A type theory for productive coprogramming via guarded recursion

Rasmus Ejlers Møgelberg
IT University of Copenhagen
Denmark
mogel@itu.dk

Abstract

To ensure consistency and decidability of type checking, proof assistants impose a requirement of productivity on corecursive definitions. In this paper we investigate a type-based alternative to the existing syntactic productivity checks of Coq and Agda, using a combination of guarded recursion and quantification over clocks. This approach was developed by Atkey and McBride in the simply typed setting, here we extend it to a calculus with dependent types. Building on previous work on the topos-of-trees model we construct a model of the calculus using a family of presheaf toposes, each of which can be seen as a multi-dimensional version of the topos-of-trees. As part of the model construction we must solve the coherence problem for modelling dependent types in locally cartesian closed categories simulatiously in a whole family of locally cartesian closed categories. We do this by embedding all the categories in a large one and applying a recent approach to the coherence problem due to Streicher and Voevodsky.

Categories and Subject Descriptors D.3.3 [*Language Constructs and Features*]: Recursion; F.3.2 [*Semantics of Programming Languages*]: Denotational semantics; F.4.1 [*Mathematical Logic and Formal Languages*]: Lambda calculus and related systems

Keywords Guarded recursion, corecursion, dependent types, denotational semantics, categorical semantics

1. Introduction

Recursive type equations are ubiquitous in semantics of programming languages. Inductive types and coinductive types are initial and final solutions to covariant type equations, and solutions to contravariant and mixed variance type equations are needed for modelling the untyped lambda calculus and programming languages with advanced type systems.

Guarded recursion [17] is a new approach to solving recursive type equations, offering unique solutions to all equations of the form $X \cong F(\blacktriangleright X)$ where $\blacktriangleright$ is a modal type operator pronounced "later". When combining guarded recursion with dependent types, one obtains a very powerful type system in which one can, for example, construct models of advanced programming languages [7], see also [4]. The models obtained are synthetic variants of step-indexing models [3], but unlike these offer a type system for expressing the model constructions. Guarded recursion has also been used to model higher order reactive programming [14].

When considering covariant type equations, guarded recursive types are a third alternative to inductive and coinductive types. In this paper, we study the relationship between guarded recursive types and coinductive types. We show that the latter can be encoded from the former, and that using a combination of the two suggests a way of building the productivity checks of corecursive definitions into the type system of proof assistants.

Consider, for example, a type of streams of natural numbers $S(\mathbb{N})$. This is a coinductive type satisfying $S(\mathbb{N}) \cong \mathbb{N} \times S(\mathbb{N})$. To ensure consistency and decidability of type checking, proof assistants require corecursive definitions of elements of type $S(\mathbb{N})$ to be productive, i.e., the n'th element of the stream should be computable in finite time. For example, the definition $xs \stackrel{\text{def}}{=} xs$ is not productive, but $zeros \stackrel{\text{def}}{=} 0 :: zeros$ is (here :: is the cons operation). Coq and Agda enforce productivity by checking that all occurences of the recursion variable are directly underneath a constructor. The syntactic checks do not interact well with higher-order programming, and this has led to research on how to "code around" these limitations, e.g., [9].

Consider on the other hand the guarded recursive solution to the same type equation: $Str_{\mathrm{GR}} \cong \mathbb{N} \times \blacktriangleright Str_{\mathrm{GR}}$. The type $\blacktriangleright Str_{\mathrm{GR}}$ should be thought of as typing streams that are only available one time-step from now, and thus we can think of Str_{GR} as a type of streams where the tail takes one time-step to compute. To define elements of Str_{GR} recursively we use a guarded fixpoint operator $\mathrm{fix}\colon (\blacktriangleright X \to X) \to X$, satisfying the fixed point equation $f(\mathrm{next}(\mathrm{fix}(f))) = \mathrm{fix}(f)$, where $\mathrm{next}\colon X \to \blacktriangleright X$ is an operation that makes anything available now also available in the future. Using fix we can define $zeros$ as $\mathrm{fix}(\lambda z : \blacktriangleright Str_{\mathrm{GR}}.\, 0 :: z)$, but $\mathrm{fix}(\lambda xs\colon Str_{\mathrm{GR}}.\, xs)$ is not defined. In fact, the type $\blacktriangleright Str_{\mathrm{GR}} \to Str_{\mathrm{GR}}$ exactly captures productive recursive stream definitions. Since elements of coinductive types are sometimes called codata, we refer to this approach as productive coprogramming via guarded recursion. Guarded recursion can be seen as a light-weight alternative to sized types [1, 18].

To give some intuition for guarded recursion we recall the topos of trees model.

1.1 The topos of trees

The topos of trees is the category of contravariant presheaves over ω, the ordered set of natural numbers. In elementary terms, this means that a type is modelled as a family of sets and maps

$$X_1 \xleftarrow{r_1^X} X_2 \xleftarrow{r_2^X} X_3 \xleftarrow{r_3^X} \dots \tag{1}$$

Intuitively, the set X_n describes the types as it looks if we have n computation steps to reason about it. For example, the type Str_{GR} is modelled as

$$\mathbb{N} \xleftarrow{\pi} \mathbb{N}^2 \xleftarrow{\pi} \mathbb{N}^3 \xleftarrow{\pi} \mathbb{N}^4 \xleftarrow{\pi} \dots \tag{2}$$

since in n computation steps we can at most access the n first elements of a stream.

The operation $\blacktriangleright$ maps X of (1) to

$$1 \longleftarrow X_1 \xleftarrow{r_1^X} X_2 \xleftarrow{r_2^X} X_3 \xleftarrow{r_3^X} \dots$$

and since products are computed pointwise, the object Str_{GR} does indeed satisfy $Str_{\mathrm{GR}} \cong \mathbb{N} \times \blacktriangleright Str_{\mathrm{GR}}$.

1.2 A calculus with universal quantification over clocks

Guarded recursive types are not a replacement for coinductive types. Sometimes the time-steps get in the way. For example, the head of the tail of a stream has type $\blacktriangleright \mathbb{N}$. In the topos of trees, the endomaps on Str_{GR} correspond to the causal stream functions, i.e., those for which the n first elements of the output stream only depend on the n first elements of the input stream.

Atkey and McBride [5] suggest a simply typed calculus that allows one to conveniently switch between guarded recursive types and coinductive types, using the former whenever constructing streams, and using the latter when taking them apart. The calculus types terms in a context of free clock variables ranged over by $\kappa, \kappa' \dots$ each corresponding to a different notion of time step. For each κ, there is a $\blacktriangleright^\kappa$ and fix^κ etc. The calculus includes universal quantification over clocks in types, i.e., if A is a type, so is $\forall\kappa.A$.

The key result relating guarded types to coinductive types is that the latter can be encoded using the former. Precisely, they show that $\nu X.F(X) \stackrel{\mathrm{def}}{=} \forall\kappa.\mathrm{Fix}^\kappa\, X.F(\blacktriangleright^\kappa X)$ is a coinductive type, if $\mathrm{Fix}^\kappa\, X.F(\blacktriangleright^\kappa X) \cong F(\blacktriangleright^\kappa(\mathrm{Fix}^\kappa\, X.F(\blacktriangleright^\kappa X)))$. The result holds for F strictly positive and κ not appearing in F, and is proved with respect to a denotational model.

For these results to be applicable in proof assistants, they must be extended to a type theory with dependent types. We do that here, and give a model using a family of categories $\mathrm{GR}[\Delta]$ indexed by clock contexts Δ each a multi-dimensional version of the topos of trees. Types and terms in the context of a single clock variable are modelled in the topos of trees, and those in empty clock context in the category of sets. Universal quantification over clocks is modelled by taking limits. Note that the limit of the sequence (2) is the set of streams.

Each $\mathrm{GR}[\Delta]$ is locally cartesian closed, and so to model dependent type theory, we must address the coherence problem, i.e., the problem that pullback, which models substitution, usually does not commute strictly with constructions such as dependent products and is not associative. To do this, we follow a recent approach by Voevodsky and Streicher using universes, but since we have not just one locally cartesian closed category but a whole family, we must first embed all the categories in a single one in a structure preserving way.

The focus of the paper is on presenting the calculus and the model construction. Atkey and McBride give a series of examples illustrating the usefulness of their calculus, including an encoding of stream processors due to Ghani et al.[10]. Since their calculus is essentially subsumed by the one presented here, all these examples can be transferred to our type theory. We shall therefore not argue further for the usefulness of the type theory, but only give a few examples, including one (Example 2) that requires dependent types and therefore cannot be expressed in the calculus of Atkey and McBride.

$$\frac{X = 0, 1, 2}{\Delta; \Gamma \vdash X : \mathrm{Type}} \qquad \frac{\Delta; \Gamma \vdash A : \mathrm{Type} \quad \Delta; \Gamma \vdash B : \mathrm{Type}}{\Delta; \Gamma \vdash A \times B : \mathrm{Type}}$$

$$\frac{\Delta; \Gamma \vdash A : \mathrm{Type} \quad \Delta; \Gamma \vdash B : \mathrm{Type}}{\Delta; \Gamma \vdash A + B : \mathrm{Type}} \qquad \frac{\Delta; \Gamma, x : A \vdash B : \mathrm{Type}}{\Delta; \Gamma \vdash \prod x : A.B : \mathrm{Type}}$$

$$\frac{\Delta; \Gamma, x : A \vdash B : \mathrm{Type}}{\Delta; \Gamma \vdash \sum x : A.B : \mathrm{Type}} \qquad \frac{\Delta; \Gamma, x : A \vdash B : \mathrm{Type}}{\Delta; \Gamma \vdash \mathrm{W}x : A.B : \mathrm{Type}}$$

Figure 1. Types of a basic dependent type theory with clock contexts

1.3 Structure of the paper

We extend the simply typed calculus of Atkey and McBride with dependent types and universes in Section 2 and prove correct the encoding of coinductive types using guarded recursive types in Section 3. As a special case we show how to encode M-types using guarded recursion (Corollary 2). We construct the family of categories $\mathrm{GR}[\Delta]$ in Section 4, and in Section 5 we show how to construct universes in these from Grothendieck universes.

The category GR subsuming all the categories $\mathrm{GR}[\Delta]$ is presented in Section 6, and Section 7 constructs the universes needed both for solving the coherence problem and to model the universes in the type theory. The interpretation of the calculus in the model is presented in Section 8.

2. Syntax

We now describe the calculus in details. To accomodate quantification over clocks the calculus has a context of clock variables. We assume given a countably infinite set CV of clock variables, and we use κ, κ' etc to range over this set. A clock context is a finite list Δ of clock variables without repetition. For every clock context Δ there are judgements of well-formed types and contexts over Δ, as well as typing judgements for terms. These rules include basic types like 0 (the empty type), 1 (unit type), 2 (a boolean type), binary products and sums, and formation of dependent sums and products and W-types, the type theoretic version of inductive types. We do not consider identity types in this paper. We consider an equality theory as corresponding to models in locally cartesian closed categories with W-types [16]. The rules are standard and for reasons of space we just include the type formation rules, see Figure 1. We write $A \to B$ (and sometimes B^A) for $\prod x : A.B$ when x is not free in B.

Figures 2 and 3 describe an extension of dependent type theory with guarded recursion and universal quantification over clocks. We consider universal quantification over clocks a binding construction and thus include (implicitly) α-equalities of universally quantified types and Λ-abstraction. We define the set of free clock variables $\mathrm{fc}(t) \subset \mathrm{CV}$ of terms, types and contexts in the obvious way. We say a term or type is closed if it can be typed in an empty context Γ (the clock context needs not be empty).

We say that two terms t, u of same type in the same context are *provably equal* if they are in the least congruence relation generated by the equalities presented here.

2.1 Guarded recursion

The type constructor $\blacktriangleright^\kappa$ is an applicative functor in the sense of McBride and Paterson [15]. Note in particular that this means that there is a term

$$\Delta; \Gamma, f : A \to B \vdash \lambda x : \overset{\kappa}{\blacktriangleright} A.\ (\mathrm{next}^\kappa f) \circledast^\kappa x : \overset{\kappa}{\blacktriangleright} A \to \overset{\kappa}{\blacktriangleright} B$$

wellformed types

$$\frac{\Delta,\kappa;\Gamma \vdash A\colon \mathrm{Type} \quad \kappa \notin \mathrm{fc}(\Gamma)}{\Delta;\Gamma \vdash \forall\kappa.A\colon \mathrm{Type}} \qquad \frac{\Delta;\Gamma \vdash A\colon \mathrm{Type} \quad \kappa \in \Delta}{\Delta;\Gamma \vdash \overset{\kappa}{\blacktriangleright} A\colon \mathrm{Type}}$$

Typing rules

$$\frac{\Delta;\Gamma \vdash t\colon A}{\Delta;\Gamma \vdash \mathrm{next}^\kappa(t)\colon \overset{\kappa}{\blacktriangleright} A} \qquad \frac{\Delta;\Gamma \vdash t\colon \overset{\kappa}{\blacktriangleright}(A \to B) \quad \Delta;\Gamma \vdash u\colon \overset{\kappa}{\blacktriangleright} A}{\Delta;\Gamma \vdash t \circledast^\kappa u\colon \overset{\kappa}{\blacktriangleright} B} \qquad \frac{\Delta;\Gamma, x\colon \overset{\kappa}{\blacktriangleright} A \vdash t\colon A}{\Delta;\Gamma \vdash \mathrm{fix}^\kappa x.t\colon A}$$

$$\frac{\kappa \notin \mathrm{fc}(\Gamma) \quad \Delta,\kappa;\Gamma \vdash A\colon \mathrm{Type} \quad \Delta,\kappa';\Gamma,\Gamma' \vdash t\colon \forall\kappa.A}{\Delta,\kappa';\Gamma,\Gamma' \vdash t[\kappa']\colon A[\kappa'/\kappa]} \qquad \frac{\Delta,\kappa;\Gamma \vdash t\colon A \quad \kappa \notin \mathrm{fc}(\Gamma)}{\Delta;\Gamma \vdash \Lambda\kappa.t\colon \forall\kappa.A}$$

Figure 2. An extension of dependent type theory with guarded recursion and universal quantification over clocks.

$$\begin{aligned}
(\Lambda\kappa.t)[\kappa'] &= t[\kappa'/\kappa] \\
\Lambda\kappa.(t[\kappa]) &= t \qquad \kappa \notin \mathrm{fc}(t) \\
\mathrm{next}^\kappa(\lambda x.x) \circledast^\kappa u &= u \\
\mathrm{next}^\kappa(\circ) \circledast^\kappa s \circledast^\kappa t \circledast^\kappa u &= s \circledast^\kappa (t \circledast^\kappa u) \\
\mathrm{next}^\kappa(t) \circledast^\kappa \mathrm{next}^\kappa(u) &= \mathrm{next}^\kappa(t(u)) \\
u \circledast^\kappa \mathrm{next}^\kappa(t) &= \mathrm{next}^\kappa(\lambda g.g(t)) \circledast^\kappa u \\
t[\mathrm{next}^\kappa(\mathrm{fix}^\kappa x.t)/x] &= \mathrm{fix}^\kappa x.t \\
\frac{t[\mathrm{next}^\kappa(u)/x] = u}{u = \mathrm{fix}^\kappa x.t}
\end{aligned}$$

Figure 3. Equational rules for an extension of dependent type theory with guarded recursion and universal quantification over clocks. In the fourth equality, $\circ$ is function composition.

which behaves functorially (i.e., preserves identities and commutes with function composition). We shall write $\blacktriangleright^\kappa f\colon \blacktriangleright^\kappa A \to \blacktriangleright^\kappa B$, whenever $f\colon A \to B$. On the other hand, $\blacktriangleright^\kappa$ is not a monad. Indeed, in the model to be described later, there is no map $\blacktriangleright^\kappa \blacktriangleright^\kappa 0 \to \blacktriangleright^\kappa 0$.

There are two ways of distributing $\blacktriangleright^\kappa$ over function types. The first one is

$$\overset{\kappa}{\blacktriangleright}(A \to B) \to \overset{\kappa}{\blacktriangleright} A \to \overset{\kappa}{\blacktriangleright} B \tag{3}$$

obtained by currying $\circledast$. This mapping is an isomorphism in the model, but not in the calculus. The other one is

$$d\colon \overset{\kappa}{\blacktriangleright}(\textstyle\prod x\colon A.B) \to \prod x\colon A.\overset{\kappa}{\blacktriangleright} B \tag{4}$$

defined as $\lambda f\colon \blacktriangleright^\kappa \prod x\colon A.B.\ \lambda x\colon A.\ \blacktriangleright^\kappa(\mathrm{ev}_x)(f)$ where ev_z is $\lambda g\colon (\prod x\colon A.B).\ g(z)$. In the case of $\kappa \notin \mathrm{fc}(A)$ this is an isomorphism in the model. Note that it satisfies $d(\mathrm{next}^\kappa f) = \lambda x\colon A.\ \mathrm{next}^\kappa(f(x))$

We say that a term $f\colon A \to B$ is *contractive*, if there is a κ and a term $g\colon \blacktriangleright^\kappa A \to B$ such that $f = g \circ \mathrm{next}^\kappa$. Note that the equational theory states that contractive terms $f\colon A \to A$ have unique fixed points.

2.2 Clock quantification

Clock abstraction $\Lambda\kappa.t$ restricts the clock κ to t, and clock application $t[\kappa']$ instantiates the restricted clock. The purpose of the assumptions in the rule for clock application is to avoid synchronisation of clocks; the clock κ' that t is instantiated with is not allowed to already occur in A, nor any variable that A depends upon. On the other hand κ' can appear in t and in the variables that t depends on, hence the two components of the context for t.

In categorical terms, universal quantification over clocks is a right adjoint as we now explain. First we construct categories of types and terms: if $\Delta;\Gamma$ is a valid context, we can consider the category whose objects are types A such that $\Delta;\Gamma \vdash A\colon \mathrm{Type}$ and morphisms are terms $\Delta;\Gamma \vdash t\colon A \to B$ considered up to provable equality. We shall call this category the slice over $(\Delta;\Gamma)$. For $\kappa \in \Delta$, the type former $\blacktriangleright^\kappa$ extends to an endofunctor on the slice over $(\Delta;\Gamma)$ as explained above, and one can show that next^κ defines a natural transformation. The following lemma introduces weakening functors between slices.

Lemma 1. *Judgements of wellformed types, contexts and typing judgements are closed under weakening in both clock contexts and variable contexts, e.g., if $\Delta;\Gamma \vdash t\colon A$ is a valid typing judgement and $\Delta,\Delta';\Gamma,\Gamma'$ is wellformed, then also $\Delta,\Delta';\Gamma,\Gamma' \vdash t\colon A$ is a valid typing judgement.*

In particular, there is a weakening functor from the slice over $(\Delta;\Gamma)$ to the slice over $(\Delta,\kappa;\Gamma)$ whenever $\kappa \notin \Delta$. On the other hand, if $\kappa \notin \Delta$ we can define a functor $\forall\kappa$ going the other way, whose action on morphisms maps $\Delta,\kappa;\Gamma \vdash t\colon A \to B$ to

$$\forall\kappa(t) \stackrel{\mathrm{def}}{=} \lambda x\colon \forall\kappa.A.\ \Lambda\kappa.t(x[\kappa])\colon \forall\kappa.A \to \forall\kappa.B$$

Proposition 1. *Let $\Delta;\Gamma$ be a well-formed context, and let $\Delta,\kappa;\Gamma \vdash A\colon \mathrm{Type}$. There is a natural bijection between terms (considered up to provable equivalence) of the form $\Delta,\kappa;\Gamma \vdash t\colon A$ and those of the form $\Delta;\Gamma \vdash s\colon \forall\kappa.A$. In categorical terms, the functor $\forall\kappa$ from the slice over $(\Delta,\kappa;\Gamma)$ to the slice over $(\Delta;\Gamma)$ is right adjoint to the weakening functor.*

The correspondence maps $\Delta,\kappa;\Gamma \vdash t\colon A$ to $\Lambda\kappa.t$ and $\Delta;\Gamma \vdash s\colon \forall\kappa.A$ to $s[\kappa]$. The unit of the adjunction is

$$\lambda x\colon A.\ \Lambda\kappa.x\colon A \to \forall\kappa.A \qquad (\kappa \notin \mathrm{fc}(A))$$

and the counit is

$$\mathrm{ev}_\kappa \stackrel{\mathrm{def}}{=} \lambda x\colon \forall\kappa.A.\ x[\kappa].$$

Naturality of ev_κ is $\mathrm{ev}_\kappa \circ \forall\kappa(f) = f \circ \mathrm{ev}_\kappa$.

2.3 Type isomorphisms

We say that two types A, B well-formed in the same context $\Delta;\Gamma$ are isomorphic, if they are isomorphic as objects of the slice over $(\Delta;\Gamma)$, i.e., if there are are terms $f\colon A \to B$ and $g\colon B \to A$ both well-typed in context $\Delta;\Gamma$ such that both compositions are provably equal to the identity. We write $A \cong B$ to mean that A and B are isomorphic. For example, the following three type

isomorphisms can be constructed in the calculus:

$$\begin{aligned} \forall\kappa.A \times B &\cong \forall\kappa.A \times \forall\kappa.B \\ \forall\kappa.\textstyle\prod x\colon A.B &\cong \textstyle\prod x\colon A.\forall\kappa.B && (\kappa \notin \mathrm{fc}(A)) \\ \forall\kappa.\forall\kappa'.A &\cong \forall\kappa'.\forall\kappa.A \end{aligned}$$

We extend the type theory with the type isomorphisms of Figure 4. The justification for these is that they hold in the model (Theorem 3). Most of these are generalisations of the type equalities of [5], but will not be identities in our model. Precisely, for all the type isomorphisms listed, the term from left to right can be defined canonically in our calculus, but we add the term going the other way, plus two equalities of terms, stating that the added term is an inverse to the existing one.

$$\begin{aligned} \forall\kappa.A &\cong A && (\kappa \notin \mathrm{fc}(A)) \\ (\forall\kappa.A) + (\forall\kappa.B) &\cong \forall\kappa.(A + B) \\ \textstyle\sum x\colon A.\forall\kappa.B &\cong \forall\kappa.\textstyle\sum x\colon A.B && (\kappa \notin \mathrm{fc}(A)) \\ \overset{\kappa'}{\blacktriangleright}\forall\kappa.A &\cong \forall\kappa.\overset{\kappa'}{\blacktriangleright} A && (\kappa \neq \kappa') \\ \forall\kappa.A &\cong \forall\kappa.\overset{\kappa}{\blacktriangleright} A \end{aligned}$$

Figure 4. Type isomorphisms. The directions from left to right are definable in the calculus

The first of these states that the unit of the adjunction of Proposition 1 is an isomorphism. Thus, we obtain the following corollary to Proposition 1.

Corollary 1. *Let* $\Delta; - \vdash A\colon \mathrm{Type}$ *be a closed type and let* $\kappa \notin \Delta$. *There is a bijective (up to provable equivalence) correspondence between terms* $\Delta, \kappa; - \vdash t\colon A$ *and terms* $\Delta; - \vdash s\colon A$.

The last isomorphism of Figure 4 is particularly important. Following [5] we name it force, and the axioms added state that force is inverse to the term

$$\lambda x\colon \forall\kappa.A.\ \Lambda\kappa.\mathrm{next}^\kappa(x[\kappa])\colon \forall\kappa.A \to \forall\kappa.\overset{\kappa}{\blacktriangleright} A$$

The term force allows us to ignore time steps in universally quantified clocks. It is used in the encoding of coinductive types using guarded recursive types (proof of Theorem 2).

Atkey and McBride state one more type equality commuting inductive type formation over $\forall\kappa$ in special cases. Since the only inductive types considered in this paper are W-types, we restrict attention to these. To best describe the type isomorphism, recall that W-types are initial algebras for polynomial functors, in particular, the W-type $\Delta; \Gamma \vdash \mathrm{W}x\colon A.B\colon \mathrm{Type}$ can be capture up to isomorphism as the initial algebra for the functor $F(X) = \sum a\colon A.\prod b\colon B.X$.

Now, consider polynomials of two variables, i.e., of the form

$$F(X, Y) = \textstyle\sum a\colon A.(\prod b\colon B_0.X) \times (\prod b\colon B_1.Y)$$

for some A, B_0, B_1 such that

$$\begin{aligned} &\Delta; \Gamma \vdash A\colon \mathrm{Type} \\ &\Delta; \Gamma, a\colon A \vdash B_0\colon \mathrm{Type} \\ &\Delta; \Gamma, a\colon A \vdash B_1\colon \mathrm{Type} \end{aligned}$$

Whenever $\Delta; \Gamma \vdash Y\colon \mathrm{Type}$, the initial algebra $\mu X.F(X, Y)$ exists for the functor $F(-, Y)$, since it is a W-Type. This also extends: if $\kappa \notin \Delta$ and $\Delta, \kappa; \Gamma \vdash Y\colon \mathrm{Type}$, also the initial algebra $\mu X.F(X, Y)$ exists for the functor $F(-, Y)$ on the slice over $(\Delta, \kappa; \Gamma)$.

We add to the type theory inverses making the canonical map

$$\mu X.F(X, \forall\kappa.Y) \to \forall\kappa.\mu X.F(X, Y) \tag{5}$$

an isomorphism.

2.4 Universes

Following [8] we use universes to introduce guarded recursive types. Universes should contain the base types and be closed under type constructors including $\blacktriangleright^\kappa$, but of course only for κ in the current clock context. Since the collection of types in the universe thus depends on the current clock context, it does not make sense to talk about a single universe closed under all these constructions, rather we must consider a family of universes U_Δ indexed by clock contexts Δ. See also Remark 1 for a semantic motivation for indexing universes by clock contexts.

Figure 5 gives the rules for universes. The $\blacktriangleright^\kappa$ in the hypothesis for the rule for $\triangleright^\kappa$ is crucial for ensuring a large collection of *contractive functors* as we shall see in Section 3.

3. Corecursion via guarded recursion

In this section we show how to encode coinductive types via guarded recursive types extending the results of [5] to dependent types. We need to speak of *internal* slice categories (as opposed to the external ones of Section 2.2) and functors between them. For readability we write X rather than $\mathrm{El}(X)$.

3.1 Internal slice categories

If $\Delta; - \vdash I\colon \mathrm{Type}$ is a valid judgement, we introduce the internal slice category over I to be the category whose objects are closed terms of type $X\colon I \to \mathrm{U}_\Delta$ and morphisms are equivalence classes of closed terms $f\colon \prod i\colon I.X(i) \to Y(i)$ considered up to provable equality. We write simply $f\colon X \to Y$ if it is clear from context that X and Y are objects of the internal slice category.

A functor between slice categories is a pair of closed terms

$$\begin{aligned} &F_0\colon (I \to \mathrm{U}_\Delta) \to (J \to \mathrm{U}_{\Delta'}) \\ &F_1\colon \textstyle\prod X, Y\colon \mathrm{U}_\Delta^I.(\prod i\colon I.Y(i)^{X(i)}) \to \prod j\colon J.F_0Y(j)^{F_0X(j)} \end{aligned}$$

preserving identity maps and composition up to provable equality. Note that F_0, F_1 must necessarily be typed in a clock context Δ'' such that $\Delta, \Delta' \subseteq \Delta''$. Which Δ'' we choose is irrelevant by Corollary 1. Following standard convention from category theory, we often write simply F for both F_0 and F_1.

For example, for any slice $(\Delta; I)$ we can define an endofunctor $\blacktriangleright^\kappa$ pointwise as $(\blacktriangleright^\kappa X)(i) = \triangleright^\kappa(\mathrm{next}^\kappa(X(i)))$. If, moreover, $\kappa \notin \Delta$, we can construct the functor $\forall\kappa$ from the slice over $(\Delta, \kappa; I)$ to that over $(\Delta; I)$ which acts on objects by

$$(\forall\kappa.X)(i) = \overline{\forall}\kappa(\Lambda\kappa.X(i))$$

The universe inclusion $\mathrm{in}_{\Delta',\Delta}\colon \mathrm{U}_{\Delta'} \to \mathrm{U}_\Delta$ induces a functor from the slice over $(\Delta'; I)$ to that over $(\Delta; I)$ which we shall also call $\mathrm{in}_{\Delta',\Delta}$. We say that an endofunctor F on the slice over $(\Delta'; I)$ *extends* to $(\Delta; I)$ if there is an endofunctor G on the slice over $(\Delta; I)$ such that $G \circ \mathrm{in}_{\Delta',\Delta} = \mathrm{in}_{\Delta',\Delta} \circ F$. We shall often simply write F also for the extension of F, since in the examples we have in mind, these are often just the same (open) type expression.

3.2 Contractive functors

We say that a functor F is *contractive* if both F_0 and F_1 are contractive maps. For example, the endofunctor $\blacktriangleright^\kappa$ is contractive, as witnessed on objects by the composite

$$\blacktriangleright^\kappa(I \to \mathrm{U}_\Delta) \xrightarrow{d} (I \to \blacktriangleright^\kappa \mathrm{U}_\Delta) \xrightarrow{(I \to \triangleright^\kappa)} (I \to \mathrm{U}_\Delta)$$

wellformed types

$$\frac{\Delta' \subseteq \Delta}{\Delta;\Gamma \vdash \mathrm{U}_{\Delta'} : \mathrm{Type}} \qquad \frac{\Delta;\Gamma \vdash t : \mathrm{U}_{\Delta'}}{\Delta;\Gamma \vdash \mathrm{El}(t) : \mathrm{Type}}$$

Typing rules

$$\frac{X = 0,1,2}{\Delta;\Gamma \vdash X : \mathrm{U}_{\Delta'}} \qquad \frac{\Delta;\Gamma \vdash A : \mathrm{U}_{\Delta'} \quad \Delta;\Gamma \vdash B : \mathrm{U}_{\Delta'}}{\Delta;\Gamma \vdash A \times B : \mathrm{U}_{\Delta'}}$$

$$\frac{\Delta;\Gamma \vdash A : \mathrm{U}_{\Delta'} \quad \Delta;\Gamma \vdash B : \mathrm{U}_{\Delta'}}{\Delta;\Gamma \vdash A + B : \mathrm{U}_{\Delta'}}$$

$$\frac{\Delta;\Gamma \vdash A : \mathrm{U}_{\Delta'} \quad \Delta;\Gamma, x : \mathrm{El}(A) \vdash B : \mathrm{U}_{\Delta'}}{\Delta;\Gamma \vdash \prod x : A.B : \mathrm{U}_{\Delta'}}$$

$$\frac{\Delta;\Gamma \vdash A : \mathrm{U}_{\Delta'} \quad \Delta;\Gamma, x : \mathrm{El}(A) \vdash B : \mathrm{U}_{\Delta'}}{\Delta;\Gamma \vdash \sum x : A.B : \mathrm{U}_{\Delta'}}$$

$$\frac{\Delta;\Gamma \vdash A : \mathrm{U}_{\Delta'} \quad \Delta;\Gamma, x : \mathrm{El}(A) \vdash B : \mathrm{U}_{\Delta'}}{\Delta;\Gamma \vdash \mathrm{W}x : A.B : \mathrm{U}_{\Delta'}}$$

$$\frac{\Delta;\Gamma \vdash A : \overset{\kappa}{\blacktriangleright} \mathrm{U}_{\Delta'} \quad \kappa \in \Delta'}{\Delta;\Gamma \vdash \overset{\kappa}{\triangleright} A : \mathrm{U}_{\Delta'}} \qquad \frac{\Delta;\Gamma \vdash A : \forall\kappa.\mathrm{U}_{\Delta',\kappa}}{\Delta;\Gamma \vdash \overline{\forall}\kappa.A : \mathrm{U}_{\Delta'}}$$

$$\frac{\Delta;\Gamma \vdash A : \mathrm{U}_{\Delta''} \quad \Delta'' \subseteq \Delta' \subseteq \Delta}{\Delta;\Gamma \vdash \mathrm{in}_{\Delta'',\Delta'}(A) : \mathrm{U}_{\Delta'}}$$

Equalities

$$\begin{aligned}
\mathrm{El}(X) &\equiv X && X = 0,1,2\\
\mathrm{El}(A \times B) &\equiv \mathrm{El}(A) \times \mathrm{El}(B)\\
\mathrm{El}(A + B) &\equiv \mathrm{El}(A) + \mathrm{El}(B)\\
\mathrm{El}(\overset{\kappa}{\triangleright}\, \mathrm{next}^\kappa(A)) &\equiv \overset{\kappa}{\blacktriangleright} \mathrm{El}(A)\\
\mathrm{El}(\overline{\forall}\kappa.(\Lambda\kappa.A)) &\equiv \forall\kappa.\mathrm{El}(A)\\
\mathrm{El}(\textstyle\sum x : A.B) &\equiv \textstyle\sum x : \mathrm{El}(A).\mathrm{El}(B)\\
\mathrm{El}(\textstyle\prod x : A.B) &\equiv \textstyle\prod x : \mathrm{El}(A).\mathrm{El}(B)\\
\mathrm{El}(\mathrm{W}x : A.B) &\equiv \mathrm{W}x : \mathrm{El}(A).\mathrm{El}(B)\\
\mathrm{El}(\mathrm{in}_{\Delta'',\Delta'}(A)) &\equiv \mathrm{El}(A)\\
\mathrm{in}_{\Delta'',\Delta'}(\mathrm{in}_{\Delta''',\Delta''}(A)) &= \mathrm{in}_{\Delta''',\Delta'}(A)\\
\mathrm{in}_{\Delta',\Delta'}(A) &= A
\end{aligned}$$

Figure 5. Rules for universes

where the first map is the d of (4), and the action on morphisms is proved contractive similarly. Since $f \circ g$ is contractive whenever f or g is, we deduce the following lemma.

Lemma 2. *For any functor F between slice categories in context Δ, and for any $\kappa \in \Delta$, the composite functor $F \circ \blacktriangleright^\kappa$ is contractive.*

The next theorem shows the interest in contractive functors.

Theorem 1 ([8])**.** *Let F be a contractive endofunctor on a slice. Then F has a fixed point $\mathrm{Fix}^\kappa\, X.F(X)$ (up to identity) which is both an initial algebra and a final coalgebra for F. Moreover $\mathrm{Fix}^\kappa\, X.F(X)$ is the unique up-to-identity fixed point, and also unique up to isomorphism among the up-to-iso fixed points.*

Proof. Since F_0 is contractive it has a unique fixed point. Given any fixed point up to isomorphism $f : FX \cong X$ and an algebra $g : FY \to Y$ the map $\lambda h : X \to Y.\, g \circ Fh \circ f^{-1}$ is contractive and thus has a unique fixed point, which is a unique algebra map from f to g. Thus, any up-to-iso fixed point for F is an initial algebra and thus all up-to-iso fixed point are isomorphic. The case of final coalgebras is similar. □

Note that from the proof we can construct a term

$$\mathrm{fold} : \textstyle\prod Z : I \to \mathrm{U}_\Delta.(FZ \to Z) \to (\mathrm{Fix}^\kappa\, X.F(X)) \to Z$$

giving the unique map from $\mathrm{Fix}^\kappa\, X.F(X)$ to any other algebra.

Example 1. *In the slice over $(\kappa, 1)$, the endofunctor given by $F(X) = \mathbb{N} \times \blacktriangleright^\kappa X$ is contractive and thus has a unique fixed point Str^κ, which is at the same time an initial algebra and final coalgebra for F. Note that even if we want to talk about types rather than codes of types, we get*

$$\begin{aligned}
\mathrm{El}(Str^\kappa) &\equiv \mathrm{El}(\mathbb{N} \times \overset{\kappa}{\triangleright}(\mathrm{next}^\kappa(Str^\kappa)))\\
&\equiv \mathbb{N} \times \overset{\kappa}{\blacktriangleright} \mathrm{El}(Str^\kappa)
\end{aligned}$$

Example 2. *We now consider a more advanced example involving dependent types. In the slice category over $\mathbb{N}$ (empty clock context), Altenkirch and Morris [2] define the functor*

$$F(X)(n) = \mathrm{Fin}(n) + X(n) \times X(n) + X(n+1)$$

where Fin *is defined by induction on n:*

$$\begin{aligned}
\mathrm{Fin}(0) &= 1\\
\mathrm{Fin}(n+1) &= \mathrm{Fin}(n) + 1\,.
\end{aligned}$$

as interpreted in set theory, the initial algebra Lam *for F has as* $\mathrm{Lam}(n)$ *the set of lambda terms with n free variables. The final coalgebra for* ILam *has* $\mathrm{ILam}(n)$ *the set of possibly infinite lambda terms with n free variables.*

We can also consider the type Lam^κ defined as the fixed point of $F \circ \blacktriangleright^\kappa$, i.e., satisfying

$$\mathrm{Lam}^\kappa(n) = \mathrm{Fin}(n) + \overset{\kappa}{\blacktriangleright} \mathrm{Lam}^\kappa(n) \times \overset{\kappa}{\blacktriangleright} \mathrm{Lam}^\kappa(n) + \overset{\kappa}{\blacktriangleright} \mathrm{Lam}^\kappa(n+1)$$

Since $\lambda n : \mathbb{N}.\mathrm{Lam}^\kappa(n+1)$ carries an algebra for $F \circ \blacktriangleright^\kappa$, and since Lam^κ is an initial algebra, we obtain an inclusion $i : \mathrm{Lam}^\kappa(n) \to \mathrm{Lam}^\kappa(n+1)$ corresponding to weakening. We can use this to construct the element

$$l = \mathrm{fix}^\kappa x : \overset{\kappa}{\blacktriangleright} \mathrm{Lam}^\kappa(n).\mathrm{in}_\lambda(\overset{\kappa}{\blacktriangleright}(i)(x))$$

where $\mathrm{in}_\lambda : \blacktriangleright^\kappa \mathrm{Lam}^\kappa(n+1) \to \mathrm{Lam}^\kappa(n)$ is the inclusion. The element l corresponds to the infinite lambda term $\lambda x.\lambda x.\lambda x.\ldots$.

3.3 Encoding coinductive types

We can now state the main theorem of this section, relating guarded recursive types to coinductive types.

Definition 1. *Let F be an endofunctor on the slice over $(\Delta; I)$, and let $\kappa \notin \Delta$. We say that F* commutes *with $\forall\kappa$, if F extends to $(\Delta, \kappa; I)$ (as in Section 3.1) and the canonical map $F(\forall\kappa.X) \to \forall\kappa.F(X)$ is an isomorphism for all X.*

Theorem 2. *Let F be an endofunctor on a slice over $(\Delta; I)$ and let $\kappa \notin \Delta$. If F commutes with $\forall\kappa$, then*

$$\nu X.F(X) \overset{\mathrm{def}}{=} \forall\kappa.\mathrm{Fix}^\kappa\, X.F(\overset{\kappa}{\blacktriangleright} X)$$

carries a final coalgebra structure for F. Moreover, there is a term

$$\mathrm{unfold} : \textstyle\prod Z : I \to \mathrm{U}_\Delta.(Z \to FZ) \to Z \to \nu X.F(X)$$

computing the unique map from any other coalgebra.

By the type isomorphisms of Section 2.3, dependent and binary sums and products commute with universal quantification over clocks. In particular, the functors F of Examples 1 and 2 commute with $\forall\kappa$. Thus $\forall\kappa.Str^\kappa$ is a coinductive type of streams, and $\forall\kappa.\mathrm{Lam}^\kappa$ is a coinductive type of infinite lambda terms. The term $\Lambda\kappa.l\colon \forall\kappa.\mathrm{Lam}^\kappa(0)$ is the element of this type corresponding to $\lambda x.\lambda x.\lambda x.\ldots$.

All polynomial functors commute with $\forall\kappa$ and since M-types by definition are final coalgebras for these we get the following.

Corollary 2. *The rule for forming M-types*

$$\frac{\Delta;\Gamma \vdash A\colon \mathrm{U}_{\Delta'} \qquad \Delta;\Gamma, x\colon \mathrm{El}(A) \vdash B\colon \mathrm{U}_{\Delta'}}{\Delta;\Gamma \vdash \mathrm{M}x\colon A.B\colon \mathrm{U}_{\Delta'}}$$

can be encoded as

$$\mathrm{M}x\colon A.B \stackrel{\mathrm{def}}{=} \forall\kappa.\mathrm{Fix}^\kappa\, X.(\textstyle\sum x\colon A.(B \to \overset{\kappa}{\blacktriangleright} X)) \qquad \kappa \notin \Delta$$

Proof of Theorem 2. For readability we define

$$Y \stackrel{\mathrm{def}}{=} \mathrm{Fix}^\kappa\, X.F(\overset{\kappa}{\blacktriangleright} X)$$
$$Z \stackrel{\mathrm{def}}{=} \forall\kappa.Y$$

Thus $Y \equiv F(\blacktriangleright^\kappa Y)$, and $F(Z) \cong Z$ by the following isomorphism

$$Z \equiv \forall\kappa.F(\blacktriangleright^\kappa Y) \xrightarrow{\cong} F(\forall\kappa.\blacktriangleright^\kappa Y) \xrightarrow{F(\mathrm{force})} F(Z)$$

We show that given a coalgebra $f\colon X \to FX$ for F, the correspondence between maps $h\colon X \to Y$ and maps $\bar{h}\colon X \to Z$ of Proposition 1 restricts to a bijective correspondence between coalgebra maps $\bar{h}$

$$\begin{array}{ccc} X & \xrightarrow{f} & FX \\ {\scriptstyle \bar{h}}\downarrow & & \downarrow{\scriptstyle F(\bar{h})} \\ Z & \xrightarrow{\mathrm{unfold}_Z} & F(Z) \end{array} \tag{6}$$

in the slice over $(\Delta; I)$ and coalgebra maps h:

$$\begin{array}{ccccc} X & \xrightarrow{f} & FX & \xrightarrow{F(\mathrm{next}^\kappa)} & F(\blacktriangleright^\kappa X) \\ {\scriptstyle h}\downarrow & & & & \downarrow{\scriptstyle F(\blacktriangleright^\kappa h)} \\ Y & & \xrightarrow{\mathrm{unfold}_Y} & & F(\blacktriangleright^\kappa Y) \end{array} \tag{7}$$

in the slice over $(\Delta, \kappa; I)$ (unfold_Y is simply the identity). The theorem then follows from Y being a final coalgebra.

Suppose first that (7) commutes. Consider the map

$$\phi_X\colon \forall\kappa.F(\overset{\kappa}{\blacktriangleright} X) \to F(\forall\kappa.X)$$

defined by composing the $F(\mathrm{force})$ with the commutativity of F and $\blacktriangleright^\kappa$. The following commutes

$$\begin{array}{ccccc} \forall\kappa.X & \xrightarrow{\forall\kappa(F(\mathrm{next}^\kappa)\circ f)} & \forall\kappa.F(\blacktriangleright^\kappa X) & \xrightarrow{\phi_X} & F(\forall\kappa.X) \\ \downarrow{\scriptstyle \forall\kappa(h)} & & \downarrow{\scriptstyle \forall\kappa(F(\blacktriangleright^\kappa h))} & & \downarrow{\scriptstyle F(\forall\kappa(h))} \\ \forall\kappa.Y & \xrightarrow{\forall\kappa(\mathrm{unfold}_Y)} & \forall\kappa.F(\blacktriangleright^\kappa Y) & \xrightarrow{\phi_Y} & F(\forall\kappa.Y) \end{array}$$

Now, the top line of this diagram is isomorphic to f via the isomorphism $X \cong \forall\kappa.X$, and thus we obtain (6).

Suppose now that (6) commutes. Let $d_X\colon F(\forall\kappa.X) \to \forall\kappa.F(X)$ denote the canonical isomorphism. Note first that

$$\begin{aligned} &\mathrm{ev}_\kappa \circ {\mathrm{unfold}_Z}^{-1} \\ &\quad = \mathrm{ev}_\kappa \circ \forall\kappa({\mathrm{unfold}_Y}^{-1}) \circ d_{\blacktriangleright^\kappa Y} \circ F(\forall\kappa(\mathrm{next}^\kappa)) \\ &\quad = \mathrm{ev}_\kappa \circ \forall\kappa({\mathrm{unfold}_Y}^{-1}) \circ \forall\kappa(F(\mathrm{next}^\kappa)) \circ d_Y \\ &\quad = {\mathrm{unfold}_Y}^{-1} \circ F(\mathrm{next}^\kappa) \circ \mathrm{ev}_\kappa \circ d_Y \\ &\quad = {\mathrm{unfold}_Y}^{-1} \circ F(\mathrm{next}^\kappa) \circ F(\mathrm{ev}_\kappa) \end{aligned}$$

so $\mathrm{unfold}_Y \circ \mathrm{ev}_\kappa = F(\mathrm{next}^\kappa) \circ F(\mathrm{ev}_\kappa) \circ \mathrm{unfold}_Z$. So now,

$$\begin{aligned} \mathrm{unfold}_Y \circ h &= \mathrm{unfold}_Y \circ \mathrm{ev}_\kappa \circ \bar{h} \\ &= F(\mathrm{next}^\kappa) \circ F(\mathrm{ev}_\kappa) \circ \mathrm{unfold}_Z \circ \bar{h} \\ &= F(\mathrm{next}^\kappa) \circ F(\mathrm{ev}_\kappa) \circ F(\bar{h}) \circ f \\ &= F(\mathrm{next}^\kappa) \circ F(h) \circ f \\ &= F(\overset{\kappa}{\blacktriangleright}(h)) \circ F(\mathrm{next}^\kappa) \circ f \end{aligned}$$

□

We end the section with an example, showing how a simple non-causal stream function can be encoded in the type theory. For more examples of coding with guarded recursion and clock quantification, see [5].

Example 3. *The type $S(\mathbb{N}) \stackrel{\mathrm{def}}{=} \forall\kappa.Str^\kappa$ was seen above to be a coinductive type of streams. We now show how to encode the function $odd\colon S(\mathbb{N}) \to S(\mathbb{N})$ returning the stream of the elements at odd indices of the input stream. This function is not causal, and so there is no function $Str^\kappa \to Str^\kappa$ that does the same.*

Following [8] we first define

$$\frac{\Delta;\Gamma, x\colon A \to \overset{\kappa}{\blacktriangleright} B \vdash t\colon A \to B}{\Delta;\Gamma \vdash \mathrm{pfix}^\kappa x.t\colon A \to B}$$

as $\mathrm{pfix}^\kappa x.t = \mathrm{fix}^\kappa y\colon \blacktriangleright^\kappa(A \to B).t[\lambda a.y \circledast^\kappa \mathrm{next}^\kappa(a)/x]$. *Note that it satisfies*

$$\mathrm{pfix}^\kappa x.t = t[\mathrm{next}^\kappa \circ (\mathrm{pfix}^\kappa x.t)/x].$$

Next define

$$\kappa; f\colon S(\mathbb{N}) \to \overset{\kappa}{\blacktriangleright} Str^\kappa \vdash \mathrm{oddrec}^\kappa(f)\colon S(\mathbb{N}) \to Str^\kappa$$

as

$$\mathrm{oddrec}^\kappa(f)(x :: y :: xs) \stackrel{\mathrm{def}}{=} x ::^\kappa f(xs)$$

Here the pattern matching syntax on the left hand side uses $S(\mathbb{N}) \cong \mathbb{N} \times S(\mathbb{N})$, and the cons operation $::^\kappa$ on the right hand side is the one of Str^κ, i.e, has type $\mathbb{N} \times \blacktriangleright^\kappa Str^\kappa \to Str^\kappa$. Now, define

$$\begin{aligned} \mathrm{odd}^\kappa &\stackrel{\mathrm{def}}{=} \mathrm{pfix}^\kappa f.(\mathrm{oddrec}^\kappa(f))\colon S(\mathbb{N}) \to Str^\kappa \\ \mathrm{odd} &\stackrel{\mathrm{def}}{=} \lambda xs\colon S(\mathbb{N}).\,(\Lambda\kappa.\mathrm{odd}^\kappa(xs))\colon S(\mathbb{N}) \to S(\mathbb{N}) \end{aligned}$$

Intuitively, odd *breaks the synchronisation between the input and output stream by using different clocks for them. This allows us to essentially ignore the timing on the input stream (treating it as an element of a coinductive type), but still use guarded recursion to construct the output stream.*

To see that odd *computes as expected, unfold the definition of the cons operation on $S(\mathbb{N})$ from the proof of Theorem 2:*

$$x :: xs = \Lambda\kappa.(x ::^\kappa \mathrm{next}^\kappa(xs[\kappa]))$$

Using this we get

$$\begin{aligned}\mathrm{odd}(x :: y :: xs) &= \Lambda\kappa.\mathrm{odd}^\kappa(x :: y :: xs)\\ &= \Lambda\kappa.\mathrm{oddrec}^\kappa(\mathrm{next}^\kappa \circ \mathrm{odd}^\kappa)(x :: y :: xs)\\ &= \Lambda\kappa.(x ::^\kappa (\mathrm{next}^\kappa(\mathrm{odd}^\kappa(xs))))\\ &= x ::(\Lambda\kappa.\mathrm{odd}^\kappa(xs))\\ &= x ::(\mathrm{odd}(xs))\end{aligned}$$

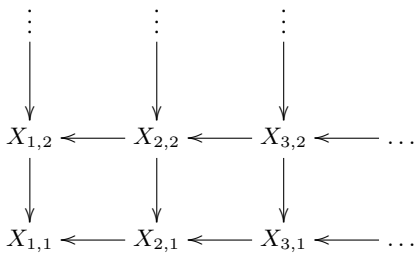
4. Model

We now construct a model of the calculus of Section 2. We start in this section, by describing a family of categories, and constructing a sketch of a model of the calculus, ignoring the coherence problem arising from interpreting dependent type theory in locally cartesian closed categories. We address this problem in Section 6.

The model is a generalisation of the topos-of-trees model of [6, 7] to more dimensions. Types and terms in clock context Δ are interpreted in the category

$$\mathrm{GR}[\Delta] \stackrel{\mathrm{def}}{=} \mathbf{Set}^{(\omega^\Delta)^{\mathrm{op}}}$$

The objects of this category are families of sets X_δ indexed by maps $\delta\colon \Delta \to \omega$ (ω are the natural numbers with the usual order), plus maps $r^X_{\delta\le\delta'}\colon X_{\delta'} \to X_\delta$ for each $\delta \le \delta'$ (pointwise order) satisfying functoriality in δ. If, for example, $\Delta = \kappa, \kappa'$ just consists of two clock variables, an object X can be drawn as a 2-dimensional lattice of sets and maps



A morphism $f\colon X \to Y$ is a family of maps $f_\delta\colon X_\delta \to Y_\delta$ such that $f_\delta \circ r^X_{\delta\le\delta'} = r^Y_{\delta\le\delta'} \circ f_{\delta'}$ for each $\delta \le \delta'$.

Since $\mathrm{GR}[\Delta]$ is a topos it is in particular locally cartesian closed, i.e., a category with pullbacks, terminal object and, for all morphisms $f\colon X \to Y$, a functor $\prod_f\colon \mathrm{GR}[\Delta]/X \to \mathrm{GR}[\Delta]/Y$, right adjoint to the functor f^* given by pullback. (Recall that the slice category $\mathrm{GR}[\Delta]/X$ has as object morphisms $Z \to X$ and commutative triangles as morphisms.) We say that $\prod_f(g)$ is the *fibred product* of g along f. Recall also that f^* always has a left adjoint $\sum_f$ given by composition with f. Since the categories $\mathrm{GR}[\Delta]$ are toposes, they moreover model W-types [16].

Locally cartesian closed categories (lccc's) almost model dependent type theory (up to the coherence problem). In our setting the lccc structure lets us model contexts $(\Delta;\Gamma)$ as objects $[\![\Delta;\Gamma]\!]$ of $\mathrm{GR}[\Delta]$, types $\Delta;\Gamma \vdash A\colon \mathrm{Type}$ as objects of the slice category $\mathrm{GR}[\Delta]/[\![\Delta;\Gamma]\!]$, i.e., as morphisms

$$p_{\Delta;\Gamma\vdash A}\colon [\![A]\!] \to [\![\Delta;\Gamma]\!]$$

and terms $\Delta;\Gamma \vdash t\colon A$ as sections of $p_{\Delta;\Gamma\vdash A}$, i.e., morphisms such that $p_{\Delta;\Gamma\vdash A} \circ [\![t]\!] = \mathrm{id}_{[\![\Delta;\Gamma]\!]}$. Dependent products are modelled using fibred products and dependent sums by composition.

4.1 Modelling guarded recursion

The operator $\blacktriangleright^\kappa$, next^κ and fix and $\circledast$ can be modelled using generalisations of the constructions of [6]. For example, in an empty context we define

$$(\overset{\kappa}{\blacktriangleright} A)(\delta) = \begin{cases} 1 & \text{if } \delta(\kappa) = 0\\ A(\delta[\kappa \mapsto (\delta(\kappa) - 1)]) & \text{else}\end{cases} \tag{8}$$

and this is extended to a family of endofunctors on slices

$$\overset{\kappa}{\underset{\Gamma}{\blacktriangleright}}\colon \mathrm{GR}[\Delta]/\Gamma \to \mathrm{GR}[\Delta]/\Gamma$$

(see [6, 7]), as needed for modelling $\blacktriangleright^\kappa$ in non-empty contexts. In each slice there is a fixed point combinator $(\blacktriangleright^\kappa_\Gamma X \to X) \to X$ giving unique fixed points.

To model quantification over clocks, we consider functors induced by inclusions of the form $i\colon \Delta \to \Delta, \kappa$ between clock contexts. Such an i induces a functor

$$\mathrm{GR}[\Delta] \xrightarrow{\ i^*\ } \mathrm{GR}[\Delta, \kappa]$$

defined on objects as $i^*(X)(\delta) = X(\delta \circ i)$. In the simple case of Δ being empty, i^* is simply the constant set functor $\mathbf{Set} \to \mathrm{GR}[\kappa]$. It is a crucial invariant of the model that the functor i^* corresponds to clock-weakening in the sense that, e.g.,

$$[\![\Delta, \kappa; \Gamma \vdash A\colon \mathrm{Type}]\!] \cong i^*[\![\Delta; \Gamma \vdash A\colon \mathrm{Type}]\!] \tag{9}$$

(in fact, this will be an equality in Lemma 12). Since, by Proposition 1 universal quantification over clocks is a right adjoint to weakening, we should model it using a right adjoint to i^*. Standard construtions from topos theory give both a right and a left adjoint to i^*.

Lemma 3. *The functor i^* has a right adjoint i_*, and a left adjoint $i_!$ defined on objects as*

$$\begin{aligned} i_*(X)(\delta) &= \varprojlim(X(\delta[\kappa \mapsto -]))\\ i_!(X)(\delta) &= X(\delta[\kappa \mapsto 1])\end{aligned}$$

The unit of the adjunction $i^* \dashv i_*$ is an isomorphism.

For example, in the case of Δ being empty, $i_*(X)$ is the limit of the sequence $X_1 \leftarrow X_2 \leftarrow X_3 \ldots$

To interpret $\forall\kappa$ in contexts, we need to extend i_* to slice categories as in the following proposition.

Proposition 2. *The functor $(i^*)^\to\colon \mathrm{GR}[\Delta]/X \to \mathrm{GR}[\Delta, \kappa]/i^*X$, mapping an object $p\colon Y \to X$ to $i^*p\colon i^*Y \to i^*X$ has a right adjoint mapping $q\colon Z \to i^*X$ to ${\eta_X}^{-1} \circ i_*q\colon i_*Z \to X$ (where η is the unit of the adjunction). Moreover, the right adjoint commutes with reindexing between slices.*

5. Universes

To model the universes U_Δ we need semantic universes in the following standard sense.

Definition 2. *Let $\mathbb{C}$ be an lccc with finite coproducts and W-types, and let $E \to U$ be a morphism in $\mathbb{C}$. Say that $B \to A$ is* small *wrt $E \to U$ if it can be presented as a pullback of $E \to U$ along some map $\overline{B}\colon A \to U$. In this case we say that $\overline{B}$ is a* code *for $B \to A$. Say X is small if $X \to 1$ is small.*

A universe *in $\mathbb{C}$ is a map $E \to U$ such that $0, 1, 1+1$ and $\mathbb{N}$ are small wrt $E \to U$ and moreover the induced notion of smallness is closed under composition, small fibered products, and W-types. The latter means that if $B \to A \to \Gamma$ are two small maps, then also the induced W-type (see [16]) is a small map into Γ.*

We will often denote a universe $E \to U$ simply by U.

If U is a Grothendieck universe in $\mathbf{Set}$, the first projection $(\coprod_{X\in U} X) \to U$ is a universe in the sense of Definition 2. A morphism $f\colon A \to B$ is small iff each fibre $f^{-1}(b)$ is isomorphic to an element of U. To construct universes in the categories $\mathrm{GR}[\Delta]$ we make the following assumption.

Assumption 1. *For the rest of the paper we will assume that we are given a Grothendieck universe U in* $\mathbf{Set}$.

Using U one can construct universes in presheaf categories such as $\mathrm{GR}[\Delta]$ via a construction going back to [12] (see also [8, 19]). The universe V_Δ is defined as the object in $\mathrm{GR}[\Delta]$, whose value at $\delta \in \omega^\Delta$ is the set of diagrams of the form

$$\{\delta' \in \omega^{\Delta'} \mid \delta' \leq \delta\}^{\mathrm{op}} \to \mathbb{U}$$

where the indexing set carries the restriction of the order from $(\omega^{\Delta'})^{\mathrm{op}}$, and $\mathbb{U}$ is the full subcategory of $\mathbf{Set}$ on the objects from U. We introduce the notation $\downarrow \delta$ for the ordered set $\{\delta' \in \omega^\Delta \mid \delta' \leq \delta\}$, thus $V_\Delta(\delta)$ is the set of objects of the functor category $\mathbb{U}^{(\downarrow\delta)^{\mathrm{op}}}$

The presheaf of elements is

$$E^V_\Delta(\delta) = \textstyle\sum_{F \in V_\Delta(\delta)} F(\delta) \tag{10}$$

with the first projection into V_Δ.

Proposition 3 ([12])**.** *Each $E^V_\Delta \to V_\Delta$ is a universe in* $\mathrm{GR}[\Delta]$ *in the sense of Definition 2. A morphism $A \to B$ is small if and only if each $A_\delta \to B_\delta$ is small.*

Remark 1. *Note that the family $(V_\Delta)_\Delta$ is not closed under reindexing, i.e., if $\Delta' \subset \Delta$, it is not the case that $i^* V_{\Delta'} \cong V_\Delta$, where $i\colon \Delta' \to \Delta$ is the inclusion. For example, in the case of $\emptyset \subset \kappa$, $i^* V_\emptyset(\delta) = U$, which is not the same as $V_\kappa(\delta)$. Since the property (9) is crucial to the model construction, this means that using a single universe in the calculus with $[\![\Delta, \kappa; - \vdash \mathrm{U} : \mathrm{Type}]\!] = V_\Delta$ is not an option. This is our semantic motivation for indexing universes by clock contexts.*

In the calculus, in clock context Δ, there is a universe type $\mathrm{U}_{\Delta'}$ for each $\Delta' \subseteq \Delta$. This will be modelled as $i^* V_{\Delta'}$, where $i\colon \Delta' \to \Delta$ is the inclusion. To model closure of $\mathrm{U}_{\Delta'}$ under type forming operations, we need to know that $i^* V_{\Delta'}$ is a universe. This follows from the following lemma.

Lemma 4. *Let $\mathbb{C}$ be a locally cartesian closed category with W-types and finite coproducts, and let $\mathbb{D}$ be a small category with a terminal object, such that the functor category $\mathbb{C}^{\mathbb{D}}$ is locally cartesian closed. Let $E \to U$ be a universe in $\mathbb{C}$, and let $D\colon \mathbb{C} \to \mathbb{C}^{\mathbb{D}}$ be the diagonal functor $D(C)(d) = C$. Then $D(E) \to D(U)$ is a universe in $\mathbb{C}^{\mathbb{D}}$.*

Note that this proves that $i^* V_{\Delta'}$ is a universe, since up to the isomorphism $\mathrm{GR}[\Delta] \cong \mathrm{GR}[\Delta']^{(\omega^{\Delta''})^{\mathrm{op}}}$ (where $\Delta = \Delta', \Delta''$), i^* is the diagonal functor.

For reasons of space we omit the proof of Lemma 4, but just state an important lemma on which the proof is based.

Lemma 5. *Under the same assumptions as Lemma 4, a map $f\colon Y \to X$ in $\mathbb{C}^{\mathbb{D}}$ is small with respect to $D(E) \to D(U)$ if and only if f_1 is small with respect to $E \to U$ and for each d in $\mathbb{D}$ the naturality square below is a pullback diagram*

$$\begin{array}{ccc} Y(d) & \longrightarrow & Y(1) \\ \downarrow & & \downarrow \\ X(d) & \longrightarrow & X(1) \end{array}$$

6. A category subsuming all $\mathrm{GR}[\Delta]$

We now turn to the coherence problem of interpretation of dependent type theory in locally cartesian closed categories, i.e., the problem that pullback, which models substitution, usually does not commute with constructions such as dependent products and is not associative [11]. Here we follow a recent approach developed independently by Voevodsky [13] and Streicher [20] using universes, see also [8, 19]. Voevodsky and Streicher show that given a universe $E \to U$ in a locally cartesian closed category, one can construct a model of dependent type theory by modelling contexts as objects and types $\Gamma \vdash A\colon \mathrm{Type}$ as morphisms $[\![\Gamma]\!] \to U$.

In our setting we have a whole family of locally cartesian closed categories $\mathrm{GR}[\Delta]$ (indexed over Δ) for which we need to *simultaneously* solve the coherence problem, and at the same time maintain the invariant that $[\![\Delta, \kappa; \Gamma]\!] \cong i^*[\![\Delta; \Gamma]\!]$. To do this we introduce a new category

$$\mathrm{GR} \stackrel{\mathrm{def}}{=} \mathbf{Set}^{(\omega^{\mathrm{CV}})^{\mathrm{op}}}$$

Definition 3. *An object X of* GR *is* supported *by Δ if whenever $\delta \leq \delta'$ and $\delta|_\Delta = \delta'|_\Delta$ (where $\delta|_\Delta$ is the restriction of δ to Δ), then $X_{\delta'} \to X_\delta$ is an isomorphism. An object X is* strictly supported *by Δ, if, under the conditions above, $X_{\delta'} \to X_\delta$ is an identity.*

Each of the categories $\mathrm{GR}[\Delta]$ is isomorphic to the full subcategory of GR consisting of the objects strictly supported by Δ and is equivalent to the category $\overline{\mathrm{GR}[\Delta]}$ of objects supported by Δ. Note that up to the equivalence $\overline{\mathrm{GR}[\Delta]} \simeq \mathrm{GR}[\Delta]$ the functors $i^*\colon \overline{\mathrm{GR}[\Delta]} \to \overline{\mathrm{GR}[\Delta, \kappa]}$ are simply inclusions of subcategories of GR.

6.1 Extending operations to GR

Lemma 6. *The embeddings $\mathrm{GR}[\Delta] \to \mathrm{GR}$ preserve the locally cartesian closed structure and W-types, likewise do $\mathrm{GR}[\Delta'] \to \mathrm{GR}[\Delta]$ for $\Delta' \subseteq \Delta$*

The proof of Lemma 6 is similar to that of Lemma 4.

Also the operations $\blacktriangleright^\kappa$ and i_* can be extended to GR. To state the properties of i_*, define the subcategory $\mathrm{GR}[\mathrm{CV} \setminus \kappa]$ of GR on objects supported by $\mathrm{CV} \setminus \kappa$.

Proposition 4. *For any κ there are functors*

$$\overset{\kappa}{\blacktriangleright}\colon \mathrm{GR} \to \mathrm{GR}$$
$$\mathrm{Lim}_\kappa\colon \mathrm{GR} \to \mathrm{GR}[\mathrm{CV} \setminus \kappa]$$

such that Lim_κ is right adjoint to the inclusion, and such that the left square below commutes and the right is a map of adjunctions whenever $\kappa \notin \Delta$.

$$\begin{array}{ccc} \mathrm{GR}[\Delta, \kappa] & \xrightarrow{\blacktriangleright^\kappa} & \mathrm{GR}[\Delta, \kappa] \\ \downarrow & & \downarrow \\ \mathrm{GR} & \xrightarrow{\blacktriangleright^\kappa} & \mathrm{GR} \end{array} \qquad \begin{array}{ccc} \mathrm{GR}[\Delta, \kappa] & \underset{i_*}{\overset{i^*}{\underset{\longrightarrow}{\overset{\longleftarrow}{\bot}}}} & \mathrm{GR}[\Delta] \\ \downarrow & & \downarrow \\ \mathrm{GR} & \underset{\mathrm{Lim}_\kappa}{\overset{\bot}{\underset{\longrightarrow}{\overset{\longleftarrow}{}}}} & \mathrm{GR}[\mathrm{CV} \setminus \kappa] \end{array}$$

Moreover, the unit of the extended adjunction is still an isomorphism, and if X is in $\mathrm{GR}[\mathrm{CV} \setminus \kappa]$ then the functor

$$\mathrm{GR}/X \to \mathrm{GR}[\mathrm{CV} \setminus \kappa]/X$$

mapping $p_Z\colon Z \to X$ to $\eta_X^{-1} \circ \mathrm{Lim}_\kappa p_Z$ is right adjoint to the inclusion. If X is in $\mathrm{GR}[\Delta]$ then this adjunction extends the one of Proposition 2.

6.2 Universes in GR

The model construction uses universes for two purposes: to solve the coherence problem as indicated above and to model the universes of the calculus. We need different set theoretic universes for each purpose and so we strengthen Assumption 1.

Assumption 2. *For the rest of the paper we will assume that we are given two Grothendieck universe U, U' in* $\mathbf{Set}$*, such that $U \subseteq U'$,*

$U \in U'$. *Moreover, we will assume that* U' *has unique names, i.e., if* $X, Y \in U'$ *and* $X \cong Y$ *then* $X = Y$.

Using these universes we can construct the universes V and V_Δ (for each Δ) in GR:

$$V(\delta) = \{X \colon (\downarrow \delta)^{\mathrm{op}} \to \mathbb{U}'\}$$
$$V_\Delta(\delta) = \{X \colon (\downarrow \delta)^{\mathrm{op}} \to \mathbb{U} \mid X \text{ strictly supported by } \Delta\}$$

where $\mathbb{U}'$ is the full subcategory of $\mathbf{Set}$ on objects of U', X as usual is required to be a functor, $\downarrow \delta$ is defined as in Section 5, and X strictly supported by Δ should be understood as in Definition 3. The projections

$$\mathrm{El}_V \colon E_V \to V \qquad \mathrm{El}_\Delta \colon E_\Delta \to V_\Delta$$

are defined as in (10). Note that $\mathrm{El}_\Delta \colon E_\Delta \to V_\Delta$ is isomorphic to the universe obtained by including that of Proposition 3 into GR.

Proposition 5. *Both* $E_V \to V$ *and* $E_\Delta \to V_\Delta$ *are universes in* GR.

Proof. That V is a universe is proved in [12]. Since $\mathrm{GR} \cong \mathrm{GR}[\Delta]^{\mathbb{D}^{\mathrm{op}}}$ (where $\mathbb{D} = \omega^{\mathrm{CV}\setminus\Delta}$), Lemma 4 shows that the universe V_Δ of $\mathrm{GR}[\Delta]$ is also a universe in GR. □

Proposition 6. *The object* V_Δ *is an object of* $\overline{\mathrm{GR}[\Delta]}$.

We will use the universe V for the coherent model, i.e., type judgements will be interpreted as morphisms with V as codomain, and V_Δ will be used to interpret U_Δ.

The requirement of uniqueness of names in Assumption 2 corresponds to the following category theoretic property.

Definition 4. *Let* $E \to U$ *be a universe in a category* $\mathbb{C}$. *We say that* $E \to U$ has unique names *if whenever* $f, g \colon \Gamma \to U$ *are codes for the same map* $A \to \Gamma$ *it is the case that* $f = g$.

Proposition 7. *Both* $E_V \to V$ *and* $E_\Delta \to V_\Delta$ *have unique names.*

Lemma 7. *Each* V_Δ *is small with respect to* V. *We write* $\overline{V}_\Delta \colon 1 \to V$ *for the code.*

7. Universe structures

It is essential to the coherent interpretation that type constructors can be encoded as morphisms on the universe V. For example, for binary product types we need a map $\overline{\times} \colon V \times V \to V$ such that if $\overline{A}, \overline{B} \colon X \to V$ are codes for A, B in the slice over X then $\overline{\times} \circ \langle \overline{A}, \overline{B} \rangle$ is a code for the product of A and B as computed in the slice category. In the cases of dependent products and sums the domains of the codes of these operations is the object $V^{(1)}$ defined as the exponential

$$(V \times V \to V)^{\mathrm{El}_V}$$

in the category $\mathbb{C}/V$. This object is essentially the interpretation of the $\sum X \colon V . V^X$ and enjoys the universal property that morphisms $X \to V^{(1)}$ correspond naturally to pairs $\langle \overline{B}, \overline{A} \rangle$ where $\overline{B} \colon X \to V$ and $\overline{A} \colon B \to V$ where B is the pullback of E_V along $\overline{B}$.

Lemma 8 ([13, 19])**.** *There are unique maps* $\prod, \sum, \mathrm{W} \colon V^{(1)} \to V$ *encoding dependent products, sums and W-types in the sense that if* $p_A \colon A \to B$ *and* $p_B \colon B \to X$ *are small with codes* $\overline{B}$ *and* $\overline{A}$ *respectively, then* $\prod \circ \langle \overline{B}, \overline{A} \rangle$, $\sum \circ \langle \overline{B}, \overline{A} \rangle$, *and* $\mathrm{W} \circ \langle \overline{B}, \overline{A} \rangle$ *are codes of* $\prod_{p_B} p_A$, $p_B \circ p_A$, *and the W-type induced by* p_A, p_B *respectively.*

The uniqueness statement of Lemma 8 follows from Proposition 7. Likewise there are unique maps encoding binary products and sums, as well as $0, 1, 1 + 1$.

Definition 5. *A* universe structure *is a universe together with a choice of codes encoding dependent products, sums, W-types, unit,* 0, $1 + 1$. *A* universe embedding *is a monomorphism* $i \colon U \to U'$ *between universes such that* i *is a code for* $E \to U$, *and which commutes with the universe structure.*

Proposition 8. *The inclusions* $\mathrm{in}_\Delta \colon V_\Delta \to V$ *and* $\mathrm{in}_{\Delta',\Delta} \colon V_{\Delta'} \to V_\Delta$ *are universe embeddings.*

Proof. This follows from [19, Remark 9.5]. □

Remark 2. *As mentioned, types in context will be modelled as morphisms into* V, *and* V_Δ *will be used to model* U_Δ. *The universe structure on* V *will be used to model the type constructors and the universe structure on* V_Δ *will be used to model the corresponding terms acting on the universe type. Proposition 8 ensures that the type equality of Figure 5 hold.*

Without the assumption of uniqueness of names in the set theoretic universes of Assumption 2 it would only be possible to model these type equalities as isomorphisms. In fact, without this assumption the canonical choice of encoding of function space as a universe map, does not commute with the universe inclusions.

7.1 Encoding operations for guarded recursion

We now show how to construct codes for the operations specific to our calculus.

Lemma 9. *There is a map* $\triangleright^\kappa \colon \blacktriangleright^\kappa V \to V$ *encoding* $\blacktriangleright^\kappa$ *in the sense that if* Γ *is an object of* GR *and* $\overline{B} \colon \Gamma \to V$ *is a code for* $B \to \Gamma$ *then* $\triangleright^\kappa \circ \mathrm{next}^\kappa \circ \overline{B}$ *is a code for* $\blacktriangleright^\kappa_\Gamma B \to \Gamma$.

The next lemma uses the notation $\widehat{f} \colon X \to \mathrm{Lim}_\kappa Y$ for the adjoint correspondent to $f \colon X \to Y$ under the adjunction of Proposition 4 whenever X in $\mathrm{GR}[\mathrm{CV} \setminus \kappa]$.

Lemma 10. *There is a map* $\overline{\forall\kappa} \colon \mathrm{Lim}_\kappa(V) \to V$ *encoding the functor of Proposition 2 for all* Δ *in the sense that if* Γ *is in* $\mathrm{GR}[\mathrm{CV} \setminus \kappa]$ *and if* $p_B \colon B \to \Gamma$ *is an object of* GR/Γ *and* $\overline{B} \colon \Gamma \to V$ *is a code for* p_B *then* $\overline{\forall\kappa} \circ \widehat{\overline{B}}$ *is a code for* $\eta^{-1} \circ i_* p_B$.

Proof. $\mathrm{Lim}_\kappa V(\delta)$ is the set of diagrams of the form

$$F \colon \{\delta' \in \omega^{\mathrm{CV}} \mid \delta'|_{\mathrm{CV}\setminus\kappa} \leq \delta|_{\mathrm{CV}\setminus\kappa}\}^{\mathrm{op}} \to \mathbb{U}'.$$

Define $\overline{\forall\kappa}(F)(\delta')$ to be the limit of $F(\delta'[\kappa \mapsto -])$. □

Both these maps restrict to V_Δ such that the following diagrams commute.

$$\begin{array}{ccc} \mathrm{Lim}_\kappa V_\Delta & \xrightarrow{\overline{\forall\kappa}} & V_\Delta \\ {\scriptstyle \mathrm{Lim}_\kappa(\mathrm{in}_\Delta)}\downarrow & & \downarrow{\scriptstyle \mathrm{in}_\Delta} \\ \mathrm{Lim}_\kappa(V) & \xrightarrow{\overline{\forall\kappa}} & V \end{array} \qquad \begin{array}{ccc} \blacktriangleright^\kappa V_\Delta & \xrightarrow{\triangleright^\kappa} & V_\Delta \\ {\scriptstyle \blacktriangleright^\kappa(\mathrm{in}_\Delta)}\downarrow & & \downarrow{\scriptstyle \mathrm{in}_\Delta} \\ \blacktriangleright^\kappa V & \xrightarrow{\triangleright^\kappa} & V \end{array}$$

8. Interpreting the calculus

We have now established the ingredients and can define the interpretation. The basic principles of the interpretation are

- Contexts $\Delta; \Gamma$ are interpreted as objects of GR
- Type judgements in context $\Delta; \Gamma \vdash A \colon \mathrm{Type}$ are interpreted as morphisms $[\![A]\!] \colon [\![\Delta; \Gamma]\!] \to V$ in GR
- The empty context is interpreted as the terminal object, and comprehension, i.e., the rule

$$\frac{\Delta; \Gamma \vdash \qquad \Delta; \Gamma \vdash A \colon \mathrm{Type}}{\Delta; \Gamma, x \colon A \vdash}$$

$$
\begin{aligned}
[\![\Delta;\Gamma \vdash X\colon \mathrm{Type}]\!] &= \overline{X}\circ ! \\
[\![\Delta;\Gamma \vdash A \times B\colon \mathrm{Type}]\!] &= \overline{\times} \circ \langle [\![A]\!], [\![B]\!]\rangle \\
[\![\Delta;\Gamma \vdash A + B\colon \mathrm{Type}]\!] &= \overline{+} \circ \langle [\![A]\!], [\![B]\!]\rangle \\
[\![\Delta;\Gamma \vdash \textstyle\prod x\colon A.B\colon \mathrm{Type}]\!] &= \prod \circ \langle [\![A]\!], [\![B]\!]\rangle \\
[\![\Delta;\Gamma \vdash \textstyle\sum x\colon A.B\colon \mathrm{Type}]\!] &= \sum \circ \langle [\![A]\!], [\![B]\!]\rangle \\
[\![\Delta;\Gamma \vdash \mathrm{W}x\colon A.B\colon \mathrm{Type}]\!] &= \mathrm{W} \circ \langle [\![A]\!], [\![B]\!]\rangle \\
[\![\Delta;\Gamma \vdash \overset{\kappa}{\blacktriangleright} A\colon \mathrm{Type}]\!] &= \overset{\kappa}{\triangleright} \circ \mathrm{next}^{\kappa} \circ [\![A]\!] \\
[\![\Delta;\Gamma \vdash \forall\kappa.A\colon \mathrm{Type}]\!] &= \overline{\forall\kappa} \circ \widehat{[\![A]\!]} \\
[\![\Delta;\Gamma \vdash \mathrm{U}_{\Delta'}\colon \mathrm{Type}]\!] &= \overline{V}_{\Delta'} \circ !
\end{aligned}
$$

Figure 6. Interpretation of types. In the first line X ranges over $0, 1, 2$, and ! is the unique map to the terminal object

is interpreted using pullback:

$$
\begin{array}{ccc}
[\![\Delta;\Gamma, x\colon A]\!] & \longrightarrow & \mathrm{El}_V \\
{\scriptstyle p_{\Delta;\Gamma,x\colon A}}\downarrow & \lrcorner & \downarrow \\
[\![\Delta;\Gamma]\!] & \xrightarrow{[\![A]\!]} & V
\end{array}
$$

- Term judgements $\Delta;\Gamma \vdash t\colon A$ are interpreted as sections of $p_{\Delta;\Gamma,x\colon A}$ in the sense of Section 4.

Figure 6 summarizes the interpretation of types using the universe structure on V constructed in Section 7.

Theorem 3 (Soundness)**.** *Under Assumption 2, the interpretation of types and contexts defined above can be extended to an interpretation of the type theory which is well-defined and sound with respect to the equality theory of Section 2. Moreover, all the terms of Figure 4 and (5) are interpreted as isomorphisms.*

For reasons of space, we omit the interpretation of terms, most of which is standard. For example, the rules for forming terms in the universe of Figure 5 are interpreted using the universe structure on V_Δ. The interpretation of a term $[\![\Delta;\Gamma \vdash t\colon \mathrm{U}_{\Delta'}]\!]$ corresponds to a map $[\![t]\!]\colon [\![\Delta;\Gamma]\!] \to V_{\Delta'}$ and $\mathrm{El}(t)$ is interpreted as $\mathrm{in}_{\Delta'} \circ [\![t]\!]\colon [\![\Delta;\Gamma]\!] \to V$. The type equalities of Figure 5 follow from the map $\mathrm{in}_{\Delta'}$ preserving the universe structure and $\triangleright^{\kappa}, \overline{\forall}\kappa$.

We end by stating the following two lemmas which are crucial to the interpretation.

Lemma 11. *If $\Delta;\Gamma$ is a well-formed context then $[\![\Delta;\Gamma]\!]$ is an object of $\overline{\mathrm{GR}[\Delta]}$.*

Lemma 12. *Let $\kappa \notin \Delta$, then*

- *if $\Delta;\Gamma$ is a well-formed context then $[\![\Delta;\Gamma]\!] = [\![\Delta,\kappa;\Gamma]\!]$*
- *If $\Delta;\Gamma \vdash A\colon \mathrm{Type}$ then*

$$[\![\Delta;\Gamma \vdash A\colon \mathrm{Type}]\!] = [\![\Delta,\kappa;\Gamma \vdash A\colon \mathrm{Type}]\!]$$

- *If $\Delta;\Gamma \vdash t\colon A$ then $[\![\Delta;\Gamma \vdash t\colon A]\!] = [\![\Delta,\kappa;\Gamma \vdash t\colon A]\!]$*

9. Conclusions and future work

Guarded recursive types are a powerful tool, whose properties are still to be fully understood. This paper not only shows how they can be used for productive coprogramming with coinductive types in a dependently typed setting, but may also be useful for development of reasoning principles for guarded recursive types.

For future work, I would like to extend the present work to the case of intensional type theory, building on the results of [8] which show how to construct intensional models of guarded recursion. Note that the model construction used in this paper constructs locally cartesian closed categories which are all models of extensional type theory. In the setting of intensional type theory it would be interesting to see if the constructions of this paper are compatible with other principles of type theory, in particular the univalence axiom.

Acknowledgment

I would like to thank Bob Atkey and Lars Birkedal for helpful discussions.

References

[1] A. Abel. Type-based termination, inflationary fixed-points, and mixed inductive-coinductive types. In *FICS*, volume 77 of *EPTCS*, pages 1–11, 2012.

[2] T. Altenkirch and P. Morris. Indexed containers. In *LICS*, pages 277–285. IEEE Computer Society, 2009.

[3] A. W. Appel and D. McAllester. An indexed model of recursive types for foundational proof-carrying code. *ACM Transactions on Programming Languages and Systems*, 23(5):657–683, September 2001.

[4] 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*, 2007.

[5] R. Atkey and C. McBride. Productive coprogramming with guarded recursion. In *ICFP*, pages 197–208. ACM, 2013.

[6] L. Birkedal, R. Møgelberg, J. Schwinghammer, and K. Støvring. First steps in synthetic guarded domain theory: step-indexing in the topos of trees. In *Proceedings of LICS*, 2011.

[7] L. Birkedal, R. Møgelberg, J. Schwinghammer, and K. Støvring. First steps in synthetic guarded domain theory: step-indexing in the topos of trees. *Logical Methods in Computer Science*, 8(4), October 2012.

[8] L. Birkedal and R.E. Møgelberg. Intensional type theory with guarded recursive types qua fixed points on universes. In *LICS*, pages 213–222. IEEE Computer Society, 2013.

[9] N.A. Danielsson. Beating the productivity checker using embedded languages. In *Proceedings of PAR*, Electronic Proceedings of Theoretical Computer Science, 2010.

[10] N. Ghani, P. Hancock, and D. Pattinson. Representations of stream processors using nested fixed points. *Logical Methods in Computer Science*, 5(3), 2009.

[11] M. Hofmann. On the interpretation of type theory in locally cartesian closed categories. In *Proc. of CSL*, 1994.

[12] M. Hofmann and T. Streicher. Lifting grothendieck universes. Unpublished manuscript.

[13] C. Kapulkin, P.L. Lumsdaine, and V. Voevodsky. The simplicial model of univalent foundations. *arXiv*, 1211.2851, 2012.

[14] N.R. Krishnaswami and N. Benton. Ultrametric semantics of reactive programs. In *Proceedings LICS*, 2011.

[15] C. McBride and R. Paterson. Applicative programming with effects. *Journal of Functional Programming*, 18(1), 2008.

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

[17] H. Nakano. A modality for recursion. In *Proc. of LICS*, 2000.

[18] J. L. Sacchini. Type-based productivity of stream definitions in the calculus of constructions. In *LICS*, pages 233–242. IEEE Computer Society, 2013.

[19] M. Shulman. Univalence for inverse diagrams and homotopy canonicity. *arxiv*, 1203.3253, 2013.

[20] T. Streicher. A model of type theory in simplicial sets. Unpublished, 2011.

A. Proof of Lemma 5

Suppose first that f is small, and consider the following commutative diagram where the right hand square is part of the code for f

$$\begin{array}{ccccc} Y(d) & \longrightarrow & Y(1) & \longrightarrow & E \\ \downarrow & & \downarrow & & \downarrow \\ X(d) & \longrightarrow & X(1) & \longrightarrow & U \end{array} \tag{11}$$

Note that the outer square must also be part of the code for f. Since pullbacks in $\mathbb{C}^{\mathbb{D}}$ are computed pointwise, both the outer square and the right square are pullbacks and so by the pullback lemma (see e.g. Lemma 5.8 in Awodey: Category theory 2nd ed) also the left hand square is a pullback.

For the other implication, consider again the square (11), where this time, the right hand square is a code for f_1 as assumed to exist. We can now construct the code for f by defining the component at d to be the composite $Y(d) \to Y(1) \to U$. This defines a code because the composite of the two pullbacks of (11) is a pullback, again by the pullback lemma. □

Remark 3. *Note that by a further application of the pullback lemma, every diagram of the form*

$$\begin{array}{ccc} Y(d') & \longrightarrow & Y(d) \\ \downarrow & & \downarrow \\ X(d') & \longrightarrow & X(d) \end{array}$$

is a pullback

B. Proof of Lemma 4

Since coproducts are given pointwise in $\mathbb{C}^{\mathbb{D}}$, it should be clear from Lemma 5 that the notion of smallness induced by $D(E) \to D(U)$ is closed under these. Likewise, since composition is given pointwise, it is closed under this too, and since all identity maps are small, 1 is small. It remains to verify that the notion of smallness is closed under small fibered products and formation of W types.

We first show that the notion of smallness is closed under fibered products. Note first that for X in $\mathbb{C}^{\mathbb{D}}$, there is a functor

$$\phi_X : \mathbb{C}/X(1) \to \mathbb{C}^{\mathbb{D}}/X$$

mapping $Z \to X(1)$ to the natural transformation whose component $Z(d) \to X(d)$ at d is the pullback of $Z \to X(1)$ along $X(d) \to X(1)$. By Lemma 5 a natural transformation $p : Y \to X$ is small wrt $D(E) \to D(U)$ if and only if there is a small q such that $p \cong \phi_X(q)$.

Suppose $f : Y \to X$ is small, and suppose we can prove the below diagram commutative (up to isomorphism).

$$\begin{array}{ccc} \mathbb{C}/Y(1) & \xrightarrow{\phi_Y} & \mathbb{C}^{\mathbb{D}}/Y \\ {\scriptstyle \Pi_{f_1}}\downarrow & & \downarrow{\scriptstyle \Pi_f} \\ \mathbb{C}/X(1) & \xrightarrow{\phi_X} & \mathbb{C}^{\mathbb{D}}/X \end{array} \tag{12}$$

where the vertical maps are the fibered product functors along f and f_1 respectively. We need to show that if $p_Z : Z \to Y$ is small, then so is $\prod_f(p_Z)$. But by the above observation and (12), if p_Z is small, then $\prod_f(p_Z) \cong \phi_X(\prod_{f_1}(q))$ for some small q and thus small.

It thus remains to show (12) commutative. Since ϕ_Y has a left adjoint ψ_Y, mapping p to p_1, and moreover $f^* \dashv \prod_f$, this is equivalent to showing the following diagram commutative.

$$\begin{array}{ccc} \mathbb{C}/Y(1) & \xleftarrow[\psi_Y]{} & \mathbb{C}^{\mathbb{D}}/Y \\ {\scriptstyle f_1^*}\uparrow & & \uparrow{\scriptstyle f^*} \\ \mathbb{C}/X(1) & \xleftarrow[\psi_X]{} & \mathbb{C}^{\mathbb{D}}/X \end{array}$$

This follows from pullbacks in $\mathbb{C}^{\mathbb{D}}$ being computed pointwise.

In fact, this argument shows that small fibered products of small maps are computed pointwise: if $f : Y \to X$ is small and $p_Z : Z \to Y$ is small then

$$\begin{aligned} (\textstyle\prod_f(p_Z))_d &\cong X(!)^*(\textstyle\prod_{f_1}(p_{Z(1)})) && \text{by (12)} \\ &\cong \textstyle\prod_{f_d}(Y(!)^* p_{Z(1)}) \\ &\cong \textstyle\prod_{f_d}(p_{Z(d)}) && \text{since } p_Z \text{ is small} \end{aligned}$$

where the second isomorphism is an instance of the Beck-Chevalley condition applied to the pullback square

$$\begin{array}{ccc} Y(d) & \xrightarrow{Y(!)} & Y(1) \\ {\scriptstyle f_d}\downarrow & & \downarrow{\scriptstyle f_1} \\ X(d) & \xrightarrow{X(!)} & X(1) \end{array}$$

which is a pullback because f is assumed small.

We now show that the notion of smallness is closed under formation of W-types. Recall first, that if

$$B \xrightarrow{p_B} A \xrightarrow{p_A} \Gamma \tag{13}$$

is a pair of maps in some lccc $\mathbb{E}$, then the W-type induced by p_B, p_A is an initial algebra of the functor

$$\mathbb{E}/\Gamma \xrightarrow{(p_A \circ p_B)^*} \mathbb{E}/B \xrightarrow{\prod_{p_B}} \mathbb{E}/A \xrightarrow{\sum_{p_A}} \mathbb{E}/\Gamma$$

Now suppose, (13) is a diagram in $\mathbb{C}^{\mathbb{D}}$, and consider the endofunctor F on $\mathbb{C}^{\mathbb{D}}/\Gamma$ induced by it. Consider on the other hand the endofunctor F_d on $\mathbb{C}/\Gamma(d)$ induced by the component over some object d in $\mathbb{D}$. Since pullbacks and dependent products and sums are given pointwise, F is given pointwise by F_d, in the sense that if $p_C : C \to \Gamma$, then

$$F(p_C)_d = F_d(p_{C(d)})$$

Let $p_{X(d)} : X(d) \to \Gamma(d)$ be the carrier of the initial algebra for F_d for all d. Note that each of these is small wrt $E \to U$. Since W-types commute with pullbacks (Hyland and Gambino: Wellfounded trees and dependent polynomial functors, Corollary 11), and since p_A and p_B are both small, it is the case that for all $f : d' \to d$ in $\mathbb{D}$, $X(d') \cong \Gamma(f)^* X(d)$, and so in fact, the $p_{X(d)}$ together constitute a small object $p_X : X \to \Gamma$ in $\mathbb{C}^{\mathbb{D}}$.

We will show that $p_{X(d)}$ is a carrier of an initial algebra for F, which will prove the lemma. First we show that the family of maps $F_d(X(d)) \to X(d)$ define a map $F(X) \to X$ in $\mathbb{C}^{\mathbb{D}}$. The fact that W-types commute with reindexing means not only that $X(d') \cong (\Gamma(f))^* X(d)$ but also that the diagram below commutes

$$\begin{array}{ccccc} F_{d'}(X(d')) & \xrightarrow{F(\cong)} & F_{d'}((\Gamma(f))^* X(d)) & \xrightarrow{\cong} & (\Gamma(f))^*(F_d(X(d))) \\ \downarrow & & \downarrow & \swarrow & \\ X(d') & \xrightarrow{\cong} & (\Gamma(f))^* X(d) & & \end{array}$$

composing this with

$$\begin{array}{ccc} (\Gamma(f))^*(F_d(X(d))) & \longrightarrow & F_d(X(d)) \\ \downarrow & & \downarrow \\ (\Gamma(f))^* X(d) & \longrightarrow & (X(d)) \end{array}$$

we get that the family $F_d(X(d)) \to X(d)$ is natural, and thus a map in $\mathbb{C}^{\mathbb{D}}$.

Suppose now, given some other algebra $F(Z) \to Z$ in $\mathbb{C}^{\mathbb{D}}/\Gamma$. Since each $X(d)$ is an initial algebra, there is a unique family of maps h_d such that the diagrams of the form

$$\begin{array}{ccc} F_d(X(d)) & \longrightarrow & X(d) \\ {\scriptstyle F_d(h_d)}\downarrow & & \downarrow{\scriptstyle h_d} \\ F_d(Z(d)) & \longrightarrow & Z(d) \end{array}$$

It remains to show that the family $(h_d)_d$ is natural, since this will show existence of an algebra map, and uniqueness of algebra maps is clear.

But to show commutativity of

$$\begin{array}{ccc} X(d) & \longrightarrow & X(d') \\ {\scriptstyle h_d}\downarrow & & \downarrow{\scriptstyle h_{d'}} \\ Z(d) & \longrightarrow & Z(d') \end{array}$$

is equivalent to showing commutativity of

$$\begin{array}{ccc} X(d) & \longrightarrow & (\Gamma(f))^* X(d') \\ {\scriptstyle h_d}\downarrow & & \downarrow{\scriptstyle (\Gamma(f))^*(h_{d'})} \\ Z(d) & \longrightarrow & (\Gamma(f))^* Z(d') \end{array}$$

Since F commutes with reindexing, this is an equality between two algebra maps for $F(d)$ from $X(d)$ to $(\Gamma(f))^* Z(d')$, and thus commutes, since $X(d)$ is the initial algebra. □

C. The map (5) is an isomorphism

We now show the following, which is one of the statements of Theorem 3.

Proposition 9. *The map (5) is interpreted as an isomorphism in the model.*

Write $\mathbb{C} = \mathrm{GR}[\Delta]/\Gamma$. In context Δ, $F(-,-)$ is interpreted as a functor

$$F_\Delta(-,-) \colon \mathbb{C} \times \mathbb{C} \to \mathbb{C}$$

Note the isomorphism

$$\mathrm{GR}[\Delta,\kappa]/i^*\Gamma \cong (\mathrm{GR}[\Delta])^{\omega^{\mathrm{op}}}/i^*\Gamma \cong (\mathrm{GR}[\Delta]/\Gamma)^{\omega^{\mathrm{op}}} = \mathbb{C}^{\omega^{\mathrm{op}}}$$

and so in context Δ, κ, $F(-,-)$ is interpreted as a functor

$$F_{\Delta,\kappa}(-,-) \colon \mathbb{C}^{\omega^{\mathrm{op}}} \times \mathbb{C}^{\omega^{\mathrm{op}}} \to \mathbb{C}^{\omega^{\mathrm{op}}}$$

Since $i^* \colon \mathbb{C} \to \mathbb{C}^{\omega^{\mathrm{op}}}$ preserves the lccc structure (Lemma 6, see also Appendix B above for an argument), $F_{\Delta,\kappa}$ is computed pointwise from F_Δ, i.e., if X, Y in $\mathbb{C}^{\omega^{\mathrm{op}}}$ and $n \leq m$, we get

$$F_{\Delta,\kappa}(X,Y)_n \cong F_\Delta(X_n, Y_n)$$

and up to this isomorphism

$$\begin{aligned} F_{\Delta,\kappa}(X,Y)_{n,m} &= F_{\Delta,\kappa}(X_{n,m}, Y_{n,m}) \\ &\quad : F_{\Delta,\kappa}(X,Y)_m \to F_{\Delta,\kappa}(X,Y)_n \end{aligned}$$

Now, given an object $X_1 \leftarrow X_2 \leftarrow X_3 \ldots$ in $\mathbb{C}^{\omega^{\mathrm{op}}}$, we must show that

$$\mu Y.F_\Delta(\varprojlim_n X_n, Y) \cong \varprojlim_n \mu Y.F_\Delta(X_n, Y)$$

To do this, we use the fact that $F_\Delta(X_n, -)$, being a polynomial functor on a slice category has rank, and so the initial algebra can be computed as a colimit. (See e.g., Abbott, Altenkirch and Ghani, Categories of Containers Theorem 5.6 and Proposition 6.6). Precisely, there is an ordinal λ such that $\mu Y.F_\Delta(X_n, Y)$ can be constructed as the colimit of a diagram $(F^\alpha_\Delta(X_n, Y))_{\alpha\in\lambda}$ where $F^\alpha_\Delta(X_n, Y)$ is defined by induction as follows

$$\begin{aligned} F^0_\Delta(X_n, 0) &= 0 \\ F^{\alpha+1}_\Delta(X_n, 0) &= F_\Delta(X_n, F^\alpha_\Delta(X_n, 0)) \\ F^\alpha_\Delta(X_n, 0) &= \varinjlim_{\beta<\alpha} F^\beta_\Delta(X_n, 0) \end{aligned}$$

where in the last case α is a limit ordinal. For $\alpha \leq \beta$, the map

$$\xi_{\alpha,\beta} \colon F^\alpha_\Delta(X_n, Y) \to F^\beta_\Delta(X_n, Y)$$

is defined by cases:

$$\begin{aligned} \xi_{0,\beta} &= ! \\ \xi_{\alpha+1,\alpha+2} &= F_\Delta(X_n, \xi_{\alpha,\alpha+1}) \\ \xi_{\alpha,\beta} &= \mathrm{in}_{\alpha,\beta} \qquad\qquad \beta \text{ limit ordinal} \end{aligned}$$

where $\mathrm{in}_{\alpha,\beta}$ is the inclusion into the colimit. The remaining case is that of $\xi_{\alpha,\alpha+1}$ for α a limit ordinal, which is defined as the unique map making the below diagram commute for all $\beta < \alpha$

$$\begin{array}{ccc} F^\beta_\Delta(X_n, 0) & \xrightarrow{\xi_{\beta,\beta+1}} & F^{\beta+1}_\Delta(X_n, 0) \\ {\scriptstyle \xi_{\beta,\alpha}}\downarrow & & \downarrow{\scriptstyle F_\Delta(X_n, \xi_{\beta,\alpha})} \\ F^\alpha_\Delta(X_n, 0) & \xrightarrow{\xi_{\alpha,\alpha+1}} & F^{\alpha+1}_\Delta(X_n, 0) \end{array}$$

In the following, we will assume that we have chosen λ large enough that all $\mu Y.F_\Delta(X_n, Y)$ and

$$\mu Y.F_\Delta(\varprojlim_n X_n, Y)$$

can be constructed this way as a colimit over λ.

Lemma 13. *If*

$$\begin{array}{ccc} A & \longrightarrow & B \\ \downarrow & & \downarrow \\ C & \longrightarrow & D \end{array}$$

is a pullback, then so is

$$\begin{array}{ccc} F_\Delta(X_{n+1}, A) & \longrightarrow & F_\Delta(X_{n+1}, B) \\ \downarrow & & \downarrow \\ F_\Delta(X_n, C) & \longrightarrow & F_\Delta(X_n, D) \end{array}$$

Proof. Follows from the fact that all polynomials preserve pullbacks. □

Lemma 14. $\xi_{\alpha+1,\beta+1} = F_\Delta(X_n, \xi_{\alpha,\beta})$.

Proof. By an easy induction on β. □

Lemma 15. *For all $\beta < \alpha$ and all n the following is a pullback.*

$$\begin{array}{ccc} F^{\beta}_{\Delta}(X_{n+1},0) & \longrightarrow & F^{\alpha}_{\Delta}(X_{n+1},0) \\ \downarrow & & \downarrow \\ F^{\beta}_{\Delta}(X_n,0) & \longrightarrow & F^{\alpha}_{\Delta}(X_n,0) \end{array}$$

In the proof and in this appendix in general we reason about limits and colimits as if they are computed in the category of sets. This is justified by the fact that $\mathbb{C}$ is a slice of a presheaf category, and thus a presheaf category itself, and limits and colimits in presheaf categories are computed pointwise.

Proof. The proof is by induction on α. Note that the case of $\beta = 0$ always holds trivially, and likewise $\alpha = 0$.

Suppose first that α is a limit ordinal. Suppose $x \in F^{\alpha}_{\Delta}(X_{n+1},0)$ and $y \in F^{\beta}_{\Delta}(X_n,0)$ map to the same element in $F^{\alpha}_{\Delta}(X_n,0)$. We must show that they derive from the same element in $F^{\beta}_{\Delta}(X_{n+1},0)$. Since $F^{\alpha}_{\Delta}(X_{n+1},0)$ is a colimit x must live in some $F^{\gamma}_{\Delta}(X_{n+1},0)$ for $\gamma < \alpha$. Since x and y map to the same element in the colimit $F^{\alpha}_{\Delta}(X_n,0)$, they must map to the same element already in some $F^{\gamma'}_{\Delta}(X_n,0)$ for $\gamma' < \alpha$. Since by the induction hypothesis the following is a pullback

$$\begin{array}{ccc} F^{\beta}_{\Delta}(X_{n+1},0) & \longrightarrow & F^{\gamma'}_{\Delta}(X_{n+1},0) \\ \downarrow & & \downarrow \\ F^{\beta}_{\Delta}(X_n,0) & \longrightarrow & F^{\gamma'}_{\Delta}(X_n,0) \end{array}$$

and so x must come from some element in $F^{\beta}_{\Delta}(X_{n+1},0)$.

Now suppose α is a successor $\alpha = \alpha' + 1$, and $\beta < \alpha'$, and consider the composite diagram

$$\begin{array}{ccccc} F^{\beta}_{\Delta}(X_{n+1},0) & \longrightarrow & F^{\beta+1}_{\Delta}(X_{n+1},0) & \longrightarrow & F^{\alpha'+1}_{\Delta}(X_{n+1},0) \\ \downarrow & & \downarrow & & \downarrow \\ F^{\beta}_{\Delta}(X_n,0) & \longrightarrow & F^{\beta+1}_{\Delta}(X_n,0) & \longrightarrow & F^{\alpha'+1}_{\Delta}(X_n,0) \end{array}$$

By induction hypothesis the diagram on the left is a pullback, and by Lemma 14 and Lemma 13 the square on the right is a pullback, so the composite square is also a pullback.

In the case of α the successor of β, again we consider cases of β. If β is a successor, then again we get a pullback by Lemma 14 and Lemma 13. If β is a limit, and $x \in F^{\alpha}_{\Delta}(X_{n+1},0)$ and $y \in F^{\beta}_{\Delta}(X_n,0)$, we know that $y \in F^{\beta'+1}_{\Delta}(X_n,0)$ already, for $\beta' < \beta$, and so we can reduce to the case of β being a successor. □

Lemma 16. *For all $\alpha \leq \lambda$ and all n, the canonical map*

$$F^{\alpha}_{\Delta}(\varprojlim_n X_n,0) \to \varprojlim_n F^{\alpha}_{\Delta}(X_n,0)$$

is an isomorphism.

Proof. By induction on α. Case $\alpha = 0$ is clear, and α being a successor follows from the fact that $F_{\Delta}(X_n,-)$ preserves ω^{op}-limits (this holds for all polynomial functors).

For the case of α a limit ordinal, suppose

$$(x_n)_n \in \varprojlim_n F^{\alpha}_{\Delta}(X_n,0),$$

then there is a sequence $\beta_1, \beta_2 \ldots$ of ordinals all smaller than α such that each

$$x_n \in F^{\beta_n}_{\Delta}(X_n,0)$$

Now, by Lemma 15 all $x_n \in F^{\beta_1}_{\Delta}(X_n,0)$, and thus

$$(x_n)_n \in F^{\beta_1}_{\Delta}(\varprojlim_n X_n,0)$$

which proves the lemma. □

Proof of Proposition 9. By Lemma 16 it suffices to show that the canonical map

$$\varinjlim_{\alpha<\lambda} \varprojlim_n F^{\alpha}_{\Delta}(X_n,0) \longrightarrow \varprojlim_n \varinjlim_{\alpha<\lambda} F^{\alpha}_{\Delta}(X_n,0)$$

is an isomorphism. This holds by an argument similar to that of Lemma 16: if

$$(x_n) \in \varprojlim_n \varinjlim_{\alpha<\lambda} F^{\alpha}_{\Delta}(X_n,0)$$

then each $x_n \in F^{\alpha_n}_{\Delta}(X_n,0)$ for some $\alpha_n < \lambda$, and so by Lemma 15 we conclude that all x_n are in $F^{\alpha_1}_{\Delta}(X_n,0)$ which proves the proposition. □