt{fold}\,(\mathtt{inr}\,\langle x, xs\rangle)$. The function $\mathtt{map}$ of type $\forall\alpha.\forall\beta.(\alpha \to \beta) \to [\alpha] \to [\beta]$ is the function applying the given function to all elements of the list in order. Additionally, we define composition of terms $f \circ g$ as the term $\lambda x.f(g(x))$ (for x not free in f and g).

We will now show that any term m of type $\forall\alpha.\forall\beta.(\alpha \to \beta) \to [\alpha] \to [\beta]$ equivalent to a term of the form $\Lambda.\Lambda.\lambda x.e$ satisfies $m[][]\,(f \circ g) =^{ctx} m[][]f \circ \mathtt{map}[][]\,g$ for all *values* f and all *deterministic and terminating* g. By this we mean that for each value v in the domain of g, there exists a *value* u in the codomain of g, such that $g\,v =^{ctx} u$. For instance, if g reduces without using choice reductions and is terminating, then g is deterministic. There are other functions that are also deterministic and terminating, though, for instance $\lambda x.\langle\rangle \oplus \langle\rangle$. In the Appendix we show that these restrictions are not superfluous.

So let m be a closed term of type $\forall\alpha.\forall\beta.(\alpha \to \beta) \to [\alpha] \to [\beta]$ and suppose further that m is equivalent to a term of the form $\Lambda.\Lambda.\lambda x.e$. Let $\tau, \sigma, \rho \in \mathfrak{T}$ be closed types and $f \in \mathbf{Val}\,(\sigma \to \rho)$ and $g \in \mathbf{Tm}\,(\tau \to \sigma)$ be a deterministic and terminating function. Then

$$\varnothing \mid \varnothing \vdash m[][](f \circ g) =^{ctx} m[][]f \circ \mathtt{map}[][]g : [\tau] \to [\rho].$$

We prove two approximations separately, starting with $\lesssim^{ctx}$. We use Theorem 4.8 multiple times. We have $\alpha, \beta \mid \varnothing \vdash m[][] : (\alpha \to \beta) \to [\alpha] \to [\beta]$. Let $R = \lambda n.\{(v,u) \mid g\, v =^{ctx} u\}$ be a member of $\mathbf{VRel}\,(\tau,\sigma)$ and $S \in \mathbf{VRel}\,(\rho,\rho)$ be the constant identity relation on $\mathbf{Val}\,(\rho)$. Let φ map α to R and β to S. Proposition 4.5 gives $(m[][], m[][]) \in [\![(\alpha \to \beta) \to [\alpha] \to [\beta]]\!]\,(\varphi)^{\top\top}(n)$ for all $n \in \mathbb{N}$.

We first claim that $(f \circ g, f) \in [\![\alpha \to \beta]\!]\,(\varphi)(n)$ for all $n \in \mathbb{N}$. Since f is a value and has a type, it must be of the form $\lambda x.e$ for some x and e. Take $j \in \mathbb{N}$, related values $(v,u) \in r(j)$, $k \le j$ and $(E,E') \in S^{\top}(k)$ two related evaluation contexts. We then have $\mathfrak{P}^{\Downarrow}\,(E'[f\,u]) = \mathfrak{P}^{\Downarrow}\,(E'[f(g\,v)])$ by Theorem 4.8 and the definition of relation R. Using the results about $\mathfrak{P}^{\Downarrow}_k\,(\cdot)$ and $\mathfrak{P}^{\Downarrow}\,(\cdot)$ proved in Section C in the Appendix this gives us

$$\mathfrak{P}^{\Downarrow}_k\,(E[f(g(v))]) \le \sum_{\pi : f(g(v)) \rightsquigarrow^* w} \mathfrak{W}\,(\pi)\,\mathfrak{P}^{\Downarrow}_k\,(E[w]) \le \sum_{\pi : f(g(v)) \rightsquigarrow^* w} \mathfrak{W}\,(\pi)\,\mathfrak{P}^{\Downarrow}\,(E'[w])$$

and the last term is equal to $\mathfrak{P}^{\Downarrow}\,(E'[f(g\,v)])$ which is equal to $\mathfrak{P}^{\Downarrow}\,(E'[f\,u])$.

From this we can conclude $(m[][]\,(f \circ g), m[][]\,f) \in [\![[\alpha] \to [\beta]]\!]\,(\varphi)^{\top\top}(n)$ for all $n \in \mathbb{N}$. Note that we have *not yet* used the fact that g is deterministic and terminating. We do so now.

Let xs be a list of elements of type τ. Then induction on the length of xs, using the assumption on g, we can derive that there exists a list ys of elements of type σ, such that $\mathtt{map}[][]\,g\,xs =^{ctx} ys$ and $(xs, ys) \in [\![[\alpha]]\!]\,(\varphi)(n)$ for all n.

This gives us $(m[][]\,(f \circ g)\,xs, m[][]\,f\,ys) \in [\![[\beta]]\!]\,(\varphi)^{\top\top}(n)$ for all $n \in \mathbb{N}$. Since the relation S is the identity relation we have for all evaluation contexts E of a suitable type, $(E,E) \in S^{\top}(n)$ for all n, which gives

$$m[][]\,(f \circ g)\,xs \lesssim^{CIU} m[][]\,f\,ys =^{ctx} m[][]\,f\,(\mathtt{map}[][]\,g\,xs) =^{ctx} (m[][]\,f \circ \mathtt{map}[][]\,g)\,xs$$

where the last equality holds because β-reduction is an equivalence.

We now conclude by using the fact that m is (equivalent to) a term of the form $\Lambda.\Lambda.\lambda x.e$ and use Lemma 4.9 to conclude $m[][]\,(f \circ g) \lesssim^{ctx} m[][]\,f \circ \mathtt{map}[][]\,g$.

For the other direction, we proceed analogously. The relation for β remains the identity relation, and the relation for R for α is $\{(v,u) \mid v =^{ctx} g\,u\}$.

6 Extension to References

We now sketch the extension of $\mathbf{F}^{\mu,\oplus}$ to include dynamically allocated references. For simplicity we add ground store only, so we do not have to solve a domain equation giving us the space of semantic types and worlds [1]. We show an equivalence using state and probabilistic choice which shows that the addition of references to the language is orthogonal to the addition of probabilistic choice. We conjecture that the extension with *higher-order* dynamically allocated references can be done as in earlier work on step-indexed logical relations [11].

We extend the language by adding the type $\mathtt{ref\,nat}$ and extend the grammar of terms with $\ell \mid \mathtt{ref}\,e \mid e_1 := e_2 \mid !e$ with ℓ being locations.

To model allocation we need to index the interpretation of types by worlds. To keep things simple a world $w \in \mathcal{W}$ is partial bijection f on locations together with, for each pair of locations $(\ell_1, \ell_2) \in f$, a relation R on numerals. We write $(\ell_1, \ell_2, R) \in w$ when the partial bijection in w relates ℓ_1 and ℓ_2 and R is the relation assigned to the pair (ℓ_1, ℓ_2). Technically, worlds are relations of type $\mathtt{Loc}^2 \times \mathcal{P}(\{\underline{n} \mid n \in \mathbb{N}\})$ satisfying the conditions described above.

The operational semantics has to be extended to include heaps, which are modeled as finite maps from locations to numerals. A pair of heaps (h_1, h_2) satisfies the world w, written $(h_1, h_2) \in \lfloor w \rfloor$, when $\forall (\ell_1, \ell_2, R) \in w, (h_1(\ell_1), h_2(\ell_2)) \in R$. The interpretation of types is then extended to include worlds. The denotation of a type is now an element of $\mathcal{W} \overset{mon}{\to} \mathbf{VRel}(\cdot, \cdot)$ where the order on $\mathcal{W}$ is inclusion. Let $\mathbf{WRel}(\tau, \tau') = \mathcal{W} \overset{mon}{\to} \mathbf{VRel}(\tau, \tau')$. We define $[\![\Delta \vdash \mathtt{ref\ nat}]\!](\varphi)(n)$ as $\lambda w. \left\{(\ell_1, \ell_2) \;\middle|\; (\ell_1, \ell_2, =) \in w\right\}$ where $=$ is the equality relation on numerals.

The rest of the interpretation stays the same, apart from some quantification over "future worlds" in the function case to maintain monotonicity. We also need to change the definition of the $\top\top$-closure to use the world satisfaction relation. For $r \in \mathbf{WRel}(\tau, \tau')$ we define an indexed relation (indexed by worlds) $r^\top$ as

$$r^\top(w)(n) \left\{ (E, E') \;\middle|\; \begin{array}{c} \forall w' \geq w, \forall k \leq n, \forall (h_1, h_2) \in \lfloor w' \rfloor, \forall v_1, v_2 \in r(w')(k), \\ \mathfrak{P}_k^{\Downarrow}(\langle h_1, E[v_1] \rangle) \leq \mathfrak{P}^{\Downarrow}(\langle h_2, E[v_2] \rangle) \end{array} \right\}$$

and analogously for $\cdot^\perp$.

We now sketch a proof that two modules, each implementing a counter by using a single internal location, are contextually equivalent. The increment method is special. When called, it chooses, uniformly, whether to increment the counter or not. The two modules differ in the way they increment the counter. One module increments the counter by 1, the other by 2. Concretely, we show that the two counters $\mathtt{pack}\,(\lambda - .\mathtt{ref}\,\underline{1}, \lambda x.!x, \lambda x.\langle\rangle \oplus (x := \mathtt{S}\,!x))$ and $\mathtt{pack}\,(\lambda - .\mathtt{ref}\,\underline{2}, \lambda x.!x\ \mathtt{div}\ \underline{2}, \lambda x.\langle\rangle \oplus (x := \mathtt{S}\,(\mathtt{S}\,!x)))$ are contextually equivalent at type $\exists\alpha.(\mathbf{1} \to \alpha) \times (\alpha \to \mathtt{nat}) \times (\alpha \to \mathbf{1})$. We have used $\mathtt{div}$ for the division function on numerals which can easily be implemented.

The interpretation of existentials $[\![\Delta \vdash \exists\alpha.\tau]\!](\varphi)(n)$ now maps world w to

$$\left\{ (\mathtt{pack}\,v, \mathtt{pack}\,v') \;\middle|\; \begin{array}{l} \exists \sigma, \sigma' \in \mathfrak{T}, \exists r \in \mathbf{WRel}(\sigma, \sigma'), \\ (v, v') \in [\![\Delta, \alpha \vdash \tau]\!](\varphi[\alpha \mapsto r])(w)(n) \end{array} \right\}$$

To prove the counters are contextually equivalent we show them directly related in the value relation. We choose the types σ and σ' to be $\mathtt{ref\,nat}$ and the relation r to be $\lambda w. \left\{(\ell_1, \ell_2) \;\middle|\; (\ell_1, \ell_2, \left\{(\underline{n}, \underline{2 \cdot n}) \;\middle|\; n \in \mathbb{N}\right\}) \in w\right\}$. We now need to check all three functions to be related at the value relation.

First, the allocation functions. We only show one approximation, the other is completely analogous. Concretely, we show that for any $n \in \mathbb{N}$ and any world $w \in \mathcal{W}$ we have $(\lambda - .\mathtt{ref}\ \underline{1}, \lambda - .\mathtt{ref}\ \underline{2}) \in [\![\mathbf{1} \to \alpha]\!](r)(w)(n)$. Let $n \in \mathbb{N}$ and $w \in \mathcal{W}$. Take $w' \geq w$ and related arguments v, v' at type $\mathbf{1}$. We know by construction that $v = v' = \langle\rangle$ so we have to show that $(\mathtt{ref}\ \underline{1}, \mathtt{ref}\ \underline{2}) \in [\![\alpha]\!](r)^{\top\top}(w')(n)$.

Let $w'' \geq w'$ and $j \leq n$ and take two related evaluation contexts (E, E') at $[\![\alpha]\!]\,(r)^{\top}(w'')(j)$ and $(h, h') \in \lfloor w'' \rfloor$. Let $\ell \notin \mathrm{dom}\,(h)$ and $\ell' \notin \mathrm{dom}\,(h')$. We have

$$\mathfrak{P}_j^{\Downarrow}\left(\langle h, E[\mathtt{ref}\ \underline{1}]\rangle\right) = \mathfrak{P}_j^{\Downarrow}\left(\langle h\,[\ell \mapsto \underline{1}], E[\ell]\rangle\right)$$

and $\mathfrak{P}^{\Downarrow}\left(\langle h', E'[\mathtt{ref}\ \underline{2}]\rangle\right) = \mathfrak{P}^{\Downarrow}\left(\langle h'\,[\ell' \mapsto \underline{2}], E'[\ell']\rangle\right)$.

Let w''' be w'' extended with (ℓ, ℓ', r). Then the extended heaps are in $\lfloor w''' \rfloor$ and $w''' \geq w''$. Thus E and E' are also related at w''' by monotonicity. Similarly we can prove that $(\ell, \ell') \in [\![\alpha]\!]\,(r)(j)(w''')$. This then allows us to conclude $\mathfrak{P}_j^{\Downarrow}\left(\langle h\,[\ell \mapsto \underline{1}], E[\ell]\rangle\right) \leq \mathfrak{P}^{\Downarrow}\left(\langle h'\,[\ell' \mapsto \underline{2}], E'[\ell']\rangle\right)$ which concludes the proof.

Lookup is simple so we omit it. Update is more interesting. Let $n \in \mathbb{N}$ and $w \in \mathcal{W}$. Let ℓ and ℓ' be related at $[\![\alpha]\!]\,(r)(w)(n)$. We need to show that $\left(\langle\rangle \oplus (\ell := \mathtt{S}\,!\ell),\, \langle\rangle \oplus (\ell' := \mathtt{S}\,(\mathtt{S}\,!\ell'))\right) \in [\![\mathbf{1}]\!]\,(r)^{\top\top}(w)(n)$. Take $w' \geq w$, $j \leq n$ and $(h, h') \in \lfloor w' \rfloor$. Take related evaluation contexts E and E' at w' and j. We have

$$\mathfrak{P}_j^{\Downarrow}\left(\langle h, E\left[\langle\rangle \oplus (\ell := \mathtt{S}\,!\ell)\right]\rangle\right) = \tfrac{1}{2}\mathfrak{P}_j^{\Downarrow}\left(\langle h, E\left[\langle\rangle\right]\rangle\right) + \tfrac{1}{2}\mathfrak{P}_j^{\Downarrow}\left(\langle h, E\left[\ell := \mathtt{S}\,!\ell\right]\rangle\right)$$
$$\mathfrak{P}^{\Downarrow}\left(\langle h', E'\left[\langle\rangle \oplus (\ell' := \mathtt{S}\,\mathtt{S}\,!\ell')\right]\rangle\right) = \tfrac{1}{2}\mathfrak{P}^{\Downarrow}\left(\langle h', E'\left[\langle\rangle\right]\rangle\right) + \tfrac{1}{2}\mathfrak{P}^{\Downarrow}\left(\langle h', E'\left[\ell' := \mathtt{S}\,\mathtt{S}\,!\ell'\right]\rangle\right)$$

Since ℓ and ℓ' are related at $[\![\alpha]\!]\,(r)(w)(n)$ and $w' \geq w$ and $(h, h') \in \lfloor w' \rfloor$ we know that $h(\ell) = \underline{m}$ and $h'(\ell') = \underline{2 \cdot m}$ for some $m \in \mathbb{N}$.

Thus $\mathfrak{P}_j^{\Downarrow}\left(\langle h, E\left[\ell := \mathtt{S}\,!\ell\right]\rangle\right) = \mathfrak{P}_j^{\Downarrow}\left(\langle h_1, E[\langle\rangle]\rangle\right)$ where $h_1 = h\,[\ell \mapsto \underline{m+1}]$. Also $\mathfrak{P}^{\Downarrow}\left(\langle h', E'\left[\ell' := \mathtt{S}\,\mathtt{S}\,!\ell'\right]\rangle\right) = \mathfrak{P}^{\Downarrow}\left(\langle h_2, E'[\langle\rangle]\rangle\right)$ where $h_2 = h'\left[\ell' \mapsto \underline{2 \cdot (m+1)}\right]$. The fact that h_1 and h_2 are still related concludes the proof.

The above proof shows that reasoning about examples involving state and choice is possible and that the two features are largely orthogonal.

7 Conclusion

We have constructed a step-indexed logical relation for a higher-order language with probabilistic choice. In contrast to earlier work, our language also features impredicative polymorphism and recursive types. We also show how to extend our logical relation to a language with dynamically allocated local state. In future work, we will explore whether the step-indexed technique can be used for developing models of program logics for probabilistic computation that support reasoning about more properties than just contextual equivalence. We are also interested in including primitives for continuous probability distributions.

Acknowledgments. We thank Filip Sieczkowski, Kasper Svendsen and Thomas Dinsdale-Young for discussions of various aspects of this work and the reviewers for their comments.

This research was supported in part by the ModuRes Sapere Aude Advanced Grant from The Danish Council for Independent Research for the Natural Sciences (FNU) and in part by Microsoft Research through its PhD Scholarship Programme.

References

1. Ahmed, A.: Semantics of Types for Mutable State. Ph.D. thesis, Princeton University (2004)
2. Ahmed, A.: Step-indexed syntactic logical relations for recursive and quantified types. In: Sestoft, P. (ed.) ESOP 2006. LNCS, vol. 3924, pp. 69–83. Springer, Heidelberg (2006)
3. Appel, A.W., McAllester, D.: An indexed model of recursive types for foundational proof-carrying code. ACM Transactions on Programming Languages and Systems 23(5) (2001)
4. Birkedal, L., Bizjak, A., Schwinghammer, J.: Step-indexed relational reasoning for countable nondeterminism. Logical Methods in Computer Science 9(4) (2013)
5. Birkedal, L., Reus, B., Schwinghammer, J., Støvring, K., Thamsborg, J., Yang, H.: Step-indexed kripke models over recursive worlds. In: Proceedings of the 38th Symposium on Principles of Programming Languages, pp. 119–132. ACM (2011)
6. Bizjak, A., Birkedal, L.: Step-indexed logical relations for probability. arXiv:1501.02623 [cs.LO] (2015), long version of this paper
7. Crubillé, R., Dal Lago, U.: On probabilistic applicative bisimulation and call-by-value λ-calculi. In: Shao, Z. (ed.) ESOP 2014 (ETAPS). LNCS, vol. 8410, pp. 209–228. Springer, Heidelberg (2014)
8. Dal Lago, U., Sangiorgi, D., Alberti, M.: On coinductive equivalences for higher-order probabilistic functional programs. In: Proceedings of 41st Symposium on Principles of Programming Languages, pp. 297–308. ACM (2014)
9. Danos, V., Harmer, R.S.: Probabilistic game semantics. ACM Transactions on Computational Logic 3(3) (2002)
10. Dreyer, D., Ahmed, A., Birkedal, L.: Logical step-indexed logical relations. Logical Methods in Computer Science 7(2) (2011)
11. Dreyer, D., Neis, G., Birkedal, L.: The impact of higher-order state and control effects on local relational reasoning. Journal of Functional Programming 22(4-5 special issue), 477–528 (2012)
12. Ehrhard, T., Pagani, M., Tasson, C.: The computational meaning of probabilistic coherence spaces. In: Proceedings of the 26th IEEE Symposium on Logic in Computer Science, pp. 87–96. IEEE (2011)
13. Ehrhard, T., Tasson, C., Pagani, M.: Probabilistic coherence spaces are fully abstract for probabilistic pcf. In: Proceedings of 41st Symposium on Principles of Programming Languages, pp. 309–320. ACM (2014)
14. Johann, P., Simpson, A., Voigtländer, J.: A generic operational metatheory for algebraic effects. In: Proceedings of the 25th Annual IEEE Symposium on Logic in Computer Science, pp. 209–218. IEEE (2010)
15. Jones, C., Plotkin, G.: A probabilistic powerdomain of evaluations. In: Proceedings of the 4th Symposium on Logic in Computer Science, pp. 186–195. IEEE (1989)
16. Lago, U.D., Zorzi, M.: Probabilistic operational semantics for the lambda calculus. RAIRO - Theoretical Informatics and Applications 46 (2012)
17. Pitts, A.M.: Parametric polymorphism and operational equivalence. Mathematical Structures in Computer Science 10(3) (2000)
18. Pitts, A.M.: Typed operational reasoning. In: Pierce, B.C. (ed.) Advanced Topics in Types and Programming Languages, ch. 7. MIT Press (2005)
19. 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. No. 10351 in Dagstuhl Seminar Proceedings (2010)

20. Ramsey, N., Pfeffer, A.: Stochastic lambda calculus and monads of probability distributions. In: Proceedings of the 29th Symposium on Principles of Programming Languages, pp. 154–165. ACM (2002)
21. Saheb-Djahromi, N.: Cpo's of measures for nondeterminism. Theoretical Computer Science 12(1) (1980)