Theoretical Computer Science 722 (2018) 1–18

Contents lists available at ScienceDirect

Theoretical Computer Science

www.elsevier.com/locate/tcs

A model of guarded recursion via generalised equilogical spaces

Aleš Bizjak *, Lars Birkedal

Department of Computer Science, Aarhus University, Denmark

ARTICLE INFO

Article history:
Received 1 October 2015
Received in revised form 6 March 2017
Accepted 12 February 2018
Available online 21 February 2018
Communicated by D. Sannella

Keywords:
Semantics
Dependent type theory
Guarded recursion
Equilogical spaces

ABSTRACT

We present a new model, called GuardedEqu, of guarded dependent type theory using generalised equilogical spaces. GuardedEqu models guarded recursive types, which can be used to program with coinductive types and we prove that GuardedEqu ensures that all definable functions on coinductive types, e.g., streams, are continuous with respect to the natural topology. We present a direct, elementary, construction of the new model, which, importantly, is coherent (split) by construction.

1. Introduction

Type theories with support for guarded recursive functions and guarded recursive types are useful for programming with coinductive types and also for serving as a meta-theory for constructing sophisticated models of programming languages with effects [1,2].

In this paper, we present a new model of guarded dependent type theory, based on a generalisation of equilogical spaces [3]. We refer to the new model as GuardedEqu.

In contrast to earlier models of guarded dependent type theory, GuardedEqu ensures that definable functions on coinductive types are suitably continuous. For example, any function definable on the type of streams is continuous with respect to the standard topology on streams. Thus, if f is such a function on streams and xs is a stream, a finite amount of the output $f(xs)$ only depends on a finite amount of the input xs. We prove that an analogous result holds for final coalgebras of arbitrary polynomial functors.

It is well-known that models of dependent type theory can be tricky to construct. A virtue of GuardedEqu is that the model construction is quite elementary and can be presented via a simple generalisation of constructions known from realizability models of type theory. An important feature of GuardedEqu is that it is coherent (split) by construction. A limitation of the model is that it does not include universes.

We now explain how GuardedEqu is related to earlier models of variations of guarded type theory.

Originally a type theory with a single $\blacktriangleright$ modality for expressing guardedness was modelled using the category $\mathrm{PSh}(\omega)$, the topos of trees [1]. The model and the type theory allows for the solution of guarded recursive domain equations. It was later realised that guarded recursion can also be used for ensuring that functions producing values of coinductive types

* Corresponding author.
E-mail addresses: abizjak@cs.au.dk (A. Bizjak), birkedal@cs.au.dk (L. Birkedal).

https://doi.org/10.1016/j.tcs.2018.02.012
0304-3975/

are productive in a precise sense. To support such encodings the type theory needs to be extended with the ability to eliminate $\blacktriangleright$ in a controlled way. This led Atkey and McBride [4] to generalise $\blacktriangleright$ to a family of modalities indexed by clocks, and to support clock quantification for controlled elimination of $\blacktriangleright$. Atkey and McBride's development was for a simply typed calculus. They developed a model of their type theory and showed that, e.g., all streams definable in the calculus were interpreted as actual streams, i.e., non bottom elements. Møgelberg [5] extended their work to a model of dependent type theory with universes. This model was subsequently refined [6] to support clock synchronisation which considerably simplified the calculus. As it currently stands this model is complex, in particular in its split form which is needed to soundly model the rules of guarded dependent type theory [7].

GuardedEqu can be seen as a generalisation of the model by Atkey and McBride to a dependent type theory. The type theory we are considering has, to be useful, certain type isomorphisms [4,5]. An example is the type isomorphism $\forall\kappa.\mathbb{N} \cong \mathbb{N}$, where $\mathbb{N}$ is the type of natural numbers. In Atkey and McBride's model these were type equalities, but in the presheaf models of guarded dependent type theory [5,6] the types $\forall\kappa.\mathbb{N}$ and $\mathbb{N}$ are only modelled as canonically[1] isomorphic types. In GuardedEqu these type isomorphisms are again type equalities. This generalises the results of Atkey and McBride to a dependent type theory.

Overview of the paper In Section 2 we introduce the basics of guarded dependent type theory, referring to previous work for more details.

Then in Section 3 we construct a general class of models of dependent type theory with dependent products, sums, and extensional equality. These models are parametrised by posets P.

In Section 4 we instantiate these general models with specific posets $\mathfrak{I}(\Delta)$ to also model the later modality and clock quantification. Moreover we show that models of dependent type theory for different $\mathfrak{I}(\Delta)$ combine into a model of polymorphic dependent type theory [8, Chapter 11], satisfying additional axioms governing the interaction between the later modality and clock quantification.

In Section 5 we show some of the advantages of GuardedEqu, namely that it shows that coinductive types definable in guarded dependent type theory are interpreted as spaces with the expected topology and that definable functions are realised by continuous functions with respect to this topology. We illustrate this concretely on the example of streams.

Finally in Section 6 we compare GuardedEqu to previous models an discuss its advantages and some of its deficiencies.

2. Guarded dependent type theory

In this section we give a very brief introduction to the syntax of guarded dependent type theory. We refer the reader to [7] for the full set of typing rules, their motivation and more detailed explanation.

Guarded dependent type theory can be seen as a version of polymorphic dependent type theory [8]. It includes two contexts. A context Δ of clock variables $\kappa, \kappa', \cdots$ and a context Γ of ordinary term variables. Types depend on clocks, that is, clocks can appear in types, but clocks are only names in the sense that there are no constructions on clocks themselves. Guarded dependent type theory has the following basic judgements.

$$\Gamma \vdash_\Delta$$

$$\Gamma \vdash_\Delta A \text{ type}$$

$$\Gamma \vdash_\Delta t : A$$

The judgement $\Gamma \vdash_\Delta$ expresses that the free clocks in Γ are contained in Δ, the judgement $\Gamma \vdash_\Delta A$ type expresses that A is a well-formed type in context $\Gamma \vdash_\Delta$ and the last judgement expresses that t has type A in context $\Gamma \vdash_\Delta$. As usual in dependent type theory, there are also judgements for type and term equality for which we refer to [7]. Clocks are used to distinguish different $\blacktriangleright$ modalities. Thus, for each clock there is a modality $\blacktriangleright^\kappa$ and a term next^κ with the typing judgement

$$\frac{\Gamma \vdash_\Delta t : A}{\Gamma \vdash_\Delta \text{next}^\kappa\, t : \blacktriangleright^\kappa A}\kappa \in \Delta$$

Clock weakening is admissible, e.g., there is a derivation of

$$\frac{\Gamma \vdash_\Delta A \text{ type}}{\Gamma \vdash_{\Delta,\kappa} A \text{ type}}$$

for $\kappa \notin \Delta$ and analogously for other judgements. Clock weakening has a right adjoint, which we write as $\forall\kappa$. The introduction rule is

$$\frac{\Gamma \vdash_\Delta \qquad \Gamma \vdash_{\Delta,\kappa} t : A}{\Gamma \vdash_\Delta \Lambda\kappa.t : \forall\kappa.A}$$

[1] Canonical means there is a term of type $\mathbb{N} \to \forall\kappa.\mathbb{N}$ definable using just the ordinary introduction rules for $\forall\kappa$ and the denotation of this term is an isomorphism.

with the usual elimination rule for products and β-η rules for judgemental equalities. Thus in the model we shall need to have *polymorphic products* of *kinds* over *types* [8, Chapter 11].

A profitable way of thinking about guarded dependent type theory (and polymorphic dependent type theory in general) is as follows.

For each clock context Δ we have a core dependent type theory with ordinary type formers Π, Σ and equality types. In Section 3.1 we show how to model this fragment by constructing categories $\mathsf{PEqu}(P)$ and associated split closed comprehension categories with strong coproducts and strong equality. The construction is parametrised by a partially ordered set P. For the model of the core fragment it does not matter what P is. Later on, in Section 4, we instantiate P with specific posets which allow us to model the $\blacktriangleright$ modality and clock quantification $\forall\kappa$. Different clock contexts Δ will give rise to different posets P.

Next, to be able to change clock contexts, e.g., to interpret weakening, we need to be able to move between categories $\mathsf{PEqu}(P)$ for different P. Section 3.2 provides the necessary results which ensure that moving from $\mathsf{PEqu}(P)$ to $\mathsf{PEqu}(Q)$ preserves all the structure, e.g., products, coproducts. Section 4 then ties it all together into one model of guarded dependent type theory with two-level indexing, one for clock contexts and one for ordinary contexts. Section 4.4 provides a high-level summary of the model construction in the framework of fibrations. In Section 5 we prove how final coalgebras for polynomial functors can be obtained via guarded recursive types, and we prove the continuity properties for functions on final coalgebras for polynomial functors mentioned in the Introduction.

GuardedEqu validates all the rules of guarded dependent type theory apart from universes. We do not show soundness of all the rules in this article but only show the main constructions needed. We organise the model using fibrations and comprehension categories which are known to support the interpretation of type theory [8], thus we rely on existing work for the precise interpretation of the type theory in the model we construct.

Remark 1. In the most recent formulation of guarded dependent type theory [7], guarded recursive types are defined via fixed points of functions on universes. Since we do not model universes in GuardedEqu, we would need some other facility for defining guarded recursive types syntactically, such as the one used in [1]. Formulating and modelling such guarded recursive types can be done without too much trouble, following [1].[2] Here, however, we do not include such a treatment, since that would not be particularly interesting, given all the other material. Instead we show in Section 5 that we can construct final coalgebras of polynomial functors using solutions of guarded type equations and we show that the solution comes equipped with the correct topology, which allows us to show expected productivity properties of functions defined on final coalgebras. ■

3. GuardedEqu

In this section we show a general construction of split closed comprehension categories parametrised by posets together with morphisms of fibrations preserving the closed comprehension category structure. These are induced by *fibrations* between the posets. In the subsequent section we specialise to particular posets $\mathfrak{I}(\Delta)$ used in previous work [6] to get a model of the calculus described in the preceding section.

3.1. Modelling core dependent type theory

Recall that an element c in a complete lattice is *compact* if for any directed subset D

$$c \leq \bigvee D \Rightarrow \exists d \in D, c \leq d$$

where $\bigvee D$ is the supremum of D.

Let P be a partially ordered set. The category $\mathsf{PEqu}(P)$ has as objects pairs $A = (|A|, R_A)$ where

- $|A|$ is an algebraic lattice, i.e., a complete lattice satisfying that every element is the supremum of compact elements below it
- R_A is a monotone map from P^{op} to PERs on $|A|$ ordered by subset inclusion. In other words, R_A is a family of partial equivalence relations on $|A|$ indexed by P such that if $p \leq q$ then $R_A(p) \supseteq R_A(q)$.

We will sometimes write $a \approx^p_A a'$ for $(a, a') \in R_A(p)$.

Morphisms from A to B in $\mathsf{PEqu}(P)$ are equivalence classes, with respect to the relation $\sim$ defined below, of continuous *non-expansive* maps $|A| \to |B|$ (i.e., morphisms in AlgLat). The function f is non-expansive if it satisfies for all $p \in P$ the property

[2] Indeed it is easy to show that $\mathsf{PEqu}(P)$ is enriched in the category of presheaves over P, which is equivalent to the category of sheaves over P equipped with the Alexandrov topology, so fits into the general framework [1] for a well-founded order P. Further, it is easy to see that the usual type constructors, $\to$, $\times$, $+$ give rise to enriched functors, and thus one can prove an existence theorem for fixed points of contractive functors following previous work [1].

$$\forall (a, a') \in R_A(p), (f(a), f(a')) \in R_B(p).$$

The equivalence relation $\sim$ is defined as

$$f \sim g \iff \forall p \in P, \forall (a, a') \in R_A(p), (f(a), g(a')) \in R_B(p).$$

Note that $\sim$ is an equivalence relation on non-expansive maps, but only a partial equivalence relation on general continuous maps. Indeed, if we define the relation $\sim$ on all maps then the non-expansive ones are precisely the ones in the domain of $\sim$.

This makes $\mathsf{PEqu}(P)$ into a category. Identities are given by equivalence classes of the identity functions in AlgLat. Composition of $[f] : A \to B$ and $[g] : B \to C$ is given by the equivalence class of $f \circ g$. It can easily be seen that this definition is independent of the choice of representatives f and g.

Remark 2. If P is the unique poset with one element then $\mathsf{PEqu}(P)$ is the category of partial equilogical spaces [3], equivalent to the category of equilogical spaces. ■

To interpret dependent types we present the slice categories in a different way, similar to *uniform families* [8] but incorporating the monotonicity requirement. The monotonicity requirement is needed later to interpret the $\mathsf{next}^\kappa\ t$ construct. See Remark 20 below.

Analogous to the way that the slice category $\mathsf{PSh}(\mathbb{C})/\Gamma$ of the category of presheaves $\mathsf{PSh}(\mathbb{C})$ over some presheaf Γ is equivalent to the category of presheaves over the category of elements we will represent the slice categories of $\mathsf{PEqu}(P)$ using an auxiliary poset $\int_P \Gamma$. We now define this poset. Let $\Gamma \in \mathsf{PEqu}(P)$. The poset $\int_P \Gamma$ has as elements pairs $(p, [\gamma]_p)$ where $p \in P$ and $[\gamma]_p$ is an equivalence class of $\gamma \in |\Gamma|$ with respect to the relation $R_\Gamma(p)$. In particular this means $(\gamma, \gamma) \in R_\Gamma(p)$. We define the order $\leq$ on $\int_P \Gamma$ as $(p, c) \leq (q, c')$ if and only if $p \leq q$ and $c \supseteq c'$. Or equivalently $(p, [\gamma]_p) \leq (q, [\gamma']_q)$ if $p \leq q$ and $(\gamma, \gamma') \in R_\Gamma(p)$.

Lemma 3. *The set $\int_P \Gamma$ with the order relation $\leq$ defined above is a poset.*

Note that in contrast to the situation with presheaves, it is not the case that the category $\mathsf{PEqu}(P)/\Gamma$ is equivalent to the category $\mathsf{PEqu}(\int_P \Gamma)$. The problem is that the latter category has too few morphisms, however its objects are precisely the ones needed. Thus we define a new category $\mathsf{Type}_P(\Gamma)$ which addresses this issue.

Definition 4. The objects of $\mathsf{Type}_P(\Gamma)$[3] are the objects of $\mathsf{PEqu}(\int_P \Gamma)$.

A morphism from A to B in $\mathsf{Type}_P(\Gamma)$ is an equivalence class of continuous functions $|\Gamma| \to |A| \to |B|$ in AlgLat with respect to the partial equivalence relation $\sim_\Gamma$ which relates f and f' if and only if

$$\forall p \in P, \forall (\gamma, \gamma') \in R_\Gamma(p), \forall (a, a') \in R_A(p, [\gamma]_p), (f\,\gamma\, a, g\,\gamma'\, a') \in R_B(p, [\gamma]_p).$$

As before such a function f is called *non-expansive* if $f \sim_\Gamma f$, i.e., if f is in the domain of $\sim_\Gamma$. Identity at the object A is given by the equivalence class of the second projection and composition of $[f] : A \to B$ and $[g] : B \to C$ is given by the equivalence class of the continuous function

$$\gamma \mapsto a \mapsto g\,\gamma\,(f\,\gamma\, a). \quad ■$$

Remark 5. Comparing to the situation with presheaves again, the morphisms in $\mathsf{Type}_P(\Gamma)$ have access to an additional parameter Γ (as compared to the morphisms in $\mathsf{PEqu}(\int_P \Gamma)$). The reason this is not needed with presheaf categories is that natural transformations are *indexed* families of functions (satisfying coherence conditions) and elements of Γ are part of the indexing poset. So in essence, the additional parameter Γ is already built into the definition of the category of presheaves over the category of elements.

In case P is the singleton poset the category $\mathsf{Type}_P(\Gamma)$ is precisely the category of uniform families of PERs over Γ [3]. ■

Theorem 6. *For any poset P and $\Gamma \in \mathsf{PEqu}(P)$ the category $\mathsf{Type}_P(\Gamma)$ is equivalent to the slice category $\mathsf{PEqu}(P)/\Gamma$.*

Proof. We define two functors $F : \mathsf{Type}_P(\Gamma) \to \mathsf{PEqu}(P)/\Gamma$ and G in the converse direction. The functor F maps

[3] We use the notation $\mathsf{Type}_P(\Gamma)$ because this category will be used to interpret dependent types in context Γ.

- the object A to the object $\pi_A : \Gamma \ltimes A \to \Gamma$ where

$$|\Gamma \ltimes A| = |\Gamma| \times |A|$$
$$R_{\Gamma \ltimes A}(p) = \left\{ \left((\gamma, a), (\gamma', a') \right) \mid (\gamma, \gamma') \in R_\Gamma(p) \wedge (a, a') \in R_A(p, [\gamma]_p) \right\}$$

  and π_A is the equivalence class of the first projection $|\Gamma| \times |A| \to |\Gamma|$.
- the morphism $[f]$ to the equivalence class of the morphism

$$(\gamma, a) \mapsto (\gamma, f\gamma a)$$

  The definition of the equivalence relation $\sim_\Gamma$ is precisely what is needed to show that such a definition is independent of the choice of representative f.

The functor G maps

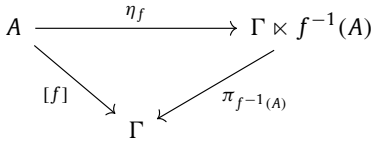
- an object $[f] : A \to \Gamma$ of the slice category to the object $f^{-1}(A)$ of $\mathsf{Type}_P(\Gamma)$ where

$$\left| f^{-1}(A) \right| = |A|$$
$$R_{f^{-1}(A)}\left(p, [\gamma]_p\right) = \left\{ (a, a') \mid (a, a') \in R_A(p) \wedge (f(a), \gamma) \in R_\Gamma(p) \right\}.$$

- a morphism $[\alpha] : [f] \to [g]$ to the equivalence class of the continuous function

$$\gamma \mapsto a \mapsto \alpha a$$

Finally we define two natural isomorphisms $\eta : \mathrm{id} \to F \circ G$ and $\varepsilon : \mathrm{id} \to G \circ F$.



Define η_f to be the equivalence class of the continuous function

$$a \mapsto (f(a), a).$$

Its inverse is the equivalence class of the second projection $\pi_2 : |\Gamma| \times |A| \to |A|$.

The component of ε and A is given by the equivalence class of

$$\gamma \mapsto a \mapsto (\gamma, a).$$

Its inverse is given by the equivalence class of the continuous function

$$\gamma \mapsto (\gamma', a) \mapsto a.$$

Checking that these are well-defined and that they satisfy the claimed properties is straightforward, albeit somewhat lengthy, unpacking of definitions. □

3.1.1. Reindexing

Given a morphism $[f] : \Gamma_1 \to \Gamma_2$ in $\mathsf{PEqu}(P)$ there is a functor $f^* : \mathsf{Type}_P(\Gamma_2) \to \mathsf{Type}_P(\Gamma_1)$ defined by "precomposition". Given an object $A \in \mathsf{Type}_P(\Gamma_2)$ the object $f^*(A)$ is

$$\left| f^*(A) \right| = |A|$$
$$R_{f^*(A)}\left(p, [\gamma]_p\right) = R_A\left(p, [f(\gamma)]_p\right).$$

The functor f^* maps a morphism realised by g to the morphism realised by $g \circ f$. More concretely, a morphism $[g]$ is by definition realised by a continuous function $g : |\Gamma_2| \to |A| \to |B|$. The morphism $f^*([g])$ is realised by the function of type $|\Gamma_1| \to |A| \to |B|$ mapping γ and a to $g(f\gamma)a$.

Lemma 7. *The operation f^* we have defined is well-defined, i.e., independent of the choice of representative f, and it is a functor.*

Moreover this construction shows that we have a functor from the opposite of the category $\mathsf{PEqu}(P)$ to the category of categories which maps Γ to $\mathsf{Type}_P(\Gamma)$ and $[f]$ to the functor f^*. So we may use the Grothendieck construction to get a fibration $p : \mathsf{UFam}(P) \to \mathsf{PEqu}(P)$. Concretely, the objects of the total category $\mathsf{UFam}(P)$ are pairs (Γ, A) where $\Gamma \in \mathsf{PEqu}(P)$ and $A \in \mathsf{Type}_P(\Gamma)$. Morphisms $(\Gamma_1, A_1) \to (\Gamma_2, A_2)$ are pairs $([f], [g])$ where $[f] : \Gamma_1 \to \Gamma_2$ is a morphism in $\mathsf{PEqu}(P)$ and $[g]$ is an equivalence class of continuous functions $|\Gamma_1| \to |A_1| \to |A_2|$ with respect to the relation $\sim_{A_1, A_2}$ which relates g and g' if and only if

$$\forall p \in P, (\gamma, \gamma') \in R_{\Gamma_1}(p), (a, a') \in R_{A_1}(p, [\gamma]_p), (g\gamma a, g'\gamma' a') \in R_{A_2}(p, [f(\gamma)]_p).$$

Since the assignment $f \mapsto f^*$ is a functor the fibration p, which simply projects the first components, is a cloven split fibration. The chosen cartesian lifting of $[f] : \Gamma_1 \to p(\Gamma_2, A)$ is

$$([f], [\gamma \mapsto a \mapsto a]) : \left(\Gamma_1, f^*(A)\right) \to (\Gamma_2, A)$$

The functor F defined in the proof of Theorem 6 can be extended to a comprehension category as we describe below. First we recall the definition of a comprehension category.

A comprehension category [8, Definition 10.4.2] is a functor $\mathcal{P} : \mathbb{E} \to \mathbb{B}^{\to}$ such that $\mathsf{cod} \circ \mathcal{P} : \mathbb{E} \to \mathbb{B}$ is a fibration and such that $\mathcal{P}$ is a morphism of fibrations, i.e., if f is cartesian in $\mathbb{E}$ then $\mathcal{P}(f)$ is a pullback square in $\mathbb{B}^{\to}$. The comprehension category $\mathcal{P}$ is *full* if $\mathcal{P}$ is a *full and faithful* functor and it is *split* if $\mathsf{cod} \circ \mathcal{P}$ is a split fibration. The comprehension category is a *comprehension category with unit* if the fibration $\mathsf{cod} \circ \mathcal{P}$ has a terminal object functor, which by definition is the right adjoint to $\mathsf{cod} \circ \mathcal{P}$.

The functor F can indeed be extended to a full split comprehension category with unit. Define the functor $\mathcal{P} : \mathsf{UFam}(P) \to \mathsf{PEqu}(P)^{\to}$ as

$$\mathcal{P}(\Gamma, A) = \pi_A$$
$$\mathcal{P}([f], [g]) = ([f], [(\gamma, a) \mapsto (f\gamma, g\gamma a)])$$

as depicted in the following diagram

$$\begin{array}{ccc} \Gamma_1 \ltimes A_1 & \xrightarrow{[(\gamma,a)\mapsto(f(\gamma),g\gamma a)]} & \Gamma_2 \ltimes A_2 \\ \downarrow{\scriptstyle \pi_{A_1}} & & \downarrow{\scriptstyle \pi_{A_2}} \\ \Gamma_1 & \xrightarrow{[f]} & \Gamma_2 \end{array}$$

Theorem 8. *The diagram*

$$\begin{array}{ccc} \mathsf{UFam}(P) & \xrightarrow{\mathcal{P}} & \mathsf{PEqu}(P)^{\to} \\ \downarrow{\scriptstyle p} & \swarrow{\scriptstyle \mathsf{cod}} & \\ \mathsf{PEqu}(P) & & \end{array}$$

commutes and p is a full split comprehension category with unit. The terminal object functor $1 : \mathsf{PEqu}(P) \to \mathsf{UFam}(P)$ maps Γ to the object $(\Gamma, 1)$ where

$$|1| = 2^{\emptyset}$$
$$R_1(p) = 2^{\emptyset} \times 2^{\emptyset}$$

and $2^{\emptyset}$ is the power set of the empty set.

Below we will extensively use the fact that the fibre over an object Γ with respect to the fibration p is isomorphic in a trivial way to the category $\mathsf{Type}_P(\Gamma)$. This reduces clutter because we do not have to carry around the first components of objects of the fibre which do not matter, since they are uniquely determined (up to equality).

3.1.2. Dependent products

Theorem 9. *The comprehension category p has (split) products satisfying (split) Beck–Chevalley condition.*

Proof. Let $\pi_A = \mathcal{P}(\Gamma, A) : \Gamma \ltimes A \to \Gamma$ be a projection. It induces a functor $\pi_A^{\blacktriangle}$ from the slice over Γ to the slice over $\Gamma \ltimes A$ which we need to show has a right adjoint. Representing slices using the categories $\mathsf{Type}_P\,(-)$ the functor $\pi_A^{\blacktriangle}$ is simply π_A^*. Thus, given an object $B \in \mathsf{Type}_P\,(\Gamma \ltimes A)$ define the object $\Pi(A, B) \in \mathsf{Type}_P\,(\Gamma)$ to have the underlying lattice $|\Pi(A, B)|$ the lattice of continuous functions $|A| \to |B|$. The family of relations $R_{\Pi(A,B)}$ is defined at $(p, [\gamma]_p)$ to be

$$\left\{(f, f') \;\middle|\; \forall q \leq p, \forall (a, a') \in R_A(q, [\gamma]_q), \left(f(a), f'(a')\right) \in R_B\left(q, [(\gamma, a)]_q\right)\right\}.$$

To show that we get a right adjoint to π_A^* we show that we have a universal morphism

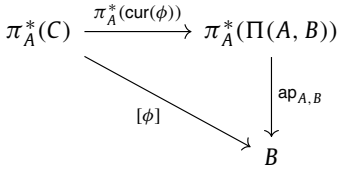
$$\mathsf{ap}_{A,B} : \pi_A^*\left(\Pi(A, B)\right) \to B$$

in $\mathsf{Type}_P\,(\Gamma \ltimes A)$. We define $\mathsf{ap}_{A,B}$ to be the equivalence class of the morphism

$$(\gamma, a) \mapsto f \mapsto fa.$$

Note that the relation $R_{\Pi(A,B)}$ states precisely what is needed to show that the function we have defined is non-expansive.

To show that $\mathsf{ap}_{A,B}$ is universal we show that for any C and any morphism $[\phi] : \pi_A^*(C) \to B$ there is a unique morphism $\mathsf{cur}\,(\phi) : C \to \Pi(A, B)$ in $\mathsf{Type}_P\,(\Gamma)$ making the following diagram commute.



Define $\mathsf{cur}\,(\phi)$ to be the equivalence class of the function

$$\gamma \mapsto c \mapsto (a \mapsto \phi(\gamma, a)c)$$

of type $|\Gamma| \to |C| \to (|A| \to |B|)$, recalling that $[\phi]$ is realised by the function ϕ of type $|\Gamma| \times |A| \to |C| \to |B|$.

Checking that all the stated properties hold is straightforward unpacking of definitions.

Finally, the Beck–Chevalley condition can be verified to hold by simple computations. □

3.1.3. Dependent sums

Theorem 10. *The comprehension category p has (split)* strong *coproducts satisfying (split) Beck–Chevalley condition.*

Proof. As in the proof of Theorem 9 given an object $B \in \mathsf{Type}_P\,(\Gamma \ltimes A)$ define an object $\Sigma(A, B) \in \mathsf{Type}_P\,(\Gamma)$ as follows. The underlying lattice $|\Sigma(A, B)|$ is the lattice $|A| \times |B|$. The family of relations $R_{\Sigma(A,B)}$ is defined at $(p, [\gamma]_p)$ to be

$$\left\{((a, b), (a', b')) \;\middle|\; (a, a') \in R_A(p, [\gamma]_p), \left(b, b'\right) \in R_B\left(p, [(\gamma, a)]_p\right)\right\}.$$

This definition comes with a morphism $\mathsf{pair}_{A,B} : B \to \pi_A^*\left(\Sigma(A, B)\right)$ in the fibre over $\Gamma \ltimes A$ which we define to be the equivalence class of the continuous function

$$(\gamma, a) \mapsto b \mapsto (a, b).$$

This morphism satisfies the property that for any C and any $[\phi] : B \to \pi_A^*(C)$ there exists a unique morphism $\mathsf{unpack}\,(\phi) : \Sigma(A, B) \to C$ satisfying $\pi_A^*(\mathsf{unpack}\,(\phi)) \circ \mathsf{pair}_{A,B} = [\phi]$. Indeed, the morphism $\mathsf{unpack}\,(\phi)$ is defined as the equivalence class of the continuous function

$$\gamma \mapsto (a, b) \mapsto \phi(\gamma, a)b.$$

All of the claimed properties are easy and straightforward to check.

With these definitions the assignment $B \mapsto \Sigma(A, B)$ extends to a functor left adjoint to π_A^*. Concretely it maps a morphism $[\phi] : B \to C$ to the equivalence class of the continuous function

$$\gamma \mapsto (a, b) \mapsto (a, \phi(\gamma, a)b).$$

Recall the domain functor $\mathsf{dom} : \mathsf{PEqu}(P)^{\to} \to \mathsf{PEqu}(P)$. A comprehension category has *strong* coproducts [8, Definition 10.5.2] if the morphism

$$\mathsf{dom}\left(\mathcal{P}(\pi_A, \mathsf{pair}_{A,B})\right)$$

is an isomorphism. In our case this follows by computation. Indeed, the morphism $\mathsf{dom}\left(\mathcal{P}(\pi_A, \mathsf{pair}_{A,B})\right)$ is the equivalence class of the continuous function

$$((\gamma, a), b) \mapsto (\gamma, (a, b)). \quad \square$$

3.1.4. Equality

Theorem 11. *The fibration p has* strong *equality.*

Proof. For any Γ and any $A \in \mathsf{Type}_P(\Gamma)$ there is a morphism in $\mathsf{PEqu}(P)$

$$\delta_A : \Gamma \ltimes A \to \Gamma \ltimes A \ltimes \pi_A^*(A)$$

which is the equivalence class of the continuous function $(\gamma, a) \mapsto ((\gamma, a), a)$.

Recall [8, Definition 10.5.1] that a comprehension category has weak equality if $\delta_A^{\dagger}$ has a left adjoint Eq_A for any A and these left adjoints satisfy the Beck–Chevalley condition.

Let us define $\mathsf{Eq}_A : \mathsf{Type}_P(\Gamma \ltimes A) \to \mathsf{Type}_P\left(\Gamma \ltimes A \ltimes \pi_A^*(A)\right)$. We proceed as we did in Theorem 10, by constructing an object $\mathsf{Eq}_A(B)$ and a universal morphism. Given $B \in \mathsf{Type}_P(\Gamma \ltimes A)$ define $\mathsf{Eq}_A(B)$ to have the underlying lattice $\left|\mathsf{Eq}_A B\right|$ the lattice $|B|$ and the family of relations $R_{\mathsf{Eq}_A(B)}$ is defined at $(p, [((\gamma, a), a')])$ to be

$$\begin{aligned} &\left\{(b, b') \;\middle|\; (b, b') \in R_B(p, [(\gamma, a)]), (a, a') \in R_A(p, [\gamma])\right\} \\ &= \begin{cases} R_B(p, [(\gamma, a)]) & \text{if } (a, a') \in R_A(p, [\gamma]) \\ \emptyset & \text{otherwise} \end{cases} \end{aligned}$$

There is a morphism $\mathsf{refl}_{A,B} : B \to \delta_A^*(\mathsf{Eq}_A B)$ which is the equivalence class of the continuous function

$$(\gamma, a) \mapsto b \mapsto b.$$

This pair satisfies the universal property stating that for any C and any $[\phi] : B \to \delta_A^*(C)$ there exists a unique morphism $\mathsf{with}(\phi) : \mathsf{Eq}_A(B) \to C$ satisfying $\delta_A^*(\mathsf{with}(\phi)) \circ \mathsf{refl}_{A,B} = [\phi]$. The morphism $\mathsf{with}(\phi)$ is the equivalence class of the continuous function

$$((\gamma, a), a') \mapsto b \mapsto \phi(\gamma, a)b.$$

To show that the equality is strong we additionally need to check [8, Definition 10.5.2] that the canonical morphism

$$\kappa : \Gamma \ltimes A \ltimes B \to \Gamma \ltimes A \ltimes \pi_A^*(A) \ltimes \mathsf{Eq}_A B$$

which is the equivalence class of the continuous function

$$((\gamma, a), b) \mapsto (((\gamma, a), a), b)$$

is an isomorphism. This is indeed the case and its inverse is given by the equivalence class of the function

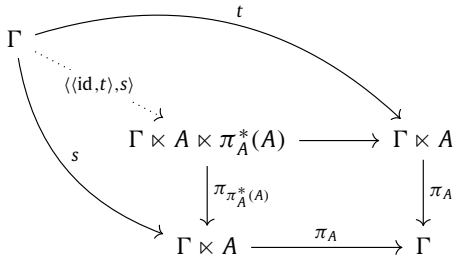
$$(((\gamma, a), a'), b) \mapsto ((\gamma, a), b). \quad \square$$

With this we have shown that for any poset P the fibration p is a model of dependent type theory with strong dependent sums and strong, i.e. extensional, equality.

A few words on how the identity type is modelled. A term t of type A in context Γ is modelled as a section of the projection $\mathcal{P}(A)$. Because $\mathcal{P}$ is a comprehension category the diagram

$$\begin{array}{ccc} \Gamma \ltimes A \ltimes \pi_A^*(A) & \longrightarrow & \Gamma \ltimes A \\ \downarrow{\scriptstyle \pi_{\pi_A^*(A)}} & & \downarrow{\scriptstyle \pi_A} \\ \Gamma \ltimes A & \xrightarrow{\pi_A} & \Gamma \end{array}$$

which is the image, $\mathcal{P}(\overline{\pi_A})$, of the cartesian lifting $\overline{\pi_A}$ of π_A, is a pullback diagram. Thus given two terms t and s of type A we get a unique morphism $\langle\langle \mathrm{id}, t\rangle, s\rangle$ making the following diagram commute.



The fibration p has a terminal object functor 1. We define the interpretation of the type $\mathsf{Id}(t,s)$ to be the object

$$\langle\langle \mathrm{id}, t\rangle, s\rangle^{\bigstar}\left(\mathsf{Eq}_A(1(\Gamma \ltimes A))\right).$$

Concretely in our model the underlying lattice of $[\![\mathsf{Id}(t,s)]\!]$ is the lattice $2^{\emptyset}$. The relation $R_{\mathsf{Id}(t,s)}$ is

$$\begin{aligned} R_{\mathsf{Id}(t,s)}(p,[\gamma]) &= \left\{(\emptyset,\emptyset) \;\middle|\; (\pi_2(t(\gamma)),\pi_2(s(\gamma))) \in R_A(p,[\gamma])\right\} \\ &= \begin{cases} |1|\times|1| & \text{if } (\pi_2(t(\gamma),\pi_2(s(\gamma)))) \in R_A(p,[\gamma]) \\ \emptyset & \text{otherwise} \end{cases} \end{aligned}$$

which is as expected. This also makes it clear why the model supports extensional equality.

3.2. Moving between $\mathsf{PEqu}(P)$ and $\mathsf{PEqu}(Q)$

Let P and Q be two posets and $\phi : Q \to P$ a monotone function. It induces a functor $\phi^\dagger : \mathsf{PEqu}(P) \to \mathsf{PEqu}(Q)$ (notice the reversed order of posets Q and P) by "precomposition" as follows. It maps an object Γ to the object $\phi^\dagger(\Gamma)$ where

$$\begin{aligned} \left|\phi^\dagger(\Gamma)\right| &= |\Gamma| \\ R_{\phi^\dagger(\Gamma)} &= R_\Gamma \circ \phi \end{aligned}$$

where we consider R_Γ as a function from P to the set of partial equivalence relations on $|\Gamma|$, so composition with ϕ is well-defined. Moreover because ϕ is monotone the composition $R_\Gamma \circ \phi$ retains the monotonicity property. The functor $\phi^\dagger$ maps a morphism realised by f to the equivalence class of the continuous function f. Note however that the equivalence relations are different, consequently $\phi^\dagger$ is in general neither full nor faithful.

Lemma 12. *The functor $\phi^\dagger$ preserves limits and colimits.*

Further, given $\Gamma \in \mathsf{PEqu}(P)$ there is an induced monotone function ϕ_Γ

$$\phi_\Gamma : \int_Q \phi^\dagger(\Gamma) \to \int_P \Gamma$$
$$\phi_\Gamma(q,[\gamma]_q) = \left(\phi(q),[\gamma]_{\phi(q)}\right).$$

Note that by the definition of $\phi^\dagger(\Gamma)$ the sets $[\gamma]_{\phi(q)}$ and $[\gamma]_q$ are equal.

This function in turn induces a functor

$$\phi^\dagger_\Gamma : \mathsf{Type}_P(\Gamma) \to \mathsf{Type}_Q\left(\phi^\dagger(\Gamma)\right)$$

which acts as follows.

objects It maps an object A to the object $(|A|, R_A \circ \phi_\Gamma)$.
morphisms It maps the morphism realised by f to the equivalence class of f, exactly as in the definition of $\phi^\dagger$.

We have two types of reindexing. The following lemma states that they commute in the appropriate way.

Lemma 13. *Let* $[f] : \Gamma_1 \to \Gamma_2$ *be a morphism in* $\mathsf{PEqu}(P)$. *The following diagram of functors commutes on the nose*

$$\begin{array}{ccc} \mathsf{Type}_P(\Gamma_2) & \xrightarrow{f^*} & \mathsf{Type}_P(\Gamma_1) \\ \downarrow{\scriptstyle \phi^\dagger_{\Gamma_2}} & & \downarrow{\scriptstyle \phi^\dagger_{\Gamma_1}} \\ \mathsf{Type}_Q\left(\phi^\dagger(\Gamma_2)\right) & \xrightarrow{\left(\phi^\dagger(f)\right)^*} & \mathsf{Type}_Q\left(\phi^\dagger(\Gamma_1)\right) \end{array}$$

Proof. Since none of these functors changes the underlying lattices or the realisers those are clearly preserved on the nose if only the relations are preserved. This is easily seen to be the case with the definitions provided. □

Combining these two functors we get a morphism of fibrations

$$\begin{array}{ccc} \mathsf{UFam}(P) & \xrightarrow{\phi^\top} & \mathsf{UFam}(Q) \\ \downarrow{\scriptstyle p_P} & & \downarrow{\scriptstyle p_Q} \\ \mathsf{PEqu}(P) & \xrightarrow{\phi^\dagger} & \mathsf{PEqu}(Q) \end{array} \tag{1}$$

The functor $\phi^\top$ maps

objects the object (Γ, A) to the object $\left(\phi^\dagger(\Gamma), \phi^\dagger_\Gamma(A)\right)$

morphisms the morphism $([f], [g])$ to the morphism $([f]', [g]')$ where we have used $'$ to highlight the fact that the equivalence relations are different.

Theorem 14. *The pair of functors just defined is a morphism of split fibrations. It maps the chosen cartesian liftings to the chosen cartesian liftings.*

Proof. The fact that it is a morphism of fibrations follows from Lemma 13 and the fact that the functors never change the underlying realisers. □

Lemma 15. *Let* $\phi^{\dagger\to} : \mathsf{PEqu}(P)^\to \to \mathsf{PEqu}(Q)^\to$ *be the functor induced by* $\phi^\dagger$ *in the natural way and let* $\mathcal{P}_P$ *and* $\mathcal{P}_Q$ *be the comprehensions belonging to fibrations* p_P *and* p_Q. *Then* $\phi^{\dagger\to} \circ \mathcal{P}_P = \mathcal{P}_Q \circ \phi^\top$. *In particular for any* $A \in \mathsf{Type}_P(\Gamma)$ *we have* $\phi^\dagger(\pi_A) = \pi_{\phi^\dagger_\Gamma A}$.

Finally, the diagram of functors

$$\begin{array}{ccc} \mathsf{UFam}(P)_\Gamma & \xrightarrow{\pi^*_A} & \mathsf{UFam}(P)_{\Gamma \ltimes A} \\ \downarrow{\scriptstyle \phi^\top} & & \downarrow{\scriptstyle \phi^\top} \\ \mathsf{UFam}(Q)_{\phi^\dagger(\Gamma)} & \xrightarrow{\pi^*_{\phi^\dagger_\Gamma A}} & \mathsf{UFam}(Q)_{\phi^\dagger(\Gamma) \ltimes \phi^\dagger_\Gamma(A)} \end{array}$$

commutes on the nose. Note that we have used the equality $\phi^\dagger(\Gamma) \ltimes \phi^\dagger_\Gamma(A) = \phi^\dagger(\Gamma \ltimes A)$ *which follows from the first part of the lemma.*

Theorem 16. *If* ϕ *is a* fibration *then the morphism of fibrations* (1) *preserves products and coproducts on the nose. This means (cf. [8, Definition 1.9.13])*

- $\phi^\dagger$ *preserves pullbacks on the nose*
- *For every* $\Gamma \in \mathsf{PEqu}(P)$ *and* $A \in \mathsf{Type}_P(\Gamma)$ *the canonical natural transformation*

$$\phi^\top \circ \Pi(A, -) \to \Pi\left(\phi^\dagger_\Gamma(A), -\right) \circ \phi^\top$$

is the identity. This in particular means that we have an equality of objects

$$\phi^\top(\Pi(A, B)) = \Pi\left(\phi^\dagger_\Gamma(A), \phi^\top(B)\right)$$

for every $B \in \mathsf{Type}_P(\Gamma \ltimes A)$.

- *For every* $\Gamma \in \mathsf{PEqu}(P)$ *and* $A \in \mathsf{Type}_P(\Gamma)$ *the canonical natural transformation*

$$\Sigma\left(\phi^\dagger_\Gamma(A), -\right) \circ \phi^\top \to \phi^\top \circ \Sigma(A, -)$$

is the identity.

Proof. Most of this is simple computation and only requires ϕ to be monotone. We need the assumption that ϕ is a fibration to show that products are preserved. To show how this assumption is used we spell out the proof of

$$\phi^\top(\Pi(A, B)) = \Pi\left(\phi^\dagger_\Gamma(A), \phi^\top(B)\right).$$

Since $\phi^\top$ or $\phi^\dagger_\Gamma$ do not change the underlying lattices they are $|A| \to |B|$ on both sides. To show that the families of relations are the same we show two inclusions.

$\subseteq$ For every $q \in Q$ and $(\gamma, \gamma) \in R_\Gamma(\phi(q))$ we show

$$R_{\phi^\top(\Pi(A,B))}(q, [\gamma]) \subseteq R_{\Pi\left(\phi^\dagger_\Gamma(A), \phi^\top(B)\right)}(q, [\gamma]).$$

Recall

$$R_{\phi^\top(\Pi(A,B))}(q, [\gamma]) = R_{\Pi(A,B)}(\phi(q), [\gamma]). \tag{2}$$

Let

$$(f, f') \in R_{\Pi(A,B)}(\phi(q), [\gamma]), \tag{3}$$

$r \leq q$ and $(a, a') \in R_{\phi^\dagger_\Gamma(A)}(r, [\gamma])$. Recall

$$R_{\phi^\dagger_\Gamma(A)}(r, [\gamma]) = R_A(\phi(r), [\gamma]).$$

Because ϕ is monotone $\phi(r) \leq \phi(q)$ holds and so from assumption (3) we get $(fa, f'a') \in R_B(\phi(r), [(\gamma, a)])$ and by definition of $\phi^\top$ we have

$$R_{\phi^\top(B)}(r, [(\gamma, a)]) = R_B(\phi(r), [(\gamma, a)])$$

which gives

$$(fa, f'a') \in R_{\phi^\top(B)}(r, [(\gamma, a)])$$

as needed. Note that we have only needed ϕ to be monotone for this direction.

$\supseteq$ For every $q \in Q$ and $(\gamma, \gamma) \in R_\Gamma(\phi(q))$ we show

$$R_{\Pi\left(\phi^\dagger_\Gamma(A), \phi^\top(B)\right)}(q, [\gamma]) \subseteq R_{\phi^\top(\Pi(A,B))}(q, [\gamma]).$$

Assume

$$(f, f') \in R_{\Pi\left(\phi^\dagger_\Gamma(A), \phi^\top(B)\right)}(q, [\gamma]). \tag{4}$$

Recalling (2) let $r \leq \phi(q)$ and $(a, a') \in R_A(r, [\gamma])$. We need to show $(fa, f'a') \in R_B(r, [(\gamma, a)])$. Because ϕ is a fibration there exists a $u(q, r) \in Q$ such that $u(q, r) \leq q$ and $\phi(u(q, r)) = r$. Thus by definition we have $(a, a') \in R_{\phi^\dagger_\Gamma(A)}(u(q, r), [\gamma])$ and so from (4) $(fa, f'a') \in R_{\phi^\top(B)}(u(q, r), [(\gamma, a)])$. By definition and the property $\phi(u(q, r)) = r$ we have

$$R_{\phi^\top(B)}(u(q, r), [(\gamma, a)]) = R_B(r, [(\gamma, a)])$$

which concludes the proof. □

Remark 17. We have not needed the full assumption that ϕ is a fibration, only that the function ϕ when restricted to $\downarrow q \to \downarrow \phi(q)$ is *surjective* for any $q \in Q$. This is equivalent to requiring that $\phi : Q \to P$ is an *open* map when Q and P are equipped with the version of the Alexandrov topology where the opens are lower sets. Indeed, this is immediate from the fact that down sets of the form $\downarrow q$, called the principal ideals, form the basis of the Alexandrov topology. However if we were to consider universes we would likely also need the additional assumption requiring that the assignment $u(q, -)$ is a monotone function [6, Lemma 3.21]. ■

4. Modelling guarded dependent type theory

In this section we use the constructions of the previous section specialised to specific posets $\mathfrak{I}(\Delta)$ indexed by finite sets Δ, and reindexing by monotone functions $\mathfrak{I}(f)$ to get a model of guarded dependent type theory.

The posets $\mathfrak{I}(\Delta)$ are as in previous work [6], but we recall their definition and basic properties of functions $\mathfrak{I}(f)$ here.

Definition 18. Let Δ be a finite set of clocks. The poset $\mathfrak{I}(\Delta)$ has as elements pairs (E, δ) where E is an equivalence relation on Δ and $\delta : \Delta \to \mathbb{N}$ is a function respecting the equivalence relation. The ordering is defined such that $(E, \delta) \leq (E', \delta')$ if $E' \subseteq E$ and $\delta \leq \delta'$, i.e., E' is finer and for all $\kappa \in \Delta$ we have $\delta(\kappa) \leq \delta'(\kappa)$.

Given a function $f : \Delta_2 \to \Delta_1$ there is a function $\mathfrak{I}(f) : \mathfrak{I}(\Delta_1) \to \mathfrak{I}(\Delta_2)$ defined as

$$\mathfrak{I}(f)(E, \delta) = \left(f^*(E), \delta \circ f\right)$$

where $f^*(E)$ is the pullback of E along f, i.e., the relation

$$\left\{(\kappa, \kappa') \;\middle|\; (f(\kappa), f(\kappa')) \in E\right\}. \quad \blacksquare$$

The most important property of $\mathfrak{I}(f)$ is that it is a *fibration* as shown in previous work [6, Lemma 3.7].

Using these posets we define $\mathcal{GR}(\Delta) = \mathsf{PEqu}(\mathfrak{I}(\Delta))$ and for a function $f : \Delta_1 \to \Delta_2$ we define $\mathcal{GR}(f) : \mathcal{GR}(\Delta_1) \to \mathcal{GR}(\Delta_2)$ to be the functor $\mathfrak{I}(f)^\dagger$ defined in the previous section. Similarly we define $\mathcal{GR}^\top(\Delta) = \mathsf{UFam}(\mathfrak{I}(\Delta))$ and $\mathcal{GR}^\top(f) : \mathsf{UFam}(\mathfrak{I}(\Delta_1)) \to \mathsf{UFam}(\mathfrak{I}(\Delta_2))$ to be the functor $\mathfrak{I}(f)^\top$ defined in the previous section. We write p_Δ for the resulting fibration. For an object $\Gamma \in \mathcal{GR}(\Delta)$ we define $\mathcal{GR}_\Gamma(\Delta) = \mathsf{Type}_{\mathfrak{I}(\Delta)}(\Gamma)$. Note that $\mathcal{GR}_\Gamma(\Delta)$ is isomorphic in a trivial way to the fibre over Γ with respect to the fibration p_Δ. Because the diagram (1) commutes the functor $\mathcal{GR}^\top(f)$, for any $f : \Delta_1 \to \Delta_2$, restricts to a functor $\mathcal{GR}_\Gamma(\Delta_1) \to \mathcal{GR}_{\mathcal{GR}(f)(\Gamma)}(\Delta_2)$. We will write $\mathcal{GR}_\Gamma(f)$ for this functor.

4.1. Clock quantification

Let Δ be a finite set of clocks, κ a clock not in Δ and $\iota : \Delta \to \Delta, \kappa$ the inclusion. Let $\Gamma \in \mathcal{GR}(\Delta)$. We define a functor $\forall\kappa : \mathcal{GR}_{\mathcal{GR}(\iota)(\Gamma)}(\Delta, \kappa) \to \mathcal{GR}_\Gamma(\Delta)$ right adjoint to the functor $\mathcal{GR}_\Gamma(\iota)$. Its definition uses the function $\iota_n^! : \mathfrak{I}(\Delta) \to \mathfrak{I}(\Delta, \kappa)$ used in previous work [6, Lemmas 3.15, 3.16]. It is defined as

$$\iota_n^!(E, \delta) = (E \cup \{(\kappa, \kappa)\}, \delta[\kappa \mapsto n]).$$

The functor $\mathcal{GR}_\Gamma(\iota)$ maps an object A to the object $\forall\kappa A$ where

$$|\forall\kappa A| = |A|$$
$$R_{\forall\kappa A}\left((E, \delta), [\gamma]_{E,\delta}\right) = \bigcap_{n \in \mathbb{N}} R_A\left(\iota_n^!(E, \delta), [\gamma]_{\iota_n^!(E,\delta)}\right)$$

The function $\iota_n^!$ satisfies $\mathfrak{I}(\iota) \circ \iota_n^! = \mathrm{id}_{\mathfrak{I}(\Delta)}$ which ensures that the sets $[\gamma]_{E,\delta}$ and $[\gamma]_{\iota_n^!(E,\delta)}$ are in fact equal, so the relation $R_{\forall\kappa A}$ is well-defined and because PERs are closed under intersection it is also a PER.

The functor $\forall\kappa$ maps a morphism realised by f to the morphism realised by f. Note however that, again, the equivalence relations with which the morphisms are constructed are not the same at the source and target, so $\forall\kappa$ is neither full nor faithful in general.

Theorem 19. *The functor $\forall\kappa$ is right adjoint to $\mathcal{GR}_\Gamma(\iota)$. It additionally satisfies the following properties.*

- $\forall\kappa \circ \mathcal{GR}_\Gamma(\iota) = id$.
- *The unit of the adjunction is the identity.*
- *The counit of the adjunction at an object A is given by the equivalence class of the continuous function*

  $$\gamma \mapsto a \mapsto a.$$

  Note however that it is not the identity.
- *(The Beck–Chevalley condition for clock substitution.) Given $u : \Delta_1 \to \Delta_2$ and $\kappa_1 \notin \Delta_1$ and $\kappa_2 \notin \Delta_2$. Let $\hat{u} : \Delta_1, \kappa_1 \to \Delta_2, \kappa_2$ be the extension of u mapping κ_1 to κ_2. Then*

  $$\mathcal{GR}_\Gamma(u) \circ \forall\kappa_1 = \forall\kappa_2 \circ \mathcal{GR}_\Gamma\left(\hat{u}\right)$$

  and moreover the canonical morphism from left to right is the identity.

- *If* $[f] : \Gamma_1 \to \Gamma_2$ *is a morphism in* $\mathfrak{GR}(\Delta)$ *then*

$$f^* \circ \forall\kappa = \forall\kappa \circ (\mathfrak{GR}(\iota)([f]))^*.$$

This item shows that substitution in terms commutes with clock quantification as it should.

Proof. The first and second items follow from the property $\mathfrak{I}(\iota) \circ \iota_n^! = \mathrm{id}_{\mathfrak{I}(\Delta)}$.

The third item follows from the property [6, Lemma 3.16]

$$\iota^!_{\delta(\kappa)}(\mathfrak{I}(\iota)(E,\delta)) \geq (E,\delta).$$

The last two items follow from the fact that the underlying lattices never change and a simple computation showing the relations are preserved. □

4.2. The delay functor

Let Δ be a finite set of clocks, $\kappa \in \Delta$ and $\Gamma \in \mathfrak{GR}(\Delta)$. There is a functor $\blacktriangleright^\kappa_\Gamma : \mathfrak{GR}_\Gamma(\Delta) \to \mathfrak{GR}_\Gamma(\Delta)$. It maps the object A to the object $\blacktriangleright^\kappa_\Gamma(A)$ where

$$\left|\blacktriangleright^\kappa_\Gamma(A)\right| = |A|$$

$$R_{\blacktriangleright^\kappa_\Gamma A}\left((E,\delta),[\gamma]_{E,\delta}\right) = \begin{cases} |A| \times |A| & \text{if } \delta(\kappa) = 1 \\ R\left((E,\delta^{-\kappa}),[\gamma]_{(E,\delta^{-\kappa})}\right) & \text{otherwise} \end{cases}$$

where $\delta^{-\kappa}$ is defined as [6, Definition 3.11]

$$\delta^{-\kappa}(\kappa') = \begin{cases} \max\{1,\delta(\kappa)-1\} & \text{if} (\kappa,\kappa') \in E \\ \delta(\kappa') & \text{otherwise} \end{cases}$$

In particular it satisfies $(E,\delta^{-\kappa}) \leq (E,\delta)$ and the assignment is monotone in (E,δ) which shows that $R_{\blacktriangleright^\kappa A}$ is well-defined. The functor $\blacktriangleright^\kappa_\Gamma$ maps a morphism realised by f to the morphism realised by f.

There is a natural transformation $\mathrm{next}^{\Gamma,\kappa} : \mathrm{id} \to \blacktriangleright^\kappa_\Gamma$ which is realised, at an object A, by the continuous function $\gamma \mapsto a \mapsto a$.

Remark 20. This is the place where monotonicity of relations R_X is needed. Without it, next would not exist. Consequently we could not state and prove the fixed point property in Proposition 23 in general. ■

Theorem 21. *The functor* $\blacktriangleright^\kappa_\Gamma$ *we have defined satisfies the following properties.*

- $\forall\kappa \circ \blacktriangleright^\kappa_\Gamma = \forall\kappa$
- *If* $u : \Delta_1 \to \Delta_2$ *and* $\kappa \in \Delta_1$ *then*

$$\blacktriangleright^{f(\kappa)}_{\mathfrak{GR}(f)(\Gamma)} \circ \mathfrak{GR}_\Gamma(u) = \mathfrak{GR}_\Gamma(u) \circ \blacktriangleright^\kappa_\Gamma$$

- *If* $[f] : \Gamma_1 \to \Gamma_2$ *is a morphism in* $\mathfrak{GR}(\Delta)$ *then*

$$f^* \circ \blacktriangleright^\kappa_{\Gamma_1} = \blacktriangleright^\kappa_{\Gamma_2} \circ f^*.$$

The last item shows that substitution for term variables commutes over ▶ *correctly.*

Remark 22. It can be shown that for each κ the functor $\blacktriangleright^\kappa : \mathsf{PEqu}(P) \to \mathsf{PEqu}(P)$ (which corresponds to the functor $\blacktriangleright^\kappa_1$) is an applicative functor in the sense of [9]. In [7] the applicative functor rules are generalised using delayed substitutions in order for ▶ to behave well with respect to dependent products. The functors $\blacktriangleright^\kappa_\Gamma$ can be shown to validate these rules. The proofs are straightforward, albeit notationally heavy. ■

4.3. Fixed points

Let Δ be a finite set of clocks, $\kappa \in \Delta$, $\Gamma \in \mathfrak{GR}(\Delta)$ and $A \in \mathfrak{GR}_\Gamma(\Delta)$.

Proposition 23. *Let* $[g] : \blacktriangleright^\kappa_\Gamma A \to A$ *be a morphism. There is a unique morphism* $[f] : 1 \to A$ *such that*

$$[g] \circ \mathrm{next}^{\Gamma,\kappa}_A \circ [f] = [f].$$

Proof. Recall that 1 is the terminal object in $\mathfrak{GR}_\Gamma(\Delta)$. The underlying lattice is $2^\emptyset$ and the relations are total relations on this lattice. To show that a fixed point exists we first construct a continuous function $f : |\Gamma| \to |1| \to |A|$ satisfying $g\gamma(f\gamma\emptyset) = f\gamma\emptyset$ for all $\gamma \in |\Gamma|$. This is easy to do using the fact that taking least fixed points is a continuous operation [10, Proposition 3.14], i.e., we define

$$f\gamma\emptyset = \bigvee_{n=0}^{\infty} (g\gamma)^n(\bot).$$

Using this realiser we have $[g] \circ \mathrm{next}^{\Gamma,\kappa} \circ [f] = [f]$, provided that f is non-expansive, i.e., that f is a realiser. We show this now. We need to show that for any $(E,\delta) \in \mathfrak{I}(\Delta)$ and any $(\gamma,\gamma') \in R_\Gamma(E,\delta)$ we have

$$(f\gamma\emptyset, f\gamma'\emptyset) \in R_A((E,\delta),[\gamma]_{E,\delta}). \tag{5}$$

We proceed by induction on $\delta(\kappa)$. To be more precise, the statement we are proving by induction is that for any $n \in \mathbb{N}$ and for any (E,δ) satisfying $\delta(\kappa) = n$ the property (5) holds.

If $\delta(\kappa) = 1$ then by the definition of $R_{\blacktriangleright^\kappa_\Gamma A}((E,\delta),[\gamma]_{E,\delta})$ we have

$$(f\gamma\emptyset, f\gamma'\emptyset) \in R_{\blacktriangleright^\kappa_\Gamma A}((E,\delta),[\gamma]_{E,\delta})$$

and so, because g is non-expansive,

$$(g\gamma(f\gamma\emptyset), g\gamma'(f\gamma'\emptyset)) \in R_A((E,\delta),[\gamma]_{E,\delta})$$

but, e.g., $g\gamma(f\gamma\emptyset) = f\gamma\emptyset$ so we have (5). The induction step should now be clear.

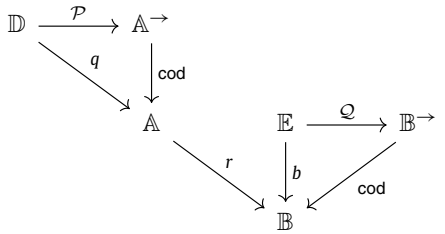
Uniqueness of the constructed fixed point follows by an analogous induction. □

Note that the underlying realisers might have many fixed points, but what we have shown is that all of them are equivalent according to the family of PERs given.

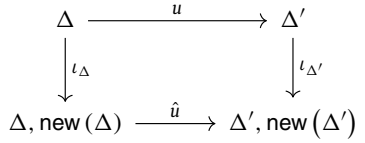
4.4. Summary of the model

Theorems 14 and 16 together with [6, Lemma 3.7] ensure that the pair $(\mathfrak{GR}^\top(f), \mathfrak{GR}(f))$ is a morphism of split fibrations preserving products and coproducts on the nose.

We can organise the constructions above into the general framework of fibrations. In particular we construct a PDTT-structure [8, Definition 11.3.1]. The model GuardedEqu can be organised in the following diagram of fibrations and comprehension categories.



We now explain what all of these are. The base category $\mathbb{B}$ and the category $\mathbb{E}$ are the same. They have as objects finite subsets of the set of clocks CV. A morphism $f : \Delta_1 \to \Delta_2$ is a function $\Delta_2 \to \Delta_1$ (note the reversal). So, briefly, we may describe $\mathbb{B}$ as the opposite of the category of finite sets and functions. The functor b is the identity functor and $\mathcal{Q}$ is defined as follows. We assume we have a function $\mathsf{new}(-)$ that given a finite set of clocks returns a clock not already in that set. The comprehension $\mathcal{Q}$ maps a set Δ to the inclusion ι_Δ from Δ to $\Delta, \mathsf{new}(\Delta)$. It maps a function $u : \Delta \to \Delta'$ to the commutative square



where $\hat{u}$ is the extension of u mapping $\mathsf{new}(\Delta)$ to $\mathsf{new}(\Delta')$. Thus we easily see that $\mathsf{cod} \circ \mathcal{Q} = \mathrm{id}_\mathbb{B} = b$.

Next, the category $\mathbb{A}$ has as objects pairs (Δ,Γ) where Δ is a finite set of clocks and Γ is an object of $\mathfrak{GR}(\Delta)$. A morphism $(\Delta_1,\Gamma_1) \to (\Delta_2,\Gamma_2)$ is a pair $(u,[f])$ where u is a function $\Delta_2 \to \Delta_1$, i.e., a morphism in $\mathbb{B}$ from Δ_1 to Δ_2, and

$[f]$ is a morphism $\Gamma_1 \to \mathfrak{GR}(u)(\Gamma_2)$ in $\mathfrak{GR}(\Delta_1)$. Equivalently, this is the total category of the Grothendieck construction applied to the functor $\mathfrak{GR}$, the evident projection $r : \mathbb{A} \to \mathbb{B}$ is a split fibration.

Finally, the category $\mathbb{D}$ can be briefly described as the total category of the Grothendieck construction applied to the functor $\mathfrak{GR}^{\top}(-)$. Concretely, its objects are pairs $(\Delta, (\Gamma, A))$ where Δ is a finite set of clocks and $(\Gamma, A) \in \mathfrak{GR}^{\top}(\Delta)$. A morphism

$$(\Delta_1, (\Gamma_1, A_1)) \to (\Delta_2, (\Gamma_2, A_2))$$

is pair morphisms $(u, ([f], [g]))$ where u is a function $\Delta_2 \to \Delta_1$ and $([f], [g])$ is a morphism in $\mathfrak{GR}^{\top}(\Delta_1)$ from (Γ_1, A_1) to $\mathfrak{GR}^{\top}(u)(\Gamma_2, A_2)$. The functor q maps the pair $(\Delta, (\Gamma, A))$ to the pair (Δ, Γ). Thus we have that $r \circ q$ is the projection associated with the Grothendieck construction. Moreover, q is also a split fibration. Indeed, the cartesian lifting of

$$(u, [f]) : (\Delta_1, \Gamma_1) \to q(\Delta_2, (\Gamma_2, A_2))$$

is the morphism

$$(u, ([f], [g])) : \left(\Delta_1, \left(\Gamma_1, f^*\left(\mathfrak{GR}_{\Gamma_2}(u)(A_2)\right)\right)\right) \to (\Delta_2, (\Gamma_2, A_2))$$

and $g : \Gamma_1 \to A_2 \to A_2$ is the function

$$\gamma \mapsto a \mapsto a.$$

Recall that $p \in \mathfrak{I}(\Delta_1)$ and so $\mathfrak{I}(u)(p) \in \mathfrak{I}(\Delta_2)$ as required of a PDTT-structure. Note that q can also be seen as arising from the Grothendieck construction mapping the object (Δ, Γ) to the category $\mathsf{Type}_{\mathfrak{I}(\Delta)}(\Gamma)$.

And at last, the comprehension $\mathcal{P}$ maps the object $(\Delta, (\Gamma, A))$ to the morphism (object of the arrow category) $(\mathrm{id}_\Delta, \pi_A)$. Hence we see immediately that all the projections are indeed r-vertical as required.

5. Continuity

In this section we show that we can construct final coalgebras of polynomial functors using solutions of guarded domain equations. In particular when we start with a polynomial functor on $\mathfrak{GR}(\emptyset)$ we can construct its final coalgebra M. By using the fact that the category of T_0 topological spaces is a full subcategory of $\mathfrak{GR}(\emptyset)$ we show that if the polynomial functor restricts to topological spaces the final coalgebra is also a topological space with the expected topology. We detail this on the case of streams of elements of some set A, where we show that functions f from streams to streams definable in the type theory are such that any finite prefix of $f(xs)$ depends only on a finite prefix of xs.

Let $A, B \in \mathfrak{GR}(\emptyset)$ and $[f] : B \to A$ a morphism. Such a morphism defines a polynomial functor $\mathfrak{P}_f : \mathfrak{GR}(\emptyset) \to \mathfrak{GR}(\emptyset)$. Abusing notation this functor can be described as

$$\mathfrak{P}_f(X) = \sum_{a:A} X^{f^{-1}(a)},$$

or more precisely, $\mathfrak{P}_f(X)$ is the total space, i.e., domain, of the exponential π_2^f in the slice over A, where $\pi_2 : X \times A \to A$ is the second projection.

As an example we have the object of streams whose elements are of type A. We take the morphism $[f] : A \to A$ to be the identity. We then have

$$\mathfrak{P}_{\mathrm{id}_A}(X) = \sum_{a:A} X^1 \cong \sum_{a:A} X = A \times X.$$

Alternatively, without abusing notation, recall that the identity on A is the terminal object in the slice category over A. Hence $\pi_2^{\mathrm{id}_A} \cong \pi_2$ as object of the slice over A and the total space of π_2 is precisely $X \times A$. The object of streams is defined to be the final coalgebra of $\mathfrak{P}_{\mathrm{id}_A}$.

A more interesting example is the type of possibly infinite lists of type A. We take the morphism $[f]$ to be the inclusion of A to $1 + A$. Then, abusing notation,

$$\mathfrak{P}_f(X) \cong \sum_{x:1} X^{f^{-1}(x)} + \sum_{x:A} X^{f^{-1}(x)} \cong \sum_{x:1} X^0 + \sum_{x:A} X^1 \cong 1 + A \times X$$

so $\mathfrak{P}_f$ indeed describes the shapes of lists. The object of lists is defined to be the initial algebra, if it exists, of this functor and the object of potentially infinite lists is defined to be the final coalgebra of this functor.

Bauer and Birkedal [11] describe the polynomial functor $\mathfrak{P}_f : \mathfrak{GR}(\emptyset) \to \mathfrak{GR}(\emptyset)$ very concretely as follows. It maps an object X to the object $\left(|A| \times (|B| \to |X|), R_{\mathfrak{P}_f(X)}\right)$ where $(a, u) \approx^{\star}_{\mathfrak{P}_f(X)} (a', u')$ if and only if $(a, a') \in R_A(\star)$ and

$$\forall (b, b') \in R_B(\star), (f(b), a) \in R_A(\star) \Rightarrow (u(b), u'(b')) \in R_X(\star).$$

We have used $\star$ to denote the unique element of $\mathfrak{I}(\emptyset)$. This functor can be lifted to the functor $\mathfrak{P}_f^\kappa$ on $\mathfrak{GR}(\kappa)$ for any clock κ by replacing f by $\mathfrak{GR}(\iota)(f)$. Concretely the functor maps the object X of $\mathfrak{GR}(\kappa)$ to the object whose underlying lattice is the same $|A| \times (|B| \to |X|)$. To describe the relations we will use the fact that there is only one equivalence relation on the set $\{\kappa\}$ to represent pairs $(E, \delta) \in \mathfrak{I}(\kappa)$ by a single natural number n. The relation $R_{\mathfrak{P}_f^\kappa(X)}$ at n relates (a, u) and (a', u') if and only if $(a, a') \in R_A(\star)$ and

$$\forall (b, b') \in R_B(\star), (f(b), a) \in R_A(\star) \Rightarrow (u(b), u'(b')) \in R_X(n)$$

The lifting is easily seen to satisfy the property

$$\mathfrak{GR}(\iota) \circ \mathfrak{P}_f = \mathfrak{P}_f^\kappa \circ \mathfrak{GR}(\iota).$$

Next, Bauer and Birkedal [11] construct the algebraic lattice

$$M = \prod_{i=0}^{\infty} (|B|^i \to |A|)$$

together with an isomorphism $\phi : |A| \times (|B| \to M) \to M$ (in AlgLat) defined component-wise (since M is a product, this makes sense). We use $\pi_i : M \to (|B|^i \to |A|)$ to denote projections.

$$\pi_0(\phi(a, u)) = \star \mapsto a$$
$$\pi_{i+1}(\phi(a, u)) = (b, \vec{b}) \in |B|^{i+1} \mapsto \pi_i(u(b))(\vec{b})$$

where we have assumed in this case that products associate to the right.

The inverse $\psi : M \to |A| \times (|B| \to M)$ to ϕ can also be given explicitly as

$$\psi(m) = \left(\pi_0(m)(\star), b \mapsto \left\{\vec{b} \mapsto \pi_{i+1}(m)(b, \vec{b})\right\}_{i=0}^{\infty}\right).$$

We will write $\psi_1 = \pi_1 \circ \psi : M \to |A|$ and $\psi_2 : \pi_2 \circ \psi : M \to (|B| \to M)$.

Next we construct the solution to $\mathfrak{P}_f^\kappa\left(\blacktriangleright^\kappa X\right) \cong X$. The underlying lattice of X is the lattice M constructed above. The family of relations R_X is defined by induction on n using the isomorphisms ϕ and ψ. We define $m \approx_X^n m'$ if and only if $(\psi_1(m), \psi_1(m')) \in R_A(\star)$ and

$$\forall (b, b') \in R_B(\star), (f(b), \psi_1(m)) \in R_A(\star) \Rightarrow (\psi_2(m)(b), \psi_2(m')(b')) \in \bigcap_{k<n} R_X(k)$$

or alternatively

$$R_X(1) = \left\{(m, m') \;\middle|\; (\psi_1(m), \psi_1(m')) \in R_A(\star)\right\}$$

and $m \approx_X^{n+1} m'$ if and only if $(\psi_1(m), \psi_1(m')) \in R_A(\star)$ and

$$\forall (b, b') \in R_B(\star), (f(b), \psi_1(m)) \in R_A(\star) \Rightarrow (\psi_2(m)(b), \psi_2(m')(b')) \in R_X(n).$$

With these definitions it is easy to see that the functions ϕ and ψ are non-expansive, and so they give rise to an isomorphism in $\mathfrak{GR}(\kappa)$.

Finally, applying $\forall\kappa$ to the object X we get the relation

$$R_M = \bigcap_{n=1}^{\infty} R_X(n).$$

Observe that the construction of R_X is precisely the chain $\Phi^n(\top)$ where Φ is the operator defined by Bauer and Birkedal [11]. Because Φ commutes with non-empty intersections, i.e.,

$$\Phi\left(\bigcap_{i \in I} R_i\right) = \bigcap_{i \in I} \Phi(R_i)$$

for all inhabited I, we have that R_M defined above is precisely the largest fixed point of Φ by Kleene's fixed point theorem.

Hence (M, R_M) is the final coalgebra of $\mathfrak{P}_f$, as needed.

To recap, what we have shown is that we can construct final coalgebras of polynomial functors using solutions of guarded domain equations.

Finally, we get more than from the presheaf models [6,5]. In these models M would just be a set with the property that it is the final coalgebra of the functor $\mathfrak{P}_f$ and we get no useful information on functions $M \to M$ definable in the type

theory, i.e., in the model functions $M \to M$ are all functions. Using GuardedEqu we get the additional property that every morphism of type $M \to M$ definable in the type theory is realised by a continuous function.

Let us look at a concrete example to see that this property gives useful information. Let A be some *set* considered as an algebraic lattice $A_{\top,\bot}$ by adding two elements $\top$ and $\bot$ with $\top$ being the top element and $\bot$ the bottom and otherwise the order is discrete. We consider A as an object of $\mathfrak{GR}(\emptyset)$ by equipping $A_{\top,\bot}$ with the PER R_A which is the identity relation on A, but *does not* relate $\bot$ or $\top$ to anything, including themselves.

The type of streams is given as the final coalgebra of the polynomial functor $\mathfrak{P}_{\mathrm{id}_{A_{\top,\bot}}}$. This is not a very convenient description so we provide another description of the type of guarded streams of type A. The underlying lattice is the product lattice $\prod_{i=0}^{\infty} A_{\top,\bot}$ and the relations are

$$R_{\mathrm{Str}^{\mathrm{g}}(A)}(n) = \left\{ \left(\{x_i\}_{i=0}^{\infty}, \{y_i\}_{i=0}^{\infty} \right) \;\middle|\; \forall k < n, (x_k, y_k) \in R_A \right\}$$

Note that n starts at 1, but the indexing of the product starts at 0, which is the reason for using the strict order relation $k < n$.

There are two continuous functions $\alpha : \prod_{i=0}^{\infty} A_{\top,\bot} \to M$ and β in the converse direction, where M is the lattice constructed above, but specialised for the functor $\mathfrak{P}_{\mathrm{id}_{A_{\top,\bot}}}$. So

$$M = \prod_{i=0}^{\infty} \left(A_{\top,\bot}^{i} \to A_{\top,\bot} \right).$$

The function α is defined as

$$\alpha(s) = \{\_ \mapsto s_i\}_{i=0}^{\infty}$$

where s_i is the i-th element of the string. The function β is defined by induction. Given $m \in M$ define the stream $\beta(m)$ by induction

$$\begin{aligned} \beta(m)_0 &= \psi_1(m) \\ \beta(m)_{n+1} &= \psi_1\left(\psi_2(m)\left(\beta(m)_n\right)\right). \end{aligned}$$

Then it is easy to see that these functions are continuous and that $\beta \circ \alpha = \mathrm{id}$. In contrast $\alpha \circ \beta$ is not the identity. However an easy calculation shows α and β are non-expansive with respect to the equivalence relations given and that $\alpha \circ \beta \sim \mathrm{id}$.

Thus because $\forall\kappa$ is a functor the lattice $\prod_{i=0}^{\infty} A_{\top,\bot}$ together with the partial equivalence relation relating only equal streams where none of the elements are $\top$ or $\bot$ is the final coalgebra for the functor $\mathfrak{P}_{\mathrm{id}_{A_{\top,\bot}}}$.

Using the equivalence between partial equilogical spaces and equilogical spaces and using the fact that Top_0 is a full subcategory of the category of equilogical spaces [3] we have the following proposition.

Proposition 24. *Let $\mathbb{S}$ be the denotation of the type of streams of elements of some set X. Any closed term t of type $\mathbb{S} \to \mathbb{S}$ is interpreted as a continuous function f which satisfies the property that for any $xs \in \mathbb{S}$, any finite prefix of the output $f(xs)$ depends only on a finite prefix of the input xs.*

Proof sketch. By using the fact that the category of topological spaces is a full subcategory of $\mathfrak{GR}(\emptyset)$ we have that functions on streams are in bijective correspondence with continuous functions on the topological space $\prod_{i=0}^{\infty} A$, where A is equipped with the discrete topology. A standard topological exercise then shows that these are precisely the functions with the property that the first m elements of an output stream only depend on the first n, for some n, elements of the input stream. □

6. Discussion

Remarks about GuardedEqu A natural question to ask is what is the relationship between $\mathsf{PEqu}(P)$ and presheaves on P valued in PEqu. Clearly if P is the singleton poset these are equivalent. In general, one can show that the category of PEqu-valued presheaves $[P^{\mathrm{op}}, \mathsf{PEqu}]$ is a reflective subcategory of $\mathsf{PEqu}(P)$ and the reflector F is faithful, but in most cases it is not full. In particular if Δ is inhabited and P is $\mathfrak{I}(\Delta)$ the reflector is not full. Based on this we *conjecture* that the categories are not equivalent, however we do not have a proof of this. However even if they were equivalent, the presentation with families of PERs is simpler to work with and does not have problems with coherence inherent in the presheaf presentation.

An inspection of the constructions used shows that the use of algebraic lattices as realisers is not essential. For instance it could be replaced by other categories of domains (such as complete pointed partial orders, Scott domains, or countably based algebraic lattices) or we could consider only PERs on a reflexive domain. The important properties are that the category is cartesian closed and that its endomorphisms have fixed points. Section 5 requires more, namely the existence of certain countable limits.

Ordered families of equivalences One can also model guarded recursive functions using (complete) ordered families of equivalences [12]. An important difference to that approach is that we require no completeness conditions on our families of PERs. The reason is precisely that the underlying category of realisers has fixed points. In contrast, functions between sets do not necessarily have fixed points, so the completeness conditions on ordered families of equivalences in [12] are needed to ensure that suitably contractive functions have fixed points.

Rule formats Guarded dependent type theory can be thought of in particular as a rich "rule format" for defining functions on coinductive types, *cf.* the work of Rutten [13] who defines a rule format for defining non-expansive stream functions. Since guarded dependent type theory is an extension of dependent type theory with types which allow us to express "non-expansiveness", the rule format is modular in the sense of [14]. Indeed in [14] it is shown that, if one defines a function, e.g., from streams to streams using a restricted set of rules then the set of rules allowed in the construction of stream functions can be extended with the newly defined function. This corresponds to the observation that the newly defined function is non-expansive, which in guarded dependent type theory corresponds to a function from *guarded* streams to *guarded* streams. Using clock quantifiers affords more expressiveness and allows us to distinguish in the type theory between functions which can be used in recursive definitions of streams (and stream functions) and functions such as the tail function which can only be used in a much more restricted way. The model constructed in this paper shows that guarded dependent type theory can also be seen as a rule format for defining continuous functions on streams and, more generally, on final coalgebras for arbitrary polynomial functors.

Acknowledgements

This research was supported in part by the ModuRes Sapere Aude Advanced Grant (no. 12-133344) from The Danish Council for Independent Research for the Natural Sciences (FNU). Aleš Bizjak was supported in part by a Microsoft Research PhD grant.

References

[1] L. Birkedal, R.E. Møgelberg, J. Schwinghammer, K. Støvring, First steps in synthetic guarded domain theory: step-indexing in the topos of trees, Log. Methods Comput. Sci. 8 (4) (2012), 45 pp.
[2] K. Svendsen, L. Birkedal, Impredicative concurrent abstract predicates, in: Z. Shao (Ed.), Programming Languages and Systems, in: Lecture Notes in Comput. Sci., vol. 8410, Springer, Berlin Heidelberg, 2014, pp. 149–168.
[3] A. Bauer, L. Birkedal, D.S. Scott, Equilogical spaces, Theoret. Comput. Sci. 315 (1) (2004) 35–59.
[4] R. Atkey, C. McBride, Productive coprogramming with guarded recursion, in: Proceedings of the 18th ACM SIGPLAN International Conference on Functional Programming, ICFP '13, ACM, New York, NY, USA, 2013, pp. 197–208.
[5] R.E. Møgelberg, A type theory for productive coprogramming via guarded recursion, in: Proceedings of the Joint Meeting of the Twenty-Third EACSL Annual Conference on Computer Science Logic (CSL) and the Twenty-Ninth Annual ACM/IEEE Symposium on Logic in Computer Science (LICS), CSL-LICS '14, ACM, New York, NY, USA, 2014, 71:1–71:10.
[6] A. Bizjak, R.E. Møgelberg, A model of guarded recursion with clock synchronisation, Electron. Notes Theor. Comput. Sci. 319 (2015) 83–101.
[7] A. Bizjak, H.B. Grathwohl, R. Clouston, R.E. Møgelberg, L. Birkedal, Guarded dependent type theory with coinductive types, in: B. Jacobs, C. Löding (Eds.), Foundations of Software Science and Computation Structures: 19th International Conference, FOSSACS 2016, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2016, Proceedings, Eindhoven, The Netherlands, April 2–8, 2016, Springer, Berlin, Heidelberg, 2016, pp. 20–35.
[8] B. Jacobs, Categorical Logic and Type Theory, Stud. Logic Found. Math., vol. 141, North Holland, Amsterdam, 1999.
[9] C. McBride, R. Paterson, Applicative programming with effects, J. Funct. Programming 18 (1) (2008) 1–13.
[10] D. Scott, Continuous lattices, in: F. Lawvere (Ed.), Toposes, Algebraic Geometry and Logic, in: Lecture Notes in Math., vol. 274, Springer, Berlin Heidelberg, 1972, pp. 97–136.
[11] A. Bauer, L. Birkedal, W-types and M-types in equilogical spaces, May 1999, manuscript, http://users-cs.au.dk/birke/papers/w-m-types-in-equ.pdf.
[12] P.D. Gianantonio, M. Miculan, A unifying approach to recursive and co-recursive definitions, in: Types for Proofs and Programs, Second International Workshop, TYPES 2002, Berg en Dal, The Netherlands, April 24–28, 2002, pp. 148–161, Selected Papers.
[13] J.J.M.M. Rutten, Behavioural differential equations: a coinductive calculus of streams, automata, and power series, Theoret. Comput. Sci. 308 (1–3) (2003) 1–53.
[14] S. Milius, L.S. Moss, D. Schwencke, Abstract GSOS rules and a modular treatment of recursive definitions, Log. Methods Comput. Sci. 9 (3:28) (2013), 52 pp.