FIRST STEPS IN SYNTHETIC GUARDED DOMAIN THEORY: STEP-INDEXING IN THE TOPOS OF TREES

LARS BIRKEDAL, RASMUS EJLERS MØGELBERG, JAN SCHWINGHAMMER, AND KRISTIAN STØVRING

IT University of Copenhagen
e-mail address: birkedal@itu.dk

IT University of Copenhagen
e-mail address: mogel@itu.dk

Saarland University
e-mail address: jan@ps.uni-saarland.de

DIKU, University of Copenhagen
e-mail address: stovring@diku.dk

Abstract. We present the topos $\mathcal{S}$ of trees as a model of guarded recursion. We study the internal dependently-typed higher-order logic of $\mathcal{S}$ and show that $\mathcal{S}$ models two modal operators, on predicates and types, which serve as guards in recursive definitions of terms, predicates, and types. In particular, we show how to solve recursive type equations involving *dependent* types. We propose that the internal logic of $\mathcal{S}$ provides the right setting for the synthetic construction of abstract versions of step-indexed models of programming languages and program logics. As an example, we show how to construct a model of a programming language with higher-order store and recursive types entirely inside the internal logic of $\mathcal{S}$. Moreover, we give an axiomatic categorical treatment of models of synthetic guarded domain theory and prove that, for any complete Heyting algebra A with a well-founded basis, the topos of sheaves over A forms a model of synthetic guarded domain theory, generalizing the results for $\mathcal{S}$.

1. Introduction

Recursive definitions are ubiquitous in computer science. In particular, in semantics of programming languages and program logics we often use recursively defined functions and relations, and also recursively defined types (domains). For example, in recent years there has been extensive work on giving semantics of type systems for programming languages with dynamically allocated higher-order store, such as general ML-like references. Models have been expressed as Kripke models over a recursively defined set of worlds (an example

1998 ACM Subject Classification: D.3.1, F.3.2.

Key words and phrases: Denotational semantics, guarded recursion, step-indexing, recursive types.

This is an expanded and revised version of a paper that appeared at LICS'11 [4]. In addition to containing more examples and proofs, it also contains a more general treatment of models of synthetic guarded domain theory.

of a recursively defined domain) and have involved recursively defined relations to interpret the recursive types of the programming language; see [5] and the references therein.

In this paper we study a topos $\mathcal{S}$, which we show models guarded recursion in the sense that it allows for guarded recursive definitions of both recursive functions and relations as well as recursive types. The topos $\mathcal{S}$ is known as the topos of trees (or forests); what is new here is our application of this topos to model guarded recursion.

The internal logic of $\mathcal{S}$ is a standard many-sorted higher-order logic extended with modal operators on both types and terms. (Recall that terms in higher-order logic include both functions and relations, as the latter are simply Prop-valued functions.) This internal logic can then be used as a language to describe semantic models of programming languages with the features mentioned above. As an example which uses both recursively defined types and recursively defined relations in the $\mathcal{S}$-logic, we present a model of $\mathsf{F}_{\mu,\mathsf{ref}}$, a call-by-value programming language with impredicative polymorphism, recursive types, and general ML-like references.

To situate our work in relation to earlier work, we now give a quick overview of the technical development of the present paper followed by a comparison to related work. We end the introduction with a summary of our contributions.

1.1. **Overview of technical development.** The topos $\mathcal{S}$ is the category of presheaves on ω, the first infinite ordinal. This topos is known as the topos of trees, and is one of the most basic examples of presheaf categories.

There are several ways to think intuitively about this topos. Let us recall one intuitive description, which can serve to understand why it models guarded recursion. An object X of $\mathcal{S}$ is a contravariant functor from ω (viewed as a preorder) to **Set**. We think of X as a variable set, i.e., a family of sets $X(n)$, indexed over natural numbers n, and with restriction maps $X(n+1) \to X(n)$. Morphisms $f : X \to Y$ are natural transformations from X to Y. The variable sets include the ordinary sets as so-called constant sets: for an ordinary set S, there is an object $\Delta(S)$ in $\mathcal{S}$ with $\Delta(S)(n) = S$ for all n. Since $\mathcal{S}$ is a category of presheaves, it is a topos, in particular it is cartesian closed category and has a subobject classifier Ω (a type of propositions). The internal logic of $\mathcal{S}$ is an extension of standard Kripke semantics: for constant sets, the truth value of a predicate is just the set of worlds (downwards closed subsets of ω) for which the predicate holds. This observation suggests that there is a modal "later" operator $\triangleright$ on predicates $\Omega^{\Delta(S)}$ on constant sets, similar to what has been studied earlier [3, 11]. Intuitively, for a predicate $\varphi : \Omega^{\Delta(S)}$ on constant set $\Delta(S)$, $\triangleright(\varphi)$ contains $n+1$ if φ contains n. (A future world is a smaller number, hence the name "later" for this operator.) A recursively specified predicate $\mu r.\varphi(r)$ is well-defined if every occurrence of the recursion variable r in φ is guarded by a $\triangleright$ modality: by definition of $\triangleright$, to know whether $n+1$ is in the predicate it suffices to know whether n is in the predicate. There is also an associated Löb rule for induction, $(\triangleright\varphi \to \varphi) \to \varphi$, as in [3].

Here we show that in fact there is a later operator not only on predicates on constant sets, but also on predicates on general variable sets, with associated Löb rule, and well-defined guarded recursive definitions of predicates.

Moreover, there is also a later operator $\blacktriangleright$ (a functor) on the variable sets themselves: $\blacktriangleright(X)$ is given by $\blacktriangleright(X)(1) = \{\star\}$ and $\blacktriangleright(X)(n+1) = X(n)$. We can show the well-definedness of recursive variable sets $\mu X.F(X)$ in which the recursion variable X is guarded by this operator $\blacktriangleright$. Intuitively, such a recursively specified variable set is well-defined since by definition of $\blacktriangleright$, to know what $\mu X.F(X)$ is at level $n+1$ it suffices to know what it is at level n.

In the technical sections of the paper, we make the above precise. In particular, we detail the internal logic and the use of later on functions / predicates and on types. We explain how one can solve mixed-variance recursive type equations, for a wide collection of types. We show how to use the internal logic of $\mathcal{S}$ to give a model of $\mathsf{F}_{\mu,\mathsf{ref}}$. The model, including the operational semantics of the programming language, is defined completely inside the internal logic; we discuss the connection between the resulting model and earlier models by relating internal definitions in the internal logic to standard (external) definitions. Since $\mathcal{S}$ is a topos, $\mathcal{S}$ also models dependent types. We give technical semantic results as needed for using later on dependent types and for recursive type-equations involving dependent types. We think of this as a first step towards a formalized dependent type theory with a later operator; here we focus on the foundational semantic issues.

To explain the relationship to some of the related work, we point out that $\mathcal{S}$ is equivalent to the category of sheaves on $\overline{\omega}$, where $\overline{\omega}$ is the complete Heyting algebra of natural numbers with the usual ordering and extended with a top element ∞. Moreover, this sheaf category, and hence also $\mathcal{S}$, is equivalent to the topos obtained by the tripos-to-topos construction [21] applied to the tripos $\mathbf{Set}(\_,\overline{\omega})$. The logic of constant sets in $\mathcal{S}$ is exactly the logic of this tripos.[1]

In the first part of this paper we work with the presentation of $\mathcal{S}$ as presheaves since it is the most concrete, but in fact many of our results generalize to sheaf categories over other complete well-founded Heyting algebras. Indeed, we include a more general axiomatic treatment of models of synthetic guarded domain theory and prove that, for any complete Heyting algebra with a well-founded basis, the topos of sheaves over the Heyting algebra yields a model of synthetic guarded domain theory. We present this generalization after the more concrete treatment of $\mathcal{S}$, since the concrete treatment of $\mathcal{S}$ is perhaps more accessible.

1.2. **Related work.**

Nakano presented a simple type theory with guarded recursive types [29] which can be modelled using complete bounded ultrametric spaces [6]. We show in Section 5 that the category *BiCBUlt* of bisected, complete bounded ultrametric spaces is a co-reflective subcategory of $\mathcal{S}$. Thus, our present work can be seen as an extension of the work of Nakano to include the full internal language of a topos, in particular dependent types, and an associated higher-order logic. Pottier [31] presents an extension of System F with recursive kinds based on Nakano's calculus; hence $\mathcal{S}$ also models the kind language of his system.

Di Gianantonio and Miculan [10] studied guarded recursive definitions of functions in certain sheaf toposes over well-founded complete Heyting algebras, thus including $\mathcal{S}$. Our work extends the work of Di Gianantonio and Miculan by also including guarded recursive definitions of *types*, by emphasizing the use of the internal logic (this was suggested as future work in [10]), and by including an extensive example application. Moreover, our general treatment of sheaf models includes sheaves over any well-founded complete Heyting algebra, whereas Di Gianantonio and Miculan restrict attention to those Heyting algebras that arise as the opens of a topological space.

[1] Recall that the tripos $\mathbf{Set}(\_,\overline{\omega})$ is a model of logic in which types and terms are interpreted as sets and functions, and predicates are interpreted as $\overline{\omega}$-valued functions.

Earlier work has advocated the use of complete bounded ultrametric spaces for solving recursive type and relation equations that come up when modelling programming languages with higher-order store [5, 7]. As mentioned above, *BiCBUlt* is a subcategory of $\mathcal{S}$, and thence our present work can be seen as an improvement of this earlier work: it is an improvement since $\mathcal{S}$ supports full higher-order logic. In the earlier work, one had to show that the functions defined in the interpretation of the programming language types were non-expansive. Here we take the synthetic approach (cf. [20]) and place ourselves in the internal logic of the topos when defining the interpretation of the programming language, see Section 3. This means that there is no need to prove properties like non-expansiveness since, intuitively, all functions in the topos are suitably non-expansive.

Dreyer *et al.* [11] proposed a logic, called LSLR, for defining step-indexed interpretations of programming languages with recursive types, building on earlier work by Appel *et al.* [3] who proposed the use of a later modality on predicates. The point of LSLR is that it provides for more abstract ways of constructing and reasoning with step-indexed models, thus avoiding tedious calculations with step indices. The core logic of LSLR is the logic of the tripos $\mathbf{Set}(\_, \overline{\omega})$ mentioned above,[2] which allows for recursively defined predicates following [3], but not recursively defined types. One point of passing from this tripos to the topos $\mathcal{S}$ is that it gives us a wider collection of types (variable sets rather than only constant sets), which makes it possible also to have mixed-variance recursively defined types.[3]

Dreyer *et al.* developed an extension of LSLR called LADR for reasoning about step-indexed models of the programming language $\mathsf{F}_{\mu,\mathsf{ref}}$ with higher-order store [13]. LADR is a specialized logic where much of the world structure used for reasoning efficiently about local state is hidden by the model of the logic; here we are proposing a general logic that can be used to construct many step-indexed models, including the one used to model LADR. In particular, in our example application in Section 3, we *define* a set of worlds inside the $\mathcal{S}$ logic, using recursively defined types.

As part of our analysis of recursive dependent types, we define a class of types, called *functorial types*. We show that functorial types are closed under nested recursive types, a result which is akin to results on nested inductive types [1, 14]. The difference is that we allow for general mixed-variance recursive types, but on the other hand we require that all occurrences of recursion variables must be guarded.

1.3. Summary of contributions.

We show how the topos $\mathcal{S}$, and, more generally, any topos of sheaves over a complete Heyting algebra with a well-founded basis, provides a simple but powerful model of guarded recursion, allowing for guarded recursive definitions of both terms and types in the internal dependently-typed higher-order logic. In particular, we

- show that the two later modalities are well-behaved on slices;
- give existence theorems for fixed points of guarded recursive terms and guarded nested dependent mixed-variance recursive types;
- detail the relation of $\mathcal{S}$ to the category of complete bounded ultrametric spaces;

[2]Dreyer *et al.* [11] presented the semantics of their second-order logic in more concrete terms, avoiding the use of triposes, but it is indeed a fragment of the internal logic of the mentioned tripos.

[3]The terminology can be slightly confusing: in [3], our notion of recursive relations were called recursive types, probably because the authors of *loc.cit.* used such to interpret recursive types of a programming language. Recursive types in our sense were not considered in [3].

- present, as an example application, a synthetic model of $\mathsf{F}_{\mu,\mathsf{ref}}$ constructed internally in $\mathcal{S}$;
- give an axiomatic treatment of a general class of models of guarded recursion.

Our general existence theorems for recursive types in Section 8 are phrased in terms of $\mathit{Sh}(A)$-categories, i.e., categories enriched in sheaves over a complete Heyting algebra A with a well-founded basis, and generalize earlier work on recursive types for categories enriched in complete bounded ultrametric spaces [9].

2. The $\mathcal{S}$ Topos

The category $\mathcal{S}$ is that of presheaves on ω, the preorder of natural numbers starting from 1 and ordered by inclusion. Explicitly, the objects of $\mathcal{S} = \mathbf{Set}^{\omega^{\mathrm{op}}}$ are families of sets indexed by natural numbers together with restriction maps $r_n \colon X(n+1) \to X(n)$. Morphisms are families $(f_n)_n$ of maps commuting with the restriction maps as indicated in the diagram

$$\begin{tikzcd} X(1) \arrow[d, "f_1"'] & X(2) \arrow[l] \arrow[d, "f_2"'] & X(3) \arrow[l] \arrow[d, "f_3"'] & \dots \arrow[l] \\ Y(1) & Y(2) \arrow[l] & Y(3) \arrow[l] & \dots \arrow[l] \end{tikzcd}$$

If $x \in X(m)$ and $n \leq m$ we write $x|_n$ for $r_n \circ \cdots \circ r_{m-1}(x)$.

As all presheaf categories, $\mathcal{S}$ is a topos, in particular it is cartesian closed and has a subobject classifier. Moreover, it is complete and cocomplete, and limits and colimits are computed pointwise. The n'th component of the exponential $Y^X(n)$ is the set of tuples $(f_1, \dots, f_n)$ commuting with the restriction maps, and the restriction maps of Y^X are given by projection. We sometimes use the notation $X \to Y$ for Y^X.

A subobject A of X is a family of subsets $A(n) \subseteq X(n)$ such that $r_n(A(n+1)) \subseteq A(n)$. The subobject classifier has $\Omega(n) = \{0, \dots, n\}$ and restriction maps $r_n(x) = \min(n, x)$. The characteristic morphism $\chi_A \colon X \to \Omega$ maps $x \in X(n)$ to the maximal m such that $x|_m \in A(m)$ if such an m exists and 0 otherwise.

The natural numbers object N in $\mathcal{S}$ is the constant set of natural numbers.

Intuitively, we can think of the set $X(n)$ as what the type X looks like, if one has at most n time steps to reason about it. The restriction maps $r_n \colon X(n+1) \to X(n)$ describe what happens to the data when one time step passes. This intuition is illustrated by the following example.

Example 2.1. We can define the object $\mathit{Str} \in \mathcal{S}$ of *(variable) streams* of natural numbers as follows:

$$\begin{tikzcd} N^1 & N^2 \arrow[l] & N^3 \arrow[l] & \dots \arrow[l] \end{tikzcd}$$

where the restriction maps $r_m : N^{m+1} \to N^m$ map $(n_1, \dots, n_m, n_{m+1})$ to $(n_1, \dots, n_m)$. Intuitively, this is the type of streams where the head is immediately available, but the tail is only available after one time step. If we have n time steps to reason about this type we can access the n first elements, hence $\mathit{Str}(n) = N^n$.

The successor function *succ* on streams, which adds one to every element in a stream, can be defined in the model by

$$\mathit{succ}_m : N^m \to N^m = (n_1, \dots, n_m) \mapsto (n_1 + 1, \dots, n_m + 1).$$

Clearly *succ* is a natural transformation from *Str* to *Str*; hence it is a well-defined map in $\mathcal{S}$. Observe that $succ_m$ can also be defined by induction as $succ_1(n) = n+1$ and $succ_{m+1}(n_1, n_2, \ldots, n_{m+1}) = (n_1+1, succ_m(n_2, \ldots, n_{m+1}))$.

The subobject $A \subseteq Str$ of increasing streams can be defined by letting $A_m \subseteq N^m$ be the set of tuples $(n_1, \ldots, n_m)$ that are increasing (i.e., $n_i > n_j$, for $i > j$). Note that A is trivially closed under the restriction maps, and thus it is a well-defined subobject of *Str*.

2.1. The $\blacktriangleright$ endofunctor.

Define the functor $\blacktriangleright\colon \mathcal{S} \to \mathcal{S}$ by $\blacktriangleright X(1) = \{\star\}$ and $\blacktriangleright X(n+1) = X(n)$. This functor, called *later*, has a left adjoint (so $\blacktriangleright$ preserves all limits) given by $\blacktriangleleft X(n) = X(n+1)$. Since limits are computed pointwise, $\blacktriangleleft$ preserves them, and so the adjunction $\blacktriangleleft \dashv \blacktriangleright$ defines a geometric morphism, in fact an embedding. However, we shall not make use of this fact in the present paper (because $\blacktriangleleft$ is not a fibred endo-functor on the codomain fibration, hence is not a useful operator in the dependent type theory; see Section 4).

There is a natural transformation $\mathrm{next}_X\colon X \to \blacktriangleright X$ whose 1st component is the unique map into $\{\star\}$ and whose $(n+1)$th component is r_n. Although next looks like a unit $\blacktriangleright$ is not a monad: there are no natural transformations $\blacktriangleright\blacktriangleright \to \blacktriangleright$.

Since $\blacktriangleright$ preserves finite limits, there is always a morphism

$$J : \blacktriangleright(X \to Y) \to (\blacktriangleright X \to \blacktriangleright Y). \tag{2.1}$$

2.2. An operator on predicates.

There is a morphism $\rhd\colon \Omega \to \Omega$ mapping $n \in \Omega(m)$ to $\min(m, n+1)$. By setting $\chi_{\rhd A} = \rhd \circ \chi_A$ there is an induced operation on subobjects, again denoted $\rhd$. This operation, which we also call "later", is connected to the $\blacktriangleright$ functor, since there is a pullback diagram

$$\begin{tikzcd} \rhd m \arrow[r] \arrow[d] \arrow[dr, phantom, "\lrcorner", very near start] & \blacktriangleright A \arrow[d, "\blacktriangleright m"] \\ X \arrow[r, "\mathrm{next}_X"] & \blacktriangleright X \end{tikzcd}$$

for any subobject $m\colon A \to X$.

2.3. Recursive morphisms.

We introduce a notion of contractive morphism and show that these have unique fixed points.

Definition 2.2. A morphism $f\colon X \to Y$ is *contractive* if there exists a morphism $g\colon \blacktriangleright X \to Y$ such that $f = g \circ \mathrm{next}_X$. A morphism $f\colon X \times Y \to Z$ is contractive in the first variable if there exists g such that $f = g \circ (\mathrm{next}_X \times \mathrm{id}_Y)$.

For instance, contractiveness of $\rhd$ on Ω is witnessed by $\mathrm{succ}\colon \blacktriangleright\Omega \to \Omega$ with $\mathrm{succ}_n(k) = k+1$.

Lemma 2.3.
(1) If $f\colon X \to Y$ and $g\colon Y \to Z$ and either f or g are contractive also gf is contractive.
(2) If $f\colon X \to Y$ and $g\colon X' \to Y'$ are contractive, so is $f \times g\colon X \times X' \to Y \times Y'$.
(3) A morphism $h\colon X \times Y \to Z$ is contractive in the first variable iff $\hat{h}\colon X \to Z^Y$ is contractive.

If $f\colon X \to Y$ is contractive as witnessed by g, the value of $f_{n+1}(x)$ can be computed from $r_n(x)$ and moreover, f_1 must be constant. If $X = Y$ we can define a fixed point $x\colon 1 \to X$ by defining $x_1 = g_1(\star)$ and $x_{n+1} = g_{n+1}(x_n)$. This construction can be generalized to include fixed points of morphisms with parameters as follows.

Theorem 2.4. There exists a natural family of morphisms $\mathrm{fix}_X\colon (\blacktriangleright X \to X) \to X$, indexed by the collection of all objects X, which computes unique fixed points in the sense that if $f\colon X\times Y \to X$ is contractive in the first variable as witnessed by g, i.e., $f = g\circ(\mathrm{next}_X \times \mathrm{id}_Y)$, then $\mathrm{fix}_X \circ \hat{g}$ is the unique $h\colon Y \to X$ such that $f \circ \langle h, \mathrm{id}_Y\rangle = h$ (here $\hat{g}$ denotes the exponential transpose of g).

2.4. **Internal logic.** We start by calling to mind parts of the Kripke-Joyal forcing semantics for $\mathcal{S}$. For $X_1, \dots, X_m$ in $\mathcal{S}$, $\varphi : X_1 \times \cdots \times X_m \to \Omega$, $n \in \omega$, and $\alpha_1 \in X_1(n), \dots, \alpha_m \in X_m(n)$, we define $n \models \varphi(\alpha_1, \dots, \alpha_m)$ iff $\varphi_n(\alpha_1, \dots, \alpha_m) = n$.

The standard clauses for the forcing relation are as follows [25, Example 9.5] (we write $\overline{\alpha}$ for a sequence $\alpha_1, \dots, \alpha_m$):

$$\begin{aligned}
n \models (s = t)\overline{\alpha} &\Leftrightarrow [\![s]\!]_n(\overline{\alpha}) = [\![t]\!]_n(\overline{\alpha})\\
n \models R(t_1, \dots, t_k)\overline{\alpha} &\Leftrightarrow [\![R]\!]_n([\![t_1]\!]_n(\overline{\alpha}), \dots, [\![t_k]\!]_n(\overline{\alpha})) = n\\
n \models (\varphi \wedge \psi)(\overline{\alpha}) &\Leftrightarrow n \models \varphi(\overline{\alpha}) \wedge n \models \psi(\overline{\alpha})\\
n \models (\varphi \vee \psi)(\overline{\alpha}) &\Leftrightarrow n \models \varphi(\overline{\alpha}) \vee n \models \psi(\overline{\alpha})\\
n \models (\varphi \to \psi)(\overline{\alpha}) &\Leftrightarrow \forall k \le n.\, k \models \varphi(\overline{\alpha}|_k) \to k \models \psi(\overline{\alpha}|_k)\\
n \models (\exists x{:}X.\varphi)(\overline{\alpha}) &\Leftrightarrow \exists \alpha \in [\![X]\!](n).\, n \models \varphi(\overline{\alpha}, \alpha)\\
n \models (\forall x{:}X.\varphi)(\overline{\alpha}) &\Leftrightarrow \forall k \le n, \alpha \in [\![X]\!](k).\, k \models \varphi(\overline{\alpha}|_k, \alpha)
\end{aligned}$$

Proposition 2.5. $\triangleright$ is the unique morphism on Ω satisfying $1 \models \triangleright\varphi(\alpha)$ and $n+1 \models \triangleright\varphi(\alpha) \Leftrightarrow n \models \varphi(\alpha|_n)$. Moreover, $\forall x, y\colon X.\, \triangleright(x = y) \leftrightarrow \mathrm{next}_X(x) = \mathrm{next}_X(y)$ holds in $\mathcal{S}$.

The following definition will be useful for presenting facts about the internal logic of $\mathcal{S}$.

Definition 2.6. An object X in $\mathcal{S}$ is *total* if all the restriction maps r_n are surjective.

Hence all constant objects $\Delta(S)$ are total, but the total objects also include many non-constant objects, e.g., the subobject classifier. The above definition is phrased in terms of the model; the internal logic can be used to give a simple characterization of when X is total and inhabited by a global element[4] : that is the case iff next_X is internally surjective in $\mathcal{S}$, i.e., iff $\forall y : \blacktriangleright X.\exists x : X.\mathrm{next}_X(x) = y$ holds in $\mathcal{S}$. The following proposition can be proved using the forcing semantics; note that the distribution rules below for $\triangleright$ generalize the ones for constant sets described in [11] (since constant sets are total).

Theorem 2.7. In the internal logic of $\mathcal{S}$ we have:

(1) (Monotonicity). $\forall p : \Omega.\, p \to \triangleright p$.
(2) (Löb rule). $\forall p : \Omega.\, (\triangleright p \to p) \to p$.
(3) $\triangleright$ commutes with the logical connectives $\top$, $\wedge$, $\to$, $\vee$, but does not preserve $\bot$.
(4) For all X, Y, and φ, we have $\exists y : Y.\, \triangleright\varphi(x, y) \to \triangleright(\exists y : Y.\, \varphi(x, y))$. The implication in the opposite direction holds if Y is total and inhabited.

[4]X is inhabited by a global element if there exists a morphism $x\colon 1 \to X$

(5) For all X, Y, and φ, we have $\triangleright(\forall y : Y.\, \varphi(x,y)) \to \forall y : Y.\, \triangleright\, \varphi(x,y)$. The implication in the opposite direction holds if Y is total.

We now define an internal notion of contractiveness in the logic of $\mathcal{S}$ which implies (in the logic) the existence of a unique fixed point for inhabited types.

Definition 2.8. The predicate Contr on Y^X is defined in the internal logic by

$$\mathrm{Contr}(f) \overset{\mathrm{def}}{\iff} \forall x, x' : X.\, \triangleright(x = x') \to f(x) = f(x').$$

For a morphism $f : X \to Y$, corresponding to a global element of Y^X, we have that if f is contractive (in the external sense of Definition 2.2), then $\mathrm{Contr}(f)$ holds in the logic of $\mathcal{S}$. The converse is true if X is total and inhabited, but not in general. We use both notions of contractiveness: the external notion provides for a simple algebraic theory of fixed points for not only morphisms but also functors (see Section 2.6), whereas the internal notion is useful when working in the internal logic.

The internal notion of contractiveness generalizes the usual metric notion of contractiveness for functions between complete bounded ultrametric spaces; see Section 5.

Theorem 2.9 (Internal Banach Fixed-Point Theorem)**.** The following holds in $\mathcal{S}$:

$$(\exists x : X.\top) \wedge \mathrm{Contr}(f) \to \exists! x : X.\, f(x) = x.$$

The above theorem (the Internal Banach Fixed-Point Theorem) is proved in the internal logic using the following lemma, which expresses a non-classical property. The lemma can be proved in the internal logic using the Löb rule (and using that N is a total object) — below we give a semantic proof using the Kripke-Joyal semantics.

Lemma 2.10. The following holds in $\mathcal{S}$:

$$\mathrm{Contr}(f) \to \exists n : N. \forall x, x' : X.\, f^n(x) = f^n(x').$$

Proof. We must show that any m forces the predicate. Unfolding the definition of the forcing relation, we see that it suffices to show that for all m and all $f \in X^X(m)$ there exists an n such that

$$m \models \mathrm{Contr}(f) \to m \models \forall x, x' : X.\, f^n(x) = f^n(x')$$

The element f is a family $(f_i \colon X(i) \to X(i))_{i \le m}$ and the condition $m \models \mathrm{Contr}(f)$ implies that $f_i^i(x) = f_i^i(y)$ for all $i \le m$ and all $x, y \in X(i)$. In particular f_i^m is constant. Therefore choosing $n = m$ makes $m \models \forall x, x' : X.\, f^n(x) = f^n(x')$ true. □

2.5. **Recursive relations.** As an example application of Theorem 2.9, we consider the definition of recursive predicates. Let $\varphi(r) : \Omega^X$ be a predicate on X in the internal logic of $\mathcal{S}$ as presented above (over non-dependent types, but possibly using $\triangleright$) with free variable r, also of type Ω^X. Note that Ω^X is inhabited by a global element. If r only occurs under a $\triangleright$ in φ, then φ defines an internally contractive map $\varphi \colon \Omega^X \to \Omega^X$ (proved by external induction on φ). Therefore, by Theorem 2.9, $\exists!\, r \colon \Omega^X. \varphi(r) = r$ holds in $\mathcal{S}$. By description (aka axiom of unique choice), which holds in any topos [25], there is then a morphism $R : 1 \to \Omega^X$ such that $\varphi(R) = R$ in $\mathcal{S}$, and since internal and external equality coincides, also $\varphi(R) = R$ externally as morphisms $1 \to \Omega^X$. In summa, we have shown the well-definedness of recursive predicates $r = \varphi(r)$ where r only occurs guarded by $\triangleright$ in φ.

Note that we have *proved* the existence of recursive guarded relations (and thus do not have to add them to the language with special syntax) since we are working with a higher-order logic.

Example 2.11. Suppose $R \subseteq X \times X$ is some relation on a set X. We can include it into $\mathcal{S}$ by using the functor $\Delta\colon \mathbf{Set} \to \mathcal{S}$, obtaining $\Delta R \subseteq \Delta X \times \Delta X$. Consider the recursive relation

$$R^\omega(x,y) \overset{\text{def}}{\iff} (x = y) \vee \exists z.(\Delta R(x,z) \wedge \rhd R^\omega(z,y))\,.$$

Now, $n+1 \models R^\omega(x,y)$ iff $(x,y) \in \cup_{0\le i \le n} R^i$ or there exists z such that $R^{n+1}(x,z)$. If R is a rewrite relation then $n+1 \models R^\omega(x,y)$ states the extent to which we can determine if x rewrites to y by inspecting all rewrite sequences of length at most n.

A variant of Example 2.11 is used in Section 3.

2.6. Recursive domain equations.

In this section we present a simplified version of our results on solutions to recursive domain equations in $\mathcal{S}$ sufficient for the example of Section 3. The full results on recursive domain equations can be found in Section 4.

Denote by $\ulcorner f \urcorner\colon 1 \to Y^X$ the curried version of $f\colon X \to Y$. Following Kock [24] we say that an endofunctor $F\colon \mathcal{S} \to \mathcal{S}$ is *strong* if, for all X, Y, there exists a morphism $F_{X,Y}\colon Y^X \to FY^{FX}$ such that $F_{X,Y} \circ \ulcorner f \urcorner = \ulcorner Ff \urcorner$ for all f.

Definition 2.12. A strong endofunctor on $\mathcal{S}$ is *locally contractive* if each $F_{X,Y}$ is contractive, i.e., there exists a family $G_{X,Y}$ such that $G_{X,Y} \circ \mathrm{next}_{X^Y} = F_{X,Y}$ and moreover G respects identity and composition, that is the following diagrams commute

$$\begin{tikzcd}
\blacktriangleright(Y^X) \times \blacktriangleright(Z^Y) \arrow[r, "\cong"] \arrow[d, "G_{X,Y} \times G_{Y,Z}"'] & \blacktriangleright(Y^X \times Z^Y) \arrow[r, "\blacktriangleright(comp)"] & \blacktriangleright(Z^X) \arrow[d, "G_{X,Z}"] \\
FY^{FX} \times FZ^{FY} \arrow[rr, "comp"] & & FZ^{FX}
\end{tikzcd}
\qquad
\begin{tikzcd}
1 \arrow[r, "\blacktriangleright\ulcorner \mathrm{id} \urcorner"] \arrow[dr, "\ulcorner id \urcorner"'] & \blacktriangleright(X^X) \arrow[d, "G_{X,X}"] \\
& X^X
\end{tikzcd}$$

This notion readily generalizes to mixed-variance endofunctors on $\mathcal{S}$.

Remark 2.13. Definition 2.12 is slightly less general than the one given in the conference version of this paper [4] where local contractiveness simply required $F_{X,Y}$ to be contractive. The definition given here greatly simplifies the proof of existence of solutions to recursive domain equations, especially in the general case as presented in Section 8, and at the same time, the extra requirements used here do not rule out any examples we know of. In particular, the syntactic conditions for well-definedness of recursive types remain unchanged.

The requirement of G commuting with composition and identity can be rephrased as G defining an enriched functor. In Section 6 we use this observation to generalise the notion of locally contractive functor.

For example, $\blacktriangleright$ is locally contractive (as witnessed by J (2.1)), and one can show that the composition of a strong functor and a locally contractive functor (in either order) is locally contractive (see Lemma 7.3 for a generalized statement). As a result, one can show that any type expression $A(X,Y)$ constructed from type variables X, Y using $\blacktriangleright$ and simple type constructors in which X occurs only negatively and Y only positively and both only under $\blacktriangleright$ gives rise to a locally contractive functor. Indeed, in Section 4 we present such syntactic conditions ensuring that a type expression in dependent type theory induces a locally contractive functor.

Theorem 2.14. Let $F\colon \mathcal{S}^{\mathrm{op}} \times \mathcal{S} \to \mathcal{S}$ be a locally contractive functor. Then there exists a unique X (up to isomorphism) such that $F(X, X) \cong X$.

Section 8 gives a detailed proof of a generalised version of this theorem. Here we just sketch a proof. We consider first the covariant case.

Lemma 2.15. Let $F\colon \mathcal{S} \to \mathcal{S}$ be locally contractive and say that $f\colon X \to Y$ is an n-isomorphism if f_i is an isomorphism for all $i \le n$. Then F maps n-isomorphisms to $n+1$-isomorphisms for all n.

Since any morphism $f\colon X \to Y$ is a 0-isomorphism $F^n(f)\colon F^nX \to F^nY$ is an n-isomorphism. Consider the sequence

$$\begin{tikzcd} F1 & F^21 \arrow[l, "F!"'] & F^31 \arrow[l, "F^2(!)"'] & F^41 \arrow[l, "F^3(!)"'] & \dots \end{tikzcd} \tag{2.2}$$

The sequence above is a sequence of morphisms and objects in $\mathcal{S}$ and so represents a diagram of sets and functions as in

$$\begin{tikzcd}
F(1)(1) & F^2(1)(1) \arrow[l, "F(!)_1"'] & F^3(1)(1) \arrow[l, "F^2(!)_1"'] & F^4(1)(1) \arrow[l, "F^3(!)_1"'] & \dots \arrow[l] \\
F(1)(2) \arrow[u] & F^2(1)(2) \arrow[l, "F(!)_2"'] \arrow[u] & F^3(1)(2) \arrow[l, "F^2(!)_2"'] \arrow[u] & F^4(1)(2) \arrow[l, "F^3(!)_2"'] \arrow[u] & \dots \arrow[l] \\
F(1)(3) \arrow[u] & F^2(1)(3) \arrow[l, "F(!)_3"'] \arrow[u] & F^3(1)(3) \arrow[l, "F^2(!)_3"'] \arrow[u] & F^4(1)(3) \arrow[l, "F^3(!)_3"'] \arrow[u] & \dots \arrow[l] \\
\vdots & \vdots & \vdots & \vdots &
\end{tikzcd} \tag{2.3}$$

By the above observation, $F^n(!)_k$ is an isomorphism for $k \le n$, in other words, after k iterations of F the first k components are fixed by further iterations of F. Intuitively, we can therefore form a fixed point for F by taking the diagonal of (2.3), i.e, the object whose k'th component is $F^k(1)(k)$. Indeed, in Section 8 we construct this object as the limit of (2.2).

Any fixed point for such an F must be at the same time an initial algebra and a final coalgebra: given any fixed point $f\colon FX \cong X$ and algebra $g\colon FY \to Y$ a morphism $h\colon X \to Y$ is a homomorphism iff $\ulcorner h \urcorner$ is a fixed point of $\xi = \lambda k\colon X \to Y.\, g \circ Fk \circ f^{-1}$. Since F is locally contractive, ξ is contractive and so must have a unique fixed point. The case of final coalgebras is similar.

Thus, $\mathcal{S}$ is algebraically compact in the sense of Freyd [15–17] with respect to locally contractive functors. The solutions to general recursive domain equations can then be established using Freyd's constructions.

Example 2.16. Recall the type *Str* of streams defined concretely in the model in Example 2.1. It can be defined in the internal language using Theorem 2.14, namely as the type satisfying the recursive domain equation

$$\mathit{Str} \cong N \times \blacktriangleright \mathit{Str}.$$

Write $i : N \times \blacktriangleright \mathit{Str} \to \mathit{Str}$ for the isomorphism. (Observe that i_m is nothing but the identity function.)

Now, we can define the successor function in the internal language as the fixed point of the following contractive function $F : (Str \to Str) \to (Str \to Str)$:

$$\begin{aligned} F(f) = \lambda s.\mathsf{let}\ (n,t) \Leftarrow i^{-1}(s) \\ \mathsf{in}\ i(n+1, J(\mathrm{next} f)(t)) \end{aligned}$$

Note that F is clearly contractive (in the external sense) since the argument f is only used under next in $F(f)$. Hence F has a fixed point, which is indeed the successor function from Example 2.1, i.e., $succ = \mathrm{fix}_{Str \to Str} F$.

3. Application to Step-Indexing

As an example, we now construct a model of a programming language with higher-order store and recursive types entirely inside the internal logic of $\mathcal{S}$. There are two points we wish to make here. First, although the programming language is quite expressive, the internal model looks—almost—like a naive, set-theoretic model. The exception is that guarded recursion is used in a few, select places, such as defining the meaning of recursive types, where the naive approach would fail. Second, when viewed externally, we recover a standard, step-indexed model. This example therefore illustrates that the topos of trees gives rise to simple, synthetic accounts of step-indexed models.

All definitions and results in Sections 3.1 to 3.4 are in the internal logic of $\mathcal{S}$. In Section 3.5 we investigate what these results mean externally.

3.1. **Language.** The types and terms of $\mathsf{F}_{\mu,\mathsf{ref}}$ are as follows:

$$\tau ::= 1 \mid \tau_1 \times \tau_2 \mid \mu\alpha.\tau \mid \forall\alpha.\tau \mid \alpha \mid \tau_1 \to \tau_2 \mid \mathsf{ref}\,\tau$$

$$\begin{aligned} t ::= x \mid l \mid () \mid (t_1,\, t_2) \mid \mathsf{fst}\,t \mid \mathsf{snd}\,t \mid \mathsf{fold}\,t \mid \mathsf{unfold}\,t \mid \\ \Lambda\alpha.t \mid t\,[\tau] \mid \overline{\lambda}x.t \mid t_1\,t_2 \mid \mathsf{ref}\,t \mid\, !t \mid t_1 := t_2 \end{aligned}$$

(The full term language also includes sum types, and can be found in Appendix A.) Here l ranges over location constants, which are encoded as natural numbers.

More explicitly, the sets OType and OTerm of possibly open types and terms are defined by induction according to the grammars above (using that toposes model W-types [27]), and then by quotienting with respect to α-equivalence.

The set OValue of syntactic values is an inductively defined subset of OTerm:

$$v ::= x \mid l \mid () \mid (v_1,\, v_2) \mid \mathsf{fold}\,v \mid \Lambda\alpha.t \mid \overline{\lambda}x.t$$

Let Term and Value be the subsets of closed terms and closed values, respectively. Let Store be the set of finite maps from natural numbers to closed values; this is encoded as the set of those finite lists of pairs of natural numbers and closed values that contain no number twice. Finally, let Config = Term $\times$ Store.

The typing judgements have the form $\Xi \mid \Gamma \vdash t : \tau$ where Ξ is a context of type variables and Γ is a context of term variables. The typing rules are standard and can be found in Appendix A. Notice, however, that there is no context of location variables and no typing judgement for stores: we only need to type-check terms that can occur in programs.

3.2. **Operational semantics.** We assume a standard one-step relation step: $\mathcal{P}(\text{Config} \times \text{Config})$ on configurations by induction, following the usual presentation of such relations by means of inference rules (see, e.g., the online appendix to [13]). For simplicity, allocation is deterministic: when allocating a new reference cell, we choose the smallest location not already in the store. Notice that the step relation is defined on untyped configurations. Erroneous configurations are "stuck."

So far, we have defined the language and operational semantics exactly as we would in standard set theory. Next comes the crucial difference. We use Theorem 2.9 to define the predicate eval: $\mathcal{P}(\text{Term} \times \text{Store} \times \mathcal{P}(\text{Value} \times \text{Store}))$,

$$\begin{aligned}&\text{eval}(t, s, Q)\\ &\quad\overset{\text{def}}{\iff} (t \in \text{Value} \ \wedge \ Q(t, s)) \vee\\ &\qquad\quad (\exists t_1 \colon \text{Term}, s_1 \colon \text{Store}.\\ &\qquad\qquad \text{step}((t, s), (t_1, s_1)) \ \wedge \ \triangleright \text{eval}(t_1, s_1, Q))\end{aligned}$$

Intuitively, the predicate Q is a post-condition, and $\text{eval}(t, s, Q)$ is a *partial* correctness specification, in the sense of Hoare logic, meaning the following: (1) The configuration (t, s) is safe, i.e., it does not lead to an error. (2) *If* the configuration (t, s) evaluates to some pair (v, s'), *then* at that point in time (v, s') satisfies Q. We shall justify this intuition in Section 3.5 below. The use of $\triangleright$ ensures that the predicate is well-defined; in effect, we postulate that one evaluation step in the programming language actually takes one unit of time in the sense of the internal logic. As we shall see below, this "temporal" semantics is essential in the proof of the fundamental theorem of logical relations.

Notice how guarded recursion is used to give a simple, coinduction-style definition of partial correctness. The Löb rule can then be conveniently used for reasoning about this definition. For example, the rule gives a very easy proof that if (t, s) is a configuration that reduces to itself in the sense that $\text{step}((t, s), (t, s))$ holds, then $\text{eval}(t, s, Q)$ holds for any Q. The Löb rule also proves the following results, which are used to show the fundamental theorem below.

Proposition 3.1. Let $Q, Q' \in \mathcal{P}(\text{Value} \times \text{Store})$ such that $Q \subseteq Q'$. Then for all t and s we have that $\text{eval}(t, s, Q)$ implies $\text{eval}(t, s, Q')$.

Proposition 3.2. For all stores s, all terms t, all evaluation contexts E such that $E[t]$ is closed, and all predicates $Q \in \mathcal{P}(\text{Value} \times \text{Store})$, we have that $\text{eval}(E[t], s, Q)$ holds iff $\text{eval}(t, s, \lambda(v_1, s_1).\, \text{eval}(E[v_1], s_1, Q))$ holds.

3.3. **Definition of Kripke worlds.** The main idea behind our interpretation of types is as in [5, 8]: Since $\mathsf{F}_{\mu,\mathsf{ref}}$ includes reference types, we use a Kripke model of types, where a semantic type is defined to be a world-indexed family of sets of syntactic values. A world is a map from locations to semantic types. This introduces a circularity between semantic types $\mathcal{T}$ and worlds $\mathcal{W}$, which can be expressed as a pair of domain equations: $\mathcal{W} = N \rightharpoonup_{\text{fin}} \mathcal{T}$ and $\mathcal{T} = \mathcal{W} \to_{\text{mon}} \mathcal{P}(\text{Value})$.

Rather than solving the above stated domain equations exactly, we solve a guarded variant. More precisely, we define the set

$$\widehat{\mathcal{T}} = \mu X.\, \blacktriangleright((N \rightharpoonup_{\text{fin}} X) \to_{\text{mon}} \mathcal{P}(\text{Value}))\ .$$

Here $N \to_{\mathrm{fin}} X$ is the set $\sum_{A\colon \mathcal{P}_{\mathrm{fin}}(N)} X^A$ where $\mathcal{P}_{\mathrm{fin}}(N) = \{A \subset N \mid \exists m \forall n \in A.\, n < m\}$. The set $\sum_{A\colon \mathcal{P}_{\mathrm{fin}}(N)} X^A$ is ordered by graph inclusion and $\to_{\mathrm{mon}}$ is the set of monotonic functions realized as a subset type on the function space.

The type $\widehat{\mathcal{T}}$ can be seen to be well-defined as a consequence of the theory of Section 4, in particular Proposition 4.10. Alternatively, observe that the corresponding functor is of the form $F = \blacktriangleright \circ G$. Here G is strong because its action on morphisms can be defined as a term $Y^X \to GY^{GX}$ in the internal logic. Now, since $\blacktriangleright$ is locally contractive so is F. Hence by Theorem 2.14, F has a unique fixed point $\widehat{\mathcal{T}}$, with an isomorphism $i\colon \widehat{\mathcal{T}} \to F(\widehat{\mathcal{T}})$. We define

$$\mathcal{W} = N \to_{\mathrm{fin}} \widehat{\mathcal{T}}\ , \qquad\qquad \mathcal{T} = \mathcal{W} \to_{\mathrm{mon}} \mathcal{P}(\mathrm{Value})\ ,$$

and $\mathcal{T}^{\mathrm{c}} = \mathcal{W} \to \mathcal{P}(\mathrm{Term})$. Notice that $\widehat{\mathcal{T}}$ is isomorphic to $\blacktriangleright \mathcal{T}$. We now define $\mathrm{app}\colon \widehat{\mathcal{T}} \to \mathcal{T}$ and $\mathrm{lam}\colon \mathcal{T} \to \widehat{\mathcal{T}}$ as follows. First, app is the isomorphism i composed with the operator $d\colon\ \blacktriangleright \mathcal{T} \to \mathcal{T}$ given by

$$d(f) = \lambda w.\lambda v.\mathrm{succ}(J(J(f)(\mathrm{next}\, w))(\mathrm{next}\, v)),$$

where J is the map in (2.1) and $\mathrm{succ}\colon\ \blacktriangleright \Omega \to \Omega$ is as defined on page 6. (This is a general way of lifting algebras for $\blacktriangleright$ to function spaces.) Here one needs to check that d is well-defined, i.e., preserves monotonicity. Second, $\mathrm{lam}\colon \mathcal{T} \to \widehat{\mathcal{T}}$ is defined by $\mathrm{lam} = i^{-1} \circ \mathrm{next}_{\mathcal{T}}$.

Define $\triangleright : \mathcal{T} \to \mathcal{T}$ as the pointwise extension of $\triangleright : \Omega \to \Omega$, i.e., for $\nu \in \mathcal{T}$, $w \in \mathcal{W}$ and $v \in \mathrm{Value}$, we have that $(\triangleright\, \nu)(w)(v)$ holds iff $\triangleright(\nu(w)(v))$ holds.

Lemma 3.3. $\mathrm{app} \circ \mathrm{lam} = \triangleright : \mathcal{T} \to \mathcal{T}$.

3.4. **Interpretation of types.** Let TVar be the set of type variables, and for $\tau \in \mathrm{OType}$, let $\mathrm{TEnv}(\tau) = \{\, \varphi \in \mathrm{TVar} \to_{\mathrm{fin}} \mathcal{T} \mid \mathrm{FV}(\tau) \subseteq \mathrm{dom}(\varphi)\,\}$. The interpretation of programming-language types is defined by induction, as a function

$$[\![\cdot]\!] : \prod_{\tau \in \mathrm{OType}} \mathrm{TEnv}(\tau) \to \mathcal{T}\,.$$

We show some cases of the definition here; the complete definition can be found in Appendix A.2.

$$\begin{aligned}
[\![\alpha]\!]\varphi &= \varphi(\alpha)\\
[\![\tau_1 \times \tau_2]\!]\varphi &= \lambda w.\, \{(v_1,\, v_2) \mid v_1 \in [\![\tau_1]\!]\varphi(w) \wedge v_2 \in [\![\tau_2]\!]\varphi(w)\}\\
[\![\mathsf{ref}\ \tau]\!]\varphi &= \lambda w.\, \{\, l \mid l \in \mathrm{dom}(w) \wedge \forall w_1 \geq w.\, \mathrm{app}(w(l))(w_1) = \triangleright([\![\tau]\!]\varphi)(w_1)\,\}\\
[\![\forall \alpha.\tau]\!]\varphi &= \lambda w.\, \{\, \Lambda\alpha.t \mid \forall \nu \in \mathcal{T}.\, \forall w_1 \geq w.\, t \in \mathit{comp}([\![\tau]\!]\varphi[\alpha \mapsto \nu])(w_1)\,\}\\
[\![\mu\alpha.\tau]\!]\varphi &= \mathit{fix}\,(\lambda\nu.\, \lambda w.\, \{\, \mathsf{fold}\ v \mid \triangleright(v \in [\![\tau]\!]\varphi[\alpha \mapsto \nu]\,(w))\})\\
[\![\tau_1 \to \tau_2]\!]\varphi &= \lambda w.\, \{\, \overline{\lambda} x.t \mid \forall w_1 \geq w.\, \forall v \in [\![\tau_1]\!]\varphi(w_1).\, t[v/x] \in \mathit{comp}([\![\tau_2]\!]\varphi)(w_1)\,\}
\end{aligned}$$

Here the operations $\mathit{comp} : \mathcal{T} \to \mathcal{T}^{\mathrm{c}}$ and $\mathit{states} : \mathcal{W} \to \mathcal{P}(\mathrm{Store})$ are given by

$$\begin{aligned}
\mathit{comp}(\nu)(w) = \{\, t \mid \forall s \in \mathit{states}(w).\, \mathit{eval}(t, s,\, \lambda(v_1, s_1).\, \exists w_1 \geq w.&\\
v_1 \in \nu(w_1)\ \wedge\ s_1 \in \mathit{states}(w_1))\,\}&\\
\mathit{states}(w) = \{\, s \mid \mathrm{dom}(s) = \mathrm{dom}(w)\ \wedge&\\
\forall l \in \mathrm{dom}(w).\, s(l) \in \mathrm{app}(w(l))(w)\,\}.&
\end{aligned}$$

Notice that this definition is almost as simple as an attempt at a naive, set-theoretic definition, except for the two explicit uses of $\triangleright$. In the definition of $[\![\mu\alpha.\tau]\!]$, the use of $\triangleright$ ensures that the fixed point is well-defined according to Theorem 2.9. As for the definition of $[\![\mathsf{ref}\,\tau]\!]$, the $\triangleright$ is needed because we have $\triangleright$ instead of the identity in Lemma 3.3. In both cases, the intuition is the usual one from step-indexing: since an evaluation step takes a unit of time, it suffices that a certain formula only holds later.

Proposition 3.4 (Fundamental theorem)**.** If $\vdash t : \tau$, then for all $w \in \mathcal{W}$ we have $t \in \mathit{comp}([\![\tau]\!]\emptyset)(w)$.

Proof. To show this, one first generalizes to open types and open terms in the standard way, and then one shows semantic counterparts of all the typing rules of the language. See Appendix A.3. To illustrate the use of $\triangleright$, we outline the case of reference lookup: $\vdash\, !t : \tau$. Here the essential proof obligation is that $v \in [\![\mathsf{ref}\,\tau]\!]\emptyset(w)$ implies $!v \in \mathit{comp}([\![\tau]\!]\emptyset)(w)$. To show this, we unfold the definition of *comp*. Let $s \in \mathit{states}(w)$ be given; we must show

$$\mathit{eval}(!v, s,\, \lambda(v_1, s_1).\, \exists w_1 \geq w.\, v_1 \in [\![\tau]\!]\emptyset(w_1) \wedge s_1 \in \mathit{states}(w_1))\,. \tag{3.1}$$

By the assumption that $v \in [\![\mathsf{ref}\,\tau]\!]\emptyset(w)$, we know that $v = l$ for some location l such that $l \in \mathrm{dom}(w)$ and $\mathrm{app}(w(l))(w_1) = \triangleright([\![\tau]\!]\emptyset)(w_1)$ for all $w_1 \geq w$. Since $s \in \mathit{states}(w)$, we know that $l \in \mathrm{dom}(s) = \mathrm{dom}(w)$ and $s(l) \in \mathrm{app}(w(l))(w)$. We therefore have $\mathrm{step}((!v, s), (s(l), s))$. Hence, by unfolding the definition of eval in (3.1) and using the rules from Proposition 2.7, it remains to show that $\exists w_1 \geq w.\, \triangleright(s(l) \in [\![\tau]\!]\emptyset(w_1)) \;\wedge\; \triangleright(s \in \mathit{states}(w_1))$. We choose $w_1 = w$. First, $s \in \mathit{states}(w)$ and hence $\triangleright(s \in \mathit{states}(w))$. Second, $s(l) \in \mathrm{app}(w(l))(w) = \triangleright([\![\tau]\!]\emptyset)(w)$, which means exactly that $\triangleright(s(l) \in [\![\tau]\!]\emptyset(w))$. □

3.5. The view from the outside.

We now return to the standard universe of sets and give external interpretations of the internal results above. One basic ingredient is the fact that the constant-presheaf functor $\Delta : \mathbf{Set} \to \mathcal{S}$ commutes with formation of W-types. This fact can be shown by inspection of the concrete construction of W-types for presheaf categories given in [27].

Let OType$'$ and OTerm$'$ be the sets of possibly open types and terms, respectively, as defined by the grammars above. Similarly, let Value$'$, Store$'$, Config$'$, and step$'$ be the external counterparts of the definitions from the previous sections.

Proposition 3.5. OType $\cong \Delta(\mathrm{OType}')$, and similarly for OTerm, Value, Store, and Config. Moreover, under these isomorphisms step corresponds to $\Delta\mathrm{step}'$ as a subobject of Config $\times$ Config.

This result essentially says that the external interpretation of the step relation is world-independent, and has the expected meaning: for all n we have that $n \models \mathrm{step}((t', s'), (t', s'))$ holds iff (t, s) actually steps to (t', s') in the standard operational semantics. We next consider the eval predicate:

Proposition 3.6. $n \models \mathrm{eval}(t, s, Q)$ iff the following property holds: for all $m < n$, if (t, s) reduces to (v, s') in m steps, then $(n - m) \models Q(v, s')$.

Using this property and the forcing semantics from Section 2.4, one obtains that the external meaning of the interpretation of types is a step-indexed model in the standard sense. In particular, note that an element of $\mathcal{P}(\mathrm{Value})(n)$ can be viewed as a set of pairs (m, v) of natural numbers $m \leq n$ and values which is downwards closed in the first component.

3.6. **Discussion.** For simplicity, we have just considered a unary model in this extended example; we believe the approach scales well both to relational models and to more sophisticated models for reasoning about local state [2, 7, 12]. In particular, we have experimented with an internal-logic formulation of parts of [7], which involve recursively defined relations on recursively defined types.

As mentioned above, the operational semantics of this example was for simplicity chosen to be deterministic. We expect that one can easily adapt the approach presented here to non-deterministic languages. For that, the evaluation predicate must be changed to quantify universally (rather than existentially) over computation steps, and errors must explicitly be ruled out, as in:

$$\begin{aligned}&\mathrm{eval}'(t, s, Q)\\ &\overset{\mathrm{def}}{\Leftrightarrow} (t \in \mathrm{Value} \to Q(t, s)) \ \wedge\ \neg\mathrm{error}(t, s) \wedge\\ &\quad (\forall t_1 : \mathrm{Term}, s_1 : \mathrm{Store}.\\ &\qquad \mathrm{step}((t, s), (t_1, s_1)) \to \rhd\, \mathrm{eval}'(t_1, s_1, Q)).\end{aligned}$$

As mentioned in the Introduction, in [5] the recursive equation for $\mathcal{T}$ was solved in the category *CBUlt* of ultrametric spaces. Using the space $\mathcal{T}$ the model was then defined in the usual universe of sets in the standard, explicit step-indexed style. Here instead we observe that the relevant part of *CBUlt* is a full subcategory of $\mathcal{S}$ (Section 5), solve the recursive equation in $\mathcal{S}$, and then *stay* within $\mathcal{S}$ to give a simpler model that does not refer to step indices. In particular, the proof of the fundamental theorem is much simpler when done in $\mathcal{S}$.

4. Dependent Types

Since $\mathcal{S}$ is a topos it models not only higher-order logic over simple type theory, but also over dependent type theory. The aim of this section is to provide the semantic foundation for extending the dependent type theory with type constructors corresponding to $\blacktriangleright$ and guarded recursive types, although we postpone a detailed syntactic formulation of such a type theory to a later paper.

Recall that dependent types in context are interpreted in slice categories,[5] in particular a type $\Gamma \vdash A$ is interpreted as an object of $\mathcal{S}/[\![\Gamma]\!]$. To extend the interpretation of dependent type theory with a type constructor corresponding to $\blacktriangleright$, we must therefore extend the definition of $\blacktriangleright$ to slice categories.

4.1. **Slice categories concretely.** Before defining $\blacktriangleright_I : \mathcal{S}/I \to \mathcal{S}/I$ we give a concrete description of the slice categories $\mathcal{S}/I$.

We first recall the construction of the category of elements for presheaves over partial orders. For B a partial order, we write $\widehat{B}$ for the category of presheaves over B, i.e., category of functors and natural transformations from B^{op} to **Set**.

Definition 4.1. Let B be a partially ordered set and let X be a presheaf over B. Define the partially ordered set of *elements* of X as $\int X = \{(b, x) \mid b \in B \wedge x \in X(b)\}$ with order defined as $(b, x) \leq (c, y)$ iff $b \leq c$ and $y|_b = x$.

[5]For now we follow the practise of ignoring coherence issues related to the interpretation of substitution in codomain fibrations since there are various ways to avoid these issues, e.g. [19]. See the end of the section for more on this issue.

Note that if one applies this construction to an object X of $\mathcal{S}$ one gets a forest $\int X$: the roots are the elements of $X(1)$ the children of the roots are the elements of $X(2)$ and so on. Indeed any forest is of the form $\int X$ for some X in $\mathcal{S}$.

Proposition 4.2. Let B be a partially ordered set and let I be a presheaf over B. Then $\widehat{B}/I \simeq \widehat{\int I}$.

Proof. This is a standard theorem of sheaf theory [26, Ex. III.8], and we just recall one direction of the equivalence. An object $p_X \colon X \to I$ of the slice category $\widehat{B}/I$ corresponds to the presheaf that maps $(b, i) \in \int I$ to ${(p_X)_b}^{-1}(i)$. □

Thus we conclude that the slices of $\mathcal{S}$ are of the form presheaves over a forest.

4.2. **Generalising ▶ to slices.** There is a simple generalisation of the ▶ functor from $\mathcal{S}$ to presheaves over any forest $\int I$: if X is a presheaf over $\int I$ then

$$\blacktriangleright_I X(n,i) = \begin{cases} 1 & \text{if } n = 1 \\ X(n-1, i|_{n-1}) & \text{if } n > 1 \end{cases}$$

In Section 8 we shall see how to generalise this even further.

The map $\mathrm{next}_X : X \to \blacktriangleright_I X$ is represented by the following natural transformation in $\widehat{\int I}$:

$$\begin{aligned} \mathrm{next}_{(1,i)}(x) &= * \\ \mathrm{next}_{(n+1,i)}(x) &= x|_{(n,i|_n)} \end{aligned}$$

The fixed point combinator also generalizes to slices. Indeed, if $f : X \to X$ in $\widehat{\int I}$ is contractive, in the sense that there exists a $g : \blacktriangleright_I X \to X$ such that $f = g \circ \mathrm{next}$, then we can construct a fixed point of f (i.e., a natural transformation $1 \to X$) by:

$$\begin{aligned} x_{(1,i)} &= g_{(1,i)}(*) \\ x_{(n+1,i)} &= g_{(n+1,i)}(x_{(n,i|_n)}). \end{aligned}$$

This construction generalises to a fixed point combinator $\mathrm{fix}_X \colon (\blacktriangleright_I X \to X) \to X$ satisfying the properties of the global fixed point operator described in Theorem 2.4.

Proposition 4.3. Let $p_Y \colon Y \to I$ be an object of $\mathcal{S}/I$. There is a map $\blacktriangleright_I Y \to \blacktriangleright Y$ making the diagram below a pullback.

$$\begin{tikzcd} \blacktriangleright_I Y \arrow[r] \arrow[d, "p_{\blacktriangleright_I Y}"'] \arrow[dr, phantom, "\lrcorner", very near start] & \blacktriangleright Y \arrow[d, "\blacktriangleright p_Y"] \\ I \arrow[r, "\mathrm{next}"] & \blacktriangleright I \end{tikzcd}$$

One could have also taken the pullback diagram of Proposition 4.3 as a definition of $\blacktriangleright_I$, and indeed we do so in our axiomatic treatment of models of guarded recursion in Section 6.

The definition above allows us to consider ▶ as a type constructor on dependent types, interpreting $[\![\Gamma \vdash \blacktriangleright A]\!] = \blacktriangleright_{[\![\Gamma]\!]}([\![\Gamma \vdash A]\!])$. The following proposition expresses that this interpretation of ▶ behaves well wrt. substitution.

Proposition 4.4. For every $u \colon J \to I$ in $\mathcal{S}$ there is a natural isomorphism $u^* \circ \blacktriangleright_I \cong \blacktriangleright_J \circ u^*$. As a consequence, the collection of functors $(\blacktriangleright_I)_{I \in \mathcal{S}}$ define a fibred endofunctor on the codomain fibration. Moreover, next defines a fibred natural transformation from the fibred identity on the codomain fibration to ▶.

We remark that each $\blacktriangleright_I$ has a left adjoint, but in Section 6.1 we prove that this family of left adjoints does not commute with reindexing. As a consequence, it does not define a well-behaved dependent type constructor.

4.3. **Recursive dependent types.** Since the slices of $\mathcal{S}$ are cartesian closed, the notions of strong functors and locally contractive functors from Definition 2.12 also make sense in slices. Thus we can formulate a version of Theorem 2.14 generalised to all slices of $\mathcal{S}$. The next theorem does that, and further generalises to parametrized domain equations, a step necessary for modelling nested recursive types.

For the statement of the theorem recall the symmetrization $\tilde{F}\colon (\mathbb{C}^{\mathrm{op}} \times \mathbb{C})^n \to \mathbb{C}^{\mathrm{op}} \times \mathbb{C}$ of a functor $F\colon (\mathbb{C}^{\mathrm{op}} \times \mathbb{C})^n \to \mathbb{C}$ defined as $\tilde{F}(\vec{X},\vec{Y}) = \langle F(\vec{Y},\vec{X}), F(\vec{X},\vec{Y})\rangle$.

Theorem 4.5. Let $F\colon ((\mathcal{S}/I)^{\mathrm{op}} \times \mathcal{S}/I)^{n+1} \to \mathcal{S}/I$ be strong and locally contractive in the $(n+1)th$ variable pair. Then there exists a unique (up to isomorphism)

$$\operatorname{Fix} F\colon ((\mathcal{S}/I)^{\mathrm{op}} \times \mathcal{S}/I)^{n} \to \mathcal{S}/I$$

such that $F \circ \langle \mathrm{id}, \widetilde{\operatorname{Fix} F}\rangle \cong \operatorname{Fix} F$. Moreover, if F is locally contractive in all variables, so is $\operatorname{Fix} F$.

We postpone the proof of this theorem to Section 8, where we prove the existence of solutions to recursive domain equations for a wider class of categories and functors.

One can prove that the fixed points obtained by Theorem 4.5 are initial dialgebras in the sense of Freyd [15–17]. This universal property generalises initial algebras and final coalgebras to mixed-variance functors, and can be used to prove mixed induction / coinduction principles [30].

The formation of recursive types is well-behaved wrt. substitution:

Proposition 4.6. If

$$\begin{tikzcd}
((\mathcal{S}/I)^{\mathrm{op}} \times \mathcal{S}/I)^{n+1} \arrow[r, "F"] \arrow[d, "u^*"'] & \mathcal{S}/I \arrow[d, "u^*"] \\
((\mathcal{S}/J)^{\mathrm{op}} \times \mathcal{S}/J)^{n+1} \arrow[r, "G"] & \mathcal{S}/J
\end{tikzcd}$$

commutes up to isomorphism, so does

$$\begin{tikzcd}
((\mathcal{S}/I)^{\mathrm{op}} \times \mathcal{S}/I)^{n} \arrow[r, "\operatorname{Fix} F"] \arrow[d, "u^*"'] & \mathcal{S}/I \arrow[d, "u^*"] \\
((\mathcal{S}/J)^{\mathrm{op}} \times \mathcal{S}/J)^{n} \arrow[r, "\operatorname{Fix} G"] & \mathcal{S}/J
\end{tikzcd}$$

For the moment, our proof of Proposition 4.6 is conditional on the existence of unique fixed points, i.e., we prove that if $\operatorname{Fix} F$ and $\operatorname{Fix} G$ exist, then they make the required diagram commute up to isomorphism.

Proof. Note that $\operatorname{Fix} G \circ u^*$ is the unique H up to isomorphism such that

$$G(u^*(\vec{X},\vec{Y}), \widetilde{H}(\vec{X},\vec{Y})) \cong H(\vec{X},\vec{Y}).$$

Now,

$$\begin{aligned} G(u^*(\vec{X},\vec{Y}),((\widetilde{u^*\circ \mathrm{Fix}}\,F)(\vec{X},\vec{Y}))) &\cong G(u^*(\vec{X},\vec{Y}),u^*(\widetilde{\mathrm{Fix}\,F}(\vec{X},\vec{Y}))) \\ &\cong u^*(F(\vec{X},\vec{Y},\widetilde{\mathrm{Fix}\,F}(\vec{X},\vec{Y}))) \\ &\cong u^*\widetilde{\mathrm{Fix}\,F}(\vec{X},\vec{Y})) \end{aligned}$$

and so we conclude $u^*\mathrm{Fix}\,F(\vec{X},\vec{Y}) \cong \mathrm{Fix}\,G(u^*(\vec{X},\vec{Y}))$ □

4.4. **A higher order dependent type theory with guarded recursion.** In this section we sketch a type theory for guarded recursive types in combination with dependent types and explain how it can be interpreted soundly in $\mathcal{S}$. Since the type theory is an extension of standard higher-order dependent type theory, which can be interpreted in any topos, we focus on the extension to guarded recursion, and refer to [22] for details on dependent higher-order type theory and its interpretation in a topos. This section is meant to illustrate how the semantic results above can be understood type theoretically; we leave a full investigation of the syntactic aspects of the type theory to future work.

Recursive types are naturally formulated using type variables, and thus we allow types to contain type variables. Hence our type judgements live in contexts Γ that can be formed using the rules below

$$\frac{}{() : \mathrm{Ctx}} \qquad \frac{\Gamma \vdash \tau : \mathrm{Type}}{(\Gamma, x\colon \tau) : \mathrm{Ctx}} \qquad \frac{\Gamma : \mathrm{Ctx}}{(\Gamma, X\colon \mathrm{Type}) : \mathrm{Ctx}}$$

Type variables can be introduced as types using the rule

$$\frac{\Gamma : \mathrm{Ctx}}{\Gamma \vdash X : \mathrm{Type}}\; X\colon \mathrm{Type} \in \Gamma$$

The exchange rule of dependent type theory should be extended to allow a type variable X to be exchanged with a term variable $x\colon \sigma$ if X does not appear in σ.

Dependent products and sums and subset types are added to the type theory in the usual way [22], but we also add a special type constructor called $\blacktriangleright$ which acts as a functor. The rules are

$$\frac{\Gamma \vdash \tau : \mathrm{Type}}{\Gamma \vdash \blacktriangleright\tau : \mathrm{Type}} \qquad \frac{\Gamma \vdash M : \sigma \to \tau}{\Gamma \vdash \blacktriangleright(M) : \blacktriangleright\sigma \to \blacktriangleright\tau}$$

and the external equality rules include equations expressing the functoriality of $\blacktriangleright$. Moreover, we add, for each pair of types σ,τ in the same context, a term of type $\blacktriangleright\sigma \times \blacktriangleright\tau \to \blacktriangleright(\sigma\times\tau)$ plus equations stating that this is inverse to $\langle \blacktriangleright(\pi_1), \blacktriangleright(\pi_2)\rangle\colon\ \blacktriangleright(\sigma\times\tau) \to \blacktriangleright\sigma\times\blacktriangleright\tau$.

The natural transformation next is introduced as follows:

$$\frac{\Gamma \vdash \tau : \mathrm{Type}}{\Gamma \vdash \mathrm{next}_\tau : \tau \to \blacktriangleright\tau}$$

plus equality rules stating that next_τ is natural in τ (i.e., $\mathrm{next}_\sigma \circ u = \blacktriangleright(u)\circ \mathrm{next}_\tau$). We omit term formation rules for fixed point terms.

We now introduce the notion of functorial contractiveness which will be used as a condition ensuring well-formedness of recursive types. The definition is a syntactic reformulation of the semantic notion of local contractiveness.

A type τ is functorial in $\vec{X}$ if there is some way to split up the occurences of the variables $\vec{X}$ in τ into positive and negative ones, in such a way that τ becomes a functor expressible in the type theory. Above, and in the exact definition below we use vectors $\vec{X}$ to denote vectors of type variables and use $\vec{x} \colon \vec{\sigma}$ to denote vectors of typing assumptions of the form $x_1 \colon \sigma_1 \dots x_n \colon \sigma_n$. An assumption of the form $\vec{f} \colon \vec{X} \to \vec{Y}$ means $f_1 \colon X_1 \to Y_1, \dots f_n \colon X_n \to Y_n$.

Definition 4.7. Let $\Gamma, \vec{X} \colon \mathrm{Type} \vdash \tau : \mathrm{Type}$ be a valid typing judgement. We say that τ is *functorial* in $\vec{X}$ if there exists some other type judgement $\Gamma, \vec{X} \colon \mathrm{Type}, \vec{Y} \colon \mathrm{Type} \vdash \tau' : \mathrm{Type}$ and a term

$$\Gamma, \vec{X}_-, \vec{Y}_-, \vec{X}_+, \vec{Y}_+, \vec{f} \colon \vec{X}_+ \to \vec{X}_-, \vec{g} \colon \vec{Y}_- \to \vec{Y}_+ \vdash \mathrm{st}(\vec{f}, \vec{g}) : \tau'(\vec{X}_-, \vec{Y}_-) \to \tau'(\vec{X}_+, \vec{Y}_+)$$

(writing $\tau'(\vec{X}_-, \vec{Y}_-)$ for $\tau'[\vec{X}_-, \vec{Y}_- / \vec{X}, \vec{Y}]$) such that $\tau'(\vec{X}, \vec{X}) = \tau$, and such that st is functorial in the sense that $\mathrm{st}(\vec{\mathrm{id}}, \vec{\mathrm{id}}) = \mathrm{id}$, $\mathrm{st}(\vec{f} \circ \vec{f}', \vec{g}' \circ \vec{g}) = \mathrm{st}(\vec{f}', \vec{g}') \circ \mathrm{st}(\vec{f}, \vec{g})$.

The definition of τ being contractively functorial in $\vec{X}$ is similar, except that the strength $\mathrm{st}(\vec{f}, \vec{g})$ must be defined for $\vec{f} \colon \blacktriangleright(\vec{X}_+ \to \vec{X}_-), \vec{g} \colon \blacktriangleright(\vec{Y}_- \to \vec{Y}_+)$. To make sense of functoriality write $f' \circ f$ for the composite

$$\blacktriangleright(X \to Y) \times \blacktriangleright(Y \to Z) \xrightarrow{\cong} \blacktriangleright((X \to Y) \times (Y \to Z)) \xrightarrow{\blacktriangleright(comp)} \blacktriangleright(X \to Z)$$

applied to f' and f.

Definition 4.8. Let $\Gamma, \vec{X} \colon \mathrm{Type} \vdash \tau : \mathrm{Type}$ be a valid typing judgement. We say that τ is *contractively functorial* in $\vec{X}$ if there exists some other type judgement $\Gamma, \vec{X} \colon \mathrm{Type}, \vec{Y} \colon \mathrm{Type} \vdash \tau' : \mathrm{Type}$ and a term

$$\Gamma, \vec{X}_-, \vec{Y}_-, \vec{X}_+, \vec{Y}_+, \vec{f} \colon \blacktriangleright(\vec{X}_+ \to \vec{X}_-), \vec{g} \colon \blacktriangleright(\vec{Y}_- \to \vec{Y}_+) \vdash \mathrm{st}(\vec{f}, \vec{g}) : \tau'(\vec{X}_-, \vec{Y}_-) \to \tau'(\vec{X}_+, \vec{Y}_+)$$

such that $\tau'(\vec{X}, \vec{X}) = \tau$, and such that st is functorial in the sense that $\mathrm{st}(\vec{\mathrm{id}}, \vec{\mathrm{id}}) = \mathrm{id}$, $\mathrm{st}(\vec{f} \circ \vec{f}', \vec{g}' \circ \vec{g}) = \mathrm{st}(\vec{f}', \vec{g}') \circ \mathrm{st}(\vec{f}, \vec{g})$.

Lemma 4.9. If τ is contractively functorial in $\vec{X}$ then it is also functorial in $\vec{X}$.

We now give the introduction rule for recursive types

$$\frac{\Gamma, X \colon \mathrm{Type} \vdash \tau : \mathrm{Type}}{\Gamma \vdash \mu X . \tau : \mathrm{Type}} \ \tau \text{ contractively functorial in } X$$

As usual, there are associated term constructors $\mathsf{fold}\, M$ and $\mathsf{unfold}\, M$ that mediate between the recursive type and its unfolding together with equations expressing that fold and unfold are each others inverses.

There is a rich supply of types contractively functorial in $\vec{X}$ as can be seen from the following proposition. Proposition 4.10 is stated compactly, and some of the items in fact cover two statements. For example, item (4) states that if σ is functorial, so are $\prod_{i \colon I} \sigma$ and $\sum_{i \colon I} \sigma$ and if σ is contractively functorial so are $\prod_{i \colon I} \sigma$ and $\sum_{i \colon I} \sigma$.

Proposition 4.10. Let $\vec{X}$ be type variables and let σ, τ be types

(1) any type variable X is functorial in $\vec{X}$
(2) if $\vec{X}$ do not appear in σ then σ is contractively functorial in $\vec{X}$
(3) if σ and τ are both (contractively) functorial in $\vec{X}$ so are $\sigma \to \tau$ and $\sigma \times \tau$

(4) if σ is (contractively) functorial in $\vec{X}$ and $\vec{X}$ do not appear in I then $\prod_{i\colon I}\sigma$ and $\sum_{i\colon I}\sigma$ are both (contractively) functorial in $\vec{X}$
(5) If σ is (contractively) functorial in $\vec{X}$ (witnessed by σ' and st_σ) and ϕ is a predicate on σ' such that
$$\phi_{\vec{X}_-,\vec{Y}_-}(x) \to \phi_{\vec{X}_+,\vec{Y}_+}(\mathrm{st}(\vec{f},\vec{g})(x))$$
then $\{x\colon \sigma \mid \phi[\vec{X}/\vec{Y}](x)\}$ is (contractively) functorial in $\vec{X}$.
(6) If σ is functorial in $\vec{X}$, then $\blacktriangleright\sigma$ is contractively functorial in $\vec{X}$.

Item (5) uses the notation $\phi_{\vec{X}_-,\vec{Y}_-}$ for $\phi[\vec{X}_-,\vec{Y}_-/\vec{X},\vec{Y}]$.

Proof. The proof is a standard construction of functors from type expressions, and we just show a few examples. For (3) if σ' and τ' along with st_σ and st_τ witness that σ and τ are functorial, then $\sigma'(\vec{Y},\vec{X}) \to \tau'$ along with $\mathrm{st}_{\sigma\to\tau}(\vec{f},\vec{g})$ defined as
$$\lambda h\colon \sigma'(\vec{Y}_-,\vec{X}_-) \to \tau'(\vec{X}_-,\vec{Y}_-).\mathrm{st}_\tau(\vec{f},\vec{g}) \circ h \circ \mathrm{st}_\sigma(\vec{g},\vec{f})$$
witness that $\sigma \to \tau$ is functorial.

For (4) the assumption gives us a type σ' plus a term
$$\Gamma, i\colon I, \vec{X}_-,\vec{Y}_-,\vec{X}_+,\vec{Y}_+,\vec{f}\colon \vec{X}_+ \to \vec{X}_-, \vec{g}\colon \vec{Y}_- \to \vec{Y}_+ \vdash \mathrm{st}_\sigma(\vec{f},\vec{g}) : \sigma'(\vec{X}_-,\vec{Y}_-) \to \sigma'(\vec{X}_+,\vec{Y}_+)$$
and we can define $\mathrm{st}_{\prod_{i\colon I}\sigma}(\vec{f},\vec{g})$ as
$$\lambda x\colon \textstyle\prod_{i\colon I}\sigma'(\vec{X}_-,\vec{Y}_-).\lambda i\colon I.\mathrm{st}_\sigma(\vec{f},\vec{g})(x(i))$$
(This uses the exchange rule mentioned earlier.)

For item 5 the assumption is exactly the condition needed to show that $\mathrm{st}_\sigma(\vec{f},\vec{g})$ restricts to a term of the type
$$\{x\colon \sigma'(\vec{X}_-,\vec{Y}_-) \mid \phi_{\vec{X}_-,\vec{Y}_-}(x)\} \to \{x\colon \sigma'(\vec{X}_+,\vec{Y}_+) \mid \phi_{\vec{X}_+,\vec{Y}_+}(x)\}$$
□

To allow for *nested* recursive types, one needs to prove that if σ is functorial in $\vec{X}$ and contractively functorial in Y, then $\mu Y.\sigma$ is functorial in $\vec{X}$. In the type theory sketched above this is not provable because in general $\mathrm{st}_{\mu Y.\sigma}(\vec{f},\vec{g})$ is not definable, but as we shall see when we sketch the interpretation of the type theory, it is safe to add $\mathrm{st}_{\mu Y.\sigma}$ as a constant, together with appropriate equations, such that nested recursive types can in fact be defined.

Remark 4.11. The rules for well-definedness of recursive types are complicated because of the subset types, which require explicit mention of the syntactic strength st. Alternatively, one could give a simple grammar for well-defined recursive types not including subset types, but including nested recursive types not mentioning st, and then show how to interpret these by inductively constructing the contractive strength in the model. We chose the above approach because it is more expressive and because the subset types are needed in applications as illustrated in Section 3.3.

4.5. **Interpreting the type theory.** The interpretation of an open type $\Gamma \vdash \sigma : \text{Type}$ is defined modulo an environment mapping the type variables in Γ to semantic types, i.e., objects in slice categories. Precisely, if Γ is of the form $\Gamma', X\colon \text{Type}, \Gamma''$ then ρ should map X to an object of $\mathcal{S}/[\![\Gamma']\!]_{\rho'}$ where ρ' is the restriction of ρ to the type variables of Γ'. The interpretation of open types is defined by induction and most of the cases are exactly as in the usual interpretation of dependent type theory [22], and we just mention the new cases. The interpretation of a type variable introduction is defined as $[\![\Gamma', X\colon \text{Type}, \Gamma'' \vdash X : \text{Type}]\!] = p^*_{\Gamma,\Gamma'}(\rho(X))$, where $p_{\Gamma,\Gamma'}$ denotes the projection $[\![\Gamma]\!]_\rho \to [\![\Gamma']\!]_\rho$. The interpretation of $\blacktriangleright$ is defined as $[\![\Gamma \vdash \blacktriangleright \sigma : \text{Type}]\!] = \blacktriangleright_{[\![\Gamma]\!]_\rho}([\![\Gamma \vdash \blacktriangleright \sigma : \text{Type}]\!])$.

For the interpretation of recursive types, note that for every type $\Gamma, \vec{X} \vdash \sigma : \text{Type}$ functorial in $\vec{X}$ and every environment ρ mapping the free type variables in Γ to semantic types, one can define a strong functor of the type

$$[\![\sigma]\!]_\rho \colon (\mathcal{S}/[\![\Gamma]\!]_\rho{}^{\text{op}} \times \mathcal{S}/[\![\Gamma]\!]_\rho)^{|\vec{X}|} \to \mathcal{S}/[\![\Gamma]\!]_\rho$$

as follows. Assuming that the functoriality of σ is witnessed by σ' and st as in Definition 4.7, the action of $[\![\sigma]\!]_\rho$ on objects is defined by the interpretation of σ'. Given objects $\vec{A}_-, \vec{A}_+, \vec{B}_-, \vec{B}_+$ of $\mathcal{S}/[\![\Gamma]\!]_\rho$ the interpretation of st is a morphism in $\mathcal{S}/[\![\Gamma]\!]_\rho$ of the type

$$A_{-,1}^{A_{+,1}} \times \cdots \times A_{-,n}^{A_{+,n}} \times B_{+,1}^{B_{-,1}} \times \cdots \times B_{+,n}^{B_{-,n}} \to [\![\sigma]\!]_\rho(\vec{A}_+, \vec{B}_+)^{[\![\sigma]\!]_\rho(\vec{A}_-, \vec{B}_-)}$$

where the products and exponentials are those of the slice $\mathcal{S}/[\![\Gamma]\!]_\rho$. The interpretation of st defines the strength of $[\![\sigma]\!]_\rho$, from which the action of $[\![\sigma]\!]_\rho$ on morphisms can be derived in the usual way.

Similarly, if σ is functorial in the n first type variables and contractively functorial in the last one then the interpretation of the witness st defines a strong functor which is locally contractive in the last variable and so we can define $[\![\mu X.\tau]\!]_\rho = \text{Fix}\,([\![\tau]\!]_\rho)$ using the fixed point given by Theorem 4.5.

There is a question of well-definedness here, since the fixed point of $[\![\sigma]\!]_\rho$ a priori could depend on the choice of σ' and st. The uniqueness of the fixed point of Theorem 4.5, however, ensures that even for different such choices, the resulting $[\![\sigma]\!]_\rho$ will be isomorphic. Usually, σ comes with a canonical choice of σ' and st as given by Proposition 4.10.

As mentioned earlier, for allowing nested recursive types in the type theory we need to add constants of the form $\text{st}_{\mu Y.\sigma}(\vec{f}, \vec{g})$. Having sketched the interpretation of the type theory we can now see that it is safe to do so: $\text{st}_{\mu Y.\sigma}(\vec{f}, \vec{g})$ can be interpreted using the strength of $\text{Fix}\,[\![\sigma]\!]_\rho$ which exists by Theorem 4.5.

4.6. **On Coherence.** Above, we have worked in the codomain fibration and ignored coherence issues, i.e., the fact that the codomain fibration and the associated fibred functors needed for the interpretation of the type theory are not split. One further advantage of the concrete representation of slices $\mathcal{S}/I$ as presheaves over $\int I$ is that the latter gives rise to a split model. The idea is to form a split indexed category $P : \mathcal{S} \to \mathbf{Cat}^{\text{op}}$, with fibre over I given by $P(I) = \widehat{\int I}$, and reindexing $P(u : I \to J)$ given by $P(u)(X)(n, i) = X(u_n(i))$. By forming the Grothendieck construction [22] on P one obtains a split fibration $\text{Fam}(\mathcal{S}) \to \mathcal{S}$ which is equivalent to the codomain fibration. Then one uses this fibration to interpret the types and terms without free type variables, and uses split fibred functors

$$(\text{Fam}(\mathcal{S})_{[\![\Gamma]\!]}{}^{\text{op}} \times \text{Fam}(\mathcal{S})_{[\![\Gamma]\!]})^{|\Theta|} \to \text{Fam}(\mathcal{S})_{[\![\Gamma]\!]}$$

to interpret open types $\Gamma \vdash \tau : \text{Type}$. Finally, one checks that the fibred constructs (e.g., right adjoints to reindexing) used to interpret the dependent type theory are split, and that $\blacktriangleright$ and the construction of recursive types is also split. The latter essentially boils down to observing that the actual construction of initial algebras in Section 8 is done fibrewise and thus preserved on-the-nose by reindexing. We omit further details.

5. Relation to metric spaces

Let *CBUlt* be the category of complete bounded ultrametric spaces and non-expansive maps. In [5–8, 33] only those spaces that were also bisected were used: a metric space is *bisected* if all non-zero distances are of the form 2^{-n} for some natural number $n \geq 0$. Let *BiCBUlt* be the full subcategory of *CBUlt* of bisected spaces, and let *BiUlt* be the category of all bisected ultrametric spaces (necessarily bounded).

Let $t\mathcal{S}$ be the full subcategory of $\mathcal{S}$ on the total objects.

Proposition 5.1. There is an adjunction between *BiUlt* and $\mathcal{S}$, which restricts to an equivalence between $t\mathcal{S}$ and *BiCBUlt*, as in the diagram:

$$\begin{tikzcd}
t\mathcal{S} \arrow[r, shift right] \arrow[d, shift left] \arrow[r, phantom, "\top"] \arrow[d, phantom, "\cong"] & \mathcal{S} \arrow[l, shift right] \arrow[d, shift left] \arrow[d, phantom, "\dashv"] \\
\mathit{BiCBUlt} \arrow[u, shift left] \arrow[r, shift right] \arrow[r, phantom, "\perp"] & \mathit{BiUlt} \arrow[l, shift right] \arrow[u, shift left, "F"]
\end{tikzcd}$$

Proof sketch. The functor $F : \mathit{BiUlt} \to \mathcal{S}$ is defined as follows. A space $(X, d) \in \mathit{BiUlt}$ gives rise to an indexed family of equivalence relations by $x =_n x' \Leftrightarrow d(x, x') \leq 2^{-n}$, which can then be viewed as a presheaf: at index n, it is the quotient $X/(=_n)$, see, e.g. [10]. One can check that F in fact maps into $t\mathcal{S}$ and that F has a right adjoint that maps into *BiCBUlt*. The right adjoint maps a variable set into a metric space on the limit of the family of variable sets; the metric expresses up to what level elements in the limit agree. The left adjoint from *BiUlt* to *BiCBUlt* is given by the Cauchy-completion. $\square$

Proposition 5.2. A morphism in *BiCBUlt* is contractive in the metric sense iff it is contractive in the internal sense of $\mathcal{S}$.

The later operator on $\mathcal{S}$ corresponds to multiplying by $\frac{1}{2}$ in ultra-metric spaces, except on the empty space. Specifically, $F(\frac{1}{2}X)$ is isomorphic to $\blacktriangleright(FX)$, for all non-empty X. For ultra-metric spaces, the formulation of existence of solutions to guarded recursive domain equations has to consider the empty space as a special case. Here, in $\mathcal{S}$, we do not have to do so, since $\blacktriangleright$ behaves better than $\frac{1}{2}$ on the empty set.

6. General models of guarded recursive terms

Having presented the specific model $\mathcal{S}$ we now turn to general models of guarded recursion. We give an axiomatic definition of what models of guarded recursion are, and in Section 8 we show that $\mathcal{S}$ is just one in a large class of models.

We start by defining a notion of model of guarded recursive terms, and showing that the class of such models is closed under taking slices. This result is not only of interest in its own right, but also needed for showing that the general models of Section 8 model guarded recursive dependent types.

Definition 6.1. A model of guarded recursive terms is a category $\mathcal{E}$ with finite products together with an endofunctor $\blacktriangleright\colon \mathcal{E} \to \mathcal{E}$ and a natural transformation $\mathrm{next}\colon \mathrm{id} \to \blacktriangleright$ such that

- for every morphism $f\colon \blacktriangleright X \to X$ there exists a unique morphism $h\colon 1 \to X$ such that $f \circ \mathrm{next} \circ h = h$.
- $\blacktriangleright$ preserves finite limits

Lemma 6.2. If $\mathcal{E}$ models guarded recursive terms then $\blacktriangleright$ is strong.

Proof. Using next one can define a strength for $\blacktriangleright$ as the composite

$$\cong \circ\, \mathrm{next} \times \mathrm{id}\colon X \times \blacktriangleright Y \to \blacktriangleright X \times \blacktriangleright Y \to \blacktriangleright(X \times Y)\,.$$

□

The notion of contractive morphism as well as Lemma 2.3 and Theorem 2.4 generalises directly to the current setting.

Theorem 6.3. If $\mathcal{E}$ is a locally cartesian closed model of guarded recursive terms, then so is every slice of $\mathcal{E}$.

To prove Theorem 6.3 we must first show how to generalise $\blacktriangleright$ to slices. We do this by taking the pullback diagram of Proposition 4.3 as a definition of $\blacktriangleright_I X$. In other words we define $\blacktriangleright_I$ as the composite

$$\mathcal{E}/I \xrightarrow{\blacktriangleright} \mathcal{E}/\blacktriangleright I \xrightarrow{\mathrm{next}^*} \mathcal{E}/I \tag{6.1}$$

where the first functor maps $p_X\colon X \to I$ to $\blacktriangleright(p_X)\colon \blacktriangleright X \to \blacktriangleright I$ and the second is given by pullback along next. Recall that next^* has a left adjoint $\coprod_{\mathrm{next}}$ mapping $p_Y\colon Y \to I$ to $\mathrm{next} \circ p_Y$ and so preserves limits. It is easy to see that also the first functor of (6.1) preserves finite limits because $\blacktriangleright$ does, and thus we have the following:

Lemma 6.4. The functor $\blacktriangleright_I\colon \mathcal{E}/I \to \mathcal{E}/I$ preserves finite limits.

We define $\mathrm{next}_I : p_Y \to p_{\blacktriangleright_I Y}$ in the slice over I as indicated in the diagram below

$$\begin{tikzcd}
Y \arrow[dr] \arrow[drr, "\mathrm{next}"] \arrow[ddr, "p_Y"'] & & \\
& \blacktriangleright_I Y \arrow[r] \arrow[d, "p_{\blacktriangleright_I Y}"] \arrow[dr, phantom, "\lrcorner", very near start] & \blacktriangleright Y \arrow[d, "\blacktriangleright p_Y"] \\
& I \arrow[r, "\mathrm{next}"] & \blacktriangleright I
\end{tikzcd}$$

It is easy to show that next_I is a natural transformation.

The following proposition states that $\blacktriangleright$ defines a fibred functor and hence can serve as a type constructor in the dependent type theory of $\mathcal{E}$.

Proposition 6.5. For every $u\colon J \to I$ in $\mathcal{E}$ the following diagram commutes up to isomorphism

$$\begin{tikzcd}
\mathcal{E}/I \arrow[r, "\blacktriangleright_I"] \arrow[d, "u^*"'] & \mathcal{E}/I \arrow[d, "u^*"] \\
\mathcal{E}/J \arrow[r, "\blacktriangleright_J"] & \mathcal{E}/J\,.
\end{tikzcd}$$

As a consequence, the collection of functors $(\blacktriangleright_I)_{I\in\mathcal{E}}$ define a fibred endofunctor on the codomain fibration.

Proof. We can write the diagram as a composite as below.

$$\begin{tikzcd}
\mathcal{E}/I \arrow[r, "\blacktriangleright"] \arrow[d, "u^*"'] & \mathcal{E}/\blacktriangleright I \arrow[r, "\mathrm{next}^*"] \arrow[d, "(\blacktriangleright u)^*"] & \mathcal{E}/I \arrow[d, "u^*"] \\
\mathcal{E}/J \arrow[r, "\blacktriangleright"] & \mathcal{E}/\blacktriangleright J \arrow[r, "\mathrm{next}^*"] & \mathcal{E}/J\,.
\end{tikzcd}$$

The square on the left commutes because $\blacktriangleright$ preserves pullbacks, the one on the right follows from the naturality square for next. □

Proposition 6.6. The collection of next morphisms defines a fibred natural transformation from the fibred identity on the codomain fibration to $\blacktriangleright$:

$$\begin{tikzcd}
\mathcal{E}^{\to} \arrow[rr, bend left, "\mathrm{id}"] \arrow[rr, bend right, "\blacktriangleright"'] \arrow[rr, phantom, "\Downarrow\ \mathrm{next}"] \arrow[dr, "\mathrm{cod}"'] & & \mathcal{E}^{\to} \arrow[dl, "\mathrm{cod}"] \\
& \mathcal{E} &
\end{tikzcd}$$

Proof. A fibred natural transformation between fibred functors is a natural transformation with vertical components. The components of next are clearly vertical, but we must show that next defines a natural transformation between the two functors on the total category $\mathcal{E}^{\to}$. So consider a morphism in $\mathcal{E}^{\to}$ from $Y \to I$ to $X \to J$, and write it as a composition

$$\begin{tikzcd}
Y \arrow[r, "g"] \arrow[dr] & f^*X \arrow[r, "\bar{f}"] \arrow[d] \arrow[dr, phantom, "\lrcorner", very near start] & X \arrow[d] \\
& I \arrow[r, "f"] & J
\end{tikzcd}$$

of a vertical morphism g and a cartesian morphism $\bar{f}$. We must verify naturality diagrams for next with respect to $\bar{f}$ and g. Naturality wrt. g is just naturality of next as a functor $\mathcal{E}/I \to \mathcal{E}/I$, and naturality wrt. $\bar{f}$ can be verified by a diagram chase that we omit. □

It remains to prove the existence (and uniqueness) of fixed points in slices. We do that by reducing those to global fixed points. In the next lemma we use internal language notation, writing $\prod_{i\colon I} X_i$ for the functor

$$\mathcal{E}/I \xrightarrow{\prod_{!\colon I\to 1}} \mathcal{E}/1 \xrightarrow{\cong} \mathcal{E}$$

applied to an object $p_X\colon X \to I$, where $\prod_{!\colon I\to 1}$ is the right adjoint to $!^*$, and using similar notation for the result of applying the same functor to morphisms.

Lemma 6.7. Suppose that $f\colon p_X \to p_Y$ is a contractive morphism in slice $\mathcal{E}/I$. Then $\prod_{i\colon I} f_i \colon \prod_{i\colon I} X_i \to \prod_{i\colon I} Y_i$ is a contractive morphism in $\mathcal{E}$. As a consequence any contractive endomorphism in $\mathcal{E}/I$ has a unique fixed point.

Proof. The assumption gives us a g such that $f = g \circ \mathrm{next}$ and from that we can derive a factorisation of $\prod_{i\colon I} f_i$ as

$$\prod_{i\colon I} X_i \xrightarrow{\prod_{i\colon I}\mathrm{next}} \prod_{i\colon I} \blacktriangleright X_i \xrightarrow{\prod_{i\colon I} g_i} \prod_{i\colon I} Y_i$$

To show $\prod_{i\colon I} f_i$ contractive, it suffices to show commutativity of the triangle

$$\begin{tikzcd} \prod_{i\colon I} X_i \arrow[rr, "\prod_{i\colon I}\mathrm{next}"] \arrow[dr, "\mathrm{next}"'] & & \prod_{i\colon I} \blacktriangleright X_i \\ & \blacktriangleright \prod_{i\colon I} X_i \arrow[ur] & \end{tikzcd} \tag{6.2}$$

Writing π_i for the term $i\colon I \vdash \lambda x\colon \prod_{i\colon I} X_i . x_i : X_i$ the adjoint correspondent of (6.2) can be expressed in the internal language of $\mathcal{E}$ as

$$i\colon I, x\colon \prod_{i\colon I} X_i \vdash \blacktriangleright(\pi_i) \circ \mathrm{next}(x) = \mathrm{next} \circ \pi_i(x) : \blacktriangleright(X_i)$$

which is simply naturality of next. This sketch in the internal language can be turned into a formal diagrammatic argument.

Now, it is easy to see that if f is an endomorphism then there is a bijective correspondence between fixed points of $\prod_{i\colon I} f_i$ in the global sense, and fixed points of f in the slice. □

Proof of Theorem 6.3. We have seen how every slice of $\mathcal{E}$ has an endofunctor $\blacktriangleright_I$ and a natural transformation $\mathrm{next}\colon \mathrm{id} \to \blacktriangleright_I$, and we have seen that $\blacktriangleright_I$ preserves finite limits (Lemma 6.4). Lemma 6.7 gives existence of the needed fixed points. □

6.1. **A left adjoint to $\blacktriangleright$.**

In our model $\mathcal{S}$, the functor $\blacktriangleright$ has a left adjoint $\blacktriangleleft$ mapping the presheaf

$$X(1) \leftarrow X(2) \leftarrow X(3) \leftarrow \dots$$

to the presheaf

$$X(2) \leftarrow X(3) \leftarrow X(4) \leftarrow \dots .$$

Moreover, $\blacktriangleleft$ preserves limits and so $\blacktriangleleft \dashv \blacktriangleright$ defines a geometric morphism from $\mathcal{S}$ to itself, in fact it is an embedding. Hence $\blacktriangleright_I$, as defined in (6.1), has a left adjoint $\blacktriangleleft_I$ because next^* has a left adjoint $\sum_{\mathrm{next}}$ and also $\blacktriangleright\colon \mathcal{E}/I \to \mathcal{E}/\blacktriangleright I$ has a left adjoint defined by mapping $p_X\colon X \to \blacktriangleright I$ to its adjoint correspondent $\blacktriangleleft X \to I$.

Even though $\blacktriangleleft$ preserves limits, $\blacktriangleleft_I$ does not. The simplest counter example is that of the terminal object id_I of $\mathcal{E}/I$ which is mapped to the adjoint correpondent $\mathrm{prev}\colon \blacktriangleleft I \to I$ of $\mathrm{next}\colon I \to \blacktriangleright I$. So, in particular, $\blacktriangleleft_I \dashv \blacktriangleright_I$ does not define a geometric morphism.

We choose not to take $\blacktriangleleft$ as part of the basic structure of a model of guarded recursion because $\blacktriangleleft$ in $\mathcal{S}$ does not define a fibred functor, and so it cannot be used in an internal language based on dependent type theory. To see why, observe that if $f\colon J \to I$ then $\blacktriangleleft_J f^*(\mathrm{id}_I) \cong \blacktriangleleft_J(\mathrm{id}_J) = \mathrm{prev}_J$ and $f^* \blacktriangleleft_I(\mathrm{id}_I) = f^*\mathrm{prev}_I$, and these two are in general not isomorphic.

Observe also that $\blacktriangleright$ does not preserve dependent products, i.e., the diagram

$$\begin{tikzcd}
\mathcal{E}/J \arrow[r, "\blacktriangleright_J"] \arrow[d, "\prod_u"'] & \mathcal{E}/J \arrow[d, "\prod_u"] \\
\mathcal{E}/I \arrow[r, "\blacktriangleright_I"] & \mathcal{E}/I\,.
\end{tikzcd}$$

does not in general commute. The reason is that the diagram obtained by taking left adjoints to all functors above is the diagram stating that $\blacktriangleleft$ is a fibred functor, which we have just established does not commute.

6.2. **An operation on predicates.** We now assume that $\mathcal{E}$ is a topos modelling guarded recursion and we shall see how to obtain the principle of Löb induction in $\mathcal{E}$.

As we have seen, $\blacktriangleright_X$ preserves limits, hence monos, and thus defines a map $\rhd\colon \mathrm{Sub}(X) \to \mathrm{Sub}(X)$ for all X, which is easily seen to be order preserving. The term next_X verifies that $m \leq \rhd m$. As a consequence of Proposition 6.5 this family is natural in X and thus, by the usual Yoneda argument, it corresponds to an operation on propositions $\rhd\colon \Omega \to \Omega$. We now embark on proving the following theorem.

Theorem 6.8 (Löb induction)**.** The reasoning principle $\forall p\colon \mathrm{Prop}.(\rhd p \to p) \to p$ is valid in $\mathcal{E}$.

To prove the theorem, we need a few lemmas. The first describes the action of $\rhd\colon \mathrm{Sub}(X) \to \mathrm{Sub}(X)$ as an action on characteristic maps.

Lemma 6.9. Let $m\colon M \to X$ be a mono and let $\chi_m\colon X \to \Omega$ be its characteristic map. Then $\mathrm{succ} \circ \blacktriangleright \chi_m \circ \mathrm{next}$ is the characteristic map of $\rhd(m)$, where $\mathrm{succ}\colon \blacktriangleright\Omega \to \Omega$ is the characteristic map of the mono $\blacktriangleright\top\colon \blacktriangleright 1 \to \blacktriangleright\Omega$.

Proof. Consider the diagram

$$\begin{tikzcd}
\rhd m \arrow[r] \arrow[d] \arrow[dr, phantom, "\lrcorner"] & \blacktriangleright M \arrow[r] \arrow[d, "\blacktriangleright m"'] \arrow[dr, phantom, "\lrcorner"] & \blacktriangleright 1 \arrow[r] \arrow[d, "\blacktriangleright \top"'] \arrow[dr, phantom, "\lrcorner"] & 1 \arrow[d, "\top"'] \\
X \arrow[r, "\mathrm{next}"] & \blacktriangleright X \arrow[r, "\blacktriangleright \chi_m"] & \blacktriangleright \Omega \arrow[r, "\mathrm{succ}"] & \Omega\,.
\end{tikzcd}$$

All the squares are pullbacks, and so also the outer square is a pullback, which proves the lemma. ☐

Subobjects of X correspond to morphisms $X \to \Omega$ which in turn correspond to global elements of Ω^X. As a consequence of Lemma 6.9, the operation $\rhd$ on subobjects corresponds to composing the global elements with the morphism $\Omega^X \to \Omega^X$ mapping χ_m to $\mathrm{succ}\circ\blacktriangleright\chi_m\circ \mathrm{next}$. Since this morphism is contractive, it has a unique fixed point.

Corollary 6.10. Let m be a subobject of X. If $\rhd(m) \leq m$ then m is the maximal subobject.

Proof of Theorem 6.8. The principle is proved using Joyal-Kripke semantics, see [25, Thm 8.4]. Using items (7) and (6) of the referenced theorem, it suffices to show that for any X and any $f\colon X \to \Omega$, if the map $\lambda x\colon X.\ \rhd f(x) \to f(x)$ factors through $\top\colon 1 \to \Omega$, then so does f. Expressing this using subobjects rather than representable maps, we must show that, for any subobject m of X, if $\rhd m \to m$ is the maximal subobject, then so is m. But $\rhd m \to m$ is maximal iff $\rhd m \leq m$, and so the principle follows from Corollary 6.10. ☐

7. General models of guarded recursive types

In this section we formulate the most general existence theorem for recursive types in models of guarded recursion. Moreover, we reduce the problem of solving general recursive domain equations to that of solving covariant domain equations using the uniqueness of fixed points in combination with Freyd's theory of algebraic compactness [15–17].

Note first that Definition 2.12 of locally contractive functor on our concrete model $\mathcal{S}$, carries over verbatim to general cartesian closed models $\mathcal{E}$ of guarded recursive terms.

Definition 7.1. A *model of guarded recursive types* is a cartesian closed model of guarded recursive terms (in the sense of Definition 6.1) $\mathcal{E}$ such that every locally contractive functor $F\colon \mathcal{E} \to \mathcal{E}$ has a fixed point (up to isomorphism). A model of guarded recursive *dependent* types is a locally cartesian closed category whose slices all are models of guarded recursive types.

As a justification of the above definition we shall prove that fixed points for locally contractive covariant functors give fixed points of general (locally contractive) mixed variance functors. In fact, we state and prove this not only for functors on $\mathcal{E}$, but, more generally, for functors on $\mathcal{E}$-enriched categories. This is in line with classical work on recursive types in O-categories [34] (categories enriched in complete partial orders) and more recent work on recursive types in M-categories [9] (categories enriched in complete bounded ultrametric spaces).

Recall that an $\mathcal{E}$-enriched category $\mathbb{C}$ is a collection of objects together with for each pair of objects X, Y of $\mathbb{C}$ an $\mathcal{E}$-object $\mathrm{Hom}_{\mathbb{C}}(X, Y)$ together with composition morphisms $\mathrm{Hom}_{\mathbb{C}}(X, Y) \times \mathrm{Hom}_{\mathbb{C}}(Y, Z) \to \mathrm{Hom}_{\mathbb{C}}(X, Z)$ and morphisms $\ulcorner \mathrm{id}_X \urcorner\colon 1 \to \mathrm{Hom}_{\mathbb{C}}(X, X)$ satisfying commutative diagrams corresponding to the rules for morphism composition in category theory [23]. To each enriched category $\mathbb{C}$ we can associate a category in the usual sense with the same objects as $\mathbb{C}$ and set of morphisms from X to Y all $\mathcal{E}$-morphisms from 1 to $\mathrm{Hom}_{\mathbb{C}}(X, Y)$. This category is called the *externalisation* of $\mathbb{C}$. Given a category $\mathbb{C}$ in the usual sense, we say that it is $\mathcal{E}$-enriched if there exists an $\mathcal{E}$-enriched category whose externalisation is $\mathbb{C}$. Any cartesian closed category $\mathbb{C}$ is self-enriched: one can take $\mathrm{Hom}_{\mathbb{C}}(X, Y)$ to be the exponent Y^X.

The notion of locally contractive functor readily generalises to $\mathcal{E}$-enriched categories: if $\mathbb{C}$ is $\mathcal{E}$-enriched consider the $\mathcal{E}$-enriched category ${}_{\blacktriangleright}\mathbb{C}$ with the same objects as $\mathbb{C}$, hom-objects $\mathrm{Hom}_{{}_{\blacktriangleright}\mathbb{C}}(X, Y) = \blacktriangleright \mathrm{Hom}_{\mathbb{C}}(X, Y)$, composition given as the composite

$$\blacktriangleright \mathrm{Hom}_{\mathbb{C}}(X, Y) \times \blacktriangleright \mathrm{Hom}_{\mathbb{C}}(Y, Z) \cong \blacktriangleright(\mathrm{Hom}_{\mathbb{C}}(X, Y) \times \mathrm{Hom}_{\mathbb{C}}(Y, Z)) \xrightarrow{\blacktriangleright(comp)} \blacktriangleright \mathrm{Hom}_{\mathbb{C}}(X, Z)$$

and identity as $\mathrm{next} \circ \ulcorner \mathrm{id} \urcorner\colon 1 \to \blacktriangleright \mathrm{Hom}_{\mathbb{C}}(X, X)$. Note that ${}_{\blacktriangleright}(\mathbb{C} \times \mathbb{D}) \cong {}_{\blacktriangleright}\mathbb{C} \times {}_{\blacktriangleright}\mathbb{D}$ and ${}_{\blacktriangleright}(\mathbb{C}^{\mathrm{op}}) \cong ({}_{\blacktriangleright}\mathbb{C})^{\mathrm{op}}$. The natural transformation next defines an enriched functor [23] $\mathbb{C} \to {}_{\blacktriangleright}\mathbb{C}$ whose action on objects is the identity and whose action on morphisms is given by $\mathrm{next}\colon \mathrm{Hom}_{\mathbb{C}}(X, Y) \to \blacktriangleright \mathrm{Hom}_{\mathbb{C}}(X, Y)$.

Definition 7.2. An enriched functor $F\colon \mathbb{D} \to \mathbb{C}$ is *locally contractive* if it factors as a composition of enriched functors

$$\begin{tikzcd} \mathbb{D} \arrow[r, "\mathrm{next}"] & {}_{\blacktriangleright}\mathbb{D} \arrow[r] & \mathbb{C} \end{tikzcd}$$

Specialising Definition 7.2 to the case of $\mathcal{S}$ as self-enriched gives Definition 2.12.

Lemma 7.3.

(1) If $F\colon \mathbb{B} \to \mathbb{C}$ and $G\colon \mathbb{C} \to \mathbb{D}$ are enriched functors and either F or G is locally contractive also GF is locally contractive.
(2) If $F\colon \mathbb{C} \to \mathbb{D}$ and $G\colon \mathbb{C}' \to \mathbb{D}'$ are locally contractive, so is $F \times G\colon \mathbb{C} \times \mathbb{C}' \to \mathbb{D} \times \mathbb{D}'$.
(3) Let $H\colon \mathbb{B} \times \mathbb{C} \to \mathbb{D}$ be enriched and suppose the enriched functor category $\mathbb{D}^{\mathbb{C}}$ exists. Then H is locally contractive in the first variable iff $\hat{H}\colon \mathbb{B} \to \mathbb{D}^{\mathbb{C}}$ is locally contractive.

Definition 7.4. An $\mathcal{E}$-enriched category $\mathbb{C}$ is *contractively complete* if any locally contractive functor $F\colon \mathbb{C} \to \mathbb{C}$ has a fixed point, i.e., an object X such that $FX \cong X$.

The isomorphism $FX \cong X$ is an isomorphism in the externalisation of $\mathbb{C}$. Similarly, the notation $f\colon X \to Y$ always refers to morphisms in the external version of $\mathbb{C}$.

We can now state the main theorem. It uses the symmetrization of $\tilde{G}$ of a mixed variance functor G defined in Section 4.3. The proof follows after a brief series of lemmas.

Theorem 7.5. Let $\mathcal{E}$ be a model of guarded recursive terms, $\mathbb{C}$ be $\mathcal{E}$-enriched and contractively complete, and let $F\colon (\mathbb{C}^{\mathrm{op}} \times \mathbb{C})^{n+1} \to \mathbb{C}$ be locally contractive in the $(n+1)th$ variable pair. Then there exists a unique (up to isomorphism) $\mathrm{Fix}\, F\colon (\mathbb{C}^{\mathrm{op}} \times \mathbb{C})^{n} \to \mathbb{C}$ such that $F \circ \langle \mathrm{id}, \widetilde{\mathrm{Fix}\, F}\rangle \cong \mathrm{Fix}\, F$. Moreover, if F is locally contractive in all variables, so is $\mathrm{Fix}\, F$. In particular, the above statement holds for $\mathbb{C}= \mathcal{E}$ if $\mathcal{E}$ is a model of guarded recursive types.

Lemma 7.6. Let $\mathbb{C}$ be $\mathcal{E}$-enriched and let $F\colon \mathbb{C} \to \mathbb{C}$ be a locally contractive functor. If $X \cong F(X)$, then the two directions of the isomorphism give an initial algebra structure and a final coalgebra structure for F on X. In particular, if $F(X) \cong X$ and $F(Y) \cong Y$, then $X \cong Y$.

Proof. Given an isomorphism $f\colon FX \to X$ and some other algebra $g\colon FZ \to Z$, $h\colon X \to Z$ is an algebra homomorphism iff the diagram

$$\begin{tikzcd}
FX \arrow[r, "Fh"] & FZ \arrow[d, "g"] \\
X \arrow[u, "f^{-1}"] \arrow[r, "h"] & Z
\end{tikzcd}$$

commutes, i.e., iff h is a fixed point of the map $h \mapsto g \circ F(h) \circ f^{-1}$, which is a contractive endomorphism on $\mathrm{Hom}_{\mathbb{C}}(X, Z)$ (as F is locally contractive). Since this map has exactly one fixed point, we conclude that there is exactly one algebra homomorphism from f to g. The argument for final coalgebras is similar. □

There is also a morphism in $\mathcal{E}$ computing the unique mediating homomorphism from the initial algebra.

Lemma 7.7. Let $\mathbb{C}$ and F be as in Lemma 7.6, and let $f\colon FX \to X$ be an isomorphism. For any Z there exists a morphism $k\colon \mathrm{Hom}_{\mathbb{C}}(FZ, Z) \to \mathrm{Hom}_{\mathbb{C}}(X, Z)$ such that $\forall g\colon \mathrm{Hom}_{\mathbb{C}}(FZ, Z).k(g) \circ f = g \circ F(k(g))$ holds in the internal language of $\mathcal{E}$.

Proof. Define k to be the fixed point of the map $\mathrm{Hom}_{\mathbb{C}}(FZ, Z)\times\mathrm{Hom}_{\mathbb{C}}(X, Z) \to \mathrm{Hom}_{\mathbb{C}}(X, Z)$ mapping g, h to $g \circ Fh \circ f^{-1}$. □

Lemma 7.8. Let $\mathbb{C}, \mathbb{D}$ be $\mathcal{E}$-enriched categories and let $F\colon \mathbb{D} \times \mathbb{C} \to \mathbb{C}$ be enriched and locally contractive in the second variable. If the functor $F(X, -)\colon \mathbb{C} \to \mathbb{C}$ has an initial algebra for all X in $\mathbb{D}$, then there is an $\mathcal{E}$-enriched functor $\mu F\colon \mathbb{D} \to \mathbb{C}$ mapping X to the carrier of the initial algebra. If, moreover, F is locally contractive in the first variable, then μF is locally contractive.

Proof. The functor μF is defined (as is standard) to map $f\colon X \to Y$ to the unique $\mu F(f)$ making the diagram

$$\begin{tikzcd}
F(X, \mu F(X)) \arrow[rr] \arrow[d, "{F(X, \mu F(f))}"'] & & \mu F(X) \arrow[d, "\mu F(f)"] \\
F(X, \mu F(Y)) \arrow[r, "{F(f, \mathrm{id})}"] & F(Y, \mu F(Y)) \arrow[r] & \mu F(Y)
\end{tikzcd} \tag{7.1}$$

commute. Now, the enrichment of μF is obtained by composing the morphism $\mathrm{Hom}_{\mathbb{D}}(X, Y) \to \mathrm{Hom}_{\mathbb{C}}(F(X, \mu F(Y)), \mu F(Y))$ mapping f to the composite in the bottom line of (7.1) with the morphism of Lemma 7.7. In the case of F being locally contractive in both variables, the first stage of this composite morphism is contractive and so μF becomes locally contractive. □

Recall that an *initial dialgebra* for $G\colon \mathbb{C}^{\mathrm{op}} \times \mathbb{C} \to \mathbb{C}$ is an initial algebra of $\tilde{G}$ [15–17].

Lemma 7.9. Let $\mathbb{C}$ be $\mathcal{E}$-enriched and $G\colon \mathbb{C}^{\mathrm{op}} \times \mathbb{C} \to \mathbb{C}$ be a locally contractive functor. If $G(X, Y) \cong Y$ and $G(Y, X) \cong X$ then the pair (X, Y) together with the isomorphisms constitute an initial dialgebra for G. In particular (X, Y) is unique up to isomorphism with this property. Moreover $X \cong Y$.

Proof. If G is locally contractive, so is $\tilde{G}$. Thence Lemma 7.6 proves that (X, Y) is an initial dialgebra. To show $X \cong Y$ note that the hypothesis of the lemma is symmetric in X and Y, so we may apply what we have just proved to conclude that (Y, X) is an initial dialgebra. By uniqueness of initial dialgebras $(X, Y) \cong (Y, X)$. □

We can now give the promised proofs of the main theorem and proposition in this section.

Proof of Theorem 7.5. Consider first the case of $n = 0$. Recall the functor $\mu F\colon \mathbb{C}^{\mathrm{op}} \to \mathbb{C}$ from Lemma 7.8 mapping X to the unique fixed point of $F(X, -)$. Define Z to be the unique fixed point of the functor $X \mapsto F(\mu F(X), X)$ and define $W = \mu F(Z)$. Then $F(W, Z) = F(\mu F(Z), Z) \cong Z$ and $F(Z, W) = F(Z, \mu F(Z)) \cong \mu F(Z) = W$, and so Lemma 7.9 applies giving the unique solution to F and proving that $W \cong Z$.

In the general case of $n \neq 0$, Lemma 7.8 applies to give the functor $\mathrm{Fix}\, F$. □

The statement and proof of Proposition 4.6 carries over verbatim from the case of $\mathcal{S}$ to the general case of $\mathcal{E}$ a model of guarded recursive dependent types.

8. A class of models of guarded recursion

The aim of this section is to establish a large class of models of guarded recursive dependent types including our main example, the topos $\mathcal{S}$. This involves showing existence of fixed points for locally contractive functors. The special case of $\mathcal{S}$, together with the results of Section 7, prove Theorem 4.5.

The class of models we consider are sheaves over a complete Heyting algebra with a well-founded basis. In this section we assume some familiarity with the basics of complete Heyting algebras and sheaves over such [26].

Definition 8.1. A partial order A is *well-founded* if there are no infinite descending sequences $a_0 > a_1 > a_2 > \dots$

Here $a > a'$ means $a \geq a'$ and $a \neq a'$ as usual. Note that any forest is well-founded.

Definition 8.2. Let A be a partial order and let $K \subseteq A$. Then K is a *basis* for A if each $a \in A$ is a least upper bound of all the base elements below it, i.e. $a = \bigvee\{k \in K \mid k \leq a\}$.

Example 8.3. If K is a well-founded partial order then the ideal completion $\mathit{Idl}(K)$ consisting of down-closed subsets of K is a complete Heyting algebra and the set $\{\downarrow k \mid k \in K\}$, where $\downarrow k = \{k' \in K \mid k' \leq k\}$ is a well-founded basis.

In the following we reserve a's and b's for elements of A and k's for elements in K. A sieve B on a in A is just a downward closed subset of $\{b \in A \mid b \leq a\}$ and it is covering if $\bigvee B = a$. If A is a complete Heyting algebra then this defines a Grothendieck topology, and the corresponding category $\mathit{Sh}(A)$ of sheaves is the full subcategory of presheaves X such that $(X(\bigvee B) \to X(b))_{b \in B}$ is a limiting cone for all $B \subseteq A$. We recall the following well-known fact.

Proposition 8.4. If A is a partial order then $\mathit{Sh}(\mathit{Idl}(A)) \simeq \widehat{A}$.

Proof. The equivalence maps X in $\widehat{A}$ to $\lambda B. \lim_{b \in B} X(b)$ (we shall write $\bar{X}$ for this sheaf) and Y in $\mathit{Sh}(\mathit{Idl}(A))$ to $\lambda a. Y(\downarrow a)$. □

Collectively Proposition 8.4 and Example 8.3 state that the general class of models we consider include all toposes of the form $\widehat{A}$ for A a well-founded partial order, in particular all slices of $\mathcal{S}$.

Theorem 8.5. Let A be a complete Heyting algebra with a well-founded base. Then $\mathit{Sh}(A)$ is a model of guarded recursive dependent types. In particular $\mathcal{S}$ and indeed any topos of the form $\widehat{A}$ for A a well-founded partial order is a model of guarded recursive dependent types.

Di Gianantonio and Miculan [10] essentially prove that $\mathit{Sh}(A)$ is a model of guarded recursive *terms* if A is the set of opens of a topological space with a well-founded basis; here we extend their results to guarded recursive *types* and, moreover, consider more general models (not necessarily arising from topological spaces).

Theorem 8.6. Let A be a complete Heyting algebra with a well-founded basis and let $\mathbb{C}$ be a $\mathit{Sh}(A)$-enriched category. If $\mathbb{C}$ is complete (precisely, the externalisation of $\mathbb{C}$ is complete in the usual sense) then it is contractively complete.

Note that the notion of completeness assumed for $\mathbb{C}$ above is the usual one (rather than the enriched notion of completeness).

In the remainder of this section we prove Theorems 8.6 and 8.5. We start by showing that $\mathit{Sh}(A)$ models guarded recursive terms.

8.1. **Modelling recursive terms.** Following [10] we give the following definition.

Definition 8.7. Define the *predecessor* map $p \colon A \to A$ by

$$p(a) = \bigvee \{k \in K \mid k < a\}\,.$$

The predecessor map induces an endofuntor on the category of presheaves on A; following standard notation, we write $p^* \colon \widehat{A} \to \widehat{A}$ for this functor, defined by $p^*(X) = X \circ p$. We define $\blacktriangleright \colon \mathit{Sh}(A) \to \mathit{Sh}(A)$ by $\blacktriangleright X = \mathbf{a}(p^* X)$, where $\mathbf{a}$ is the associated sheaf functor. Define $\mathrm{next}^{pre} \colon X \to p^* X$ by $\mathrm{next}^{pre}_a(x \in X(a)) = x|_{p(a)}$ and define $\mathrm{next} = \mathbf{a}(\mathrm{next}^{pre}) \colon X \to \blacktriangleright X$ for all sheaves X.

Note that

$$\mathrm{next} = \eta \circ \mathrm{next}^{pre} \tag{8.1}$$

where η is the unit of the adjunction $\mathbf{a} \dashv I$, with $I \colon \mathit{Sh}(A) \to \widehat{A}$ the inclusion of sheaves into presheaves. This can be seen by applying $\mathbf{a}$ to both sides of the equation since $\mathbf{a}$ fixes maps between sheaves and because $\mathbf{a}(\eta)$ is the identity.

Remark 8.8. The use of the associated sheaf functor $\mathbf{a}$ in the definition of $\blacktriangleright$ is necessary, because $p^* X$ needs not be a sheaf. Consider, for example, the situation where A is the powerset of a 2-element set $\{a, b\}$. Then a sheaf is a presheaf X such that $X(\emptyset) = 1$ and $X(\{a, b\}) = X(\{a\}) \times X(\{b\})$. The map p is

$$\begin{aligned} p(\emptyset) &= \emptyset & \qquad p(\{a\}) &= \emptyset \\ p(\{b\}) &= \emptyset & \qquad p(\{a, b\}) &= \{a, b\} \end{aligned}$$

So $p^* X(\{a, b\}) = X(\{a, b\})$, but $p^* X(\{a\}) = p^* X(\{b\}) = 1$, in particular $p^* X$ is in general not a sheaf. On the other hand $\blacktriangleright X = 1$.

Lemma 8.9. The functor $\blacktriangleright$ preserves finite limits.

We will now show that the above definition of $\blacktriangleright$ generalises the definition of $\blacktriangleright$ from Section 4.2 on slices of $\mathcal{S}$, see Proposition 8.11 below. For that we first need a lemma:

Lemma 8.10. Let A be a partial order. The composite

$$\widehat{\mathit{Idl}(A)} \xrightarrow{\mathbf{a}} \mathit{Sh}(\mathit{Idl}(A)) \xrightarrow{\simeq} \widehat{A}$$

maps P to $\lambda b. P(\downarrow b)$. In other words $\mathbf{a}P(\downarrow b) = P(\downarrow b)$.

Proof. Since $\mathbf{a}$ is left adjoint to the inclusion, the composite sought for is left adjoint to the functor $P \mapsto \bar{P}$, and it is easy to check that the functor of the lemma satisfies this condition. □

Proposition 8.11. Let I be an object of $\mathcal{S}$. The composite

$$\widehat{\textstyle\int I} \simeq \mathit{Sh}(\mathit{Idl}(\textstyle\int I)) \xrightarrow{\blacktriangleright} \mathit{Sh}(\mathit{Idl}(\textstyle\int I)) \simeq \widehat{\textstyle\int I}$$

which we shall also call $\blacktriangleright$ agrees with $\blacktriangleright_I$ as defined in Section 4.2

Proof. We compute

$$\begin{aligned} \blacktriangleright P(n, i) &= \blacktriangleright \bar{P}(\downarrow (n, i)) \\ &= (\mathbf{a}\, p^* \bar{P})(\downarrow (n, i)) \\ &= (p^* \bar{P})(\downarrow (n, i)) \\ &= \bar{P}(p(\downarrow (n, i))) \end{aligned}$$

Now, it is easy to see that if $n = 1$ then $p(\downarrow (n,i)) = \emptyset$ so that $\blacktriangleright P(1,i) = \bar{P}(\emptyset) = 1$ and and otherwise

$$\begin{aligned} \bar{P}(p(\downarrow (n,i))) &= \bar{P}(\downarrow (n-1, i|_{n-1})) \\ &= P(n-1, i|_{n-1}) \end{aligned}$$

which implies the result. □

Using the well-founded basis we can reason by well-founded induction over A as the following easy lemma shows.

Lemma 8.12. Let $\phi(a)$ be a predicate on A. If

$$\forall a \in A.(\forall k\colon K.k < a \to \phi(k)) \to \phi(a)$$

then $\phi(a)$ holds for all a in A.

Proof. First use well-founded induction to conclude that $\phi(k)$ holds for all $k \in K$, then use the condition again to conclude that $\phi(a)$ holds for all a. □

We now aim to show that any morphism $f\colon \blacktriangleright X \to X$ has a unique fixed point. Since the associated sheaf functor is left adjoint to the inclusion of sheaves into presheaves such morphisms correspond bijectively to morphisms of presheaves $\hat{f}\colon p^*X \to X$ (where $\hat{f} = f \circ \eta$), and we shall start by constructing fixed points of morphisms of the latter form.

Lemma 8.13. Let X be a sheaf and let $f\colon p^*X \to X$ and $a \in A$. Then there exists a unique family $(x_b)_{b\leq a}$ such that

(1) $x_a|_b = x_b$ for all $b \leq a$
(2) $f_b(x_{pb}) = x_b$ for all $b \leq a$

Proof. The proof is by well-founded induction on a using Lemma 8.12. Thus suppose the lemma holds for all $k < a$, i.e., for any $k < a$ there exists a unique family $(x_{k,b})_{b\leq k}$ satisfying the requirements. Note that by uniqueness, if $b \leq k' \leq k$ then $x_{k,b} = x_{k',b}$, so for any $b < a$ we can define x_b to be the unique amalgamation of the family $(x_{k,k})_{k\leq b}$. This gives us a compatible family $(x_b)_{b<a}$, i.e., $x_{b'} = x_b|_{b'}$ if $b' < b$. To see that this family also satisfies (2), for all $b < a$, note that it suffices to show that $f_b(x_{pb})|_k = x_k$, for all $k \leq b$. But

$$\begin{aligned} f_b(x_{pb})|_k &= f_k(x_{pk}) \\ &= x_k \end{aligned}$$

since the family $(x_{k,b})_{b\leq k}$ satisfied (2).

It only remains to extend this family with a component x_a. By the sheaf condition there is a unique y in $X(p(a))$ such that $y|_b = x_b$. Define $x_a = f_a(y)$. We must check that the extended family $(x_b)_{b\leq a}$ satisfies the conditions, and all that remains to prove is the case of $b = a$.

For (1) we must show that $x_a|_b = x_b$ for all $b < a$.

$$\begin{aligned} x_a|_b &= f_a(y)|_b \\ &= f_b(y|_{pb}) \\ &= f_b(x_{pb}) \\ &= x_b \end{aligned}$$

For (2) we branch on whether $a = pa$ or not (using classical reasoning). If $pa < a$ then $y = x_{pa}$, and we are done. If $a = pa$ then, by the sheaf condition, it suffices to prove that $f_a(x_a)|_b = x_b$ for all $b < a$. But

$$\begin{aligned} f_a(x_a)|_b &= f_b((f_a(y))|_{p(b)}) \\ &= f_b(f_{p(b)}(y|_{pp(b)})) \\ &= f_b(f_{p(b)}(x_{pp(b)})) \\ &= f_b(x_{pb}) \\ &= x_b \end{aligned}$$

For the proof of uniqueness, we must show that x_a as defined above gives the unique extension of $(x_b)_{b<a}$ satisfying the conditions. Again we branch on $pa = a$ or $pa < a$. In the first case, (1) together with the sheaf condition gives uniqueness and in the second it is (2) that gives uniqueness. □

Theorem 8.14. If A is a complete Heyting algebra with a well-founded basis then every slice of $\mathit{Sh}(A)$ is a model of guarded recursive terms.

Proof. By Theorem 6.3 it suffices to show that $\mathit{Sh}(A)$ is a model of guarded recursive terms, and for this it remains to show that if $f\colon \blacktriangleright X \to X$, then there exists a unique $\mathrm{fix}(f)\colon 1 \to X$ such that $f \circ \mathrm{next} \circ \mathrm{fix}(f) = \mathrm{fix}(f)$

Consider first $f \circ \eta\colon p^*X \to X$. The family $(x_b)_{b \leq \bigvee A}$ given by Lemma 8.13 defines a map $\mathrm{fix}(f)\colon 1 \to X$: the naturality condition needed to have a map in $\widehat{A}$ is (1) and (2) states

$$f \circ \eta \circ \mathrm{next}^{pre} \circ \mathrm{fix}(f) = \mathrm{fix}(f) \tag{8.2}$$

which by (8.1) is equivalent to $f \circ \mathrm{next} \circ \mathrm{fix}(f) = \mathrm{fix}(f)$. In fact we see that to give a map $\mathrm{fix}(f)\colon 1 \to X$ satisfying the (8.2) is the same as giving a family $(x_b)_{b \leq \bigvee A}$ and so the uniqueness statement of Lemma 8.13 shows that $\mathrm{fix}(f)\colon 1 \to X$ is the unique such map. □

8.2. Recursive types in sheaf models.

Having proved that $\mathit{Sh}(A)$ models guarded recursive terms, we now show that it models guarded recursive dependent types. We first prove Theorem 8.6 and then show how Theorem 8.5 follows from it. So in the following, let $\mathbb{C}$ be a complete $\mathit{Sh}(A)$-enriched category.

In the technical development it is simpler to work with presheaves and p^* than it is to work with sheaves and $\blacktriangleright$, so we first reformulate the definition of local contractiveness in terms of p^*. Note that we can define ${}_{p^*}\mathbb{C}$ in the same way as we defined ${}_{\blacktriangleright}\mathbb{C}$, using p^* rather than $\blacktriangleright$. This gives us an $\widehat{A}$-enriched category rather than a $\mathit{Sh}(A)$-enriched one. Any $\mathit{Sh}(A)$-enriched category is also $\widehat{A}$-enriched and so in particular, $\mathbb{C}$ and ${}_{\blacktriangleright}\mathbb{C}$ are $\widehat{A}$-enriched.

There is a commutative diagram of $\widehat{A}$-enriched functors

$$\begin{tikzcd} \mathbb{C} \arrow[r, "\mathrm{next}^{pre}"] \arrow[dr, "\mathrm{next}"] & {}_{p^*}\mathbb{C} \arrow[d, "\eta"] \\ & {}_{\blacktriangleright}\mathbb{C} \end{tikzcd}$$

and the following lemma tells us that we can proceed to work with p^* and presheaves rather than $\blacktriangleright$ and sheaves.

Lemma 8.15. An enriched functor $F\colon \mathbb{C} \to \mathbb{C}$ is *locally contractive* iff there exist a $\widehat{A}$-enriched functor $H\colon {}_{p^*}\mathbb{C} \to \mathbb{C}$ such that $H \circ \mathrm{next}^{pre} = F$.

Proof. If F is locally contractive and G is a witness of this, we can construct H by precomposing G with η. On the other hand, given H as above we can construct G by applying $\mathbf{a}$ to each hom-action of H. $\square$

Now suppose $F\colon \mathbb{C} \to \mathbb{C}$ is locally contractive. We will construct a fixed point for F by a sufficiently large induction. To determine the height of the induction we start by assigning to each element a of A an ordinal by well-founded induction on a. We use ordinals (rather than just the elements of A) to get a linear diagram to take limits over when constructing the fixed point for F.

Definition 8.16. Define for each $a \in A$ the ordinal $\mathit{Ord}(a) = \sup\{\mathit{Ord}(k) + 1 \mid k < a \wedge k \in K\}$.

Lemma 8.17. Definition 8.16 defines an order preserving map $\mathit{Ord}(-)\colon A \to \mathit{Ord}(\bigvee A)$. If $k < a$ and $k \in K$ then $\mathit{Ord}(k) < \mathit{Ord}(a)$.

We shall use $p\colon \mathit{Ord}(\bigvee A) \to \mathit{Ord}(\bigvee A)$ defined as $p(\alpha) = \bigvee\{\beta \mid \beta < \alpha\}$.

In the following we distinguish notationally between ordinals and elements of A by using Greek letters for the former and latin letters for the latter.

Next we generalise the notion of n-isomorphism of Lemma 2.15. Recall that a morphism $f\colon X \to Y$ in $\mathbb{C}$ is the same as a morphism $1 \to \mathrm{Hom}_{\mathbb{C}}(X,Y)$ in $\mathit{Sh}(A)$, which is the same as a family $(f_a)_{a\in A}$ with $f_a \in \mathrm{Hom}_{\mathbb{C}}(X,Y)_a$ such that $f_a|_b = f_b$ for all a and $b \leq a$. We say that f_a is an isomorphism if there exists $g_a \in \mathrm{Hom}_{\mathbb{C}}(X,Y)_a$ such that $\mathit{comp}_a(f_a, g_a) = \mathrm{id}_a$ and $\mathit{comp}_a(g_a, f_a) = \mathrm{id}_a$. In the following we shall simply write $f_a \circ g_a$ for $\mathit{comp}_a(g_a, f_a)$.

Definition 8.18. Let $f\colon X \to Y$ be a morphism in $\mathbb{C}$, let $a \in A$ and let α be an ordinal. We say that f is an *a-isomorphism* if for all $b \leq a$ the component f_b is an isomorphism. We say that f is an *α-isomorphism* if it is a b-isomorphism, for all b such that $\mathit{Ord}(b) \leq \alpha$.

Lemma 8.19. Let $F\colon \mathbb{C} \to \mathbb{C}$ be locally contractive, and suppose $f\colon X \to Y$ is a b-isomorphism for all $b < a$. Then Ff is an a-isomorphism. As a consequence, Ff is an α-isomorphism if f is a β-isomorphism, for all $\beta < \alpha$, or, equivalently, if f is a $p(\alpha)$ isomorphism.

Proof. Formulating the assumption of local contractiveness using the equivalent condition of Lemma 8.15 we get maps $H_{X,Y}\colon p^*\mathrm{Hom}_{\mathbb{C}}(X,Y) \to \mathrm{Hom}_{\mathbb{C}}(FX,FY)$ such that

$$(Ff)_b = H_b(f_{p(b)})$$

The functoriality conditions on H are commutative diagrams in $\widehat{A}$. These amount to the following equations required to hold for each b in A

$$H_b(f_{p(b)} \circ g_{p(b)}) = H_b(f_{p(b)}) \circ H_b(g_{p(b)}) \tag{8.3}$$

$$H_b(\mathrm{id}_{p(b)}) = \mathrm{id}_b \tag{8.4}$$

Now, suppose $f\colon X \to Y$ is a b-isomorphism, for all $b < a$. Define $f^{-1}_{p(a)}$ to be the unique amalgamation of $(f_b^{-1})_{b<a}$. Then ${f_{p(a)}}^{-1}$ is an inverse to $f_{p(a)}$: to show $f^{-1}_{p(a)} \circ f_{p(a)} = \mathrm{id}_{p(a)}$ it suffices to show $(f^{-1}_{p(a)} \circ f_{p(a)})|_b = \mathrm{id}_b$ for all $b < a$, which is clear since composition commutes with restriction.

So f_b has an inverse f_b^{-1} for all $b \leq p(a)$, in particular $f_{p(b)}$ has an inverse, for all $b \leq a$. Equations (8.3) and (8.4) then say that $H_b(f_{p(b)}^{-1})$ is an inverse of $F(f)_b$, for all $b \leq a$.

For the last statement, suppose f is a β-isomorphism for all $\beta < \alpha$, and suppose $\mathit{Ord}(a) \leq \alpha$. We must show that Ff is an a-isomorphism. By what we have just proved, it suffices to show that f is a b-isomorphism, for all $b < a$, and for this, by the sheaf property, it suffices to show that f is a k-isomorphism, for all $k < a$, $k \in K$. But this is true because $\mathit{Ord}(k) < \mathit{Ord}(a) \leq \alpha$. □

Remark 8.20. The strengthening of the definition of locally contractive functor compared to the definition used in the conference version of this paper [4] was introduced in order to make Lemma 8.19 true, also with the weaker notion of a-isomorphism used here. Without the requirement of functoriality of H, equation (8.3) only holds for families $(f_b)_{b \leq p(a)}$, $(g_b)_{b \leq p(a)}$ in the image of next, i.e., families that extend to families $(f_b)_{b \leq a}$, $(g_b)_{b \leq a}$

We construct, by well-founded induction, for every $\alpha \leq \mathit{Ord}(\bigvee A)$ a $\mathbb{C}$-object X_α and maps

$$\phi_\alpha \colon F(X_\alpha) \to X_\alpha \qquad \text{and} \qquad \pi_{\alpha,\beta} \colon X_\alpha \to X_\beta, \quad \text{for } \beta < \alpha$$

by

$$X_\alpha = \lim_{\beta < \alpha} F(X_\beta)$$

and

$$\begin{tikzcd} \pi_{\alpha,\beta} : \lim\limits_{\beta' < \alpha} F(X_{\beta'}) \arrow[r, "\pi_\beta"] & F(X_\beta) \arrow[r, "\phi_\beta"] & X_\beta \\ \phi_\alpha : F(\lim\limits_{\beta < \alpha} F(X_\beta)) \arrow[r, "F(\lim_{\beta<\alpha} \phi_\beta)"] & F(\lim\limits_{\beta < \alpha}(X_\beta)) \arrow[r] & \lim\limits_{\beta < \alpha} F(X_\beta) \end{tikzcd}$$

Precisely, each α is an ordered set and so can be considered a category. We define X_α as the limit of a diagram indexed over α mapping an inequality $\beta' \leq \beta < \alpha$ to $F(\pi_{\beta,\beta'}) \colon F(X_\beta) \to F(X_{\beta'})$.

Theorem 8.21. Each $\pi_{\alpha,\beta}$ is a β-isomorphism and each ϕ_α is an α-isomorphism. In particular, $\phi_{\mathit{Ord}(\bigvee A)} \colon F(X_{\mathit{Ord}(\bigvee A)}) \to X_{\mathit{Ord}(\bigvee A)}$ is an isomorphism.

Before we give the proof we record the following simple lemma.

Lemma 8.22. Let $\alpha > \beta$ and let $(Y_{\beta'})_{\beta' < \alpha}$ be a diagram over α considered a category. The morphism $\lim_{\beta' < \alpha} Y_{\beta'} \to \lim_{\beta \leq \beta' < \alpha} Y_{\beta'}$ given by diagram inclusion is an isomorphism.

Lemma 8.23. $\alpha \leq \bigvee\{\gamma \mid p\gamma < \alpha\}$

Proof. Recall that for ordinals $\beta < \gamma$ is equivalent to $\beta \in \gamma$, and so $p(\gamma) = \bigcup_{\beta \in \gamma} \beta$. For the lemma we must show that if $x \in \alpha$ also $x \in \bigcup_{p\gamma \in \alpha} \gamma$, i.e., there exists a γ such that $x \in \gamma$ and $(\bigcup_{\beta \in \gamma} \beta) \in \alpha$. Take $\gamma = \{\beta \mid \beta \leq x\}$. □

Proof of Theorem 8.21. The theorem is proved by induction on α, but the induction hypothesis must be strengthened with the following two statements.

(1) For all $\beta < \alpha$, the projection

$$\pi_\beta \colon \lim_{\beta' < \alpha} X_{\beta'} \to X_\beta$$

is a β-isomorphism.

(2) For all $\beta < \alpha$ and all γ such that $p\gamma \leq \beta$, the projection
$$\pi_\beta \colon \lim_{\beta' < \alpha} F(X_{\beta'}) \to F(X_\beta)$$
is a γ-isomorphism. In particular, each π_β is a β-isomorphism.

We now give the induction steps of the inductive proof, proving each part of the induction hypothesis in turn.

For (1) note first that by Lemma 8.22 we may replace the limit $\lim_{\beta'<\alpha} X_{\beta'}$ by $\lim_{\beta\leq\beta'<\alpha} X_{\beta'}$. By the induction hypothesis, all morphisms of the form $\pi_{\beta',\beta''} \colon X_{\beta'} \to X_{\beta''}$ for $\beta \leq \beta'' < \beta' < \alpha$ are β-isomorphisms. Therefore the limit $\lim_{\beta\leq\beta'<\alpha} X_{\beta'}$ is a limit of a diagram of β-isomorphisms. Since limits are computed pointwise, the projections are β-isomorphisms.

For (2) we reason similarly and conclude by Lemma 8.19 that each $F(\pi_{\beta',\beta''})$ is a γ-isomorphism. So in this case the limit $\lim_{\beta\leq\beta'<\alpha} F(X_{\beta'})$ is a limit of a diagram of γ-isomorphisms and each projection π_β is a γ-isomorphisms.

Now consider the case of $\pi_{\alpha,\beta} = \phi_\beta \circ \pi_\beta$. By (2) above and the induction hypothesis, this is a β-isomorphism.

We will now show that ϕ_α is an α-isomorphism. Consider the following commutative diagram

$$\begin{tikzcd}
\lim\limits_{\beta'<\alpha} F(X_{\beta'}) \arrow[r, "\lim_{\beta'<\alpha}\phi_{\beta'}"] \arrow[d, "\pi_\beta"] & \lim\limits_{\beta'<\alpha} (X_{\beta'}) \arrow[d, "\pi_\beta"] \\
F(X_\beta) \arrow[r, "\phi_\beta"'] & X_\beta
\end{tikzcd}$$

Since (1) and (2) state that both projections π_β are β-isomorphisms and by induction hypothesis ϕ_β is a β-isomorphism, also $\lim_{\beta'<\alpha}\phi_{\beta'}$ must be a β-isomorphism. Since this holds for all $\beta < \alpha$, by Lemma 8.19 also $F(\lim_{\beta<\alpha}\phi_\beta)$ must be an α-isomorphism.

Now, consider the diagram

$$\begin{tikzcd}
F(\lim\limits_{\beta'<\alpha} X_{\beta'}) \arrow[r] \arrow[dr, "F(\pi_\beta)"'] & \lim\limits_{\beta'<\alpha} F(X_{\beta'}) \arrow[d, "\pi_\beta"] \\
 & F(X_\beta)
\end{tikzcd}$$

It only remains to show that the vertical map is an α-isomorphism. By induction hypothesis (2) the maps $F(\pi_\beta)$ and π_β are γ-isomorphisms for any γ such that $p\gamma \leq \beta$. Since this holds for all β, the vertical map is a $\bigvee\{\gamma \mid p\gamma < \alpha\}$-isomorphism, and we conclude by Lemma 8.23. □

Proof of Theorem 8.6. We must show that any locally contractive endofunctor $F \colon \mathbb{C} \to \mathbb{C}$ has a fixed point, but Theorem 8.21 gives such a fixed point. □

For Theorem 8.5 it remains to show that any slice of $\mathit{Sh}(A)$ is a model of guarded recursive types. We do that by reducing to Theorem 8.6, using the fact that slices of $\mathit{Sh}(A)$ are all $\mathit{Sh}(A)$-enriched. Indeed this holds for any locally cartesian closed category $\mathcal{E}$, because one can take as homobject from p_X to p_Y the object $\prod_{i\colon I} {Y_i}^{X_i}$ (using internal language notation as in Lemma 6.7). Since each slice $\mathcal{E}/I$ is also self-enriched, this gives us two possible notions of local contractiveness. The next lemma states a relation between the two.

Lemma 8.24. Let $\mathcal{E}$ be a locally cartesian closed model of guarded recursive terms, and let $F\colon \mathcal{E}/I \to \mathcal{E}/I$ be a functor. If F is locally contractive in the $\mathcal{E}/I$-enriched sense then it is also locally contractive in the $\mathcal{E}$-enriched sense.

Proof. The assumption gives an $\mathcal{E}/I$-enrichment of F as a composite

$$p_Y{}^{p_X} \xrightarrow{\text{next}} \blacktriangleright_I(p_Y{}^{p_X}) \xrightarrow{G_{p_X,p_Y}} p_{FY}{}^{p_{FX}}$$

Lemma 6.7 then tells us that each $\prod_{i\colon I} F_{X_i,Y_i}$ is contractive in the $\mathcal{E}$-enriched sense. To show that F is locally contractive in the $\mathcal{E}$-enriched sense one must check that the derived witness of contractiveness commutes with composition and identity, but this follows from naturality of the morphism $\blacktriangleright \prod_{i\colon I} X_i \to \prod_{i\colon I} \blacktriangleright X_i$ used in Lemma 6.7. □

Proof of Theorem 8.5. We have already shown (Theorem 8.14) that every slice of $\mathit{Sh}(A)$ is a model of guarded recursive terms. It remains to show that any functor $F\colon \mathit{Sh}(A)/I \to \mathit{Sh}(A)/I$, which is locally contractive in the $\mathit{Sh}(A)/I$-enriched sense, has a fixed point. Since $\mathit{Sh}(A)$ is complete [26, Prop. III.4.4], its slices $\mathit{Sh}(A)/I$ are also complete and thus the required follows from Theorem 8.6 and Lemma 8.24. □

9. Conclusion and Future Work

We have shown that the topos of sheaves over a complete Heyting algebra with a well-founded basis, in particular $\mathcal{S}$, the topos of trees, provides a model for an extension of higher-order logic over dependent type theory with guarded recursive types and terms. Moreover, we have argued that this logic provides the right setting for the synthetic construction of step-indexed models of programming languages and program logics, by constructing a model of the programming language $\mathsf{F}_{\mu,\mathsf{ref}}$ in the logic.

In this paper we have focused solely on guarded recursion. As future work, it would be interesting to study further the connections between guarded and unguarded recursion in $\mathcal{S}$. For example, it might be possible to show the existence of recursive types in which only negative occurrences of the recursion variable were guarded.

We plan to make a tool for formalized reasoning in the internal logic of $\mathcal{S}$. We have conducted some initial experiments by adding axioms to Coq and used it to formalize some of the proofs from [7] involving recursively defined relations on recursively defined types. These experiments suggest that it will be important to have special support for the manipulation of the isomorphisms involved in recursive type equations, such as the coercions and canonical structures of [18]. An alternative approach, inspired by the conference version of the present paper, has recently been proposed by Jaber et. al. [?], who show how to internalize the construction of the topos of trees in Coq and thus model guarded recursive types. Future work includes investigating how easy or difficult it is in practice to develop and work with step-indexed models using that approach.

Future work also includes studying further applications of guarded recursion in connection with step-indexed models. In particular, we plan to give a synthetic account of a recent step-indexed model by the first and third author for a language with countable non-determinism [32]. That model uses step-indexing over ω_1, the first uncountable ordinal, so would naturally live in sheaves over ω_1. Indeed, this was part of the motivation for generalizing the study of models of guarded recursion from $\mathcal{S}$ to general sheaf categories $Sh(A)$.

It could also be interesting to study predicative models of guarded recursive dependent type theory, thus extending the work of Moerdijk and Palmgren [27, 28] on "predicative toposes".

9.1. **Acknowledgments.** We thank Andy Pitts and Paul Blain Levy for encouraging discussions. This work was supported in part by grants from the Danish research council (Birkedal and Møgelberg) and from the Carlsberg Foundation (Støvring).

References

[1] M. Abbott, T. Altenkirch, and N. Ghani. Representing nested inductive types using W-types. In *Proc. of ICALP*, pages 59–71. Springer LNCS 3142, 2004.

[2] A. Ahmed, D. Dreyer, and A. Rossberg. State-dependent representation independence. In *Proc. of POPL*, pages 340–353. ACM, 2009.

[3] A.W. Appel, P.-A. Melliès, C.D. Richards, and J. Vouillon. A very modal model of a modern, major, general type system. In *Proc. of POPL*, pages 109-122. ACM 2007.

[4] L. Birkedal, R. Møgelberg, J. Schwinghammer, and K. Støvring. First steps in synthetic guarded domain theory: Step-indexing in the topos of trees. In *Proc. of LICS*, pages 55-64. IEEE Computer Society, 2011.

[5] L. Birkedal, B. Reus, J. Schwinghammer, K. Støvring, J. Thamsborg, and H. Yang. Step-indexed Kripke models over recursive worlds. In *Proc. of POPL*, pages 119-132. ACM, 2011.

[6] L. Birkedal, J. Schwinghammer, and K. Støvring. A metric model of lambda calculus with guarded recursion. Presented at FICS, 2010.

[7] L. Birkedal, J. Schwinghammer, and K. Støvring. A step-indexed Kripke model of hidden state via recursive properties on recursively defined metric spaces. In *Proc. of FOSSACS*, pages 305-319. Springer LNCS 6604, 2011.

[8] L. Birkedal, K. Støvring, and J. Thamsborg. Realisability semantics of parametric polymorphism, general references and recursive types. *Math. Struct. Comp. Sci.*, 20(4):655–703, 2010.

[9] Lars Birkedal, Kristian Støvring, and Jacob Thamsborg. The category-theoretic solution of recursive metric-space quations. *Theoretical Computer Science*, 411(47):4102–4122, 2010.

[10] P. Di Gianantonio and M. Miculan. Unifying recursive and co-recursive definitions in sheaf categories. In *Proc. of FOSSACS*, pages 136-150. Springer LNCS 2987, 2004.

[11] D. Dreyer, A. Ahmed, and L. Birkedal. Logical step-indexed logical relations. In *Proc. of LICS*, pages 71-80. IEEE Computer Society, 2009.

[12] D. Dreyer, G. Neis, and L. Birkedal. The impact of higher-order state and control effects on local relational reasoning. In *Proc. of ICFP*, pages 143-156. ACM, 2010.

[13] D. Dreyer, G. Neis, A. Rossberg, and L. Birkedal. A relational modal logic for higher-order stateful ADTs. In *Proc. of POPL*, pages 185-198. ACM, 2010.

[14] P. Dybjer. Representing inductively defined sets by wellorderings in Martin-Löf's type theory. *Theor. Comput. Sci.*, 176(1-2):329–335, 1997.

[15] P.J. Freyd. Recursive types reduced to inductive types. In *Proc. of LICS*, pages 498-507. IEEE Computer Society Press, 1990.

[16] P.J. Freyd. Algebraically complete categories. In *Proc. of the 1990 Como Category Theory Conference*, pages 95–104. Springer Verlag LNM 1488, 1991.

[17] P.J. Freyd. Remarks on algebraically compact categories. In *Applications of Categories in Computer Science*, volume 177 of *LMS Lecture Note Series*, pages 95-106, 1991.

[18] F. Garillot, G. Gonthier, A. Mahboubi, and L. Rideau. Packaging mathematical structures. In *Proc. of TPHOLs*, pages 327-342. Springer LNCS 5674, 2009.

[19] M. Hofmann. On the interpretation of type theory in locally cartesian closed categories. In *Proc. of CSL*, pages 427–441. Springer LNCS 933, 1994.

[20] J.M.E. Hyland. First steps in synthetic domain theory. In *Proc. of the 1990 Como Category Theory Conference*, pages 131-156. Springer LNM 1488, 1991.

[21] J.M.E. Hyland, P.T. Johnstone, and A.M. Pitts. Tripos theory. *Math. Proc. of the Cambridge Philosophical Society*, 88:205–232, 1980.

[22] B. Jacobs. *Categorical Logic and Type Theory*, volume 141 of *Studies in Logic and the Foundations of Mathematics*. Elsevier, 1999.

[23] G.M. Kelly. *Basic Concepts of Enriched Categories.* Number 4 in Lecture Notes in Mathematics. Cambridge University Press, 1982. Available in TAC reprint series at `http://www.tac.mta.ca/tac/reprints/articles/10/tr10.pdf`.

[24] A. Kock. Strong functors and monoidal monads. *Arch. Math.*, 23:113–120, 1972.

[25] J. Lambek and P.J. Scott. *Introduction to higher order categorical logic.* Cambridge University Press, 1986.

[26] S. Mac Lane and I. Moerdijk. *Sheaves in Geometry and Logic. A First Introduction to Topos Theory.* Springer, 1992.

[27] I. Moerdijk and E. Palmgren. Wellfounded trees in categories. *Annals of Pure and Appl. Logic*, 104:189–218, 2000.

[28] I. Moerdijk and E. Palmgren. Type theories, toposes and constructive set theory: Predicative aspects of ast. *Annals of Pure and Applied Logic*, 114:155–201, 2002.

[29] H. Nakano. A modality for recursion. In *Proc. of LICS*, pages 255–266. IEEE Computer Society, 2000.

[30] A.M. Pitts. Relational properties of domains. *Inf. Comput.*, 127(2):66–90, 1996.

[31] F. Pottier. A typed store-passing translation for general references. In *Proc. of POPL*, pages 147-158. ACM, 2011.

[32] J. Schwinghammer and L. Birkedal. Step-indexed relational reasoning for countable nondeterminism. In *Proc. of CSL*, pages 512-524. Schloss Dagstuhl, 2011.

[33] J. Schwinghammer, H. Yang, L. Birkedal, F. Pottier, and B. Reus. A semantic foundation for hidden state. In *Proc. of FOSSACS*, pages 2-17. Springer LNCS 6014, 2010.

[34] M.B. Smyth and G.D. Plotkin. The category-theoretic solution of recursive domain equations. *SIAM Journal on Computing*, 11(4):761–783, 1982.

$$\begin{array}{llll}
\text{Types:} & \tau & ::= & 1 \mid \tau_1 \times \tau_2 \mid 0 \mid \tau_1 + \tau_2 \mid \mu\alpha.\tau \mid \forall\alpha.\tau \mid \alpha \mid \tau_1 \to \tau_2 \mid \mathsf{ref}\,\tau \\
\text{Terms:} & t & ::= & x \mid l \mid () \mid \langle t_1, t_2\rangle \mid \mathsf{fst}\,t \mid \mathsf{snd}\,t \mid \mathsf{void}\,t \mid \mathsf{inl}\,t \mid \mathsf{inr}\,t \\
 & & & \mid \mathsf{case}\,t_0\ x_1.t_1\ x_2.t_2 \mid \mathsf{fold}\,t \mid \mathsf{unfold}\,t \\
 & & & \mid \Lambda\alpha.t \mid t\,[\tau] \mid \lambda x{:}t. \mid t_1\,t_2 \mid \mathsf{fix}\,f.\lambda x.t \mid \mathsf{ref}\,t \mid !t \mid t_1 := t_2
\end{array}$$

Typing rules:

$$\frac{}{\Xi \mid \Gamma \vdash x : \tau}\ (\Xi \vdash \Gamma,\ \Gamma(x) = \tau) \qquad \frac{}{\Xi \mid \Gamma \vdash () : 1}\ (\Xi \vdash \Gamma)$$

$$\frac{\Xi \mid \Gamma \vdash t_1 : \tau_1 \qquad \Xi \mid \Gamma \vdash t_2 : \tau_2}{\Xi \mid \Gamma \vdash \langle t_1, t_2\rangle : \tau_1 \times \tau_2} \qquad \frac{\Xi \mid \Gamma \vdash t : 0}{\Xi \mid \Gamma \vdash \mathsf{void}\,t : \tau}\ (\Xi \vdash \tau)$$

$$\frac{\Xi \mid \Gamma \vdash t : \tau_1 \times \tau_2}{\Xi \mid \Gamma \vdash \mathsf{fst}\,t : \tau_1} \qquad \frac{\Xi \mid \Gamma \vdash t : \tau_1 \times \tau_2}{\Xi \mid \Gamma \vdash \mathsf{snd}\,t : \tau_2}$$

$$\frac{\Xi \mid \Gamma \vdash t : \tau_1}{\Xi \mid \Gamma \vdash \mathsf{inl}\,t : \tau_1 + \tau_2}\ (\Xi \vdash \tau_2) \qquad \frac{\Xi \mid \Gamma \vdash t : \tau_2}{\Xi \mid \Gamma \vdash \mathsf{inr}\,t : \tau_1 + \tau_2}\ (\Xi \vdash \tau_1)$$

$$\frac{\Xi \mid \Gamma \vdash t_0 : \tau_1 + \tau_2 \qquad \Xi \mid \Gamma, x_i : \tau_i \vdash t_i : \tau \quad (i = 1, 2)}{\Xi \mid \Gamma \vdash \mathsf{case}\,t_0\ x_1.t_1\ x_2.t_2 : \tau}$$

$$\frac{\Xi \mid \Gamma \vdash t : \tau[\mu\alpha.\tau/\alpha]}{\Xi \mid \Gamma \vdash \mathsf{fold}\,t : \mu\alpha.\tau} \qquad \frac{\Xi \mid \Gamma \vdash t : \mu\alpha.\tau}{\Xi \mid \Gamma \vdash \mathsf{unfold}\,t : \tau[\mu\alpha.\tau/\alpha]}$$

$$\frac{\Xi, \alpha \mid \Gamma \vdash t : \tau}{\Xi \mid \Gamma \vdash \Lambda\alpha.t : \forall\alpha.\tau}\ (\Xi \vdash \Gamma) \qquad \frac{\Xi \mid \Gamma \vdash t : \forall\alpha.\tau_0}{\Xi \mid \Gamma \vdash t\,[\tau_1] : \tau_0[\tau_1/\alpha]}\ (\Xi \vdash \tau_1)$$

$$\frac{\Xi \mid \Gamma, x : \tau_0 \vdash t : \tau_1}{\Xi \mid \Gamma \vdash \lambda x{:}t. \ : \tau_0 \to \tau_1} \qquad \frac{\Xi \mid \Gamma \vdash t_1 : \tau \to \tau' \qquad \Xi \mid \Gamma \vdash t_2 : \tau}{\Xi \mid \Gamma \vdash t_1\,t_2 : \tau'}$$

$$\frac{\Xi \mid \Gamma, f : \tau_0 \to \tau_1, x : \tau_0 \vdash t : \tau_1}{\Xi \mid \Gamma \vdash \mathsf{fix}\,f.\lambda x.t : \tau_0 \to \tau_1} \qquad \frac{\Xi \mid \Gamma \vdash t : \tau}{\Xi \mid \Gamma \vdash \mathsf{ref}\,t : \mathsf{ref}\,\tau}$$

$$\frac{\Xi \mid \Gamma \vdash t : \mathsf{ref}\,\tau}{\Xi \mid \Gamma \vdash !t : \tau} \qquad \frac{\Xi \mid \Gamma \vdash t_1 : \mathsf{ref}\,\tau \qquad \Xi \mid \Gamma \vdash t_2 : \tau}{\Xi \mid \Gamma \vdash t_1 := t_2 : 1}$$

Figure 1: Programming language

Appendix A. More details on the application to step-indexing

Here are some more details on the application in Section 3. Everything is this appendix should be understood within the logic of $\mathcal{S}$.

A.1. **Language.** The full language considered in the application is shown in Figure 1.

A.2. **Interpretation of types.** Recall that we have

$$\begin{aligned}\mathcal{W} &= N \to_{\text{fin}} \widehat{\mathcal{T}}\\ \mathcal{T} &= \mathcal{W} \to_{\text{mon}} \mathcal{P}(\text{Value})\\ \mathcal{T}^{\text{c}} &= \mathcal{W} \to \mathcal{P}(\text{Term})\end{aligned}$$

and

$$\text{app} : \widehat{\mathcal{T}} \to \mathcal{T}, \qquad \text{lam} : \mathcal{T} \to \widehat{\mathcal{T}}$$

with $\text{app} \circ \text{lam} = \triangleright : \mathcal{T} \to \mathcal{T}$.

Let TVar be the set of type variables, and for $\tau \in \text{OType}$, let $\text{TEnv}(\tau) = \{\, \varphi \in \text{TVar} \to_{\text{fin}} \mathcal{T} \mid \text{FV}(\tau) \subseteq \text{dom}(\varphi) \,\}$. The interpretation of programming-language types is defined by induction, as a function

$$[\![\cdot]\!] : \prod_{\tau \in \text{OType}} \text{TEnv}(\tau) \to \mathcal{T}\,.$$

$$\begin{aligned}
[\![\alpha]\!]\varphi &= \varphi(\alpha)\\
[\![1]\!]\varphi &= \lambda w.\, \{()\}\\
[\![0]\!]\varphi &= \lambda w.\, \emptyset\\
[\![\tau_1 \times \tau_2]\!]\varphi &= \lambda w.\, \{\, (v_1,\, v_2) \mid v_1 \in [\![\tau_1]\!]\varphi(w) \wedge v_2 \in [\![\tau_2]\!]\varphi(w) \,\}\\
[\![\tau_1 + \tau_2]\!]\varphi &= \lambda w.\, \{\, \mathsf{inl}\ v_1 \mid v_1 \in [\![\tau_1]\!]\varphi(w) \,\} \cup \{\, \mathsf{inr}\ v_2 \mid v_2 \in [\![\tau_2]\!]\varphi(w) \,\}\\
[\![\mathsf{ref}\,\tau]\!]\varphi &= \lambda w.\, \{\, l \mid l \in \text{dom}(w) \wedge \forall w_1 \geq w.\, \text{app}(w(l))(w_1) = \triangleright([\![\tau]\!]\varphi)(w_1) \,\}\\
[\![\forall \alpha.\tau]\!]\varphi &= \lambda w.\, \{\, \Lambda\alpha.t \mid \forall \nu \in \mathcal{T}.\, \forall w_1 \geq w.\, t \in \mathit{comp}([\![\tau]\!]\varphi[\alpha \mapsto \nu])(w_1) \,\}\\
[\![\mu\alpha.\tau]\!]\varphi &= \mathit{fix}(\lambda\nu.\, \lambda w.\, \{\, \mathsf{fold}\, v \mid \triangleright(v \in [\![\tau]\!]\varphi[\alpha \mapsto \nu]\,(w)) \,\})\\
[\![\tau_1 \to \tau_2]\!]\varphi &= \lambda w.\, \{\, \overline{\lambda} x.t \mid \forall w_1 \geq w.\, \forall v \in [\![\tau_1]\!]\varphi(w_1).\, t[v/x] \in \mathit{comp}([\![\tau_2]\!]\varphi)(w_1) \,\}
\end{aligned}$$

Here the operations $\mathit{comp} : \mathcal{T} \to \mathcal{T}^{\text{c}}$ and $\mathit{states} : \mathcal{W} \to \mathcal{P}(\text{Store})$ are given by

$$\begin{aligned}
\mathit{comp}(\nu)(w) &= \{\, t \mid \forall s \in \mathit{states}(w).\, \mathit{eval}(t, s,\, \lambda(v_1, s_1).\, \exists w_1 \geq w.\\
&\qquad\qquad v_1 \in \nu(w_1) \ \wedge\ s_1 \in \mathit{states}(w_1)) \,\}\\
\mathit{states}(w) &= \{\, s \mid \text{dom}(s) = \text{dom}(w) \ \wedge\\
&\qquad\qquad \forall l \in \text{dom}(w).\, s(l) \in \text{app}(w(l))(w) \,\}\,.
\end{aligned}$$

A.3. **Soundness and the fundamental theorem.** Given Ξ and Γ such that Γ is well-formed in Ξ, and given $\varphi \in \mathcal{T}^{\Xi}$, define

$$[\![\Gamma]\!]\varphi(w) = \{\rho : \text{Value}^{\text{dom}(\Gamma)} | \forall (x, \tau) \in \Gamma.\, \rho(x) \in [\![\tau]\!]\varphi(w)\}.$$

Abbreviate $[\![\tau]\!]^{\text{c}}\varphi = \mathit{comp}([\![\tau]\!]\varphi)$.

Now we define semantic validity. The notation

$$\Xi \mid \Gamma \models t : \tau$$

means: For all $w \in W$, all $\varphi \in \mathcal{T}^{\Xi}$, and all $\rho \in [\![\Gamma]\!]\varphi(w)$, we have $\rho(t) \in [\![\tau]\!]^{\text{c}}\varphi(w)$. (Here $\rho(t)$ is ρ acting by substitution on t.)

To show the fundamental theorem, we must show semantic counterparts of all the typing rules. First we need some "monadic" properties of the *comp* operator. For $\nu \in \mathcal{T}$

and $\xi \in \mathcal{T}^{\mathrm{c}}$ and $w \in \mathcal{W}$, let $\nu \multimap_w \xi$ be the set of closed evaluation contexts E that satisfy the following property: for all $w_1 \geq w$ and $v \in \nu(w_1)$ we have $E[v] \in \xi(w_1)$.

Lemma A.1.

(1) If $v \in \nu(w)$, then $v \in \mathit{comp}(\nu)(w)$.
(2) If $t \in \mathit{comp}(\nu_1)(w)$ and $E \in \nu_1 \multimap_w \mathit{comp}(\nu_2)$, then $E[t] \in \mathit{comp}(\nu_2)(w)$.

Proof. The first part follows immediately from the definitions of *comp* and eval. As for the second part, let $t \in \mathit{comp}(\nu_1)(w)$ and $E \in \nu_1 \multimap_w \mathit{comp}(\nu_2)$ be given; we must show that $E[t] \in \mathit{comp}(\nu_2)(w)$. We unfold the definition of *comp*. Let $s \in \mathit{states}(w)$ be given; we must show that $\mathrm{eval}(E[t], s, Q)$ where

$$Q(v_2, s_2) = \exists w_2 \geq w.\, v_2 \in \nu_2(w_2) \ \wedge \ s_2 \in \mathit{states}(w_2)).$$

By Proposition 3.2, it suffices to show

$$\mathrm{eval}(t, s,\ \lambda(v_1, s_1).\, \mathrm{eval}(E[v_1], s_1, Q)). \tag{A.1}$$

Since $t \in \mathit{comp}(\nu_1)(w)$ and $s \in \mathit{states}(w)$ we know that

$$\begin{aligned}\mathrm{eval}(t, s,\ \lambda(v_1, s_1).\, &\exists w_1 \geq w.\\ &v_1 \in \nu_1(w_1) \ \wedge \ s_1 \in \mathit{states}(w_1)).\end{aligned}$$

We can therefore use Proposition 3.1 to show (A.1): it suffices to show that $\exists w_1 \geq w.\, v_1 \in \nu_1(w_1) \ \wedge \ s_1 \in \mathit{states}(w_1))$ implies $\mathrm{eval}(E[v_1], s_1, Q)$. So let $w_1 \geq w$ be given and assume that $v_1 \in \nu_1(w_1)$ and $s_1 \in \mathit{states}(w_1)$. Then, since $E \in \nu_1 \multimap_w \mathit{comp}(\nu_2)$, we have $E[v_1] \in \mathit{comp}(\nu_2)(w_1)$ and hence

$$\begin{aligned}\mathrm{eval}(E[v_1], s_1,\ \lambda(v_2, s_2).\, &\exists w_2 \geq w_1.\\ &v_2 \in \nu_2(w_2) \ \wedge \ s_2 \in \mathit{states}(w_2)).\end{aligned}$$

Since $w_1 \geq w$, another use of Proposition 3.1 gives $\mathrm{eval}(E[v_1], s_1, Q)$, which is what we had to show. □

Proof of Proposition 3.4 (fundamental theorem). We show four key cases.

A.4. **Case "allocation":** If $\Xi \mid \Gamma \models t : \tau$, then $\Xi \mid \Gamma \models \mathsf{ref}\, t : \mathsf{ref}\, \tau$.

Let $w \in \mathcal{W}$ and $\varphi \in \mathcal{T}^{\Xi}$ and $\rho \in [\![\Gamma]\!]\varphi$ be given; we must show that $\rho(\mathsf{ref}\, t) \in [\![\mathsf{ref}\, \tau]\!]^{\mathrm{c}}\varphi(w)$. Since $\Xi \mid \Gamma \models t : \tau$ holds we know that $\rho(t) \in [\![\tau]\!]^{\mathrm{c}}\varphi(w)$. Therefore, by Lemma A.1, it suffices to show that $\mathsf{ref}\, \text{-} \in [\![\tau]\!]\varphi \multimap_w [\![\mathsf{ref}\, \tau]\!]^{\mathrm{c}}\varphi$. To that end, let $w_1 \geq w$ and $v \in [\![\tau]\!]\varphi(w_1)$ be given. We must show that $\mathsf{ref}\, v \in [\![\mathsf{ref}\, \tau]\!]^{\mathrm{c}}\varphi(w_1)$.

Let $s \in \mathit{states}(w_1)$ be given. By definition of *comp* we must show

$$\mathrm{eval}(\mathsf{ref}\, v, s,\ \lambda(v_1, s_1).\, \exists w_2 \geq w_1.\, v_1 \in [\![\mathsf{ref}\, \tau]\!]\varphi(w_2) \wedge s_1 \in \mathit{states}(w_2)).$$

Let l be the smallest location not in s. Then we have $\mathit{step}((\mathsf{ref}\, v, s), (l, s_1))$ where $s_1 = s[l \mapsto v]$. Therefore, by definition of eval and Proposition 2.7, it suffices to show

$$\exists w_2 \geq w_1.\, l \in [\![\mathsf{ref}\, \tau]\!]\varphi(w_2) \wedge s_1 \in \mathit{states}(w_2).$$

(In fact, we are only required to show $\triangleright$ applied to this formula, which is weaker by Proposition 2.7(1).) To that end, we choose $w_2 = w_1[l \mapsto \mathrm{lam}([\![\tau]\!]\varphi)]$. It remains to show

$$l \in [\![\mathsf{ref}\, \tau]\!]\varphi(w_2) \tag{A.2}$$

$$s_1 \in \mathit{states}(w_2). \tag{A.3}$$

As for (A.2), we expand the definition of $[\![\mathsf{ref}\,\tau]\!]$. Clearly we have $l \in \mathit{dom}(w_2)$ as required. Now let $w_3 \geq w_2$ be given; Lemma 3.3 gives

$$\begin{aligned}\mathrm{app}(w_2(l))(w_3) &= \mathrm{app}(\mathrm{lam}([\![\tau]\!]\varphi))(w_3)\\ &= \triangleright([\![\tau]\!]\varphi)(w_3)\end{aligned}$$

as required.

As for (A.3), we first have that $\mathrm{dom}(s_1) = \mathrm{dom}(w_2)$ since $s \in \mathit{states}(w_1)$. Second, we must show that $s_1(l') \in \mathrm{app}(w_2(l'))(w_2)$ for all $l' \in \mathrm{dom}(s_1)$. For $l' = l$ we have $\mathrm{app}(w_2(l))(w_2) = \triangleright([\![\tau]\!]\varphi)(w_2)$ as above. But $s_1(l) = v$, and we know that $v \in [\![\tau]\!]\varphi(w_1)$ where

$$[\![\tau]\!]\varphi(w_1) \subseteq [\![\tau]\!]\varphi(w_2) \subseteq \triangleright([\![\tau]\!]\varphi)(w_2)$$

by monotonicity and Proposition 2.7(1). We conclude that $s_1(l) \in \mathrm{app}(w_2(l))(w_2)$.

For $l' \neq l$ we have $s_1(l') = s(l')$. Since $s \in \mathit{states}(w_1)$ we know that $s(l') \in \mathrm{app}(w_1(l'))(w_1)$. But

$$\begin{aligned}\mathrm{app}(w_1(l'))(w_1) &= \mathrm{app}(w_2(l'))(w_1)\\ &\subseteq \mathrm{app}(w_2(l'))(w_2)\end{aligned}$$

by monotonicity. Therefore $s_1(l') \in \mathrm{app}(w_2(l'))(w_2)$, which completes the proof of (A.3).

A.5. **Case "lookup":** If $\Xi \mid \Gamma \models t : \mathsf{ref}\,\tau$ then $\Xi \mid \Gamma \models\ !t : \tau$.

Let $w \in \mathcal{W}$ and $\varphi \in \mathcal{T}^{\Xi}$ and $\rho \in [\![\Gamma]\!]\varphi$ be given; we must show that $\rho(!t) \in [\![\tau]\!]^{\mathrm{c}}\varphi(w)$. Since $\Xi \mid \Gamma \models t : \mathsf{ref}\,\tau$ holds we know that $\rho(t) \in [\![\mathsf{ref}\,\tau]\!]^{\mathrm{c}}\varphi(w)$. Therefore, by Lemma A.1, it suffices to show that $!- \in [\![\tau]\!]\varphi \multimap_w [\![\mathsf{ref}\,\tau]\!]^{\mathrm{c}}\varphi$. This is essentially what was done in the proof sketch in the main text, but for completeness we repeat the argument here.

Let $w_1 \geq w$ and $v \in [\![\mathsf{ref}\,\tau]\!]\varphi(w_1)$ be given. We must show that $!v \in \mathit{comp}([\![\tau]\!]\varphi)(w_1)$ We unfold the definition of comp. Let $s \in \mathit{states}(w_1)$ be given; we must show

$$\mathit{eval}(!v, s, \lambda(v_2, s_2).\, \exists w_2 \geq w_1.\, v_2 \in [\![\tau]\!]\varphi(w_2) \ \wedge\ s_2 \in \mathit{states}(w_2))\,. \tag{A.4}$$

By the assumption that $v \in [\![\mathsf{ref}\,\tau]\!]\varphi(w_1)$, we know that $v = l$ for some location l such that $l \in \mathrm{dom}(w_1)$ and $\mathrm{app}(w_1(l))(w_2) = \triangleright([\![\tau]\!]\varphi)(w_2)$ for all $w_2 \geq w_1$. Since $s \in \mathit{states}(w_1)$, we know that $l \in \mathrm{dom}(s) = \mathrm{dom}(w_1)$ and $s(l) \in \mathrm{app}(w_1(l))(w_1)$. We therefore have $\mathrm{step}((!v, s), (s(l), s))$. Hence, by unfolding the definition of eval in (A.4) and using the rules from Proposition 2.7, it remains to show that

$$\exists w_2 \geq w_1.\ \triangleright(s(l) \in [\![\tau]\!]\varphi(w_2)) \ \wedge\ \triangleright(s \in \mathit{states}(w_2)).$$

To that end, choose $w_2 = w_1$. First, $s \in \mathit{states}(w_1)$ and hence $\triangleright(s \in \mathit{states}(w_1))$. Second,

$$s(l) \in \mathrm{app}(w_1(l))(w_1) = \triangleright([\![\tau]\!]\varphi)(w_1),$$

which means exactly that $\triangleright(s(l) \in [\![\tau]\!]\varphi(w_1))$.

A.6. **Case "assignment":** If $\Xi \mid \Gamma \models t_1 : \mathsf{ref}\,\tau$ and $\Xi \mid \Gamma \models t_2 : \tau$, then $\Xi \mid \Gamma \models t_1 := t_2 : 1$.

Here we must use Lemma A.1 twice. Let $w \in \mathcal{W}$ and $\varphi \in \mathcal{T}^{\Xi}$ and $\rho \in [\![\Gamma]\!]\varphi$ be given; we must show that

$$\rho(t_1 := t_2) \in [\![1]\!]^{\mathrm{c}}\varphi(w).$$

Since $\Xi \mid \Gamma \models t_1 : \mathsf{ref}\,\tau$ holds we know that $\rho(t_1) \in [\![\mathsf{ref}\,\tau]\!]^{\mathrm{c}}\varphi(w)$. Therefore, by Lemma A.1, it suffices to show that

$$(- := \rho(t_2)) \in [\![\mathsf{ref}\,\tau]\!]\varphi \multimap_w [\![1]\!]^{\mathrm{c}}\varphi.$$

So let $w_1 \geq w$ and $v_1 \in [\![\mathsf{ref}\,\tau]\!]\varphi(w_1)$ be given; we must show that $(v_1 := \rho(t_2)) \in [\![1]\!]^{\mathrm{c}}\varphi(w_1)$. By assumption we have $\rho(t_2) \in [\![\tau]\!]^{\mathrm{c}}\varphi(w_1)$, so by Lemma A.1 again, it suffices to show that

$$(v_1 := -) \in [\![\tau]\!]\varphi \multimap_{w_1} [\![1]\!]^{\mathrm{c}}\varphi.$$

Therefore, let $w_2 \geq w_1$ and $v_2 \in [\![\tau]\!]\varphi(w_2)$ be given. The final proof obligation is to show that

$$(v_1 := v_2) \in [\![1]\!]^{\mathrm{c}}\varphi(w_2).$$

We unfold the definition of *comp*. Assume that $s \in \mathit{states}(w_2)$ is given; we must show

$$\mathrm{eval}((v_1 := v_2), s,\ \lambda(v_3, s_3).\, \exists w_3 \geq w_2.\, v_3 \in [\![1]\!]\varphi(w_3)\ \wedge\ s_3 \in \mathit{states}(w_3)).$$

By monotonicity we have $v_1 \in [\![\mathsf{ref}\,\tau]\!]\varphi(w_2)$, and therefore $v_1 = l$ for some $l \in \mathrm{dom}(w_2)$ such that

$$\mathrm{app}(w_2(l))(w_3) = \triangleright([\![\tau]\!]\varphi)(w_3) \quad \text{for all } w_3 \geq w_2. \tag{A.5}$$

Furthermore, since $s \in \mathit{states}(w_2)$ we know that $\mathrm{dom}(s) = \mathrm{dom}(w_2)$ and hence that $l \in \mathrm{dom}(s)$. Therefore $\mathrm{step}((v_1 := v_2,\, s),\ ((),\, s[l \mapsto v_2]))$ holds. By definition of eval and Proposition 2.7, it then suffices to show

$$\exists w_3 \geq w_2.\, () \in [\![1]\!]\varphi(w_3)\ \wedge\ s[l \mapsto v_2] \in \mathit{states}(w_3).$$

We choose $w_3 = w_2$. Now $() \in [\![1]\!]\varphi(w_2)$ holds trivially, and it remains to show that $s[l \mapsto v_2] \in \mathit{states}(w_2)$. For $l' \neq l$ we have

$$(s[l \mapsto v_2])(l') = s(l') \in \mathrm{app}(w_2(l'))(w_2)$$

since $s \in \mathit{states}(w_2)$. Furthermore,

$$(s[l \mapsto v_2])(l) = v_2 \in [\![\tau]\!]\varphi(w_2),$$

and therefore $\triangleright(v_2 \in [\![\tau]\!]\varphi(w_2))$ by Proposition 2.7(1). But this means exactly that $v_2 \in \triangleright([\![\tau]\!]\varphi)(w_2)$. We conclude from (A.5) that $v_2 \in \mathrm{app}(w_2(l))(w_2)$ as required.

A.7. **Case "unfold":** If $\Xi \mid \Gamma \models t : \mu\alpha.\tau$, then $\Xi \mid \Gamma \models \mathsf{unfold}\,t : \tau[(\mu\alpha.\tau)/a]$.

Abbreviate $\tau_1 = \tau[(\mu\alpha.\tau)/a]$. Let $w \in \mathcal{W}$ and $\varphi \in \mathcal{T}^{\Xi}$ and $\rho \in [\![\Gamma]\!]\varphi$ be given; we must show that $\rho(\mathsf{unfold}\,t) \in [\![\tau_1]\!]^{\mathrm{c}}\varphi(w)$. Since $\Xi \mid \Gamma \models t : \mu\alpha.\tau$ holds we know that $\rho(t) \in [\![\mu\alpha.\tau]\!]^{\mathrm{c}}\varphi(w)$. Therefore, by Lemma A.1, it suffices to show that $\mathsf{unfold}\,- \in [\![\mu\alpha.\tau]\!]\varphi \multimap_w [\![\tau_1]\!]^{\mathrm{c}}\varphi$. To that end, let $w_1 \geq w$ and $v \in [\![\mu\alpha.\tau]\!]\varphi(w_1)$ be given. We must show that $\mathsf{unfold}\,v \in [\![\tau_1]\!]^{\mathrm{c}}\varphi(w_1)$.

Let $s \in \mathit{states}(w_1)$ be given. By definition of *comp* we must show

$$\mathrm{eval}(\mathsf{unfold}\,v, s,\ \lambda(v_1, s_1).\, \exists w_2 \geq w_1.\, v_1 \in [\![\tau_1]\!]\varphi(w_2)\ \wedge\ s_1 \in \mathit{states}(w_2)). \tag{A.6}$$

By definition of $[\![\mu\alpha.\tau]\!]$ we know that $v = \mathsf{fold}\,v_0$ for some v_0 such that $\triangleright(v_0 \in [\![\tau]\!]\varphi[\alpha \mapsto [\![\mu\alpha.\tau]\!]\varphi](w_1))$ holds. By Proposition 2.7 and a substitution lemma (shown by an easy induction on types), this means that $\triangleright(v_0 \in [\![\tau_1]\!]\varphi(w_1))$ holds.

Since $v = \mathsf{fold}\, v_0$ we have step$((\mathsf{unfold}\, v, s), (v_0, s))$. Therefore, by unfolding the definition of eval in (A.6) and using Proposition 2.7, it suffices to show

$$\exists w_2 \geq w_1.\ \triangleright(v_0 \in [\![\tau_1]\!]\varphi(w_2)) \wedge \triangleright(s \in \mathit{states}(w_2)).$$

We choose $w_2 = w_1$. We have already shown that $\triangleright(v_0 \in [\![\tau_1]\!]\varphi(w_1))$ holds, and Proposition 2.7(1) gives that $s \in \mathit{states}(w_1)$ implies $\triangleright(s \in \mathit{states}(w_1))$, as required. □

As an immediate corollary of the fundamental theorem we get a type-safety result for the "temporal" semantics given by the eval predicate. This is formulated by means of a trivial post-condition.

Corollary A.2 (Type safety)**.** Assume that $\vdash t : \tau$ holds. Then eval$(t, s_{\text{init}}, \top)$ holds where s_{init} is the empty store.

Proof. Follows directly from the fundamental theorem (using the empty world $\emptyset \in \mathcal{W}$) and Proposition 3.1. □