Completeness of Conversion between Reactive Programs for Ultrametric Models

Paula Severi and Fer-Jan de Vries

Computer Science, University of Leicester
{ps56,fdv1}@mcs.le.ac.uk

Abstract. In 1970 Friedman proved completeness of beta eta conversion in the simply-typed lambda calculus for the set-theoretical model. Recently Krishnaswami and Benton have captured the essence of Hudak's reactive programs in an extension of simply typed lambda calculus with causal streams and a temporal modality and provided this typed lambda calculus for reactive programs with a sound ultrametric semantics.
We show that beta eta conversion in the typed lambda calculus of reactive programs is complete for the ultrametric model.

1 Introduction

Krishnaswami and Benton have recently introduced a typed lambda calculus for reactive programs [1, 2]. Their basic idea was to have "a lambda calculus with types not only for data, but also indexed with time." This led them to extend simply typed lambda calculus with causal streams and a temporal modality and secondly, to define an ultrametric semantics for reactive programs. In the ultrametric model, types are interpreted as ultrametric spaces and terms as non-expansive maps [1, 3, 4]. They demonstrated the soundness of this extension for the ultrametric semantics.

This raises the natural question of *completeness*. In this paper we show that two terms typable in the calculus of reactive programs are $\beta\eta$-convertible if and only if they have the same interpretation in the model of ultrametric spaces.

Completeness has been well studied for simply typed lambda calculus. It has been proved for the set-theoretical model [5], the model of CPOs and the model of modest sets [6, 7]. Towards completeness for the *ultrametric semantics*, we introduce the notions of *step-indexed applicative structure* and *Henkin model for reactive programs*. We show that the term model (consisting of reactive programs modulo conversion) and the ultrametric model can be seen as step-indexed applicative structures and also as Henkin models for reactive programs. Since for the ultrametric model, a stream is a function on natural numbers, we need a strong notion of extensionality that requires that two streams are equal if all their components are equal. Strong extensionality of the term model is not so easy to prove. It does not follow immediately from the η-rule but from the fact that our calculus is confluent and strongly normalising.

Actually, we show two completeness results. The first one, called *completeness (of $\beta\eta$-conversion) for Henkin models*, says that there exists a Henkin model for

reactive programs satisfying exactly the theory of $\beta\eta$-conversion. The second one, mentioned before, is about *completeness (of $\beta\eta$-conversion) for the ultrametric model*. The latter is proved by constructing a partial surjective step-indexed logical relation between the ultrametric model and the term model.

One interesting aspect of our paper is that we consider (ultra) metric spaces in a proof of completeness. We show that on the term model of the typed lambda calculus for reactive programs an ultrametric d can be defined for which the equivalence classes of terms of type $\sigma \to \tau$ are *non-expansive*, i.e. for M of type $\sigma \to \tau$ we have that $d([MP],[MQ]) \leq d([P],[Q])$.

This paper is organised as follows. Section 2 defines the typed lambda calculus for reactive programs. Section 3 introduces the notions of (step-indexed) applicative structure, Henkin model and (step-indexed) logical relation for reactive programs. Section 4 proves strong normalisation and confluence. Section 5 defines the term model and proves completeness for Henkin models. Section 6 shows that the model of ultrametric spaces is a Henkin model. Section 7 defines an ultrametric on the term model and shows that this metric is well-behaved. Section 8 shows completeness for the ultrametric model.

2 Typed Lambda Calculus for Reactive Programs

We recall the typed lambda calculus λ^{RP} for reactive programs as defined in [1]. It comes with a syntax, rewriting rules and typing rules.

Definition 1 (Syntax for reactive programs). *We define the set* P *of reactive programs (or terms) and the set* T *of types as follows.*

$$\begin{array}{l} \mathsf{P} \ni M ::= x \mid \mathsf{hd}(M) \mid \mathsf{tl}(M) \mid \mathsf{cons}(M,M) \mid \mathsf{await}(M) \mid \circ(M) \mid \lambda x{:}\sigma.M \mid MM \\ \mathsf{T} \ni \sigma \ ::= \mathsf{b} \mid (\sigma \to \sigma) \mid \bullet\sigma \mid \mathsf{S}(\sigma) \end{array}$$

where the parameter x ranges over a set V *of variables,* b *over a set* B *of basic types. A type declaration is a statement of the form $x :_i \tau$. A context is a finite set of type declarations with only distinct variables as subjects.*

Definition 2 (Reduction for reactive programs). *The β-rule is defined by:*

$$\begin{array}{llll} (\lambda x{:}\sigma.N)M & \to N[x := M]\ (\beta) & \mathsf{await}(\circ(M)) & \to M\ (\beta) \\ \mathsf{hd}(\mathsf{cons}(M,N)) & \to M\ (\beta) & \mathsf{tl}(\mathsf{cons}(M,N)) & \to N\ (\beta) \end{array}$$

The η-rule is defined by:

$$\begin{array}{lll} \lambda x{:}\sigma.Mx & \to M,\ \textit{if } x \notin \mathsf{fv}(M)\ (\eta) & \circ(\mathsf{await}(M)) \to M\ (\eta) \\ \mathsf{cons}(\mathsf{hd}(M),\mathsf{tl}(M)) & \to M\ (\eta) & \end{array}$$

Let $\rho \in \{\beta, \eta, \beta\eta\}$. The relation $\to_\rho$ is defined as the smallest relation on P *that is closed under under contexts and the ρ-rule(s). The reflexive and transitive closure of $\to_\rho$ is denoted by $\twoheadrightarrow_\rho$. The reflexive, symmetric and transitive closure of $\to_\rho$ is denoted by $=_\rho$, called* ρ-conversion. *The ρ-normal form of a term M is N, if $M \twoheadrightarrow_\rho N$ and N is in ρ-normal form. If the ρ-normal form of M exists, we denote it by $\mathsf{nf}_\rho(M)$.*

Definition 3 (Typing rules for reactive programs). *A type declaration is a statement of the form $x :_i \tau$. A context is a finite set of type declarations with only distinct variables as subjects. A type declaration $M :_i \tau$ is derivable from the context Γ, if the typing judgement $\Gamma \vdash M :_i \sigma$ can be derived from the following typing rules:*

$$\frac{x{:_i}\sigma \in \Gamma \quad j \geq i}{\Gamma \vdash x :_j \sigma}\,(\mathsf{var}) \qquad \frac{\Gamma \vdash M :_i \mathsf{S}(\sigma)}{\Gamma \vdash \mathsf{hd}(M) :_i \sigma}\,(\mathsf{head})$$

$$\frac{\Gamma \vdash M :_i \mathsf{S}(\sigma)}{\Gamma \vdash \mathsf{tl}(M) :_{i+1} \mathsf{S}(\sigma)}\,(\mathsf{tail}) \qquad \frac{\Gamma \vdash M :_i \sigma \quad \Gamma \vdash N :_{i+1} \mathsf{S}(\sigma)}{\Gamma \vdash \mathsf{cons}(M,N) :_i \mathsf{S}(\sigma)}\,(\mathsf{cons})$$

$$\frac{\Gamma, x{:_i}\sigma \vdash N :_i \tau}{\Gamma \vdash \lambda x{:}\sigma.N :_i (\sigma \to \tau)}\,(\to \mathsf{I}) \qquad \frac{\Gamma \vdash M :_{i+1} \sigma}{\Gamma \vdash \circ(M) :_i \bullet\sigma}\,(\bullet\mathsf{I})$$

$$\frac{\Gamma \vdash N :_i (\sigma \to \tau) \quad \Gamma \vdash M :_i \sigma}{\Gamma \vdash NM :_i \tau}\,(\to \mathsf{E}) \qquad \frac{\Gamma \vdash M :_i \bullet\sigma}{\Gamma \vdash \mathsf{await}(M) :_{i+1} \sigma}\,(\bullet\mathsf{E})$$

If a judgement $\Gamma \vdash M :_i \sigma$ is derivable from these rules, we call M a typable term.

The intuition is that $M :_i \sigma$ expresses that at time stamp i we know about the existence of a term M with type σ. If time stamp 0 represents 'now', then time stamp i represents 'i steps from now into the future.' To observe the tail N of a stream $\mathsf{cons}(M,N)$ of which we can see the head M now, we must wait one time step. We cannot force the future into the present.

Lemma 1 (Time adjustment [1]). *If $\Gamma, \Delta \vdash M :_i \sigma$ then $\Gamma, \Delta_{+n} \vdash M :_{i+n} \sigma$, where Δ_{+n} is obtained from Δ by raising the indexing time by n in all type declarations in Δ. Moreover, the derivations of $\Gamma, \Delta \vdash M :_i \sigma$ and $\Gamma, \Delta_{+n} \vdash M :_{i+n} \sigma$ have the same size.*

The following lemma is proved by induction on the derivation.

Lemma 2 (Subject reduction for reactive programs). *If $\Gamma \vdash M :_i \sigma$ and $M \twoheadrightarrow_{\beta\eta} N$ then $\Gamma \vdash N :_i \sigma$.*

3 Applicative Structures, Henkin Models and Logical Relations for Reactive Programs

In this section, we extend the notions of applicative structure, Henkin model and logical relations as defined for the simply typed lambda calculus, e.g., [7], to reactive programs. The time indices $i \in \mathbb{N}$ will play a similarly crucial role as they did in the typing rules for reactive programs.

Definition 4 (Applicative structures for reactive programs). *A (step-indexed) applicative structure for reactive programs is a tuple*

$$\mathcal{A} = \langle \{\mathcal{A}_i^\sigma\}, \{\delta_{ij}^\sigma\}, \{\mathit{hd}_i^\sigma, \mathit{tl}_i^\sigma, \mathit{await}_i^\sigma, \mathit{app}_i^{\sigma\tau}\} \rangle$$

of families of sets and functions indexed by types from T *such that for all* $\sigma, \tau \in \mathsf{T}$ *and all* $i, j \in \mathbb{N}$ *with* $i \leq j$ *we have:*

1. $\mathcal{A}_i^\sigma$ *is a set,*
2. $\delta_{ij}^\sigma \in \mathcal{A}_i^\sigma \to \mathcal{A}_j^\sigma$ *(expressing "delay"),*
3. $\delta_{ii}^\sigma = id : \mathcal{A}_i^\sigma \to \mathcal{A}_i^\sigma$ *and* $\delta_{jk}^\sigma \circ \delta_{ij}^\sigma = \delta_{ik}^\sigma$,
4. $\mathit{hd}_i^\sigma \in \mathcal{A}_i^{\mathsf{S}(\sigma)} \to \mathcal{A}_i^\sigma$, $\mathit{tl}_i^\sigma \in \mathcal{A}_i^{\mathsf{S}(\sigma)} \to \mathcal{A}_{i+1}^{\mathsf{S}(\sigma)}$,
   $\mathit{await}_i^\sigma \in \mathcal{A}_i^{\bullet\sigma} \to \mathcal{A}_{i+1}^\sigma$, $\mathit{app}_i^{\sigma\tau} \in \mathcal{A}_i^{\sigma\to\tau} \to \mathcal{A}_i^\sigma \to \mathcal{A}_i^\tau$.
5. $\delta_{ij}^\sigma(\mathit{hd}_i^\sigma(a)) = \mathit{hd}_j^\sigma(\delta_{ij}^\sigma(a))$ *for all* $a \in \mathcal{A}_i^{\mathsf{S}(\sigma)}$,
   $\delta_{(i+1)j}^\sigma(\mathit{tl}_i^\sigma(a)) = \mathit{tl}_j^\sigma(\delta_{ij}^\sigma(a))$ *for all* $a \in \mathcal{A}_i^{\mathsf{S}(\sigma)}$ *and* $i+1 \leq j$,
   $\delta_{(i+1)j}^\sigma(\mathit{await}_i^\sigma(a)) = \mathit{await}_j^\sigma(\delta_{ij}^\sigma(a))$ *for all* $a \in \mathcal{A}_i^{\bullet\sigma}$ *and* $i+1 \leq j$,
   $\delta_{ij}^\tau(\mathit{app}_i^{\sigma\tau}(f, b)) = \mathit{app}_j^{\sigma\tau}((\delta_{ij}^{\sigma\to\tau} f), (\delta_{ij}^\sigma b))$ *for all* $f \in \mathcal{A}_i^{\sigma\to\tau}$, $b \in \mathcal{A}_i^\sigma$.

To define extensional applicative structures, we have to define when the element of all three kind of types $\mathcal{A}_i^{\sigma\to\tau}$, $\mathcal{A}_i^{\bullet\sigma}$ and $\mathcal{A}_i^{\mathsf{S}(\sigma)}$ are extensional. For the first two this is straightforward. However for the latter we consider a strong version of extensionality that views streams as functions from natural numbers: two streams are equal if all their components are equal.

Definition 5 (Extensional applicative structure for reactive programs). *We say that an applicative structure for reactive programs is extensional if it satisfies the following conditions:*

1. **Extensionality on** $\sigma \to \tau$**.** *For all* $j \geq i$ *and all* $a, b \in \mathcal{A}_j^{\sigma\to\tau}$,
   if $\mathit{app}_j^{\sigma\tau}(\delta_{ij}^{\sigma\to\tau}(a), d) = \mathit{app}_j^{\sigma\tau}(\delta_{ij}^{\sigma\to\tau}(b), d)$ *for all* $d \in \mathcal{A}_j^\sigma$*, then* $a = b$.
2. **Extensionality on** $\bullet\sigma$**.** *For all* $a, b \in \mathcal{A}_i^{\bullet\sigma}$*, if* $\mathit{await}_i^\sigma(a) = \mathit{await}_i^\sigma(b)$ *then* $a = b$.
3. **Extensionality on** $\mathsf{S}(\sigma)$**.** *For all* $a, b \in \mathcal{A}_i^{\mathsf{S}(\sigma)}$*, if for all* $n \in \mathbb{N}$ *we have that* $\mathit{hd}_{i+n}^\sigma(\mathit{tl}_{i+n}^\sigma(\ldots(\mathit{tl}_i^\sigma(a)))) = \mathit{hd}_{i+n}^\sigma(\mathit{tl}_{i+n}^\sigma(\ldots(\mathit{tl}_i^\sigma(b))))$ *then* $a = b$.

Extensionality for the arrow type requires that the applications are equal for all $j \geq i$. This is clearly stronger than having the same condition for just i. However, for the other two cases, the formulations with $j \geq i$ and just i are equivalent.

It is easy to show that extensionality implies the next weaker notion.

Definition 6 (Weak extensional applicative structure for reactive programs). *We say that an applicative structure for reactive programs is weakly extensional if extensionality on* $\mathsf{S}(\sigma)$ *is replaced by the weaker condition:*

3'. For all $MN \in \mathcal{A}_i^{\mathsf{S}(\sigma)}$*, if* $\mathit{hd}_i^\sigma(M) = \mathit{hd}_i^\sigma(N)$ *and* $\mathit{tl}_i^\sigma(M) = \mathit{tl}_i^\sigma(N)$ *then* $M = N$.

Let $\mathcal{A}$ be an applicative structure for reactive programs and Γ be a context. An environment ρ is a function from the set of variables V to the union of all $\mathcal{A}_i^\sigma$. For $a \in \mathcal{A}_i^\sigma$, the update environment $\rho[x \leftarrow a]$ is the environment mapping x to a and all other variables $y \neq x$ to $\rho(y)$. We write $\rho \models \Gamma$ if $\rho(x) \in \mathcal{A}_i^\sigma$ holds for all $x :_i \sigma \in \Gamma$. A *meaning function* for an applicative structure $\mathcal{A}$ is a (total) function that maps any derivation $\Gamma \vdash M :_i \sigma$ and any environment ρ, to an element $[\![\Gamma \vdash M :_i \sigma]\!]_\rho^{\mathcal{A}}$ in $\mathcal{A}_i^\sigma$.

Definition 7 (Henkin models for reactive programs). *Let $\rho \models \Gamma$. An extensional applicative structure $\mathcal{A}$ for reactive programs is called a* Henkin model *if there exists a meaning function satisfying the following conditions (all together called the* environment model condition*):*

- $\delta^\sigma_{ij}([\![\Gamma \vdash M :_i \sigma]\!]^{\mathcal{A}}_\rho) = [\![\Gamma \vdash M :_j \sigma]\!]^{\mathcal{A}}_\rho$
- $[\![\Gamma \vdash x :_j \sigma]\!]^{\mathcal{A}}_\rho = \delta^\sigma_{ij}(\rho(x))$ *for all* $x :_i \sigma \in \Gamma$
- $[\![\Gamma \vdash MN :_i \tau]\!]^{\mathcal{A}}_\rho = \mathit{app}^{\sigma\tau}_i([\![\Gamma \vdash M :_i \sigma \to \tau]\!]^{\mathcal{A}}_\rho,\ [\![\Gamma \vdash N :_i \sigma]\!]^{\mathcal{A}}_\rho)$
- $[\![\Gamma \vdash \lambda x{:}\sigma.M :_i \sigma \to \tau]\!]^{\mathcal{A}}_\rho = \begin{cases} \text{the unique } f \in \mathcal{A}^{\sigma\to\tau}_i \text{ such that} \\ \text{for all } j \geq i,\ d \in \mathcal{A}^\sigma_j \\ \mathit{app}^{\sigma\tau}_j(\delta^{\sigma\to\tau}_{ij}(f), d) = [\![\Gamma, x :_j \sigma \vdash M :_j \tau]\!]_{\rho[x:=d]} \end{cases}$
- $[\![\Gamma \vdash \mathsf{await}(M) :_{i+1} \sigma]\!]^{\mathcal{A}}_\rho = \mathit{await}^\sigma_i([\![\Gamma \vdash M :_i \sigma]\!]^{\mathcal{A}}_\rho)$
- $[\![\Gamma \vdash \circ(M) :_i \bullet\sigma]\!]^{\mathcal{A}}_\rho = \begin{cases} \text{the unique } a \in \mathcal{A}^{\bullet\sigma}_i \text{ such that} \\ \mathit{await}^\sigma_i(a) = [\![\Gamma \vdash M :_{i+1} \sigma]\!]^{\mathcal{A}}_\rho \end{cases}$
- $[\![\Gamma \vdash \mathsf{hd}(M) :_i \sigma]\!]^{\mathcal{A}}_\rho = \mathit{hd}^\sigma_i([\![\Gamma \vdash M :_i \mathsf{S}(\sigma)]\!]^{\mathcal{A}}_\rho)$
- $[\![\Gamma \vdash \mathsf{tl}(M) :_{i+1} \mathsf{S}(\sigma)]\!]^{\mathcal{A}}_\rho = \mathit{tl}^\sigma_i([\![\Gamma \vdash M :_i \mathsf{S}(\sigma)]\!]^{\mathcal{A}}_\rho)$
- $[\![\Gamma \vdash \mathsf{cons}(M,N) :_i \mathsf{S}(\sigma)]\!]^{\mathcal{A}}_\rho = \begin{cases} \text{the unique } s \in \mathcal{A}^{\mathsf{S}(\sigma)}_i \text{ such that} \\ \mathit{hd}^\sigma_i(s) = [\![\Gamma \vdash M :_i \sigma]\!]^{\mathcal{A}}_\rho \text{ and} \\ \mathit{tl}^\sigma_i(s) = [\![\Gamma \vdash N :_{i+1} \mathsf{S}(\sigma)]\!]^{\mathcal{A}}_\rho \end{cases}$

We will use the notation

$$\mathcal{A} \models \Gamma \vdash M = N :_i \sigma$$

if $\Gamma \vdash M :_i \sigma$, $\Gamma \vdash N :_i \sigma$ *and* $[\![\Gamma \vdash M :_i \sigma]\!]^{\mathcal{A}}_\rho = [\![\Gamma \vdash N :_i \sigma]\!]^{\mathcal{A}}_\rho$ *for all* ρ *with* $\rho \models \Gamma$.

Lemma 3 (Soundness of Henkin models for reactive programs).

1. *If* $\Gamma \vdash M :_i \sigma$, *then* $[\![\Gamma \vdash M :_i \sigma]\!]^{\mathcal{A}}_\rho \in \mathcal{A}^\sigma_i$ *for all* $\rho \models \Gamma$.
2. *If* $\Gamma \vdash M :_i \sigma$ *and* $\Gamma \vdash N :_i \sigma$ *then* $M =_{\beta\eta} N$ *implies* $\mathcal{A} \models \Gamma \vdash M = N :_i \sigma$.

Both items of the lemma can be proved by induction on the size of the derivation. By It is enough to consider one step $\to_{\beta\eta}$ in the proof of the second item. .

Definition 8 (Logical relations for reactive programs). *A* (step-indexed) logical relation for reactive programs $\mathcal{R}$ *between two applicative structures for reactive programs* $\mathcal{A}$ *and* $\mathcal{B}$ *is a family* $\{\mathcal{R}^\sigma_i\}$ *of indexed relations such that*

- $\mathcal{R}^\sigma_i \subseteq \mathcal{A}^\sigma_i \times \mathcal{B}^\sigma_i$ *for each* σ *and* i,
- *if* $\mathcal{R}^\sigma_i(a,b)$ *then* $\mathcal{R}^\sigma_j(\delta^\sigma_{ij}(a), \delta^\sigma_{ij}(b))$ *for all* $j \geq i$,
- $\mathcal{R}^{\bullet\sigma}_i(a,b)$ *iff* $\mathcal{R}^\sigma_{i+1}(\mathit{await}^\sigma_i(a), \mathit{await}^\sigma_i(b))$,
- $\mathcal{R}^{\mathsf{S}(\sigma)}_i(a,b)$ *iff* $\mathcal{R}^\sigma_{i+n}(\mathit{hd}^\sigma_{i+n}(\mathit{tl}^\sigma_{i+n} \dots \mathit{tl}^\sigma_i(a)), \mathit{hd}^\sigma_{i+n}(\mathit{tl}^\sigma_{i+n} \dots \mathit{tl}^\sigma_i(b)))$ *for all* n,
- $\mathcal{R}^{\sigma\to\tau}_i(f,g)$ *iff* $\forall j \geq i. \forall a \in \mathcal{A}^\sigma_j. \forall b \in \mathcal{B}^\sigma_j. \mathcal{R}^\sigma_j(a,b) \Rightarrow \mathcal{R}^\tau_j(\mathit{app}^{\sigma\tau}_j(\delta^{\sigma\to\tau}_{ij} f)\ a, \mathit{app}^{\sigma\tau}_j(\delta^{\sigma\to\tau}_{ij} g)\ b)$.

A logical relation $\mathcal{R}$ *between* $\mathcal{A}$ *and* $\mathcal{B}$ *is called a* logical partial (surjective) function *from* $\mathcal{A}$ *to* $\mathcal{B}$ *if each* $\mathcal{R}^\sigma_i$ *is a partial (surjective) function.*

The definition of binary logical relations generalises easily to any arity. In this paper we will define a logical relation of arity one (a logical predicate) and one of arity two.

Lemma 4 (Basic lemma on logical relations for reactive programs). *Let $\mathcal{R}$ be a logical relation for reactive programs between two Henkin models $\mathcal{A}$ and $\mathcal{B}$. Let $\rho_{\mathcal{A}}$ and $\rho_{\mathcal{B}}$ be environments for $\mathcal{A}$ and $\mathcal{B}$ respectively, such that $\rho_{\mathcal{A}} \models \Gamma$, $\rho_{\mathcal{B}} \models \Gamma$ and $\mathcal{R}^{\tau}_{j}(\rho_{\mathcal{A}}(x), \rho_{\mathcal{B}}(x))$ for all $x :_j \tau$ in Γ.*

If $\Gamma \vdash M :_i \sigma$ then $\mathcal{R}^{\sigma}_{i}([\![\Gamma \vdash M :_i \sigma]\!]^{\mathcal{A}}_{\rho_{\mathcal{A}}}, [\![\Gamma \vdash M :_i \sigma]\!]^{\mathcal{B}}_{\rho_{\mathcal{B}}})$.

The above lemma is proved by induction on the size of the derivation.

The theory induced by a Henkin model $\mathcal{A}$, denoted by $\mathsf{Th}(\mathcal{A})$ is the set $\{(M, N) \mid \mathcal{A} \models \Gamma \vdash M = N :_i \sigma\}$.

Lemma 5 (Theory inclusion). *Let $\mathcal{A}, \mathcal{B}$ be Henkin models for reactive programs. If there is a logical partial function from $\mathcal{A}$ to $\mathcal{B}$, then $\mathsf{Th}(\mathcal{A}) \subseteq \mathsf{Th}(\mathcal{B})$.*

This lemma is proved similarly as [7, Lemma 8.2.17].

4 Confluence and Strong Normalisation

In this section, we prove confluence and strong normalisation of $\beta\eta$ for the typed lambda calculus of reactive programs.

Failure of confluence of $\beta\eta$ on untypable terms has several causes. One cause is the presence of explicit types in the abstractions. Nederpelt's term $\lambda x{:}\sigma.(\lambda y{:}\tau.y)x$ provides a counterexample [8]. Another cause is the non-left linear η-rule for streams. This is shown through a variation of Klop's counterexample on surjective pairs [9, 10]. Define

$$\mathsf{D} = \lambda x{:}\sigma.\lambda y{:}\tau.(\mathsf{cons}(\mathsf{hd}(\lambda z.zx), \mathsf{tl}(\lambda z.zy))\lambda z.u).$$

Then, we have that $\mathsf{D}MM \twoheadrightarrow_{\beta\eta} u$ for any M. Note that the η-step creates a β-redex that cannot be performed earlier (this shows that η cannot be postponed over β on untypable terms). Next, we define $\mathsf{E} = Y\ (\lambda f{:}\sigma'.\lambda x{:}\tau'.\mathsf{D}\ x\ (f\ x))$ and $\mathsf{F} = Y\ (\lambda f{:}\sigma''.\mathsf{E}\ f)$. We have that $\mathsf{E} \twoheadrightarrow_{\beta} \lambda x{:}\tau'.\mathsf{D}\ x\ (\mathsf{E}\ x)$ and $\mathsf{F} \twoheadrightarrow_{\beta} \mathsf{E}\ \mathsf{F}$. So, $\mathsf{F} \twoheadrightarrow_{\beta\eta} u$ and $\mathsf{F} \twoheadrightarrow_{\beta\eta} \mathsf{E}\ u$. But u and $\mathsf{E}\ u$ do not have a common reduct.

We will show that the typable terms are $\beta\eta$-strongly normalising using a logical predicate similar to the ones used in strong normalisation proofs of the simply typed lambda calculus (e.g., see Section 8.3.2 of [7]). For this proof, we use an applicative structure $\mathcal{T}$ constructed from typable terms.

Notation 1 *From now on, we assume the existence of a family $\{\mathsf{V}^{\sigma}_{i}\}$ of pairwise disjoint, infinite sets of pairwise distinct variables. We define Γ_{∞} to be the infinite context consisting of all type declarations of the form $x :_i \sigma$ with $x \in \mathsf{V}^{\sigma}_{i}$ for some type σ and index i.*

Definition 9 (Term applicative structure). *For each type σ and index i, let $\mathcal{T}_i^\sigma$ be the set $\{M \mid \Gamma \vdash M :_i \sigma$, for some σ and finite $\Gamma \subset \Gamma_\infty\}$ of all terms that can be typed with σ at "stage" i with some finite subcontext of Γ_∞. We define* the term applicative structure *as the applicative structure*

$$\mathcal{T} = \langle \{\mathcal{T}_i^\sigma\}, \{\delta_{ij}^\sigma\}, \{\mathit{hd}_i^\sigma, \mathit{tl}_i^\sigma, \mathit{await}_i^\sigma, \mathit{app}_i^{\sigma\tau}\}\rangle$$

where $\mathit{app}_i^{\sigma\tau}(M,N) = MN$, $\mathit{hd}_i^\sigma(M) = \mathsf{hd}(M)$, $\mathit{await}_i^\sigma(M) = \mathsf{await}(M)$ and $\mathit{tl}_i^\sigma(M) = \mathsf{tl}(M)$. We take set inclusion for δ_{ij}^σ when $i \leq j$. This is well-defined by the Time Adjustment Lemma. As meaning function, we take $[\![\Gamma \vdash M :_i \sigma]\!]_\rho^{\mathcal{T}} = \rho(M)$ where $\rho(M)$ is the result of performing the substitution ρ to M.

Note that $[\![\Gamma \vdash M :_i \sigma]\!]_\rho^{\mathcal{T}} \in \mathcal{T}_i^\sigma$. This meaning function does not satisfy the environment model condition.

Definition 10 (Logical predicate of strongly normalizing terms). *Let $\mathcal{SN}$ be the set of $\beta\eta$-strongly normalising terms. We define the family of predicates $\mathcal{P}_i^\sigma \subseteq \mathcal{T}_i^\sigma$ by induction on σ:*

$$\begin{array}{ll}
\mathcal{P}_i^{\mathsf{b}} & = \{M \in \mathcal{T}_i^{\mathsf{b}} \mid M \in \mathcal{SN}\} \\
\mathcal{P}_i^{\sigma\to\tau} & = \{M \in \mathcal{T}_i^{\sigma\to\tau} \mid \forall j \geq i, N \in \mathcal{P}_j^\sigma, MN \in \mathcal{P}_j^\tau\} \\
\mathcal{P}_i^{\bullet\sigma} & = \{M \in \mathcal{T}_i^{\bullet\sigma} \mid \mathsf{await}(M) \in \mathcal{P}_{i+1}^\sigma\} \\
\mathcal{P}_i^{\mathsf{S}(\sigma)} & = \{M \in \mathcal{T}_i^{\mathsf{S}(\sigma)} \mid \forall n \in \mathbb{N}, \mathsf{hd}(\mathsf{tl}^n(M)) \in \mathcal{P}_{i+n}^\sigma\}
\end{array}$$

where $\mathsf{tl}^n(M)$ is the term $\mathsf{tl}(\mathsf{tl}(\ldots(\mathsf{tl}(M))))$ consisting of n applications of tl.

It is easy to see that $\mathcal{P}$ is a step-indexed logical relation for reactive programs of arity one. Note that $\mathcal{P}_i^\sigma \subseteq \mathcal{P}_j^\sigma$ if $i \leq j$. We define the elimination contexts $\mathcal{E}[\,]$ with one hole with the grammar $\mathcal{E}[\,] := [\,] \mid \mathcal{E}[\,]M \mid \mathsf{hd}(\mathsf{tl}^n(\mathcal{E}[\,])) \mid \mathsf{await}(\mathcal{E}[\,])$. We write $\mathcal{E}[\,] \in \mathcal{SN}$ if all terms used in the construction of $\mathcal{E}[\,]$ are $\beta\eta$-strongly normalising.

Lemma 6. *1. $\mathcal{P}_i^\sigma \subseteq \mathcal{SN}$. 2. If $\mathcal{E}[\,] \in \mathcal{SN}$ and $\mathcal{E}[x] \in \mathcal{T}_i^\sigma$ then $\mathcal{E}[x] \in \mathcal{P}_i^\sigma$.*

The two statements are proved simultaneously by induction on the type σ. For the base case $\sigma = \mathsf{b}$, it is easy to see that $\mathcal{E}[x] \in \mathcal{SN}$ because $\mathcal{E}[\,] \in \mathcal{SN}$. The cases in the next lemma all follow by induction on σ.

Lemma 7 (Closure under β-expansion inside context $\mathcal{E}$).

- *If $\mathcal{E}[(\lambda x.M)N] \in \mathcal{T}_i^\sigma$ and $\mathcal{E}[M[x := N]] \in \mathcal{P}_i^\sigma$ then $\mathcal{E}[(\lambda x.M)N] \in \mathcal{P}_i^\sigma$.*
- *If $\mathcal{E}[\mathsf{await}(M)] \in \mathcal{T}_i^{\bullet\sigma}$ and $\mathcal{E}[M] \in \mathcal{P}_i^\sigma$ then $\mathcal{E}[\mathsf{await}(M)] \in \mathcal{P}_i^{\bullet\sigma}$.*
- *If $\mathcal{E}[\mathsf{hd}(\mathsf{cons}(M,N))] \in \mathcal{T}_i^\sigma$ and $\mathcal{E}[M] \in \mathcal{P}_i^\sigma$ then $\mathcal{E}[\mathsf{hd}(\mathsf{cons}(M,N))] \in \mathcal{P}_i^\sigma$.*
- *If $\mathcal{E}[\mathsf{tl}(\mathsf{cons}(M,N))] \in \mathcal{T}_i^\sigma$ and $\mathcal{E}[N] \in \mathcal{P}_i^\sigma$ then $\mathcal{E}[\mathsf{tl}(\mathsf{cons}(M,N))] \in \mathcal{P}_i^\sigma$.*

Lemma 8 (Soundness for logical predicate $\mathcal{P}$). *Let $\Gamma \vdash M :_i \sigma$ and $\rho(x) \in \mathcal{P}_i^\sigma$ for all $x :_i \sigma \in \Gamma$. Then, $[\![\Gamma \vdash M :_i \sigma]\!]_\rho^{\mathcal{T}} \in \mathcal{P}_i^\sigma$.*

Proof. By induction on the derivation using Lemma 7. □

Remark 1 (Alternative proof of Lemma 8). It is possible to give a general version of the basic lemma for logical relations instead of Lemma 8. For that we would have to introduce more notational machinery like the notions of acceptable meaning function and admissible relation, as on pages 540-541 in [7].

Theorem 1 (Strong $\beta\eta$-Normalisation on typable terms). *If $\Gamma \vdash M :_i \sigma$ then M is $\beta\eta$-strongly normalising.*

Proof. Suppose $\Gamma \vdash M :_i \sigma$. As environment ρ, we take identity. Now $\rho \models \Gamma$, since $\rho(x) = x \in \mathcal{P}_i^\tau$ for all $x :_i \tau \in \Gamma$ by the second item of Lemma 6. From Lemma 8 we obtain $[\![\Gamma \vdash M :_i \sigma]\!]_\rho^{\mathcal{T}} = M \in \mathcal{P}_i^\sigma$. Using the first item of Lemma 6 we find $\mathcal{P}_i^\sigma \subseteq \mathcal{SN}$. Hence M is $\beta\eta$-strongly normalising. □

Theorem 2 (Confluence of $\beta\eta$ on typable terms). *The $\beta\eta$-reduction is confluent on typable terms.*

Proof. We apply Newman's Lemma [11, Theorem 1.2.1]. Since, by Theorem 1, $\beta\eta$ is strongly normalising, it is sufficient to verify that $\beta\eta$-reduction is locally confluent. This is straightforward. □

5 Term Model

In this section, we construct the term model $\mathcal{T}/{=_{\beta\eta}}$ from the term applicative structure $\mathcal{T}$ by quotienting over $\beta\eta$ conversion. We prove that $\mathcal{T}/{=_{\beta\eta}}$ is extensional. This gives us our first completeness result, i.e. there exists a Henkin model for reactive programs satisfying exactly the theory of $\beta\eta$-conversion.

We write $[M]$ to denote the set of terms that are $\beta\eta$ convertible to M.

Definition 11 (Term model). *For each type σ and index i, let $(\mathcal{T}/{=_{\beta\eta}})_i^\sigma = \{[M] \mid M \in \mathcal{T}_i^\sigma\}$. We define the applicative structure $\mathcal{T}/{=_{\beta\eta}}$ as*

$$\langle \{(\mathcal{T}/{=_{\beta\eta}})_i^\sigma\}, \{\delta_{ij}^\sigma\}, \{\mathit{hd}_i^\sigma, \mathit{tl}_i^\sigma, \mathit{await}_i^\sigma, \mathit{app}_i^{\sigma\tau}\} \rangle$$

with $\mathit{app}_i^{\sigma\tau}([M],[N]) = [MN]$, $\mathit{hd}_i^\sigma([M]) = [\mathsf{hd}(M)]$, $\mathit{await}_i^\sigma([M]) = [\mathsf{await}(M)]$ and $\mathit{tl}_i^\sigma([M]) = [\mathsf{tl}(M)]$. We take set inclusion for δ_{ij}^σ when $i \leq j$, and define as meaning function $[\![\Gamma \vdash M :_i \sigma]\!]_\rho = [M[x_1{:=}N_1, \ldots, x_n{:=}N_n]]$ where $\rho(x_i) = [N_i]$ for all $1 \leq i \leq n$ and all x_i occur in Γ.

We write $\mathsf{size}(M)$, $\mathsf{size}(\sigma)$ and $\mathsf{size}(\Gamma)$ for the number of symbols of M, σ and all types in Γ, respectively.

Lemma 9 (Shape of β-normal forms). *Let M be a typable β-normal form. Then, M is of the form $\lambda x_1{:}\sigma_1 \ldots x_n{:}\sigma_n.N$ where N satisfies one of the clauses:*

1. *N is of the form $\mathsf{cons}(P,Q)$ where P and Q are in β-normal form.*
2. *N is of the form $\circ P$ where P is in β-normal form.*

3. *N belongs to the grammar:*

$$\mathsf{X} \ni X := x \mid XP \mid \mathsf{hd}(X) \mid \mathsf{tl}(X) \mid \mathsf{await}(X)$$

where P is in β-normal form and $x \in \mathsf{V}$. Instead of a variable X ranging over X we may occasionally write $X[P_1, \ldots, P_n]$ to list explicitly all arguments P used in the construction of X. For such X we have that if $\Gamma \vdash X :_j \tau$ then $\mathsf{size}(\tau) \leq \mathsf{size}(\Gamma)$ and $\mathsf{size}(P_i) < \mathsf{size}(\Gamma)$ for all $1 \leq i \leq n$.

The previous lemma can be proved by induction on the derivation.

Note that weak extensionality of the term model is a direct consequence of the η-rule. To prove extensionality we need more, namely that λ^{RP} is confluent and strongly normalising. The assumption made for Γ_∞ in Notation 1 is important in the proof of extensionality on $\sigma \to \tau$, as it allows us to pick an $x \in \mathcal{T}_i^\sigma$.

Lemma 10 (Extensionality of term model). *The applicative structure for the term model is extensional.*

Proof. We only prove extensionality on $\mathsf{S}(\sigma)$ and leave the other cases to the reader. Let $M, N \in \mathcal{T}_i^{\mathsf{S}(\sigma)}$ be in $\beta\eta$-normal form. We now analyse the shape of these $\beta\eta$-normal forms. Suppose that $\mathsf{hd}(\mathsf{tl}^n(M)) =_{\beta\eta} \mathsf{hd}(\mathsf{tl}^n(N))$ for all $n \in \mathbb{N}$. We prove that $M =_\eta N$ by induction on the number of cons that appear in M and N. We distinguish cases depending on the shape of M and N by Lemma 9.

1. Case $M, N \in \mathsf{X}$. Then $\mathsf{hd}(M)$ and $\mathsf{hd}(N)$ are in $\beta\eta$-normal form. By Confluence of $\beta\eta$ (Theorem 2) we find $\mathsf{hd}(M) = \mathsf{hd}(N)$. Hence $M = N$.
2. Case $M = \mathsf{cons}(P, Q)$ and $N = \mathsf{cons}(P', Q')$. We have $P =_{\beta\eta} \mathsf{hd}(M) =_{\beta\eta} \mathsf{hd}(N) =_{\beta\eta} P'$. Since P and P' are in $\beta\eta$-normal form, we have $P = P'$ by confluence of $\beta\eta$. We also have $\mathsf{hd}(\mathsf{tl}^n(Q)) =_{\beta\eta} \mathsf{hd}(\mathsf{tl}^n(Q'))$ for all $n \in \mathbb{N}$. Since Q and Q' have fewer number of cons than M and N, $Q =_\eta Q'$ by induction hypothesis.
3. Case $M = \mathsf{cons}(P, Q)$ and $N \in \mathsf{X}$. Then P, $\mathsf{hd}(N)$ and $\mathsf{tl}(N)$ are all in $\beta\eta$-normal form. We get $P =_{\beta\eta} \mathsf{hd}(M) =_{\beta\eta} \mathsf{hd}(N)$. By confluence of $\beta\eta$ we conclude $P = \mathsf{hd}(N)$. We also have $\mathsf{hd}(\mathsf{tl}^n(Q)) =_{\beta\eta} \mathsf{hd}(\mathsf{tl}^n(\mathsf{tl}(N)))$ for all $n \in \mathbb{N}$. Applying the induction hypothesis to Q and $\mathsf{tl}(N)$ we get $Q =_\eta \mathsf{tl}(N)$.

□

Theorem 3 (Completeness for Henkin models of reactive programs). *There exists a Henkin model for reactive programs satisfying exactly the theory of $\beta\eta$-conversion.*

Proof. It is routine to show that the meaning function of $\mathcal{T}/{=_{\beta\eta}}$ satisfies the environment condition. The term model trivially satisfies the theory of $\beta\eta$-conversion, i.e., $M =_{\beta\eta} N$ iff $\mathcal{T}/{=_{\beta\eta}} \models \Gamma \vdash M = N :_i \sigma$ whenever $\Gamma \vdash M, N :_i \sigma$.

□

6 Ultrametric Model for Reactive Programs

In this section, we present the ultrametric model of [1] as a Henkin Model for reactive programs.

A *complete 1-bounded ultrametric space* is a tuple (U, d_U), where U is a set and the distance function $d_U : U \times U \to [0,1]$ satisfies: 1. $d_U(u,v) = 0$ iff $u = v$, 2. $d_U(u,v) = d_U(v,u)$, 3. $d_U(u,z) \leq \max(d_U(u,v), d_U(v,z))$, 4. every Cauchy sequence in U has a limit in X. A function $f : U \to V$ between ultrametric spaces is *non-expansive* if $d_V(f(u_1), f(u_2)) \leq d_U(u_1, u_2)$. It is well-known that the complete 1-bounded ultrametric spaces and nonexpansive functions form a cartesian-closed category. The *shrink functor* $\frac{1}{2}$ maps (U, d_U) to $(U, \frac{1}{2}d_U)$ and a non-expansive function $f \in U \to V$ to the non-expansive function $\frac{1}{2}(f) \in \frac{1}{2}U \to \frac{1}{2}V$ where $\frac{1}{2}(f)(u) = f(u)$.

Definition 12 (Ultrametric applicative structure). *An* ultrametric applicative structure *is an applicative structure*

$$\mathcal{U} = \langle \{\mathcal{U}_i^\sigma\}, \{\delta_{ij}^\sigma\}, \{\mathit{hd}_i^\sigma, \mathit{tl}_i^\sigma, \mathit{await}_i^\sigma, \mathit{app}_i^{\sigma\tau}\} \rangle$$

1. $\mathcal{U}_i^\sigma = \frac{1}{2}^i \mathcal{U}_0^\sigma$ *where* $\mathcal{U}_0^\sigma$ *is defined by induction on* σ*:*
   (a) $\mathcal{U}_0^{\mathsf{b}}$ *is some ultrametric space* (U, d_U).
   (b) $\mathcal{U}_0^{\sigma\to\tau}$ *is the set of nonexpansive maps from* $\mathcal{U}_0^\sigma$ *to* $\mathcal{U}_0^\tau$*, equipped with the supremum metric:* $d_{\mathcal{U}_0^\sigma \to \mathcal{U}_0^\tau}(f,g) = \sup\{d_{\mathcal{U}_0^\tau}(f(x), g(x)) \mid x \in \mathcal{U}_0^\sigma\}$.
   (c) $\mathcal{U}_0^{\bullet\sigma} = \frac{1}{2}\mathcal{U}_0^\sigma$.
   (d) $\mathcal{U}_0^{\mathsf{S}(\sigma)}$ *is the set of total functions from* $\mathbb{N}$ *to* $\mathcal{U}_0^\sigma$*, equipped with the stream metric:* $d_{\mathcal{U}_0^{\mathsf{S}(\sigma)}}(f,g) = \sup\{\frac{1}{2}^n d_{\mathcal{U}_0^\sigma}(f(n), g(n)) \mid n \in \mathbb{N}\}$.
2. $\delta_{ij}^\sigma \in \mathcal{U}_i^\sigma \to \frac{1}{2}^{j-i}\mathcal{U}_j^\sigma$ *is defined by* $\delta_{ij}^\sigma(u) = u$.
3. $\mathit{app}_i^{\sigma\tau}(f,a) = f(a)$, $\mathit{await}_i^\sigma(a) = a$, $\mathit{hd}_i^\sigma(f) = f(0)$ *and* $\mathit{tl}_i^\sigma(f)(n) = f(n+1)$ *for all* $n \geq 0$.

It is easy to see that an ultrametric applicative structure is extensional. We define $\mathit{cons}_i^\sigma(a,f) = g$ where $g(0) = a$ and $g(n+1) = f(n)$ for $n \geq 0$.

Lemma 11 (Ultrametric model). *Let* $\rho \models \Gamma$*. The ultrametric applicative structure together with the meaning function defined as*

- $[\![\Gamma \vdash x :_j \sigma]\!]_\rho = \delta_{ij}^\sigma(\rho(x))$ *if* $x{:_i}\sigma \in \Gamma$,
- $[\![\Gamma \vdash MN :_i \tau]\!]_\rho = \mathit{app}_i^{\sigma\tau}([\![\Gamma \vdash M :_i \sigma \to \tau]\!]_\rho, [\![\Gamma \vdash N :_i \tau]\!]_\rho)$,
- $[\![\Gamma \vdash \lambda x{:}\sigma.M :_i \sigma \to \tau]\!]_\rho = \{(a, [\![\Gamma, x :_i \sigma \vdash M :_i \tau]\!]_{\rho[x:=a]}) \mid a \in \mathcal{U}_i^\sigma\}$,
- $[\![\Gamma \vdash \mathsf{await}(M) :_{i+1} \sigma]\!]_\rho = [\![\Gamma \vdash M :_i \bullet\sigma]\!]_\rho$,
- $[\![\Gamma \vdash \circ(M) :_i \bullet\sigma]\!]_\rho = [\![\Gamma \vdash M :_{i+1} \sigma]\!]_\rho$,
- $[\![\Gamma \vdash \mathsf{hd}(M) :_i \mathsf{S}(\sigma)]\!]_\rho = \mathit{hd}_i^\sigma([\![\Gamma \vdash M :_i \mathsf{S}(\sigma)]\!]_\rho)$,
- $[\![\Gamma \vdash \mathsf{tl}(M) :_{i+1} \mathsf{S}(\sigma)]\!]_\rho = \mathit{tl}_i^\sigma([\![\Gamma \vdash M :_i \mathsf{S}(\sigma)]\!]_\rho)$,
- $[\![\Gamma \vdash \mathsf{cons}(M,N) :_i \mathsf{S}(\sigma)]\!]_\rho = \mathit{cons}_i^\sigma([\![\Gamma \vdash M :_i \sigma]\!]_\rho, [\![\Gamma \vdash N :_{i+1} \sigma]\!]_\rho)$,

is a Henkin model for reactive programs called the ultrametric model.

7 Metric on the Term Model

In this section, we define an ultrametric d on the term model for which the equivalence classes of terms of type $\sigma \to \tau$ are *non-expansive*, i.e. for M of type $\sigma \to \tau$ we have that $d([MP],[MQ]) \leq d([P],[Q])$.

We recall the notions of depth, truncation and metric on terms of [12]. The depth of N in argument positions in $\mathsf{cons}(M,N)$ and $\circ(N)$ is counted one deeper than the depth of the terms $\mathsf{cons}(M,N)$ and $\circ(N)$ themselves. To define truncation, we extend the syntax with a constant $\bot$.

Definition 13 (Truncation). *The truncation of M at depth n, denoted by M^n, is defined by induction as follows.*

$$\begin{array}{rlrl} M^0 &= \bot & x^{n+1} &= x \\ (\lambda x.M)^{n+1} &= \lambda x.M^{n+1} & (M\ N)^{n+1} &= (M^{n+1}\ N^{n+1}) \\ (\circ(M))^{n+1} &= \circ(M^n) & (\mathsf{await}(M))^{n+1} &= \mathsf{await}(M^{n+1}) \\ (\mathsf{hd}(M))^{n+1} &= \mathsf{hd}(M^{n+1}) & (\mathsf{tl}(M))^{n+1} &= \mathsf{tl}(M^{n+1}) \\ (\mathsf{cons}(M,N))^{n+1} &= \mathsf{cons}(M^{n+1},N^n) & & \end{array}$$

Definition 14 (Metric on terms). *Define $d : \mathsf{P} \times \mathsf{P} \to [0,1]$ as $d(M,N) = 0$, if $M = N$ and $d(M,N) = 2^{-m}$ otherwise, where $m = \max\{n \in \mathbb{N} \mid M^n = N^n\}$.*

Note that d is not invariant under $\beta\eta$-conversion. The metric on equivalence classes defined by $d([M],[N]) = d(\mathsf{nf}_{\beta\eta}(M), \mathsf{nf}_{\beta\eta}(N))$ is not the right one since there may be elements in $(\mathcal{T}/{=_{\beta\eta}})_i^{\sigma\to\tau}$ that are not non-expansive. For example, $d([MP],[MQ]) = 1$ and $d([P],[Q]) = 1/2$ if $M = \lambda x{:}\mathsf{S}(\sigma).\mathsf{cons}(\mathsf{hd}(y),\mathsf{tl}(x))$, $P = y$ and $Q = z$. The distance between $[MP] = [y]$ and $[MQ] = [\mathsf{cons}(\mathsf{hd}(y),\mathsf{tl}(z))]$ should be $\frac{1}{2}$ and not 1 for $[M]$ to be non-expansive.

To define the right metric, we introduce the notions of infinite term and extensional long normal form. The notion of extensional long normal form does not coincide with the notion of eta long normal form. In order to define the notion of extensional long normal, we express a function f on natural numbers as an infinite term $\mathsf{cons}(M_1, \mathsf{cons}(M_2, \ldots))$ where M_1 corresponds to $f(1)$, M_2 to $f(2)$, etc. For example, the extensional long normal form of a variable x of type $\mathsf{S}(\sigma)$ is the infinite term $\mathsf{cons}(\mathsf{hd}(x), \mathsf{cons}(\mathsf{hd}(\mathsf{tl}(x)), \ldots))$.

Definition 15 (Infinitary terms). *We define the set P^∞ of infinitary terms as the metric completion of (P, d).*

Definition 16 (Extensional long normal form). *Let $\Gamma \vdash M :_i \sigma$. The (extensional) long normal form of M is a term in P^∞ denoted by $\mathsf{L}(M)$ and defined as follows. If M is not in β-normal form, we define $\mathsf{L}(M) = \mathsf{L}(\mathsf{nf}_\beta(M))$. If M is in β-normal form then we define it by induction on the pair $(\mathsf{size}(\Gamma) + \mathsf{size}(\sigma), \mathsf{size}(M))$ with the lexicographic order as follows.*

1. *Case σ is a base type. Then $M = X[M_1, \ldots, M_n] \in \mathsf{X}$. We define $\mathsf{L}(M) = X[\mathsf{L}(M_1), \ldots, \mathsf{L}(M_n)]$.*

2. *Case* σ *is* $\sigma_1 \to \sigma_2$. *If* $M = \lambda x.N$ *then we define* $\mathsf{L}(M) = \lambda x.\mathsf{L}(N)$. *Otherwise,* $M \in \mathsf{X}$ *and we define* $\mathsf{L}(M) = \lambda y.\mathsf{L}(M\ y)$.
3. *Case* σ *is* $\bullet\tau$. *If* $M = \circ(P)$, *we define* $\mathsf{L}(M) = \circ(\mathsf{L}(P))$. *Otherwise,* $M \in \mathsf{X}$ *and we define* $\mathsf{L}(M) = \circ(\mathsf{L}(\mathsf{await}(M)))$.
4. *Case* $\sigma = \mathsf{S}(\tau)$. *Either* $M = \mathsf{cons}(P, Q)$ *and we put* $\mathsf{L}(M) = \mathsf{cons}(\mathsf{L}(P), \mathsf{L}(Q))$, *or* $M \in \mathsf{X}$ *and we define* $\mathsf{L}(M) = \mathsf{cons}(\mathsf{L}(\mathsf{hd}(M)), \mathsf{cons}(\mathsf{L}(\mathsf{hd}(\mathsf{tl}(M))), \ldots))$.

Lemma 12. 1. *Let* M *be in* β*-normal form such that* $\Gamma \vdash M :_i \sigma$. *If* $M \to_\eta N$, *then* $\mathsf{L}(M) = \mathsf{L}(N)$.
2. *If* $\Gamma \vdash M :_i \sigma$ *and* $M =_{\beta\eta} N$, *then* $\mathsf{L}(M) = \mathsf{L}(N)$.

The first statement is proved by induction on $(\mathsf{size}(\Gamma) + \mathsf{size}(\sigma), \mathsf{size}(M))$. The second is proved using confluence, strong normalisation and the first one.

Definition 17 (Metric on equivalence classes of typable terms). *We define a metric* $d : (\mathcal{T}/=_{\beta\eta})^\sigma_i \times (\mathcal{T}/=_{\beta\eta})^\sigma_i \to [0,1]$ *as* $d([M],[N]) = d(\mathsf{L}(M), \mathsf{L}(N))$.

We define $\Gamma \vdash^n M :_i \sigma$ by adding to the typing rules of Definition 3 the rule $\Gamma \vdash^n \bot :_i \sigma$ if $n \geq i$ for all $i \in \mathbb{N}$ and all types σ. It is easy to show that this typing rules satisfy subject reduction, confluence and strong normalisation. The notion of extensional long normal form is extended to terms with $\bot$ and typable in $\vdash^n$. The follow lemma is proved by induction on the derivation.

Lemma 13. *If* $\Gamma \vdash^{n+i} M :_i \sigma$ *then the* $\bot$*'s in* M *all occur at depth greater or equal than* n.

The following three lemmas are proved by induction on $(\mathsf{size}(\Gamma) + \mathsf{size}(\sigma), n)$.

Lemma 14. *If* $\Gamma \vdash M :_i \sigma$ *then* $\Gamma \vdash^{n+i} (\mathsf{L}(M))^n :_i \sigma$.

Lemma 15. *Let* M *be in* β*-normal form such that* $\Gamma \vdash M :_i \sigma$. *Then, there exists* N *in* β*-normal form such that* $N \twoheadrightarrow_\eta M$ *and* $(\mathsf{L}(M))^n = N^n$.

Lemma 16. *Let* M, N *be in* β*-normal form such that* $\Gamma \vdash M, N :_i \sigma$. *If* $M^n = N^n$ *then* $(\mathsf{L}(M))^n = (\mathsf{L}(N))^n$.

We write $M \prec N$ if M is the result of replacing some subterms of N by $\bot$.

Lemma 17. *If* $M \prec N$ *and* $M \to_\beta M'$ *then* $M' \prec N'$ *and* $N \to_\beta N'$ *for some* N'.

Theorem 4 (Non-expansiveness). *Let* $M \in \mathcal{T}_i^{\sigma\to\tau}$ *and* $P, Q \in \mathcal{T}_i^\sigma$. *Then:*

1. $(\mathsf{L}(MP))^n = (\mathsf{L}(M(\mathsf{L}(P))^n))^n$. 2. $d([MP],[MQ]) \leq d([P],[Q])$.

Proof. (1) Assume M, P are in β-normal form. By Lemma 15, there exists P' in β-normal form such that $P' \twoheadrightarrow_\eta P$ and $(\mathsf{L}(P))^n = (P')^n$. Then, $M(\mathsf{L}(P))^n \prec MP'$. By Lemma 14, $\Gamma \vdash^{n+i} (\mathsf{L}(P))^n :_i \sigma$ and hence, $\Gamma \vdash^{n+i} M(\mathsf{L}(P))^n :_i \sigma$. By Subject reduction, $\Gamma \vdash^{n+i} N :_i \sigma$ where $N = \mathsf{nf}_\beta(M(\mathsf{L}(P))^n)$. By Lemma 17, there exists N' such that $N \prec N'$ and $MP' \twoheadrightarrow_\beta N'$. By Lemma 13, the $\bot$'s in N occur at depth greater than n. Hence, $N^n = (N')^n = (\mathsf{nf}_\beta(MP'))^n$ since N is in β-normal form. By Lemma 16, $(\mathsf{L}(N))^n = (\mathsf{L}(\mathsf{nf}_\beta(MP')))^n$. By Lemma 12, $(\mathsf{L}(MP))^n = (\mathsf{L}(M(\mathsf{L}(P))^n))^n$.
(2) Suppose $(\mathsf{L}(P))^n = (\mathsf{L}(Q))^n$. By the first part, $(\mathsf{L}(MP))^n = (\mathsf{L}(M(\mathsf{L}(P))^n))^n = (\mathsf{L}(M(\mathsf{L}(Q))^n))^n = (\mathsf{L}(MQ))^n$. □

Theorem 5 (Well-behaviour of metric).

1. $d([M],[N]) = \sup\{d([MP],[NP]) \mid P \in \mathcal{T}_i^\sigma\}$.
2. $d([M],[N]) = \frac{1}{2}d([\mathsf{await}(M)],[\mathsf{await}(N)])$.
3. $d([M],[N]) = \sup\{\frac{1}{2}^n d([\mathsf{hd}(\mathsf{tl}^n(M))],[\mathsf{hd}(\mathsf{tl}^n(N))]) \mid n \in \mathbb{N}\}$.

The proof of this theorem is similar to the one for Theorem 4.

8 Completeness for the Ultrametric Model

In this section we show that $\beta\eta$-conversion captures semantic equality between reactive programs, i.e. two terms typable in the calculus of reactive programs are $\beta\eta$-convertible if and only if they have the same interpretation in the ultrametric model. Our proof follows closely the proof of completeness of the simply typed lambda calculus given in [7, Section 8].

An ultrametric frame is an ultrametric applicative structure where $\mathcal{U}_i^{\sigma\to\tau}$ is a subset of the set of non-expansive maps from $\mathcal{U}_i^\sigma$ to $\mathcal{U}_i^\tau$.

Theorem 6. *There exists an ultrametric frame isomorphic to the term model.*

Proof. We define $\mathcal{U}_i^\sigma = \{\phi_i^\sigma([M]) \mid M \in \mathcal{T}_i^\sigma\}$ where ϕ_i^σ is a function from $(\mathcal{T}/=_{\beta\eta})_i^\sigma$ to $\mathcal{U}_i^\sigma$ defined by induction on σ.

$$\begin{array}{ll} \phi_i^{\mathsf{b}}([M]) = [M] & \phi_i^{\sigma\to\tau}([M]) = \{(a, \phi_i^\tau([M(\phi_i^\sigma)^{-1}(a)]) \mid a \in \mathcal{U}_i^\sigma\} \\ \phi_i^{\bullet\sigma}([M]) = \phi_{i+1}^\sigma([\mathsf{await}(M)]) & \phi_i^{\mathsf{S}(\sigma)}([M]) = \{(n, \phi_i^\sigma([\mathsf{hd}(\mathsf{tl}^n(a))])) \mid n \in \mathbb{N}\} \end{array}$$

Surjectivity of ϕ_i^σ is trivial. Injectivity follows by induction on σ using Lemma 10. That ϕ_i^σ and its inverse are non-expansive follows by induction on σ using Theorem 5. It remains to prove that $\phi_i^{\sigma\to\tau}([M])$ is non-expansive. This follows from Theorem 4 and the fact that ϕ_i^τ and the inverse of ϕ_i^σ are non-expansive. □

Definition 18 (Embedding). *Let (U, d_U), (V, d_V) be ultrametric spaces. An embedding from V to U is a pair (ϕ, ψ) of non-expansive maps $\phi : V \to U$ and $\psi : U \to V$ such that $\psi \circ \phi = id_V$.*

Lemma 18 (Partial Surjections for ultrametric spaces). *Let $\mathcal{U}$ be an ultrametric applicative structure and $\mathcal{V}$ be an ultrametric frame. If there exists an embedding from $\mathcal{V}_i^{\mathsf{b}}$ to $\mathcal{U}_i^{\mathsf{b}}$ for each constant type b and $i \in \mathbb{N}$, then there exists a partial logical surjective function $\mathcal{R}$ from $\mathcal{U}$ to $\mathcal{V}$ and two families of non-expansive maps ϕ_i^σ, ψ_i^σ such that*

1. *$\mathcal{R}_i^\sigma(\phi_i^\sigma(v), v)$ for all $v \in \mathcal{V}_i^\sigma$.* 2. *$\mathcal{R}_i^\sigma(u, \psi_i^\sigma(u))$ for all $u \in dom(\mathcal{R}_i^\sigma)$.*

Proof. By induction on σ. We do only the case $\mathsf{S}(\sigma)$ for streams. Then we define:

$$\begin{array}{ll} \phi_i^{\mathsf{S}(\sigma)}(v) = \{(n, \phi_{i+n}^\sigma(v(n))) \mid n \in \mathbb{N}\} & \forall v \in \mathcal{V}_i^{\mathsf{S}(\sigma)} \\ \psi_i^{\mathsf{S}(\sigma)}(u) = \{(n, \psi_{i+n}^\sigma(u(n))) \mid n \in \mathbb{N}\} & \forall u \in dom(\mathcal{R}_i^{\mathsf{S}(\sigma)}) \\ \mathcal{R}_i^{\mathsf{S}(\sigma)} = \{(f, g) \mid (f(n), g(n)) \in \mathcal{R}_{i+n}^\sigma\} & \end{array}$$

Statements (1) and (2) follow from induction hypothesis. Surjectivity follows from (1). That $\mathcal{R}_i^{\mathsf{S}(\sigma)}$ is a function follows by extensionality. Surjectivity plays a role only in the arrow type case, for proving that $\mathcal{R}_i^{\sigma\to\tau}$ is a function. □

Theorem 7 (Completeness for the ultrametric model). *Suppose there exists an embedding from* $(\mathcal{T}/{=_{\beta\eta}})_i^{\mathsf{b}}$ *to* $\mathcal{U}_i^{\mathsf{b}}$ *for each constant type* b *and* $i \in \mathbb{N}$. *Let* $\Gamma \vdash M, N :_i \sigma$. *Then,* $M =_{\beta\eta} N$ *if and only if* $\mathcal{U} \models \Gamma \vdash M = N :_i \sigma$.

Proof. ($\Rightarrow$) [1, Theorem 4]. Alternatively, it also follows from Soundness for Henkin Models (Lemma 3) and the fact that the ultrametric model is a Henkin one (Lemma 11). ($\Leftarrow$). Let $\mathcal{V}$ be the ultrametric frame isomorphic to $\mathcal{T}/{=_{\beta\eta}}$ of Theorem 6. By Lemma 18, there exists a logical partial function from $\mathcal{U}$ to $\mathcal{V} \sim \mathcal{T}/{=_{\beta\eta}}$. By Lemma 11 and Theorem 3, the ultrametric model and the term model are Henkin models. Hence, $\mathsf{Th}(\mathcal{U}) \subseteq \mathsf{Th}(\mathcal{T}/{=_{\beta\eta}})$ by Lemma 5. □

9 Conclusions and Future Work

As a natural sequel, we are currently studying the theory induced by the ultrametric model for the typed lambda calculus of reactive programs extended with the fixpoint operator of [1].

Statman's 1-Section Theorem [13, 14, 15] generalises Friedman's result by giving necessary and sufficient conditions for a model to satisfy completeness of $\beta\eta$-conversion on terms typable in the simply typed lambda calculus. It will be interesting to prove a similar result to Statman's 1-Section Theorem for the typed lambda calculus of reactive programs.

Our step-indexed applicative structures are in fact Kripke lambda models over the partial order $(\mathbb{N}, \leq)$ in the terminology of Mitchell and Moggi [16]. By using natural numbers, the additional operators for streams such as tl_i^σ can move from time i to $i+1$. However, the notion of Henkin model for reactive programs is not a particular case of Kripke model as defined in [16]. Our environment and meaning function do not have the natural number i as argument since that information is provided by the judgement $\Gamma \vdash M :_i \sigma$.

The notion of step indexed logical relation for recursive types in [17, 18, 19] use the index in a different way from ours. In our definition of logical relation on the type $\sigma \to \tau$ we quantify over $j \geq i$ for some given i. While in the definition of logical relation for recursive types, the quantification is over $j \leq i$ for some given i. The choice to quantify over $j \geq i$ is essential for our proofs to go through. The logical predicate of strongly normalising terms should satisfy $\mathcal{P}_i^{\sigma\to\tau} \subseteq \mathcal{P}_j^{\sigma\to\tau}$ for $i \leq j$ and similarly, the logical surjective function defined in Lemma 18 should satisfy $\mathcal{R}_i^{\sigma\to\tau} \subseteq \mathcal{R}_j^{\sigma\to\tau}$ for $i \leq j$. This holds trivially when we quantify over $j \geq i$ but it does not hold if the quantification is done reversing the order.

Our step-indexed notion of applicative structure can be described as families of *covariant* functors from $(\mathbb{N}, \leq)$ to the category **Set** of sets and functions. The topos of trees [20] that Birkedal and coworkers use for step-indexing models of various programming languages consists of the *contravariant* functors from $(\mathbb{N}, \leq)$ to **Set**. These are functors from $(\mathbb{N}, \leq)^{op}$, that is $(\mathbb{N}, \geq)$, to **Set**.

10 Acknowledgements

We thank Lars Birkedal for asking us a question during ICFP 2012 that led to this paper.

References

[1] N. R. Krishnaswami and N. Benton, "Ultrametric semantics of reactive programs," in *LICS*, 2011, pp. 257–266.
[2] ——, "A semantic model for graphical user interfaces," in *ICFP*, 2011, pp. 45–57.
[3] L. Birkedal, J. Schwinghammer, and K. Støvring, "A metric model of lambda calculus with guarded recursion," 2010, presented at FICS 2010.
[4] M. Escardó, "A metric model of PCF," presented at the Workshop on Realizability Semantics and Applications, June 30-July 1, 1999.
[5] H. Friedman, "Equality between functionals," in *Logic Colloquium*, ser. Springer Lecture Notes, vol. 453, 1975, pp. 22–37.
[6] G. Plotkin, "Notes on completeness of the full continuous hierarchy," 1982, unpublished Manuscript.
[7] J. C. Mitchell, *Foundations for programming languages*, ser. Foundation of computing series. MIT Press, 1996.
[8] R. P. Nederpelt, "Strong normalization in a typed lambda calculus," Ph.D. dissertation, Technische Universiteit Eindhoven, 1973.
[9] J. W. Klop, "Combinatory reduction systems," Ph.D. dissertation, Rijksuniversiteit Utrecht, 1980.
[10] H. P. Barendregt, *The Lambda Calculus: Its Syntax and Semantics*, 2nd ed. Amsterdam: North-Holland, 1984.
[11] Terese, Ed., *Term Rewriting Systems*, ser. Cambridge Tracts in Theoretical Computer Science. Cambridge University Press, 2003, vol. 55.
[12] P. Severi and F.-J. de Vries, "Pure type systems with corecursion on streams: from finite to infinitary normalisation," in *ICFP*, 2012, pp. 141–152.
[13] R. Statman, "Completeness, invariance and λ-definability," *Journal of Symbolic Logic*, 1982.
[14] ——, "Equality between functionals, revisted," in *Harvey Friedman's Research on the Foundations of Mathematics*, 1985, pp. 331–338.
[15] J. G. Riecke, "Statman's 1-section theorem," *Inf. Comput.*, vol. 116, no. 2, pp. 294–303, 1995.
[16] J. C. Mitchell and E. Moggi, "Kripke-style models for typed lambda calculus," *Ann. Pure Appl. Logic*, vol. 51, no. 1-2, pp. 99–124, 1991.
[17] A. J. Ahmed, "Step-indexed syntactic logical relations for recursive and quantified types," in *ESOP*, 2006, pp. 69–83.
[18] L. Birkedal, B. Reus, J. Schwinghammer, K. Støvring, J. Thamsborg, and H. Yang, "Step-indexed Kripke models over recursive worlds," in *POPL*, 2011, pp. 119–132.
[19] D. Dreyer, A. Ahmed, and L. Birkedal, "Logical step-indexed logical relations," *Logical Methods in Computer Science*, vol. 7, no. 2, 2011.
[20] L. Birkedal, R. E. Møgelberg, J. Schwinghammer, and K. Støvring, "First steps in synthetic guarded domain theory: Step-indexing in the topos of trees," in *LICS*, 2011, pp. 55–64.