A Model of Countable Nondeterminism in Guarded Type Theory

Aleš Bizjak[1] and Lars Birkedal[1] and Marino Miculan[2]

[1] Aarhus University {abizjak,birkedal}@cs.au.dk
[2] University of Udine marino.miculan@uniud.it

Abstract. We show how to construct a logical relation for countable nondeterminism in a guarded type theory, corresponding to the internal logic of the topos $\mathbf{Sh}(\omega_1)$ of sheaves over ω_1. In contrast to earlier work on abstract step-indexed models, we not only construct the logical relations in the guarded type theory, but also give an internal proof of the adequacy of the model with respect to standard contextual equivalence. To state and prove adequacy of the logical relation, we introduce a new propositional modality. In connection with this modality we show why it is necessary to work in the logic of $\mathbf{Sh}(\omega_1)$.

1 Introduction

Countable nondeterminism arises naturally when modeling properties of concurrent systems or systems with user input, etc. Still, semantic models for reasoning about *must-contextual equivalence* of higher-order programming languages with countable nondeterminism are challenging to construct [3, 7, 1, 10–13, 17]. Recently, it was shown how step-indexed logical relations, indexed over the first uncountable ordinal ω_1, can be used to give a simple model of a higher-order programming language $\mathbf{F}^{\mu,?}$ with recursive types and countable nondeterminism [4], allowing one to reason about must-contextual equivalence. Using step-indexed logical relations is arguably substantially simpler than using other models, but still involves some tedious reasoning about indices, as is characteristic of any concrete step-indexed model.

In previous work [8, 5], the guarded type theory corresponding to the internal logic of the topos $\mathbf{Sh}(\omega)$ of sheaves[3] on ω has been proved very useful for developing abstract accounts of step-indexed models indexed over ω. Such abstract accounts eliminate much of the explicit tedious reasoning about indices. We recall that the internal logic of $\mathbf{Sh}(\omega)$ can be thought of as a logic of discrete time, with time corresponding to ordinals and smaller ordinals being the future. In the application to step-indexed logical relations, the link between steps in the operational semantics and the notion of time provided by the internal logic of $\mathbf{Sh}(\omega)$ is made by defining the operational semantics using guarded recursion [5].

In this paper we show how to construct a logical relation for countable nondeterminism in a guarded type theory GTT corresponding to the internal logic

[3] Considered as sheaves on the topological space ω equipped with the Alexandrov topology.

of the topos $\mathbf{Sh}(\omega_1)$ of sheaves over ω_1. For space reasons we only consider the case of must-equivalence; the case for may-equivalence is similar. In contrast to earlier work on abstract step-indexed models [8, 5], we not only construct the logical relation in the guarded type theory, but also give an internal proof of the adequacy of the model with respect to must-contextual equivalence. To state and prove adequacy of the logical relation we introduce a new propositional modality $\Box$: intuitively, $\Box\varphi$ holds if φ holds at all times. Using this modality we give a logical explanation for why it is necessary to work in the logic of $\mathbf{Sh}(\omega_1)$: a certain logical equivalence involving $\Box$ holds in the internal logic of $\mathbf{Sh}(\omega_1)$ but not in the internal logic of $\mathbf{Sh}(\omega)$ (see Lemma 4).

To model *must*-equivalence, we follow [4] and define the logical relation using *biorthogonality*. Typically, biorthogonality relies on a definition of convergence; in our case, it would be must-convergence. In an abstract account of step-indexed models, convergence would need to be defined by guarded recursion (to show the fundamental lemma). However, that is not possible in the logic of $\mathbf{Sh}(\omega_1)$. There are two ways to understand that. If one considers the natural guarded-recursive definition of convergence,[4] using Löb induction one could show that a non-terminating computation would converge! Another way to understand this issue is in terms of the model. The stratified convergence predicate $\Downarrow_\beta$ from [4] is not a well-defined subobject in $\mathbf{Sh}(\omega_1)$. Intuitively, the reason is that all predicates in GTT are closed wrt. the future (smaller ordinals), but if an expression converges to a value in, say, 15 computation steps, then it does not necessarily converge to a value in 14 steps. Instead we observe that the dual of stratified must-convergence, the stratified *may-divergence*, is a subobject of $\mathbf{Sh}(\omega_1)$ and can easily be defined as a predicate in GTT using guarded recursion. Thus we use the stratified may-divergence predicate to define biorthogonality, modifying the definition accordingly.

The remainder of the paper is organized as follows. In Section 2 we explain the guarded type theory GTT, which we use to define the operational semantics of the higher-order programming language $\mathbf{F}^{\mu,?}$ with countable nondeterminism (Section 3) and to define the adequate logical relation for reasoning about contextual equivalence (Section 4). We include an example to demonstrate how reasoning in the resulting model avoids tedious step-indexing. Finally, in Section 5 we show that the guarded type theory GTT is consistent by providing a model thereof in $\mathbf{Sh}(\omega_1)$. Thus, most of the paper can be read without understanding the details of the model $\mathbf{Sh}(\omega_1)$. For reasons of space, most proofs have been omitted; they can be found in the accompanying technical report [6].

2 The logic GTT

The logic GTT is the internal logic of $\mathbf{Sh}(\omega_1)$. In this section we explain some of the key features of the logic; in the subsequent development we will also use a couple of additional facts, which will be introduced as needed.

[4] must-converge$(e) \leftrightarrow \forall e', e \rightsquigarrow e' \to \triangleright(\text{must-converge}(e'))$.

The logic is an extension of a multisorted intuitionistic higher-order logic with two modalities $\triangleright$ and $\Box$, pronounced "later" and "always" respectively. Types (aka sorts) are ranged over by X, Y; we denote the type of propositions by Ω and the function space from X to Y as Y^X. We write $\mathcal{P}(X) = \Omega^X$ for the type of the power set of X. We think of types as *variable* sets (although in the logic we will not deal with indices explicitly). There is a subset of types which we call *constant sets*; given a set a, we denote by $\Delta(a)$ the type which is constantly equal to a. Constant sets are closed under product and function space. For each type X there is a type $\blacktriangleright X$ and a function symbol $\mathrm{next}^X : X \to \blacktriangleright X$. Intuitively $\blacktriangleright X$ is "one time step later" than the type X, so we can only use it *later*, i.e. after one time step and $\mathrm{next}^X(x)$ freezes x for a time step so it is only available later.

We also single out the space of *total* types. Intuitively, these are the types whose elements at each stage have evolved from some elements from previous stages, i.e. they do not appear out of nowhere.

Definition 1. *For a type X we define* $\mathrm{Total}(X)$ *to mean that* next^X *is surjective*

$$\mathrm{Total}(X) \overset{\triangle}{\leftrightarrow} \forall x : \blacktriangleright X, \exists x' : X, \mathrm{next}^X(x') = x$$

and say that X is total *when* $\mathrm{Total}(X)$ *holds.*

Note that for each X, $\mathrm{Total}(X)$ is a formula of the logic, but Total itself is not a predicate of the logic. Constant sets $\Delta(a)$ for an inhabited a are total. For simplicity, we do not formalize how to construct constant sets. In the following, we shall instead just state for some of the types that we use that they are constant; these facts can be shown using the model in Section 5.

We will adopt the usual "sequent-in-context" judgment of the form $\Gamma \mid \Xi \vdash \varphi$ for saying that the formula φ is a consequence of formulas in Ξ, under the typing context Γ.

The $\triangleright$ modality on *formulas* is used to express that a formula holds only "later", that is, after a time step. More precisely, there is a function symbol $\triangleright : \Omega \to \Omega$ which we extend to formulas by composition. We require $\triangleright$ to satisfy the following properties (Γ is an arbitrary context).

1. (Monotonicity) $\Gamma \mid \varphi \vdash \triangleright\varphi$
2. (Löb induction rule) $\Gamma \mid (\triangleright\varphi \to \varphi) \vdash \varphi$
3. $\triangleright$ commutes over $\top$, $\wedge$, $\to$ and $\vee$ (but does not preserve $\bot$).
4. For all X, Y and φ we have $\Gamma, x : X \mid \exists y : Y, \triangleright\varphi(x, y) \vdash \triangleright(\exists y : Y, \varphi(x, y))$.
5. For all X, Y and φ we have $\Gamma, x : X \mid \triangleright(\forall y : Y, \varphi(x, y)) \vdash \forall y : Y, \triangleright\varphi(x, y)$. The converse entailment in the last rule holds if Y is total.

Following [5, Definition 2.8] we define a notion of contractiveness which will be used to construct unique fixed points of morphisms on total types.

Definition 2. *We define the predicate* Contr *on* Y^X *as*

$$\mathrm{Contr}(f) \overset{\triangle}{\leftrightarrow} \forall x, x' : X, \triangleright(x = x') \to f(x) = f(x')$$

and we say that f is (internally) contractive *if* $\mathrm{Contr}(f)$ *holds.*

Intuitively, a function f is contractive if $f(x)$ *now* depends only on the value of x later, in the future. The following theorem holds in the logic.

Theorem 1 (Internal Banach's fixed point theorem). *Internally, any contractive function f on a total object X has a unique fixed point. More precisely, the following formula is valid in the logic of* $\mathbf{Sh}(\omega_1)$*:*

$$\mathrm{Total}(X) \to \forall f : X^X, \mathrm{Contr}(f) \to \exists! x : X, f(x) = x.$$

We will use Theorem 1 in Section 4 on a function of type $\mathcal{P}(X) \to \mathcal{P}(X)$ for a constant set X. We thus additionally assume that $\mathrm{Total}(\mathcal{P}(X))$ holds for any constant set X.

The $\square$ modality is used to express that a formula holds for all time steps. It is thus analogous to the $\square$ modality in temporal logic. It is defined as the *right* adjoint to the $\neg\neg$-closure operation on formulas and behaves as an interior operator. More precisely, for a formula φ in context Γ, $\square\varphi$ is another formula in context Γ. In contrast to the $\triangleright$ modality, $\square$ on formulas does not arise from a function on Ω and consequently does not commute with substitution, i.e., in general $(\square\varphi)[t/x]$ is not equivalent to $\square(\varphi[t/x])$, although $(\square\varphi)[t/x]$ always implies $\square(\varphi[t/x])$ which is useful for instantiating universally quantified assumptions. Thus, to be precise, we would have to annotate the $\square$ with the context in which it is used. However, restricting to contexts consisting of constant types, $\square$ does commute with substitution and since we will only use it in such contexts we will omit explicit contexts.

The basic rules for the $\square$ modality are the following. In particular, note the first rule which characterizes $\square$ as the right adjoint to the $\neg\neg$-closure.

$$\frac{\Gamma \mid \neg\neg\varphi \vdash \psi}{\Gamma \mid \varphi \vdash \square\psi} \qquad \frac{\Gamma \mid \varphi \vdash \psi}{\Gamma \mid \square\varphi \vdash \square\psi} \qquad \frac{}{\Gamma \mid \square\varphi \vdash \varphi}$$

$$\frac{}{\Gamma \mid \square\varphi \vdash \square\square\varphi} \qquad \frac{}{\Gamma \mid \neg\neg(\square\varphi) \vdash \square\varphi} \qquad \frac{}{\Gamma \mid \neg\neg\varphi \vdash \square(\neg\neg\varphi)}$$

Note that some of the rules can be derived from others. A simple consequence of the rules is that $\neg\neg\varphi \leftrightarrow \square(\neg\neg\varphi)$ and $\neg\neg(\square\varphi) \leftrightarrow \square\varphi$. Thus one way to understand $\square\varphi$ is as the largest predicate that implies φ and is $\neg\neg$-closed.

Proposition 1. *Using the rules for $\square$ stated above we can prove the following in the logic.*

$$\square\top \leftrightarrow \top \text{ and } \square\bot \leftrightarrow \bot \qquad \Gamma \mid \emptyset \vdash \square(\varphi \wedge \psi) \leftrightarrow \square\varphi \wedge \square\psi$$
$$\Gamma \mid \emptyset \vdash \square(\forall x : X, \varphi) \leftrightarrow \forall x : X, \square\varphi \qquad \Gamma \mid \emptyset \vdash \square(\varphi \to \psi) \to \square\varphi \to \square\psi$$

A useful derived introduction rule for the $\square$ modality is the well-known $\square$-introduction rule for S4. It states that if we can prove φ using only $\square$'ed facts, then we can also conclude $\square\varphi$. Formally:

$$\frac{\Gamma \mid \Xi \vdash \varphi}{\Gamma \mid \Xi \vdash \square\varphi} \Xi = \square\varphi_1, \square\varphi_2, \ldots, \square\varphi_n$$

$$\begin{aligned}
\tau &::= \alpha \mid \mathbf{1} \mid \tau_1 \times \tau_2 \mid \tau_1 + \tau_2 \mid \tau_1 \to \tau_2 \mid \mu\alpha.\tau \mid \forall\alpha.\tau \mid \exists\alpha.\tau \\
e &::= x \mid \langle\rangle \mid \langle e_1, e_2\rangle \mid \mathtt{inl}\ e \mid \mathtt{inr}\ e \mid \lambda x.e \mid \Lambda.e \mid \mathtt{pack}\, e \mid \mathtt{unfold}\, e \mid \mathtt{fold}\, e \\
&\quad \mid \mathtt{?} \mid \mathtt{proj}_i\, e \mid e_1\, e_2 \mid \mathtt{case}\,(e, x_1.e_1, x_2.e_2) \mid e[] \mid \mathtt{unpack}\ e_1\ \mathtt{as}\ x\ \mathtt{in}\ e_2 \\
E &::= - \mid \langle E, e\rangle \mid \langle v, E\rangle \mid \mathtt{inl}\ E \mid \mathtt{inr}\ E \mid \mathtt{pack}\, E \mid \mathtt{proj}_i\, E \mid E\, e \mid v\, E \mid E[] \\
&\quad \mid \mathtt{case}\,(E, x_1.e_1, x_2.e_2) \mid \mathtt{unpack}\ E\ \mathtt{as}\ x\ \mathtt{in}\ e \mid \mathtt{unfold}\, E \mid \mathtt{fold}\, E
\end{aligned}$$

Fig. 1. Syntax of $\mathbf{F}^{\mu,\mathtt{?}}$: types τ, terms e and evaluation contexts E. $\mathtt{inl}\ e$ and $\mathtt{inr}\ e$ introduce terms of sum type. $\mathtt{case}\,(e, x_1.e_1, x_2.e_2)$ is the pattern matching construct that eliminates a term e of the sum type with the left branch being e_1 and right branch e_2. $\mathtt{pack}\, e$ and $\mathtt{unpack}\ e_1\ \mathtt{as}\ x\ \mathtt{in}\ e_2$ introduce and eliminate terms of existential types and $\Lambda.e$ and $e[]$ introduce and eliminate terms of universal types.

3 The language $\mathbf{F}^{\mu,\mathtt{?}}$

In this section we introduce $\mathbf{F}^{\mu,\mathtt{?}}$, a call-by-value functional language akin to System F, i.e., with impredicative polymorphism, existential and general recursive types, extended with a countable choice expression `?`. We work informally in the logic outlined above except where explicitly stated.

Syntax We assume disjoint, countably infinite sets of *type variables*, ranged over by α, and *term variables*, ranged over by x. The syntax of types, terms and evaluation contexts is defined in Figure 1. Values v and contexts (terms with a hole) C can be defined in the usual way. The free type variables in a type $\mathit{ftv}(\tau)$ and free term variables in a term $\mathit{fv}(e)$, are defined in the usual way. The notation $\sigma[\boldsymbol{\tau}/\boldsymbol{\alpha}]$ denotes the simultaneous capture-avoiding substitution of types $\boldsymbol{\tau}$ for the free type variables $\boldsymbol{\alpha}$ in the type σ; similarly, $e[\boldsymbol{v}/\boldsymbol{x}]$ denotes simultaneous capture-avoiding substitution of values $\boldsymbol{v}$ for the free term variables $\boldsymbol{x}$ in e. We define the type of natural numbers as $\mathtt{nat} = \mu\alpha.1 + \alpha$ and the corresponding numerals as $\underline{0} = \mathtt{fold}\,(\mathtt{inl}\ \langle\rangle)$ and $\underline{n+1} = \mathtt{fold}\,(\mathtt{inr}\ \underline{n})$ by induction on n.

The judgment $\Delta \vdash \tau$ expresses $\mathit{ftv}(\tau) \subseteq \Delta$. The typing judgment $\Delta \mid \Gamma \vdash e : \tau$ expresses that e has type τ in type variable context Δ and term variable context Γ. Typing rules are the same as for system F with recursive types, apart from the typing of the `?`, which has type `nat` in any well-formed context.

We write $\mathbf{Type}$ for the set of closed types τ, i.e. types τ satisfying $\mathit{ftv}(\tau) = \emptyset$. We write $\mathbf{Val}\,(\tau)$ and $\mathbf{Tm}\,(\tau)$ for the sets of closed values and terms of type τ, respectively. $\mathbf{Stk}\,(\tau)$ denotes the set of evaluation contexts E with the hole of type τ. The typing of evaluation contexts can be defined as in [4] by an inductive relation. We write $\mathbf{Val}$ and $\mathbf{Tm}$ for the set of all closed values and closed terms, respectively, and $\mathbf{Stk}$ for the set of all evaluation contexts.

Using the model in Section 5, we can show that the types of terms, values, evaluation contexts and contexts are constant sets. We use this fact in the proof of adequacy in Section 4.

Operational semantics The operational semantics of $\mathbf{F}^{\mu,\mathtt{?}}$ is given in Figure 2 by a one-step reduction relation $e \rightsquigarrow e'$. The rules are standard apart from the rule for `?` which states that the countable choice expression `?` evaluates

$$\begin{array}{rlrl}
\mathtt{proj}_i\,\langle v_1, v_2\rangle &\longmapsto v_i & \mathtt{unfold}\,(\mathtt{fold}\,v) &\longmapsto v \\
(\lambda x.e)\,v &\longmapsto e[v/x] & \mathtt{unpack}\,(\mathtt{pack}\,v)\ \mathtt{as}\ x\ \mathtt{in}\ e &\longmapsto e[v/x] \\
(\Lambda.e)[] &\longmapsto e & \mathtt{case}\,(\mathtt{inl}\,v, x_1.e_1, x_2.e_2) &\longmapsto e_1[v/x_1] \\
\mathtt{?} &\longmapsto \underline{n} \quad (n \in \mathbb{N}) & \mathtt{case}\,(\mathtt{inr}\,v, x_1.e_1, x_2.e_2) &\longmapsto e_2[v/x_2]
\end{array}$$

$$E[e] \rightsquigarrow E[e'] \qquad \text{if } e \longmapsto e'$$

Fig. 2. Operational semantics of $\mathbf{F}^{\mu,?}$: basic reductions $\longmapsto$ and one step reduction $\rightsquigarrow$.

nondeterministically to any numeral $\underline{n}$ ($n \in \mathbb{N}$). We extend basic reduction $\longmapsto$ to the single step reduction relation $\rightsquigarrow$ using evaluation contexts E.

To define the logical relation we need further restricted reduction relations. These will allow us to ignore most reductions in the definition of the logical relation, except the ones needed to prove the fundamental property (Corollary 1).

Let $\rightsquigarrow^*$ be the reflexive transitive closure of $\rightsquigarrow$. Following [4] we call *unfold-fold* reductions those of the form $\mathtt{unfold}\,(\mathtt{fold}\,v) \longmapsto v$, and *choice* reductions those of the form $\mathtt{?} \longmapsto \underline{n}$ $(n \in \mathbb{N})$. Choice reductions are important because these are the only ones that do not preserve equivalence. We define

- $e \overset{p}{\rightsquigarrow} e'$ if $e \rightsquigarrow^* e'$ and *none* of the reductions is a choice reduction;
- $e \overset{0}{\rightsquigarrow} e'$ if $e \rightsquigarrow^* e'$ and *none* of the reductions is an unfold-fold reduction;
- $e \overset{1}{\rightsquigarrow} e'$ if $e \rightsquigarrow^* e'$ and *exactly one* of the reductions is an unfold-fold reduction;
- $e \overset{p,0}{\rightsquigarrow} e'$ if $e \overset{p}{\rightsquigarrow} e'$ and $e \overset{0}{\rightsquigarrow} e'$;
- $e \overset{p,1}{\rightsquigarrow} e'$ if $e \overset{p}{\rightsquigarrow} e'$ and $e \overset{1}{\rightsquigarrow} e'$.

The $\overset{1}{\rightsquigarrow}$ reduction relation will be used in the stratified definition of divergence and the other reduction relations will be used to state additional properties of the logical relation in Lemma 1. Note that although some of the relations are described informally using negation they can be described constructively in a positive way. For instance, $\overset{p}{\rightsquigarrow}$ can be defined in the same way as the $\rightsquigarrow^*$ but using a subset of the one step relation $\rightsquigarrow$.

Divergence relations We define the logical relation using biorthogonality. As we explained in the introduction we use two *may-divergence* predicates, which are, informally, the negations of the two *must-convergence* relations from [4]. Thus we define, in the logic, the *stratified may-divergence predicate* $\Uparrow$ as the unique fixed point of $\Psi : \mathcal{P}(\mathbf{Tm}) \to \mathcal{P}(\mathbf{Tm})$ given as

$$\Psi(A) = \left\{ e : \mathbf{Tm} \;\middle|\; \exists e' : \mathbf{Tm}, e \overset{1}{\rightsquigarrow} e' \wedge \triangleright\,(e' \in A) \right\}.$$

Ψ is internally contractive and since $\mathbf{Tm}$ is a constant set $\mathcal{P}(\mathbf{Tm})$ is total. By Theorem 1, Ψ has a unique fixed point.

We also define the non-stratified may-divergence predicate $\uparrow$ as the *greatest* fixed-point of $\Phi : \mathcal{P}(\mathbf{Tm}) \to \mathcal{P}(\mathbf{Tm})$ given as

$$\Phi(A) = \left\{ e : \mathbf{Tm} \;\middle|\; \exists e' : \mathbf{Tm}, e \rightsquigarrow e' \wedge e' \in A \right\}.$$

Since Φ is monotone and $\mathcal{P}(\mathbf{Tm})$ is a complete lattice, the greatest fixed point exists by Knaster-Tarski's fixed-point theorem, which holds in our logic.[5] Observe that Ψ is almost the same as $\Phi \circ \triangleright$, apart from using a different reduction relation. We write $e\!\uparrow$ and $e\!\updownarrow$ for $e \in \uparrow$ and $e \in \updownarrow$, respectively.

The predicates $\updownarrow$ and $\uparrow$ are closed under some, but not all, reductions.

Lemma 1. *Let $e, e' : \mathbf{Tm}$. The following properties hold in the logic GTT.*

$$\text{if } e \overset{p}{\rightsquigarrow} e' \text{ then } e\!\uparrow \leftrightarrow e'\!\uparrow \qquad \text{if } e \overset{p,0}{\rightsquigarrow} e' \text{ then } e\!\updownarrow \leftrightarrow e'\!\updownarrow$$
$$\text{if } e \overset{0}{\rightsquigarrow} e' \text{ then } e'\!\updownarrow \to e\!\updownarrow \qquad \text{if } e \overset{1}{\rightsquigarrow} e' \text{ then } \triangleright(e'\!\updownarrow) \to e\!\updownarrow$$

Must-contextual approximation Contexts can be typed as second-order terms, by means of a typing judgment of the form $C : (\Delta \mid \Gamma \Rrightarrow \tau) \hookrightarrow (\Delta' \mid \Gamma' \Rrightarrow \sigma)$, stating that whenever $\Delta \mid \Gamma \vdash e : \tau$ holds, $\Delta' \mid \Gamma' \vdash C[e] : \sigma$ also holds. The typing of contexts can be defined as an inductive relation defined by suitable typing rules, which we omit here due to lack of space; see [2]. We write $C : (\Delta \mid \Gamma \Rrightarrow \tau)$ to mean there exists a type σ, such that $C : (\Delta \mid \Gamma \Rrightarrow \tau) \hookrightarrow (\varnothing \mid \varnothing \Rrightarrow \sigma)$ holds.

We define *contextual must-approximation* using the may-divergence predicate. This is in contrast with the definition in [4] which uses the must-convergence predicate. However externally, in the model, the two definitions coincide.

Definition 3 (Must-contextual approximation). *In GTT, we define must-contextual approximation $\Delta \mid \Gamma \vdash e_1 \lesssim^{ctx}_{\Downarrow} e_2 : \tau$ as*

$$\Delta \mid \Gamma \vdash e_1 : \tau \wedge \Delta \mid \Gamma \vdash e_2 : \tau \ \wedge \forall C, (C : (\Delta \mid \Gamma \Rrightarrow \tau)) \wedge C[e_2]\!\uparrow \to C[e_1]\!\uparrow.$$

Note the order in the implication: if $C[e_{\mathbf{2}}]$ may-diverges then $C[e_{\mathbf{1}}]$ may-diverges. This is the contrapositive of the definition in [4] which states that if $C[e_{\mathbf{1}}]$ must-converges then $C[e_{\mathbf{2}}]$ must-converges. Must-contextual approximation defined explicitly using contexts can be shown to be the largest compatible adequate and transitive relation, so it coincides with contextual approximation in [4].

4 Logical relation

In this section we give an abstract account of the concrete step-indexed model from [4] by defining a logical relation interpretation of types in GTT. The result is a simpler model without a proliferation of step-indices, as we will demonstrate in the example at the end of the section.

Relational interpretation of types Let $\mathbf{Type}(\Delta) = \{\tau \mid \Delta \vdash \tau\}$ be the set of types well-formed in context Δ. Given $\tau, \tau' \in \mathbf{Type}$ let $\mathbf{VRel}(\tau, \tau') = \mathcal{P}(\mathbf{Val}(\tau) \times \mathbf{Val}(\tau'))$, $\mathbf{TRel}(\tau, \tau') = \mathcal{P}(\mathbf{Tm}(\tau) \times \mathbf{Tm}(\tau'))$ and $\mathbf{SRel}(\tau, \tau') = \mathcal{P}(\mathbf{Stk}(\tau) \times \mathbf{Stk}(\tau'))$. We implicitly use the inclusion $\mathbf{VRel}(\tau, \tau') \subseteq \mathbf{TRel}(\tau, \tau')$. For a type variable context Δ, we define $\mathbf{VRel}(\Delta)$ to be

$$\left\{(\varphi_1, \varphi_2, \varphi_r) \;\middle|\; \varphi_1, \varphi_2 : \Delta \to \mathbf{Type}, \forall \alpha \in \Delta, \varphi_r(\alpha) \in \mathbf{VRel}(\varphi_1(\alpha), \varphi_2(\alpha))\right\}$$

[5] Knaster-Tarski's fixed point theorem holds in the internal language of any topos.

$$
\begin{aligned}
[\![\Delta \vdash \alpha]\!]\,(\varphi) &= \varphi_r(\alpha)\\
[\![\Delta \vdash \mathbf{1}]\!]\,(\varphi) &= \mathbf{Id_1}\\
[\![\Delta \vdash \tau_1 \times \tau_2]\!]\,(\varphi) &= \{(\langle v,u\rangle, \langle v',u'\rangle) \mid (v,v') \in [\![\Delta \vdash \tau_1]\!]\,(\varphi)\,, (u,u') \in [\![\Delta \vdash \tau_2]\!]\,(\varphi)\}\\
[\![\Delta \vdash \tau_1 + \tau_2]\!]\,(\varphi) &= \{(\mathtt{inl}\, v, \mathtt{inl}\, v') \mid (v,v') \in [\![\Delta \vdash \tau_1]\!]\,(\varphi)\} \cup\\
&\qquad \{(\mathtt{inr}\, u, \mathtt{inr}\, u') \mid (u,u') \in [\![\Delta \vdash \tau_2]\!]\,(\varphi)\}\\
[\![\Delta \vdash \tau_1 \to \tau_2]\!]\,(\varphi) &= \{(\lambda x.e, \lambda y.e') \mid \forall (v,v') \in [\![\Delta \vdash \tau_1]\!]\,(\varphi),\\
&\qquad \left(e[v/x], e'[v'/y]\right) \in [\![\Delta \vdash \tau_2]\!]\,(\varphi)^{\top\top}\}\\
[\![\Delta \vdash \forall\alpha.\tau]\!]\,(\varphi) &= \{(\Lambda.e, \Lambda.e') \mid \forall \sigma,\sigma' \in \mathbf{Type}, \forall s \in \mathbf{VRel}\left(\sigma,\sigma'\right),\\
&\qquad \left(e,e'\right) \in [\![\Delta, \alpha \vdash \tau]\!]\,(\varphi\left[\alpha \mapsto (\sigma,\sigma',s)\right])^{\top\top}\}\\
[\![\Delta \vdash \exists\alpha.\tau]\!]\,(\varphi) &= \{(\mathtt{pack}\, v, \mathtt{pack}\, v') \mid \exists \sigma,\sigma' \in \mathbf{Type}, \exists s \in \mathbf{VRel}\left(\sigma,\sigma'\right),\\
&\qquad \left(v,v'\right) \in [\![\Delta, \alpha \vdash \tau]\!]\,(\varphi\left[\alpha \mapsto (\sigma,\sigma',s)\right])\}\\
[\![\Delta \vdash \mu\alpha.\tau]\!]\,(\varphi) &= \mathtt{fix}\left(\lambda s.\{(\mathtt{fold}\, v, \mathtt{fold}\, v') \mid \triangleright\left((v,v') \in [\![\Delta, \alpha \vdash \tau]\!]\,(\varphi\left[\alpha \mapsto s\right])\right)\}\right)
\end{aligned}
$$

where the $\cdot^{\top\top} : \mathbf{VRel}\,(\tau,\tau') \to \mathbf{TRel}\,(\tau,\tau')$ is defined with the help of $\cdot^{\top} : \mathbf{VRel}\,(\tau,\tau') \to \mathbf{SRel}\,(\tau,\tau')$ as follows

$$
\begin{aligned}
r^{\top} &= \{(E,E') \mid \forall (v,v') \in r, E'[v']\!\uparrow \to E[v]\!\uparrow\}\\
r^{\top\top} &= \{(e,e') \mid \forall (E,E') \in r^{\top}, E'[e']\!\uparrow \to E[e]\!\uparrow\}.
\end{aligned}
$$

Fig. 3. Interpretation of types. All the relations are on *typeable* terms and contexts.

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 shown in Figure 3. The definition is by induction on the judgement $\Delta \vdash \tau$. Given 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 φ_1 and φ_2 are the first two components of φ, and $\varphi_i(\tau)$ denotes substitution of types in φ_i for free type variables in τ. Since we are working in the logic GTT, the interpretations of all type constructions are simple and intuitive. For instance, functions are related when they map related values to related results, two values of universal type are related if they respect all value relations. In particular, there are no admissibility requirements on the relations, nor any step-indexing — but just a use of $\triangleright$ in the interpretation of recursive types, to make it well-defined as a consequence of Theorem 1, using that the type $\mathcal{P}\,(\mathbf{Tm} \times \mathbf{Tm})$ is total.

The definition of $\top\top$-closure is where we connect operational semantics and the $\triangleright$ modality, using the stratified may-divergence predicate $\uparrow$. $\top\top$-closed relations are closed under some reductions. More precisely, the following holds.

Lemma 2. *Let* $\tau,\tau' : \mathbf{Type}$ *and* $r \in \mathbf{VRel}\,(\tau,\tau')$.

- *If* $e \overset{p,0}{\rightsquigarrow} e_1$ *and* $e' \overset{p}{\rightsquigarrow} e'_1$ *then* $(e,e') \in r^{\top\top} \leftrightarrow (e_1,e'_1) \in r^{\top\top}$.
- *If* $e \overset{1}{\rightsquigarrow} e_1$ *then for all* $e' : \mathbf{Tm}$, *if* $\triangleright((e_1,e') \in r^{\top\top})$ *then* $(e,e') \in r^{\top\top}$.

We use this fact extensively in the proofs of the fundamental property and example equivalences.

In order to define logical relations, we need first to extend the interpretation of types to the interpretation of contexts (note that in particular, related substitutions map into well-typed values):

$$\llbracket \Delta \vdash \Gamma \rrbracket(\varphi) = \{(\gamma,\gamma') \mid \gamma,\gamma' : \mathbf{Val}^{\mathrm{dom}(\Gamma)}, \\ \forall x \in \mathrm{dom}(\Gamma), (\gamma(x),\gamma'(x)) \in \llbracket \Delta \vdash \Gamma(x) \rrbracket(\varphi)\}$$

The logical relation and its fundamental property We define the logical relation on open terms by reducing it to relations on closed terms by substitution.

Definition 4 (Logical relation). $\Delta \mid \Gamma \vdash e_1 \lesssim^{log}_{\Downarrow} e_2 : \tau$ *if*

$$\forall \varphi \in \mathbf{VRel}(\Delta), \forall (\gamma,\gamma') \in \llbracket \Delta \vdash \Gamma \rrbracket(\varphi), (e_1\gamma, e_2\gamma') \in \llbracket \Delta \vdash \tau \rrbracket(\varphi)^{\top\top}.$$

To prove the fundamental property of logical relations and connect the logical relation to contextual-must approximation we start with some simple properties relating evaluation contexts and relations. All the lemmata are essentially of the same form: given two related evaluation contexts at a suitable type, the contexts extended with an elimination form are also related at a suitable type. We only state the case for `unfold`, since it shows the interplay between unfold-fold reductions and the stratified may divergence predicate.

Lemma 3. *If* $(E,E') \in \llbracket \Delta \vdash \tau[\mu\alpha.\tau/\alpha] \rrbracket(\varphi)^{\top}$ *then*

$$(E \circ (\mathtt{unfold}\,[]), E' \circ (\mathtt{unfold}\,[])) \in \llbracket \Delta \vdash \mu\alpha.\tau \rrbracket(\varphi)^{\top}.$$

Proof. Given $(\mathtt{fold}\,v, \mathtt{fold}\,v') \in \llbracket \Delta \vdash \mu\alpha.\tau \rrbracket(\varphi)$ suppose $E'[\mathtt{unfold}\,(\mathtt{fold}\,v')]\!\uparrow$. By Lemma 1 we have $E'[v']\!\uparrow$ and so $\triangleright(E'[v']\!\uparrow)$. By definition of interpretation of recursive types we have $\triangleright((v,v') \in \llbracket \Delta \vdash \tau[\mu\alpha.\tau/\alpha] \rrbracket(\varphi))$. Thus $\triangleright(E[v]\!\updownarrow)$ and so by Lemma 1 we have $E[\mathtt{unfold}\,(\mathtt{fold}\,v)]\!\updownarrow$. □

Note that the proof would not work, were we to use the $\uparrow$ relation in place of $\updownarrow$ in the definition of the $\top\top$ closure since the last implication would not hold.

Proposition 2. *The logical approximation relation is compatible with the typing rules (see also [6, Prop.* 2.4.17*]).*

Proof. We only give two cases, to show how to use the context extension lemmata.
Elimination of recursive types: we need to show

$$\frac{\Delta \mid \Gamma \vdash e \lesssim^{log}_{\Downarrow} e' : \mu\alpha.\tau}{\Delta \mid \Gamma \vdash \mathtt{unfold}\,e \lesssim^{log}_{\Downarrow} \mathtt{unfold}\,e' : \tau[\mu\alpha.\tau/\alpha]}.$$

So take $\varphi \in \mathbf{VRel}(\Delta)$ and $(\gamma,\gamma') \in \llbracket \Delta \vdash \Gamma \rrbracket(\varphi)$. Let $f = e\gamma$ and $f' = e'\gamma'$. We have to show $(\mathtt{unfold}\,f, \mathtt{unfold}\,f') \in \llbracket \Delta \vdash \tau[\mu\alpha.\tau/\alpha] \rrbracket(\varphi)^{\top\top}$. So take $(E,E') \in \llbracket \Delta \vdash \tau[\mu\alpha.\tau/\alpha] \rrbracket(\varphi)^{\top}$. By assumption $(f,f') \in \llbracket \Delta \vdash \mu\alpha.\tau \rrbracket(\varphi)^{\top\top}$ so it suffices to show $(E \circ (\mathtt{unfold}\,[]), E' \circ (\mathtt{unfold}\,[])) \in \llbracket \Delta \vdash \mu\alpha.\tau \rrbracket(\varphi)^{\top}$ and this is exactly the content of Lemma 3.

The `?` *expression:* we need to show $\Delta \mid \Gamma \vdash \texttt{?} \lesssim^{log}_{\Downarrow} \texttt{?} : \texttt{nat}$. It is easy to see by induction that for all $n \in \mathbb{N}$, $(\underline{n}, \underline{n}) \in [\![\vdash \texttt{nat}]\!]$. So take $(E, E') \in [\![\vdash \texttt{nat}]\!]^{\top}$ and assume $E'[\texttt{?}]\uparrow$. By definition of the $\uparrow$ relation there exists an e', such that $\texttt{?} \rightsquigarrow e'$ and $E'[e']\uparrow$. Inspecting the operational semantics we see that $e' = \underline{n}$ for some $n \in \mathbb{N}$. This implies $E[\underline{n}]\updownarrow$ and so by Lemma 1 we have $E[\texttt{?}]\updownarrow$. □

Corollary 1 (Fundamental property of logical relations). *If* $\Delta \mid \Gamma \vdash e : \tau$ *then* $\Delta \mid \Gamma \vdash e \lesssim^{log}_{\Downarrow} e : \tau$

Proof. By induction on the typing derivation $\Delta \mid \Gamma \vdash e : \tau$, using Prop. 2. □

We need the next corollary to relate the logical approximation relation to must-contextual approximation.

Corollary 2. *For any expressions* e, e' *and context* C*, if* $\Delta \mid \Gamma \vdash e \lesssim^{log}_{\Downarrow} e' : \tau$ *and* $C : (\Delta \mid \Gamma \Rrightarrow \tau) \leadsto (\Delta' \mid \Gamma' \Rrightarrow \sigma)$ *then* $\Delta' \mid \Gamma' \vdash C[e] \lesssim^{log}_{\Downarrow} C[e'] : \tau'$.

Proof. By induction on the judgment $C : (\Delta \mid \Gamma \Rrightarrow \tau) \leadsto (\Delta' \mid \Gamma' \Rrightarrow \sigma)$, using Proposition 2. □

Adequacy We now wish to show soundness of the logical relation with respect to must-contextual approximation. However, the implication

$$\Delta \mid \Gamma \vdash e \lesssim^{log}_{\Downarrow} e' : \tau \to \Delta \mid \Gamma \vdash e \lesssim^{ctx}_{\Downarrow} e' : \tau$$

does not hold, due to the different divergence relations used in the definition of the logical relation. To see precisely where the proof fails, let us attempt it. Let $\Delta \mid \Gamma \vdash e \lesssim^{log}_{\Downarrow} e' : \tau$ and take a well-typed closing context C with result type σ. Then by Corollary 2, $\varnothing \mid \varnothing \vdash C[e] \lesssim^{log}_{\Downarrow} C[e'] : \sigma$. Unfolding the definition of the logical relation we get $(C[e], C[e']) \in [\![\varnothing \vdash \sigma]\!]^{\top\top}$. It is easy to see that $(-,-) \in [\![\varnothing \vdash \sigma]\!]^{\top}$ and so we get by definition of $\top\top$ that $C[e']\uparrow \to C[e]\updownarrow$. However the definition of contextual equivalence requires the implication $C[e']\uparrow \to C[e]\uparrow$, which is not a consequence of the previous one.

Intuitively, the gist of the problem is that $\uparrow$ defines a time-independent predicate, whereas $\updownarrow$ is time-dependent, since it is defined by guarded recursion. However, in the model in Section 5, we can show the validity of a formula expressing a connection between $\uparrow$ and $\updownarrow$:

Lemma 4. $e : \mathbf{Tm} \mid \varnothing \vdash \Box(e\updownarrow) \to e\uparrow$ *holds in the logic GTT.*

Thus we additionally assume this principle in our logic. Note that this lemma is *not* valid in the logic of the topos of trees [5] and this is the reason we must work in the logic of $\mathbf{Sh}(\omega_1)$. We sketch a proof of the lemma at the end of Section 5 which shows the role of $\Box$ and why the lemma does not hold in the topos of trees. Using Lemma 4 we are led to the following corrected statement of adequacy using the $\Box$ modality.[6]

[6] Readers who are familiar with concrete step-indexed models will note that the $\Box$ modality captures the universal quantification over all steps used in the the definition of concrete step-indexed logical relations.

Theorem 2 (Adequacy). *If e and e' are of type τ in context $\Delta \mid \Gamma$ then $\Box(\Delta \mid \Gamma \vdash e \lesssim_{\Downarrow}^{log} e' : \tau)$ implies $\Delta \mid \Gamma \vdash e \lesssim_{\Downarrow}^{ctx} e' : \tau$.*

To prove this theorem we first observe that all the lemmata used in the proof of Corollary 2 are proved in constant contexts, using only other constant facts. Hence, Corollary 2 can be strengthened, yielding the following restatement.

Proposition 3. $\Box[\forall \Delta, \Delta', \Gamma, \Gamma', \tau, \sigma, C, e, e', C : (\Delta \mid \Gamma \Rrightarrow \tau) \nrightarrow (\Delta' \mid \Gamma' \Rrightarrow \sigma) \to \Delta \mid \Gamma \vdash e \lesssim_{\Downarrow}^{log} e' : \tau \to \Delta' \mid \Gamma' \vdash C[e] \lesssim_{\Downarrow}^{log} C[e'] : \tau']$.

Note that all the explicit universal quantification in the proposition is over constant types. One additional ingredient we need to complete the proof is the fact that $\uparrow$ is $\neg\neg$-closed, i.e. $e\uparrow \leftrightarrow \neg\neg(e\uparrow)$. We can show this in the logic using the fact that $\uparrow$ is the greatest post-fixed point by showing that $\neg\neg\uparrow$ is another one. This fact further means that $\Box(e\uparrow) \leftrightarrow (e\uparrow)$ (using the adjoint rule relating $\neg\neg$ and $\Box$ in Section 2). We are now ready to proceed with the proof of Theorem 2.

Proof (Theorem 2). Continuing the proof we started above we get, using Proposition 1, that $\Box(C[e']\uparrow \to C[e]\updownarrow)$ and thus also $\Box(C[e']\uparrow) \to \Box(C[e]\updownarrow)$. Moreover, $\Box(C[e']\uparrow) \leftrightarrow C[e']\uparrow$ and, by Lemma 4, $\Box(C[e]\updownarrow) \to C[e]\uparrow$. We thus conclude $C[e']\uparrow \to C[e]\uparrow$, as required. □

Thus, if we can prove that e and e' are logically related relying only on constant facts we can use this theorem to conclude that e must-contextually approximates e'. In particular, the fundamental property (Corollary 1) can be strengthened to a "boxed" statement.

Completeness As in [4] we also get completeness with respect to contextual approximation. The proof proceeds as in [4] via the notion of CIU-approximation [15, 4]. This property relies on the fact that we have built the logical relation using biorthogonality and using typeable realizers.

Theorem 3. *For any Δ, Γ, e, e' and τ,*

$$\Delta \mid \Gamma \vdash e \lesssim_{\Downarrow}^{CIU} e' : \tau \leftrightarrow \Delta \mid \Gamma \vdash e \lesssim_{\Downarrow}^{ctx} e' : \tau \leftrightarrow \Box(\Delta \mid \Gamma \vdash e \lesssim_{\Downarrow}^{log} e' : \tau)$$

Applications We can now use the logical relation to prove contextual equivalences. Indeed, the accompanying technical report [6] provides internal proofs of all the examples done in the concrete step-indexed model in [4]; these proofs are simpler than the ones in [4]. As an example, in this paper we include the proof of syntactic minimal invariance for *must*-equivalence. Remarkably, the proof below is just as simple as the proof of the minimal invariance property in the abstract account of a step-indexed model for the *deterministic* language $\mathbf{F}^{\mu}$ [8].

Let $\mathtt{fix} : \forall\alpha,\beta.((\alpha \to \beta) \to (\alpha \to \beta)) \to (\alpha \to \beta)$ be the term $\Lambda.\Lambda.\lambda f.\delta_f(\mathtt{fold}\,\delta_f)$ where δ_f is the term $\lambda y.\mathtt{let}\ y' = \mathtt{unfold}\, y\ \mathtt{in}\ f\,(\lambda x.y'\,y\,x)$.

Consider the type $\tau = \mu\alpha.\mathtt{nat} + \alpha \to \alpha$. Let $id = \lambda x.x$ and consider the term

$$f \equiv \lambda h, x.\mathtt{case}\,(\mathtt{unfold}\,x, y.\mathtt{fold}\,(\mathtt{inl}\,y), g.\mathtt{fold}\,(\mathtt{inr}\,\lambda y.h(g(h\,y))))\,.$$

We show that $\mathtt{fix}[][]\, f \lesssim^{log}_{\Downarrow} id : \tau \to \tau$. The other direction is essentially the same. Since we prove this in the context of constant facts we can use Theorem 3 to conclude that the terms are contextually equivalent.

We show by Löb induction that $(\mathtt{fix}[][]\, f, id) \in [\![\tau \to \tau]\!]^{\top\top}$. It is easy to see that $\mathtt{fix}[][]\, f \overset{p,1}{\rightsquigarrow} \lambda x.\mathtt{case}\,(\mathtt{unfold}\, x, y.\mathtt{fold}\,(\mathtt{inl}\, y), g.\mathtt{fold}\,(\mathtt{inr}\, \lambda y.h(g(h\, y))))$ where $h = \lambda x.\delta_f(\mathtt{fold}\, \delta_f)\, x$. Let

$$\varphi = \lambda x.\mathtt{case}\,(\mathtt{unfold}\, x, y.\mathtt{fold}\,(\mathtt{inl}\, y), g.\mathtt{fold}\,(\mathtt{inr}\, \lambda y.h(g(h\, y)))) .$$

We now show directly that $(\varphi, id) \in [\![\tau \to \tau]\!]$ which suffices by Lemma 2.

Let us take $(u, u') \in [\![\tau]\!]$. By the definition of the interpretation of recursive and sum types there are two cases:

- $u = \mathtt{fold}\,(\mathtt{inl}\, \underline{n})$ and $u' = \mathtt{fold}\,(\mathtt{inl}\, \underline{n})$ for some $n \in \mathbb{N}$: immediate.
- $u = \mathtt{fold}\,(\mathtt{inr}\, g)$, $u' = \mathtt{fold}\,(\mathtt{inr}\, g')$ for some g, g' such that $\triangleright((g, g') \in [\![\tau \to \tau]\!])$. We then have that $\varphi\, u \overset{p,1}{\rightsquigarrow} \mathtt{fold}\,(\mathtt{inr}\, \lambda y.h(g(h\, y)))$ and $id\, u' \overset{p}{\rightsquigarrow} u'$ and so it suffices to show $\triangleright\,(\lambda y.\,(h(g(h\, y)), g') \in [\![\tau \to \tau]\!])$. We again show that these are related as values so take $\triangleright((v, v') \in [\![\tau]\!])$ and we need to show $\triangleright\left((h(g(h\, v)), g'\, v') \in [\![\tau]\!]^{\top\top}\right)$. Take $\triangleright((E, E') \in [\![\tau]\!]^{\top})$. Löb induction hypothesis gives us that $\triangleright((h', id) \in [\![\tau \to \tau]\!]^{\top\top})$, where h' is the body of h, i.e $h = \lambda x.h'\, x$. It is easy to see that this implies $\triangleright((h, id) \in [\![\tau \to \tau]\!]^{\top\top})$ and so by extending the contexts three times using lemmata analogous to Lemma 3 we get $\triangleright\left((E[h\,(g\,(h\,[]))], E'[g'\,[]]) \in [\![\tau]\!]^{\top}\right)$.
  So, assuming $\triangleright(E'[g'\, v']\!\uparrow)$ we get $\triangleright(E[h\,(g\,(h\, v))]\!\uparrow)$, concluding the proof.

5 The model for GTT

In this section, we present a model for the logic GTT, where all the properties we have used in the previous sections are justified. The model we consider is the topos of sheaves over the first uncountable ordinal ω_1 (in fact, any ordinal $\alpha \geq \omega_1$ would suffice). We assume some basic familiarity with topos theory, on the level described in [14]. We briefly recall the necessary definitions.

The objects of $\mathbf{Sh}\,(\omega_1)$ are sheaves over ω_1 considered as a topological space equipped with the Alexandrov topology. Concretely, this means that objects of $\mathbf{Sh}\,(\omega_1)$ are continuous functors from $(\omega_1 + 1)^{\mathrm{op}}$ to $\mathbf{Set}$. We think of ordinals as time, with smaller ordinals being the future. The restriction maps then describe the evolution of elements through time.

$\mathbf{Sh}\,(\omega_1)$ is a full subcategory of the category of presheaves $\mathbf{PSh}\,(\omega_1 + 1)$. The inclusion functor i has a left adjoint $\mathbf{a} : \mathbf{PSh}\,(\omega_1 + 1) \to \mathbf{Sh}\,(\omega_1)$ called the *associated sheaf functor*. Limits and exponentials are constructed as in presheaf categories. Colimits *are not* constructed pointwise as in presheaf categories, but they require also the application of the associated sheaf functor.

There is an essential geometric morphism $\Pi_1 \dashv \Delta \dashv \Gamma : \mathbf{Sh}\,(\omega_1) \to \mathbf{Set}$, with Δ the *constant sheaf* functor, Γ the global sections functor and $\Pi_1(X) =$

$X(1)$ the evaluation at 1 (we consider 0 to be the first ordinal). Given a set a, the constant sheaf $\Delta(a)$ is not the constant presheaf: rather it is equal to the singleton set 1 at stage 0, and to a at all other stages. For a sheaf X, an element $\xi \in X(\nu)$ and $\beta \leq \nu$ we write $\xi|_\beta$ for the restriction $X(\beta \leq \nu)(\xi)$.

Analogously to the topos of trees [5], there is a "later" modality on *types*, i.e. a functor $\blacktriangleright : \mathbf{Sh}(\omega_1) \to \mathbf{Sh}(\omega_1)$ defined as (we consider 0 a limit ordinal)

$$\blacktriangleright X(\nu+1) = X(\nu), \qquad \blacktriangleright X(\alpha) = X(\alpha) \text{ for } \alpha \text{ limit ordinal.}$$

There is an obvious natural transformation $\mathrm{next}^X : X \to \blacktriangleright X$.

The subobject classifier Ω is given by $\Omega(\nu) = \{\beta \mid \beta \leq \nu\}$ and its restriction maps are given by minimum. There is a natural transformation $\triangleright : \Omega \to \Omega$ given as $\triangleright_\nu(\beta) = \min\{\beta+1, \nu\}$.

Kripke-Joyal semantics [9] is a way to translate formulas in the logic to statements about objects and morphisms of $\mathbf{Sh}(\omega_1)$; we refer to [14, Section VI.5] for a detailed introduction and further references. We now briefly explain the Kripke-Joyal semantics of GTT.

Let X be a sheaf and φ, ψ formulas in the internal language with a free variable of type X. Intuitively, for an ordinal ν and an element $\xi \in X(\nu)$, $\nu \Vdash \varphi(\xi)$ means that φ holds for ξ at stage ν. A formula φ is *valid* if it holds for all ξ and at all stages.

Let $\nu \leq \omega_1$ and $\xi \in X(\nu)$. The rules of Kripke-Joyal semantics are the usual ones (see, e.g., [14, Theorem VI.7.1]), specialized for our particular topology:

- $\nu \Vdash \bot$ iff $\nu = 0$;
- $\nu \Vdash \top$ always;
- $\nu \Vdash \varphi(t)(\xi)$ iff $[\![\varphi]\!]_\nu([\![t]\!]_\nu(\xi)) = \nu$, for a predicate symbol φ on X;
- $\nu \Vdash \varphi(\xi) \wedge \psi(\xi)$ iff $\nu \Vdash \varphi(\xi)$ and $\nu \Vdash \psi(\xi)$;
- $\nu \Vdash \varphi(\xi) \vee \psi(\xi)$ iff $\nu \Vdash \varphi(\xi)$ or $\nu \Vdash \psi(\xi)$;
- $\nu \Vdash \varphi(\xi) \to \psi(\xi)$ iff for all $\beta \leq \nu, \beta \Vdash \varphi(\xi|_\beta)$ implies $\beta \Vdash \psi(\xi|_\beta)$;
- $\nu \Vdash \neg\varphi(\xi)$ iff for all $\beta \leq \nu, \beta \Vdash \varphi(\xi|_\beta)$ implies $\beta = 0$.

Note that $0 \Vdash \varphi$ for any φ, as is usual in Kripke-Joyal semantics for sheaves over a space: intuitively, the stage 0 represents the impossible world. Moreover, if φ is a formula with free variables $x : X$ and $y : Y$, $\nu \leq \omega_1$ and $\xi \in X(\nu)$ then:

- For ν a successor ordinal: $\nu \Vdash \exists y : Y, \varphi(\xi, y)$ iff there exists $\xi' \in Y(\nu)$ such that $\nu \Vdash \varphi(\xi, \xi')$;
- For ν a limit ordinal: $\nu \Vdash \exists y : Y, \varphi(\xi, y)$ iff for all $\beta < \nu$ there exists $\xi_\beta \in Y(\beta)$ such that $\beta \Vdash \varphi(\xi|_\beta, \xi_\beta)$;
- $\nu \Vdash \forall y : Y, \varphi(\xi, y)$ iff for all $\beta \leq \nu$ and for all $\xi_\beta \in Y(\beta)$: $\beta \Vdash \varphi(\xi|_\beta, \xi_\beta)$.

The semantics of $\triangleright$ is as follows. Let φ be a predicate on X, then

$$\nu \Vdash \triangleright\varphi(\alpha) \text{ iff for all } \beta < \nu, \beta \Vdash \varphi(\alpha|_\beta).$$

For successor ordinals $\nu = \nu' + 1$ this reduces to

$$\nu + 1 \Vdash \triangleright\varphi(\alpha) \text{ iff } \nu' \Vdash \varphi(\alpha|_{\nu'}).$$

The predicate $\mathrm{Total}\,(X)$ in Definition 1 internalizes the property that all X's restriction maps are surjections which intuitively means that elements at any stage β evolve from elements in the past. Total sheaves are also called *flabby* in homological algebra literature, but we choose to use the term total since it was used in previous work on guarded recursion to describe an analogous property.

The properties of $\triangleright$ stated in Section 2 can be proved easily using the Kripke-Joyal semantics. The rules are similar to the rules in [5, Theorem 2.7], except the case of the existential quantifier in which the converse implication *does not hold*, even if we restrict to total and inhabited types, or even to constant sets. As a consequence, we cannot prove the internal Banach's fixed point theorem in the logic in the same way as in the topos of trees, cf. [5, Lemma 2.10].

In contrast to that in the topos of trees [5, Theorem 2.9], which requires the type X only to be inhabited, the internal Banach's fixed point theorem in $\mathbf{Sh}\,(\omega_1)$ (Theorem 1) has stronger assumptions: we require X to be total, which implies that it is inhabited. The additional assumption seems to be necessary and is satisfied in all the instances where we use the theorem. In particular, for a constant X, $\mathcal{P}\,(X)$ is total.

The operator $\neg\neg : \Omega \to \Omega$ gives rise to a function $\neg\neg_X$ on the lattice of subobjects $\mathbf{Sub}\,(X)$. In $\mathbf{Sh}\,(\omega_1)$, $\neg\neg_X$ preserves suprema[7] on each $\mathbf{Sub}\,(X)$ and therefore has a right adjoint $\Box_X : \mathbf{Sub}\,(X) \to \mathbf{Sub}\,(X)$ defined as

$$\Box_X P = \bigvee \{Q \mid \neg\neg Q \leq P\}\,.$$

If $X = \Delta(a)$ then $\Box_X P$ has a simpler description:

$$\Box_{\Delta(a)}(P)(\nu) = \begin{cases} 1 & \text{if } \nu = 0 \\ \bigcap_{\beta=1}^{\omega_1} P(\beta) & \text{otherwise.} \end{cases}$$

Thus for a predicate P on a constant set $\Delta(a)$, $\Box(P)$ contains only those elements for which P holds at all stages.

However, in contrast to $\neg\neg$ which commutes with reindexing, $\Box$ does not. There is a general reason for this: in any category with pullbacks, any deflationary operation $\Box$ that preserves the top element and *is natural*, i.e. commutes with reindexing, is necessarily the identity [16, Proposition 4.2]. However $\Delta(f)^*\left(\Box_{\Delta(a)}\,(P)\right) = \Box_{\Delta(b)}\,(\Delta(f)^*\,(P))$ for any $f : a \to b$ in $\mathbf{Set}$ and since Δ preserves products we do get that $\Box$ in the logic commutes with substitution when restricted to constant contexts.

The external interpretation of $\uparrow$ is exactly the negation of the must-convergence predicate $\Downarrow$ from [4]. In particular, $\uparrow$ is a constant predicate. In contrast, $\Uparrow(\nu)$ is a set of expressions e such that there exists a reduction of length at least ν starting with e. This can easily be seen using the description of Kripke-Joyal semantics above. Thus, $\Uparrow$ is externally the pointwise complement of the stratified must-convergence predicate $\{\Downarrow_\beta\}_{\beta<\omega_1}$ from [4]. Then, the proof that $\Box\Uparrow \to \uparrow$ corresponds to the proof that $\Downarrow \subseteq \bigcup_{\beta<\omega_1} \Downarrow_\beta$ in [4]. Here we technically see the need for indexing over ω_1.

[7] Recall that this is *not* the case in every topos.

Acknowledgments

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. Agha, G., Mason, I.A., Smith, S.F., Talcott, C.L.: A foundation for actor computation. Journal of Functional Programming 7(1), 1–72 (1997)
2. Ahmed, A.: Step-indexed syntactic logical relations for recursive and quantified types. Tech. rep., Harvard University (2006), http://www.ccs.neu.edu/home/amal/papers/lr-recquant-techrpt.pdf
3. Apt, K.R., Plotkin, G.D.: Countable nondeterminism and random assignment. Journal of the ACM 33(4), 724–767 (1986)
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., Møgelberg, R.E., Schwinghammer, J., Støvring, K.: First steps in synthetic guarded domain theory: step-indexing in the topos of trees. Logical Methods in Computer Science 8(4) (2012)
6. Bizjak, A., Birkedal, L., Miculan, M.: A model of countable nondetermism in guarded type theory. Available at http://cs.au.dk/~abizjak/documents/trs/cntbl-gtt-tr.pdf (2014)
7. Di Gianantonio, P., Honsell, F., Plotkin, G.D.: Uncountable limits and the lambda calculus. Nordic Journal of Computing 2(2), 126–145 (1995)
8. Dreyer, D., Ahmed, A., Birkedal, L.: Logical step-indexed logical relations. Logical Methods in Computer Science 7(2) (2011)
9. Lambek, J., Scott, P.: Introduction to Higher-Order Categorical Logic. Cambridge Studies in Advanced Mathematics, Cambridge University Press (1988)
10. Lassen, S.B.: Relational Reasoning about Functions and Nondeterminism. Ph.D. thesis, University of Aarhus (1998)
11. Lassen, S.B., Moran, A.: Unique fixed point induction for McCarthy's amb. In: Mathematical Foundations of Computer Science. pp. 198–208 (1999)
12. Lassen, S.B., Pitcher, C.: Similarity and bisimilarity for countable non-determinism and higher-order functions. Electronic Notes in Theoretical Computer Science 10 (1997)
13. Levy, P.B.: Infinitary Howe's method. In: Coalgebraic Methods in Computer Science. pp. 85–104 (2006)
14. MacLane, S., Moerdijk, I.: Sheaves in Geometry and Logic: A First Introduction to Topos Theory. Mathematical Sciences Research Institute Publications, Springer New York (1992)
15. Mason, I.A., Talcott, C.L.: Equivalence in functional languages with effects. Journal of Functional Programming 1(3), 287–327 (1991)
16. Reyes, G., Zolfaghari, H.: Bi-heyting algebras, toposes and modalities. Journal of Philosophical Logic 25(1), 25–43 (1996)
17. Sabel, D., Schmidt-Schauß, M.: A call-by-need lambda calculus with locally bottom-avoiding choice: context lemma and correctness of transformations. Mathematical Structures in Computer Science 18(3), 501–553 (2008)