Categorical Semantics for Functional Reactive Programming with Temporal Recursion and Corecursion

Wolfgang Jeltsch
TTÜ Küberneetika Instituut
Tallinn, Estonia
wolfgang@cs.ioc.ee

Functional reactive programming (FRP) makes it possible to express temporal aspects of computations in a declarative way. Recently we developed two kinds of categorical models of FRP: abstract process categories (APCs) and concrete process categories (CPCs). Furthermore we showed that APCs generalize CPCs. In this paper, we extend APCs with additional structure. This structure models recursion and corecursion operators that are related to time. We show that the resulting categorical models generalize those CPCs that impose an additional constraint on time scales. This constraint boils down to ruling out ω-supertasks, which are closely related to Zeno's paradox of Achilles and the tortoise.

1 Introduction

Functional reactive programming (FRP) makes it possible to express temporal aspects of computations in a declarative way. Traditional FRP is based on behaviors and events, which denote time-varying values and values attached to times, respectively. There is a Curry–Howard correspondence between traditional FRP and an intuitionistic temporal logic with "always" and "eventually" modalities [6, 5]. Thereby the type constructor for behaviors corresponds to "always," and the type constructor for events corresponds to "eventually."

Extending the temporal logic with "until" operators leads to an extended variant of FRP. Thereby proofs of "until" propositions correspond to a new class of FRP constructs, which we call processes. Processes in the FRP sense combine continuous and discrete aspects and generalize behaviors and events [7, Section 2]. We give an introduction to FRP with processes in Section 2.

We have developed two kinds of models of FRP with processes, which can also be used to model temporal logic with "until:"

Abstract process categories (APCs) [8] are defined purely axiomatically. They are an extension of temporal categories [6], which in turn build on categorical models of intuitionistic S4 [9, 2].

Concrete process categories (CPCs) [7] are not defined in a purely axiomatic manner, but use concrete constructions to express time-dependence of type inhabitance and causality of FRP operations.

Abstract process categories generalize concrete process categories. We describe APCs in Section 3 and CPCs in Section 5.

The aim of this paper is to extend APCs and CPCs in order to model recursion and corecursion on processes. We make the following contributions:

- In Section 4, we develop APCs with recursion and corecursion ($\mathcal{R}$-APCs). $\mathcal{R}$-APCs extend APCs with recursive comonad and completely iterative monad structures that model recursion and corecursion on processes. The extensions to APCs arise naturally as extensions of ideal comonad and ideal monad structures that APCs already contain.

To appear in EPTCS.

- In Section 6, we develop CPCs with recursion and corecursion ($\mathcal{R}$-CPCs). $\mathcal{R}$-CPCs differ from CPCs in that they impose an additional constraint on time scales. This constraint boils down to ruling out ω-supertasks [11]. We show that $\mathcal{R}$-APCs generalize $\mathcal{R}$-CPCs.

We discuss related work in Section 7 and give conclusions and an outlook on further work in Section 8.

2 Functional Reactive Programming with Processes

The FRP dialect we consider here is based on a linear notion of time. However the time scale does not need to be discrete; so the time domain can be any totally ordered set. Type inhabitance is time-dependent, which means that every FRP type corresponds to a time-indexed family of sets of inhabitants.

Besides the ordinary type constructors 1, 0, $\times$, $+$, and $\to$, our FRP dialect has two binary type constructors $\rhd''_0$ and $\rhd''_1$, which we call the strong and the weak basic process type constructor. Both have higher precedence than the other binary type constructors.

A value that inhabits a type $\tau_1 \rhd''_0 \tau_2$ at a time t corresponds to a tuple with the following elements:

- a time $t' > t$
- a function h that maps each time t'' with $t < t'' < t'$ to a value that inhabits τ_1 at t''
- a value y that inhabits τ_2 at t'

The function h denotes a time-varying value that exists between times t and t'. We call this time-varying value the continuous part of the process. The pair (t', y) denotes an event that occurs at time t' and carries the value y. We call this event the terminal event of the process.

A value that inhabits a type $\tau_1 \rhd''_1 \tau_2$ at a time t is either a process that inhabits $\tau_1 \rhd''_0 \tau_2$ at t or a time-varying value of type τ_1 that starts immediately after t and persists forever. We regard constructs of the latter kind as special processes that never terminate and thus have an infinite continuous part and no terminal event.

An inhabitant of a type $\tau_1 \rhd''_0 \tau_2$ or $\tau_1 \rhd''_1 \tau_2$ has a continuous part that assigns values only to future times. We define type constructors $\rhd'_0$ and $\rhd'_1$ for processes that start at the present time:

$$\tau_1 \rhd'_0 \tau_2 = \tau_1 \times \tau_1 \rhd''_0 \tau_2 \qquad\qquad \tau_1 \rhd'_1 \tau_2 = \tau_1 \times \tau_1 \rhd''_1 \tau_2 \tag{1}$$

A pair (x, p) represents a process that starts with x as the initial value of its continuous part, and continues like the process p.

An inhabitant of a type $\tau_1 \rhd'_0 \tau_2$ or $\tau_1 \rhd'_1 \tau_2$ can only terminate in the future. We define type constructors $\rhd_0$ and $\rhd_1$ for processes that may terminate at the present time:

$$\tau_1 \rhd_0 \tau_2 = \tau_2 + \tau_1 \rhd'_0 \tau_2 \qquad\qquad \tau_1 \rhd_1 \tau_2 = \tau_2 + \tau_1 \rhd'_1 \tau_2 \tag{2}$$

A value $\iota_1(y)$ represents a process whose terminal event occurs immediately and carries the value y, while a value $\iota_2(p)$ represents the process p, which does not terminate immediately.

From the process type constructors, we derive the type constructors $\Box'$, $\Box$, $\Diamond'$, and $\Diamond$ as follows:

$$\Box'\tau = \tau \rhd''_1 0 \qquad\qquad \Box\tau = \tau \rhd'_1 0 \tag{3}$$

$$\Diamond'\tau = 1 \rhd'_0 \tau \qquad\qquad \Diamond\tau = 1 \rhd_0 \tau \tag{4}$$

Inhabitants of types $\Box'\tau$ and $\Box\tau$ are non-terminating time-varying values and are called behaviors. Inhabitants of types $\Diamond'\tau$ and $\Diamond\tau$ are events.

The FRP dialect described above corresponds to an intuitionistic temporal logic via a Curry–Howard isomorphism. Thereby time-dependent type inhabitance is related to time-dependent trueness of temporal propositions. The type constructors $\rhd_0$ and $\rhd_1$ correspond to the strong and weak “until” operators $\mathcal{U}$ and $\mathcal{W}$ from linear-time temporal logic (LTL), and the type constructors $\Box$ and $\Diamond$ correspond to the “always” and “eventually” modalities.

3 Abstract Process Categories

Abstract process categories (APCs) are axiomatically defined categorical models of FRP with processes, which we developed recently [8]. They are an extension of temporal categories, which are models of FRP with behaviors and events, but without processes [6]. In this section, we give an introduction to APCs.

3.1 The Basics

An APC is a category $\mathcal{C}$ with some additional structure. FRP types are modeled by objects of $\mathcal{C}$. If objects A and B model FRP types τ_1 and τ_2, the morphisms from A to B model operations that turn every value that inhabits τ_1 at some time into a value that inhabits τ_2 at the same time.

Since FRP has the usual type constructors for finite products, finite sums, and function spaces, we require $\mathcal{C}$ to be a cartesian closed category (CCC) with finite coproducts.

3.2 Temporal Functors

Let $\mathbf{2}$ denote the interval category, that is, the category with exactly two objects 0 and 1 and a single non-identity morphism $w : 0 \to 1$. Let furthermore $\mathcal{Q}$ be the category $\mathbf{2} \times \mathcal{C} \times \mathcal{C}$. We require the existence of a functor $\rhd'' : \mathcal{Q} \to \mathcal{C}$, which we call the basic temporal functor. We use the notation $A \rhd''_W B$ for $\rhd''(W, A, B)$. We model the process type constructors $\rhd''_0$ and $\rhd''_1$ by the partial functor applications $\rhd''_0$ and $\rhd''_1$, which are themselves functors from $\mathcal{C} \times \mathcal{C}$ to $\mathcal{C}$.

Every inhabitant of a type $\tau_1 \rhd''_0 \tau_2$ corresponds to an inhabitant of $\tau_1 \rhd''_1 \tau_2$. So there should be an operation that performs a type conversion from $\rhd''_0$ to $\rhd''_1$. We use the natural transformation $\rhd''_w : \rhd''_0 \to \rhd''_1$ to model this operation.

From the functor $\rhd''$, we derive functors $\rhd', \rhd : \mathcal{Q} \to \mathcal{C}$ that model the type constructors $\rhd'_0$, $\rhd'_1$, $\rhd_0$, and $\rhd_1$ as well as functors $\Box', \Box, \Diamond', \Diamond : \mathcal{C} \to \mathcal{C}$ that model the type constructors for behaviors and events:

$$A \rhd'_W B = A \times A \rhd''_W B \qquad A \rhd_W B = B + A \rhd'_W B \tag{5}$$

$$\Box' A = A \rhd''_1 0 \qquad \Box A = A \rhd'_1 0 \tag{6}$$

$$\Diamond' B = 1 \rhd'_0 B \qquad \Diamond B = 1 \rhd_0 B \tag{7}$$

We call the functors introduced in (5) through (7) derived temporal functors.

APCs do not necessarily use the category $\mathbf{2}$ to model how weak a process type constructor is. Any category $\mathcal{W}$ with finite products is allowed instead. In particular, we can use any category that corresponds to a partially ordered set with finite meets. This makes it possible to model more advanced termination properties. An example of such a property is termination with an upper bound on the termination time. We discuss this example in Subsection 5.2.

3.3 Process Expansion and Joining

An APC contains a natural transformation θ'' with

$$\theta''_{W,A,B} : A \rhd''_W B \to \left(A \rhd'_W B\right) \rhd''_W B \quad .$$

This natural transformation models an FRP operation that turns a process p into a process p' such that the following holds:

- The process p' terminates if and only if p terminates.
- The terminal event of p', if any, is the terminal event of p.
- The value of the continuous part of p' at a time t is the suffix of p that starts at t.

We call this FRP operation process expansion. We derive variants of θ'' that work with $\rhd'$ and $\rhd$ instead of $\rhd''$:

$$\begin{aligned} \theta'_{W,A,B} &: A \rhd'_W B \to \left(A \rhd'_W B\right) \rhd'_W B \\ \theta'_{W,A,B} &= \left\langle \mathrm{id}_{A \rhd'_W B}, \theta''_{W,A,B} \circ \pi_2 \right\rangle \end{aligned} \tag{8}$$

$$\begin{aligned} \theta_{W,A,B} &: A \rhd_W B \to \left(A \rhd'_W B\right) \rhd_W B \\ \theta_{W,A,B} &= \mathrm{id}_B + \theta'_{W,A,B} \end{aligned} \tag{9}$$

Besides the natural transformation θ'', an APC also contains a natural transformation ϑ'' with

$$\vartheta''_{W,A,B} : A \rhd''_W (A \rhd_W B) \to A \rhd''_W B \quad .$$

This natural transformation models an FRP operation that turns a process p into a process p' such that the following holds:

- If p terminates, and its terminal event carries a process p^*, then p' is the result of concatenating the continuous part of p and the process p^*.
- If p does not terminate, then p' does not terminate, and the continuous part of p' is the continuous part of p.

We call this FRP operation process joining. We derive variants of ϑ'' that work with $\rhd'$ and $\rhd$ instead of $\rhd''$:

$$\begin{aligned} \vartheta'_{W,A,B} &: A \rhd'_W (A \rhd_W B) \to A \rhd'_W B \\ \vartheta'_{W,A,B} &= \mathrm{id}_A \times \vartheta''_{W,A,B} \end{aligned} \tag{10}$$

$$\begin{aligned} \vartheta_{W,A,B} &: A \rhd_W (A \rhd_W B) \to A \rhd_W B \\ \vartheta_{W,A,B} &= \left[\mathrm{id}_{A \rhd_W B}, \iota_2 \circ \vartheta'_{W,A,B}\right] \end{aligned} \tag{11}$$

The natural transformations θ'' and ϑ'' have to fulfill certain coherence conditions. Regarding θ'', we require that for all $W \in \mathrm{Ob}(\mathcal{W})$ and $B \in \mathrm{Ob}(\mathcal{C})$, $\left(- \rhd''_W B, \theta''_{W,-,B}\right)$ is an ideal comonad.

Definition 1 (Ideal comonad)**.** Let $\mathcal{C}$ be a category with binary products, let U' be an endofunctor on $\mathcal{C}$, and let $\delta' : U' \to U'(\mathrm{Id} \times U')$ be a natural transformation. We define U, ε, and δ as follows:

$$\begin{aligned} U &: \mathcal{C} \to \mathcal{C} & \varepsilon &: U \to \mathrm{Id} & \delta &: U \to UU \\ U &= \mathrm{Id} \times U' & \varepsilon &= \pi_1 & \delta &= \langle \mathrm{id}_U, \delta' \circ \pi_2 \rangle \end{aligned} \tag{12}$$

The pair (U', δ') is an ideal comonad on $\mathcal{C}$ if and only if the diagram in Figure 1 commutes.

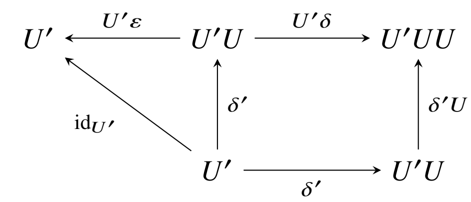

Figure 1: Coherence of an ideal comonad

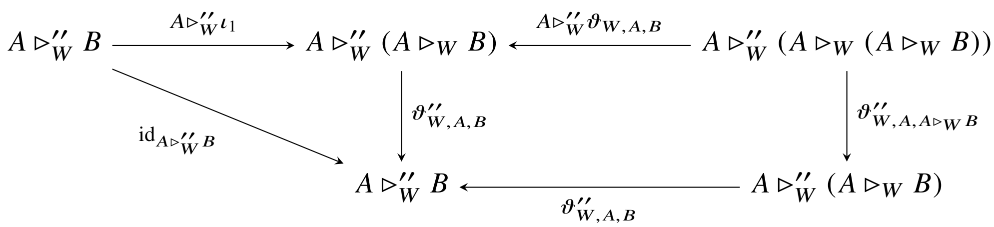

Figure 2: Coherence of process joining

Regarding ϑ'', we require that the diagram in Figure 2 commutes. This coherence condition implies that every pair $\left(A \rhd'_W -, \vartheta'_{W,A,-}\right)$ is an ideal monad, that is, an ideal comonad on $\mathcal{C}^{\mathrm{op}}$. Note that we obtain a diagram for the coherence condition of such an ideal monad by taking the diagram in Figure 2 and replacing $\rhd''$ and ϑ'' with $\rhd'$ and ϑ'.

Finally we require that the diagram in Figure 3 commutes. This ensures that θ'' and ϑ'' interact properly.

3.4 Process Merging and the Canonical Nonterminating Process

There are additional constraints on APCs, which ensure that an APC also models two other operations: process merging and the construction of the sole value of $1 \rhd''_1 0$, which is called the canonical nonterminating process. We do not discuss these additional constraints here, because they are not fundamentally related to our extension of APCs. We just mention them in the definition of APCs in the next subsection.

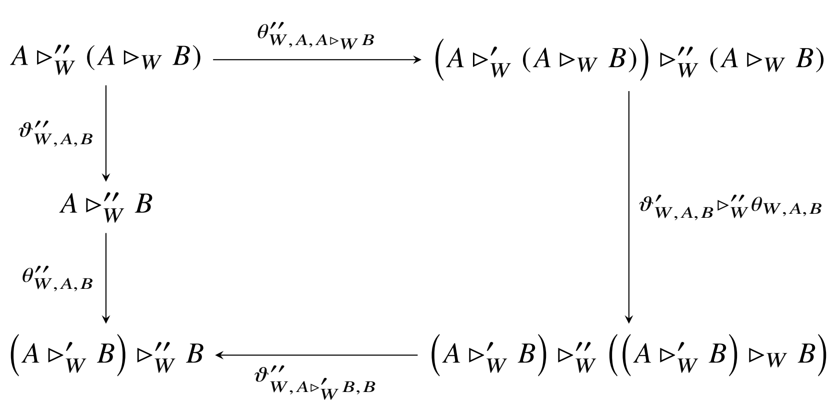

Figure 3: Coherence between process expansion and joining

3.5 Summary

Let us now state the complete definition of abstract process categories.

Definition 2 (Abstract process category)**.** Let $\mathcal{W}$ be a category with finite products, and let $\mathcal{C}$ be a CCC with finite coproducts. Furthermore let $\rhd''$ be a functor from $\mathcal{W} \times \mathcal{C} \times \mathcal{C}$ to $\mathcal{C}$, let $\rhd'$ and $\rhd$ be defined as in (5), and let θ'' and ϑ'' be natural transformations with the following typings:

$$\theta''_{W,A,B} : A \rhd''_W B \to \left(A \rhd'_W B\right) \rhd''_W B$$
$$\vartheta''_{W,A,B} : A \rhd''_W (A \rhd_W B) \to A \rhd''_W B$$

The tuple $(\mathcal{W}, \mathcal{C}, \rhd'', \theta'', \vartheta'')$ is an abstract process category (APC) if and only if the following propositions hold:

- For all $W \in \mathrm{Ob}(\mathcal{W})$ and $B \in \mathrm{Ob}(\mathcal{C})$, $\left(- \rhd''_W B, \theta''_{W,-,B}\right)$ is an ideal comonad.
- For all $W \in \mathrm{Ob}(\mathcal{W})$ and $A, B \in \mathrm{Ob}(\mathcal{C})$, the diagrams in Figures 2 and 3 commute where θ, ϑ', and ϑ are defined as in (9), (10), and (11).
- The natural transformation $\left\langle \chi''_1, \chi''_2 \right\rangle$ with

$$\begin{aligned} \chi''_i \; : \; & (A_1 \times A_2) \rhd''_{W_1 \times W_2} \left(B_1 \times B_2 + B_1 \times A_2 \rhd'_{W_2} B_2 + A_1 \rhd'_{W_1} B_1 \times B_2\right) \\ & \to A_i \rhd''_{W_i} B_i \\ \chi''_i \; = \; & \vartheta''_{W_i, A_i, B_i} \circ \pi_i \rhd''_{\pi_i} [\iota_1 \circ \pi_i, \iota_i \circ \pi_i, \iota_{3-i} \circ \pi_i] \end{aligned} \quad (13)$$

  is an isomorphism.
- The morphism $!_{1 \rhd''_1 0}$ is an isomorphism.

The third and fourth of the above propositions refer to process merging and the canonical nonterminating process mentioned in Subsection 3.4.

4 APCs with Recursion and Corecursion

In Subsection 3.3, we saw that every APC comprises ideal comonads and ideal monads based on temporal functors. There exists a special kind of ideal comonads, called recursive comonads, which captures a form of comonadic recursion. Likewise there exists a special kind of ideal monads, called completely iterative monads, which captures a form of monadic corecursion. In this section, we introduce APCs with recursion and corecursion ($\mathcal{R}$-APCs), which comprise recursive comonads and completely iterative monads that model recursion and corecursion on processes.

4.1 Corecursion on Processes

If $(\mathcal{W}, \mathcal{C}, \rhd'', \theta'', \vartheta'')$ is an APC, then every pair $\left(A \rhd'_W -, \vartheta'_{W,A,-}\right)$ is an ideal monad. We extend the structure of APCs by requiring that every such pair is a completely iterative monad. Completely iterative monads are defined, for example, by Milius [12, Definition 5.5]. We use a definition that is different from the one by Milius, but nevertheless equivalent to it. We compare both definitions in Appendix A, where we also list the advantages that our definition has in our opinion.

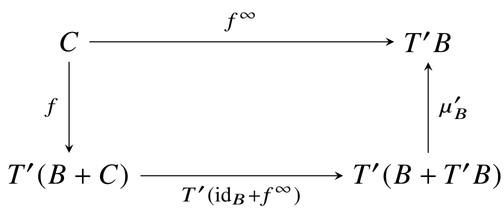

Figure 4: Condition for a morphism f^∞ of a completely iterative monad

Definition 3 (Completely iterative monad)**.** Let $\mathcal{C}$ be a category with binary coproducts. A pair (T', μ') is a completely iterative monad on $\mathcal{C}$ if and only if it is an ideal monad on $\mathcal{C}$, and for every morphism $f : C \to T'(B + C)$, there exists a unique morphism $f^\infty : C \to T'B$ for which the diagram in Figure 4 commutes.

By requiring that every pair $\left(A \rhd'_W -, \vartheta'_{W,A,-}\right)$ is a completely iterative monad, we ensure that for every morphism $f : C \to A \rhd'_W (B + C)$, there is a corresponding morphism $f^\infty : C \to A \rhd'_W B$. Let u and u^∞ be the FRP operations that f and f^∞ model, and let z be a value from the domain of u and u^∞. The diagram in Figure 4 tells us how the process $u^\infty(z)$ is defined:

- If $u(z)$ terminates, and its terminal event carries a value $\iota_1(y)$, then $u^\infty(z)$ terminates at the same time as $u(z)$, the continuous part of $u^\infty(z)$ is the continuous part of $u(z)$, and the terminal event of $u^\infty(z)$ carries the value y.
- If $u(z)$ terminates, and its terminal event carries a value $\iota_2(z')$, then $u^\infty(z)$ is the result of concatenating the continuous part of $u(z)$ and the process $u^\infty(z')$.
- If $u(z)$ does not terminate, then $u^\infty(z)$ does not terminate, and the continuous part of $u^\infty(z)$ is the continuous part of $u(z)$.

Note that in the second case, $u^\infty(z)$ is defined in terms of $u^\infty(z')$, but only the suffix of $u^\infty(z)$ that starts at the termination time of $u(z)$ actually depends on $u^\infty(z')$. The operator $-^\infty$ models corecursion on processes.

We derive a variant of $-^\infty$ that works with $\rhd$ instead of $\rhd'$. We use the notation $-^\infty$ also for this variant. This overloading of notation is possible, because we can always deduce from the types which variant of $-^\infty$ is meant. The $\rhd$-variant turns every morphism $f : C \to B + A \rhd'_W C$ into a morphism f^∞ that is defined as follows:

$$f^\infty : C \to A \rhd_W B$$
$$f^\infty = \left(\mathrm{id}_B + \left(\mathrm{id}_A \rhd'_{\mathrm{id}_W} f\right)^\infty\right) \circ f \tag{14}$$

Note that the $-^\infty$ on the right-hand side refers to the original $\rhd'$-variant. There is also a variant of $-^\infty$ that works with $\rhd''$. It turns every morphism $f : C \to A \rhd''_W (B + A \times C)$ into a morphism f^∞ that is defined as follows:

$$f^\infty : C \to A \rhd''_W B$$
$$f^\infty = \vartheta''_{W,A,B} \circ \left(\mathrm{id}_A \rhd''_{\mathrm{id}_W} (\mathrm{id}_B + (\mathrm{id}_A \times f)^\infty)\right) \circ f \tag{15}$$

Again the $-^\infty$ on the right-hand side refers to the original $\rhd'$-variant.

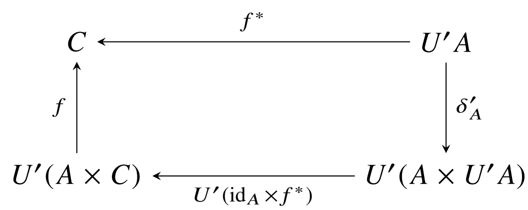

Figure 5: Condition for a morphism f^* of a recursive comonad

4.2 Recursion on Processes

We saw in Subsection 3.3 that every pair $\left(- \rhd''_W B, \theta''_{W,-,B}\right)$ is an ideal comonad. We extend the structure of APCs by requiring that every such pair is a recursive comonad. The notion of recursive comonad is dual to the notion of completely iterative monad. So a recursive comonad on $\mathcal{C}$ is just a completely iterative monad on $\mathcal{C}^{\mathrm{op}}$. We give an explicit definition nevertheless.

Definition 4 (Recursive comonad)**.** Let $\mathcal{C}$ be a category with binary products. A pair (U', δ') is a recursive comonad on $\mathcal{C}$ if and only if it is an ideal comonad on $\mathcal{C}$, and for every morphism $f : U'(A \times C) \to C$, there exists a unique morphism $f^* : U'A \to C$ for which the diagram in Figure 5 commutes.

By requiring that every pair $\left(- \rhd''_W B, \theta''_{W,-,B}\right)$ is a recursive comonad, we ensure that for every morphism $f : (A \times C) \rhd''_W B \to C$, there is a corresponding morphism $f^* : A \rhd''_W B \to C$. Let u and u^* be the FRP operations that f and f^* model, and let p be a process from the domain of u^*. The diagram in Figure 5 tells us that the value $u^*(p)$ is $u\left(p^\dagger\right)$ where $p^\dagger$ is defined by the following statements:

- The process $p^\dagger$ terminates if and only if p terminates.
- The terminal event of $p^\dagger$, if any, is the terminal event of p.
- The value of the continuous part of $p^\dagger$ at a time t is $(x, u^*(p'))$ where x is the value of the continuous part of p at t, and p' is the suffix of p that follows t.

Note that $u^*(p)$ is defined in terms of $u^*(p')$, but p' is a proper suffix of p. The operator $-^*$ models recursion on processes.

We derive a variant of $-^*$ that works with $\rhd'$ instead of $\rhd''$. We use the notation $-^*$ also for this variant. The $\rhd'$-variant turns every morphism $f : A \times C \rhd''_W B \to C$ into a morphism f^* that is defined as follows:

$$f^* \ : \ A \rhd'_W B \to C$$
$$f^* = f \circ \left(\mathrm{id}_A \times \left(f \rhd''_{\mathrm{id}_W} \mathrm{id}_B\right)^*\right) \tag{16}$$

Note that this construction is dual to the construction of the $\rhd$-variant of $-^\infty$.

4.3 Summary

We formulate the definition of $\mathcal{R}$-APCs based on the above explanations.

Definition 5 (APC with recursion and corecursion)**.** An APC with recursion and corecursion ($\mathcal{R}$-APC) is an APC $(\mathcal{W}, \mathcal{C}, \rhd'', \theta'', \vartheta'')$ for which the following holds:

- For all $W \in \mathrm{Ob}(\mathcal{W})$ and $B \in \mathrm{Ob}(\mathcal{C})$, $\left(- \rhd''_W B, \theta''_{W,-,B}\right)$ is a recursive comonad.
- For all $W \in \mathrm{Ob}(\mathcal{W})$ and $A \in \mathrm{Ob}(\mathcal{C})$, $\left(A \rhd'_W -, \vartheta'_{W,A,-}\right)$ is a completely iterative monad.

5 Concrete Process Categories

Concrete process categories (CPCs) are models of FRP with processes that use concrete categorical constructions to express time-dependence of type inhabitance and causality of FRP operations. They are described in detail in an earlier publication of ours [7, Section 3]. Here we only give a short introduction to CPCs.

5.1 Core Structure

Let $(T, \leq)$ be a totally ordered set, and let $\mathcal{B}$ be a CCC with finite coproducts. We use $(T, \leq)$ to model the time scale, and $\mathcal{B}$ to model ordinary types and functions. Based on $(T, \leq)$, we construct a category $\mathcal{I}$, which we call the temporal index category of $(T, \leq)$.

Definition 6 (Temporal index category)**.** The temporal index category of a totally ordered set $(T, \leq)$ is the category $\mathcal{I}$ for which the following holds:

$$\mathrm{Ob}(\mathcal{I}) = \{\, (t, t_o) \in T \times T \mid t \leq t_o \,\} \tag{17}$$

$$\hom_{\mathcal{I}}\left((t', t'_o), (t, t_o)\right) = \begin{cases} \{(t, t_o, t'_o)\} & \text{if } t = t' \text{ and } t_o \leq t'_o \\ \emptyset & \text{otherwise} \end{cases} \tag{18}$$

We model FRP types and FRP operations by the functor category $\mathcal{B}^{\mathcal{I}}$. Let A be an object of $\mathcal{B}^{\mathcal{I}}$ that models an FRP type τ. For every object (t, t_o) of $\mathcal{I}$, the object $A(t, t_o)$ of $\mathcal{B}$ deals with the FRP values that inhabit τ at t; it describes the type whose inhabitants give the information we have about these FRP values at t_o. We call t_o the observation time. For every morphism (t, t_o, t'_o) of $\mathcal{I}$, the morphism $A\,(t, t_o, t'_o)$ models the function that turns information we have at t'_o into information we have at t_o by forgetting all information acquired after t_o

The definition of CPCs requires the category $\mathcal{B}$ to have some additional properties, which are necessary for modeling function types and process types in FRP.

Definition 7 (Concrete process category)**.** Let $(T, \leq)$ be a totally ordered set, let $\mathcal{I}$ be its temporal index category, and let $\mathcal{B}$ be a CCC with finite coproducts that has all products and coproducts of families indexed by intervals $(t, t') \subseteq T$ and all ends of the form $\int_{t'' \in [t, t']} B(t, t'')^{A(t, t'')}$ where $A, B \in \mathcal{B}^{\mathcal{I}}$. Then the tuple $(T, \leq, \mathcal{B})$ is a concrete process category (CPC). The actual category that $(T, \leq, \mathcal{B})$ denotes is the functor category $\mathcal{B}^{\mathcal{I}}$.

5.2 The Basic Temporal Functor

The basic temporal functor of a CPC [8, Subsection 4.3] models the weak basic process type constructor as well as process type constructors that put upper bounds on termination times.

We define the totally ordered set $(T_\infty, \leq_\infty)$ such that $T_\infty = T \cup \{\infty\}$ and

$$t_1 \leq_\infty t_2 \Leftrightarrow t_1 \leq t_2 \vee t_2 = \infty \quad .$$

Let $\mathcal{W}$ be the category of $(T_\infty, \leq_\infty)$. We use $\mathcal{W}$ to model constraints regarding termination. Thereby any object $t_b \in T$ stands for termination at or before the time t_b, and the object ∞ stands for the absence of any termination guarantees. We define the basic temporal functor as follows.

Definition 8 (Basic temporal functor of a CPC)**.** Let $(T, \leq, \mathcal{B})$ be a CPC, and let $\mathcal{W}$ be defined as above. The basic temporal functor of $(T, \leq, \mathcal{B})$ is the functor $\rhd'' : \mathcal{W} \times \mathcal{B}^{\mathcal{I}} \times \mathcal{B}^{\mathcal{I}} \to \mathcal{B}^{\mathcal{I}}$ with

$$\left(A \rhd''_{t_\mathrm{b}} B\right)(t, t_\mathrm{o}) = \begin{cases} 0 & \text{if } t_\mathrm{b} < t \\ S_{t_\mathrm{b}} & \text{if } t \leq t_\mathrm{b} \leq t_\mathrm{o} \\ S_{t_\mathrm{o}} + \prod_{t' \in (t, t_\mathrm{o}]} A(t', t_\mathrm{o}) & \text{if } t_\mathrm{o} <_\infty t_\mathrm{b} \end{cases} \tag{19}$$

where for every $t^* \in T$, S_{t^*} is defined as follows:

$$S_{t^*} = \coprod_{t' \in (t, t^*]} \left(\left(\prod_{t'' \in (t, t')} A(t'', t_\mathrm{o}) \right) \times B(t', t_\mathrm{o}) \right) \tag{20}$$

The above definition is actually incomplete. A complete definition would also define the following:

- morphisms $\left(A \rhd''_W B\right) i$ where $W \in \mathrm{Ob}(\mathcal{W})$, $A, B \in \mathrm{Ob}\left(\mathcal{B}^{\mathcal{I}}\right)$, and $i \in \mathrm{Mor}(\mathcal{I})$
- morphisms $\left(f \rhd''_{\mathrm{id}_W} g\right)_I$ where $W \in \mathrm{Ob}(\mathcal{W})$, $f, g \in \mathrm{Mor}\left(\mathcal{B}^{\mathcal{I}}\right)$, and $I \in \mathrm{Ob}(\mathcal{I})$
- morphisms $(\mathrm{id}_A \rhd''_w \mathrm{id}_B)_I$ where $w \in \mathrm{Mor}(\mathcal{W})$, $A, B \in \mathrm{Ob}\left(\mathcal{B}^{\mathcal{I}}\right)$, and $I \in \mathrm{Ob}(\mathcal{I})$

However defining these things is a mostly mechanical task. In particular, the definition of morphisms $\left(f \rhd''_{\mathrm{id}_W} g\right)_I$ can be derived from (19) and (20) by replacing 0 with id_0 and object expressions of the form $C(t, t_\mathrm{o})$ with morphism expressions of the form $h_{(t, t_\mathrm{o})}$.

5.3 Relationship to Abstract Process Categories

APCs generalize CPCs. This is expressed by the following theorem.

Theorem 1. *Let $(T, \leq, \mathcal{B})$ be a CPC, let $\mathcal{I}$ be the temporal index category of $(T, \leq)$, let $\rhd''$ be the basic temporal functor of $(T, \leq, \mathcal{B})$, and let $\mathcal{W}$ be the category of the totally ordered set $(T_\infty, \leq_\infty)$ that is defined as described in Subsection 5.2. Then there exist natural transformations θ'' and ϑ'' such that $\left(\mathcal{W}, \mathcal{B}^{\mathcal{I}}, \rhd'', \theta'', \vartheta''\right)$ is an APC.*

Please see our earlier work [8, proof of Theorem 10] for a proof of this theorem.

6 CPCs with Recursion and Corecursion

According to Theorem 1, every CPC gives rise to an APC. However there is no guarantee that this APC is also an $\mathcal{R}$-APC. In this section, we develop CPCs with recursion and corecursion ($\mathcal{R}$-CPCs). $\mathcal{R}$-CPCs are a special kind of CPCs whose corresponding APCs are $\mathcal{R}$-APCs.

6.1 Enabling Corecursion on Processes

Let $(T, \leq, \mathcal{B})$ be a CPC. Let furthermore $\left(\mathcal{W}, \mathcal{B}^{\mathcal{I}}, \rhd'', \theta'', \vartheta''\right)$ be the APC it induces according to Theorem 1. This APC models corecursion on processes if and only if for every morphism $f : C \to A \rhd'_{t_\mathrm{b}} (B + C)$, there exists a unique morphism f^∞ for which the following holds:

$$\begin{aligned} f^\infty &: C \to A \rhd'_{t_\mathrm{b}} B \\ f^\infty &= \vartheta'_{t_\mathrm{b}, A, B} \circ \left(\mathrm{id}_A \rhd'_{\mathrm{id}_{t_\mathrm{b}}} (\mathrm{id}_B + f^\infty)\right) \circ f \end{aligned} \tag{21}$$

From the definition of $\triangleright'$ in (5) and the definition of the basic temporal functor in Subsection 5.2, it follows that (21) is true if and only if for all $t, t_{\mathrm{o}} \in T$ with $t \leq t_{\mathrm{o}}$, we have

$$f^{\infty}_{(t,t_{\mathrm{o}})} = \left(\vartheta'_{t_{\mathrm{b}},A,B}\right)_{(t,t_{\mathrm{o}})} \circ (\mathrm{id}_A \times h) \circ f_{(t,t_{\mathrm{o}})} \tag{22}$$

where h is defined as follows:

$$h = \begin{cases} \mathrm{id}_0 & \text{if } t_{\mathrm{b}} < t \\ \coprod_{t' \in (t,t_{\mathrm{b}}]} \left(\mathrm{id} \times \left(\mathrm{id} + f^{\infty}_{(t',t_{\mathrm{o}})}\right)\right) & \text{if } t \leq t_{\mathrm{b}} \leq t_{\mathrm{o}} \\ \left(\coprod_{t' \in (t,t_{\mathrm{o}}]} \left(\mathrm{id} \times \left(\mathrm{id} + f^{\infty}_{(t',t_{\mathrm{o}})}\right)\right)\right) + \mathrm{id} & \text{if } t_{\mathrm{o}} <_{\infty} t_{\mathrm{b}} \end{cases} \tag{23}$$

Equations (22) and (23) define $f^{\infty}_{(t,t_{\mathrm{o}})}$ in terms of morphisms $f^{\infty}_{(t',t_{\mathrm{o}})}$ with $t' > t$. So if the order $\geq$ on the set $\{\, t \in T \mid t \leq t_{\mathrm{o}} \,\}$ is well-founded, (22) and (23) define $f^{\infty}_{(t,t_{\mathrm{o}})}$ by well-founded recursion with recursion parameter t. As a consequence, there exists a unique f^{∞} that fulfills (21).

Based on the above considerations, we require for an $\mathcal{R}$-CPC that for all $t \in T$, the order $\geq$ on $\{\, t' \in T \mid t' \leq t \,\}$ is well-founded, which means that we cannot have an infinite ascending sequence of times that all lie before a certain time. This makes ω-supertasks [11] impossible, which, for example, implies that the sequence of actions described in Zeno's paradox of Achilles and the tortoise cannot occur.

Note however that we still allow time scales that are quite different from the discrete time scale $(\mathbb{N}, \leq)$. For example, $\{\, z + 1/n \mid z \in \mathbb{Z} \wedge n \in \mathbb{N} \setminus \{0\} \,\}$ with the usual ordering of rational numbers is still a perfectly acceptable time scale.

6.2 Enabling Recursion on Processes

The APC $\left(\mathcal{W}, \mathcal{B}^{\mathcal{I}}, \triangleright'', \theta'', \vartheta''\right)$ induced by a CPC $(T, \leq, \mathcal{B})$ models recursion on processes if and only if for every morphism $f : (A \times C) \triangleright''_{t_{\mathrm{b}}} B \to C$, there exists a unique morphism f^* for which the following holds:

$$\begin{aligned} f^* &: A \triangleright''_{t_{\mathrm{b}}} B \to C \\ f^* &= f \circ \left((\mathrm{id}_A \times f^*) \triangleright''_{\mathrm{id}_{t_{\mathrm{b}}}} \mathrm{id}_B\right) \circ \theta''_{t_{\mathrm{b}},A,B} \end{aligned} \tag{24}$$

Equation (24) is true if and only if for all $t, t_{\mathrm{o}} \in T$ with $t \leq t_{\mathrm{o}}$, we have

$$f^*_{(t,t_{\mathrm{o}})} = f_{(t,t_{\mathrm{o}})} \circ h \circ \left(\theta''_{t_{\mathrm{b}},A,B}\right)_{(t,t_{\mathrm{o}})} \tag{25}$$

where h is defined as follows:

$$h = \begin{cases} \mathrm{id}_0 & \text{if } t_{\mathrm{b}} < t \\ \coprod_{t' \in (t,t_{\mathrm{b}}]} s_{t'} & \text{if } t \leq t_{\mathrm{b}} \leq t_{\mathrm{o}} \\ \coprod_{t' \in (t,t_{\mathrm{o}}]} s_{t'} + \prod_{t' \in (t,t_{\mathrm{o}}]} \left(\mathrm{id} \times f^*_{(t',t_{\mathrm{o}})}\right) & \text{if } t_{\mathrm{o}} <_{\infty} t_{\mathrm{b}} \end{cases} \tag{26}$$

$$s_{t'} = \left(\prod_{t'' \in (t,t')} \left(\mathrm{id} \times f^*_{(t'',t_{\mathrm{o}})}\right)\right) \times \mathrm{id} \tag{27}$$

Equations (25) through (27) define $f^*_{(t,t_{\mathrm{o}})}$ by well-founded recursion, assuming that the order $\geq$ on the set $\{\, t \in T \mid t \leq t_{\mathrm{o}} \,\}$ is well-founded. As a consequence, there exists a unique f^* that fulfills (24).

6.3 Summary

We formulate the definition of $\mathcal{R}$-CPCs based on the above explanations.

Definition 9 (CPC with recursion and corecursion)**.** A CPC $(T, \leq, \mathcal{B})$ is a CPC with recursion and corecursion ($\mathcal{R}$-CPC) if and only if for all $t \in T$, the order $\geq$ on $\{ t' \in T \mid t' \leq t \}$ is well-founded.

We have developed $\mathcal{R}$-CPCs such that they give rise to $\mathcal{R}$-APCs. We state the relationship between $\mathcal{R}$-CPCs and $\mathcal{R}$-APCs in the following theorem.

Theorem 2. *The APC that is induced by an $\mathcal{R}$-CPC according to Theorem 1 is an $\mathcal{R}$-APC.*

7 Related Work

We can build $\mathcal{R}$-APCs by taking the categorical semantics of the intuitionistic modal logic IK [1] and successively adding more structure. This leads us first to intuitionistic S4 categories [6, Section 4], which are closely related to the categorical semantics of intuitionistic S4 variants by Kobayashi [9] and Bierman and de Paiva [2]. Subsequent additions to intuitionistic S4 categories lead us to temporal categories [6, Sections 5 and 6], from there to APCs [8], and finally to $\mathcal{R}$-APCs.

Krishnaswami and Benton [10] model FRP categorically based on 1-bounded ultrametric spaces. Their semantics ensures that morphisms only model causal functions, which is also a key property of CPCs. However they model only behaviors, not events or even processes. Furthermore they require the time scale to be discrete. Although this is a restriction, it makes it possible to model recursion on behaviors.

Cave et al. [4] present an FRP calculus that features inductive and coinductive types and a type constructor that corresponds to the "next" modality of LTL. The authors show that process types are definable within their calculus. We strongly conjecture that their calculus also allows for the definition of recursion and corecursion operators on processes. The downside of their solution is that it is fundamentally tied to discrete time.

Jeffrey [5, Section 2] defines categories $\boxminus$**RSet** and $\trianglerighteq$**RSet**. The morphisms of $\boxminus$**RSet** model causal functions on time-varying values, and the morphisms of $\trianglerighteq$**RSet** model causal functions that produce output values based only on a finite history. Jeffrey also introduces a fixed point operator **ifix**, which is similar to our $-^*$ operator for recursion, but uses time-varying values that lie in the past, while $-^*$ uses processes, whose continuous parts lie in the future. Consequently Jeffrey requires that $\leq$, and not $\geq$, is well-founded on certain intervals.

Birkedal et al. [3] model guarded recursion by the topos of trees. The topos of trees is the category $\mathcal{S} = \mathbf{Set}^{\mathcal{N}^{\mathrm{op}}}$, where $\mathcal{N}$ is the category of the ordered set $(\mathbb{N}, \leq)$. If $A \in \mathrm{Ob}(\mathcal{S})$ models a type τ, an object An with $n \in \mathbb{N}$ models the type whose inhabitants give the information we have about inhabitants of τ after n recursion steps, and a morphism Av with $v : m \to n$ models the function that forgets the information acquired during recursion steps $n + 1$ through m. There is a close connection between this development and CPCs. Consider the CPC $(\mathbb{N}, \leq, \mathbf{Set})$. Let $\mathcal{I}$ be the temporal index category of this CPC, and let $\mathcal{E}$ be the full subcategory of $\mathbf{Set}^{\mathcal{I}}$ that is induced by those objects that model types with time-independent inhabitance, that is, by those $A \in \mathrm{Ob}\left(\mathbf{Set}^{\mathcal{I}}\right)$ with $\forall k, l, n \in \mathbb{N} \,.\, A(k, k+n) = A(l, l+n)$ and a similar condition for morphisms. Then the categories $\mathcal{S}$ and $\mathcal{E}$ are isomorphic. Birkedal et al. define a fixed point operator based on a "later" type constructor that corresponds to the "next" modality of temporal logic. Studying the relationship between this fixed point operator and our structure for recursion and corecursion on processes remains a task for the future.

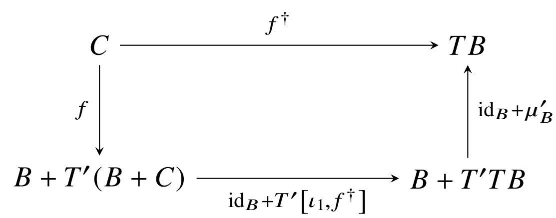

Figure 6: Condition for a morphism $f^\dagger$ of a Milius-style completely iterative monad

8 Conclusions and Further Work

We have developed $\mathcal{R}$-APCs and $\mathcal{R}$-CPCs, which are categorical models of FRP with recursion and corecursion on processes. $\mathcal{R}$-APCs are defined purely axiomatically, while $\mathcal{R}$-CPCs use concrete structure to express time-dependence of type inhabitance and causality of FRP operations. We have furthermore shown that $\mathcal{R}$-APCs generalize $\mathcal{R}$-CPCs.

In the future, we want to develop categorical models of FRP with support for mutable state. Another goal of ours is to implement FRP in mainstream functional programming languages such that the interface directly reflects the abstract categorical semantics.

A Changes in the Completely Iterative Monad Definition

Milius [12, Definition 5.5] defines completely iterative monads differently from us. A simplified version of his definition is as follows.

Definition 10 (Milius-style completely iterative monad)**.** Let $\mathcal{C}$ be a category with binary coproducts. A pair (T', μ') is a Milius-style completely iterative monad on $\mathcal{C}$ if and only if it is an ideal monad on $\mathcal{C}$, and for every morphism $f : C \to B + T'(B + C)$, there exists a unique morphism $f^\dagger : C \to TB$ with $T = \mathrm{Id} + T'$ for which the diagram in Figure 6 commutes.

We do not use this definition in this paper, because in our opinion, it has disadvantages:

- The morphism $f : C \to B + T'(B + C)$ gives us two points per iteration where iterating can come to an end. These two points correspond to the two sums in the codomain of f. However only one such point is necessary.
- The codomain of $f^\dagger$ uses the derived functor T instead of the actual ideal monad functor T', weakening the type of $f^\dagger$.

However our own definition is equivalent to the one by Milius. This fact is expressed by the following two theorems.

Theorem 3. *If (T', μ') is a completely iterative monad on a category $\mathcal{C}$ according to Definition 3, then it is a Milius-style completely iterative monad on $\mathcal{C}$.*

Proof. Let (T', μ') be a completely iterative monad according to Definition 3, and let g be an arbitrary morphism with $g : C \to B + T'(B + C)$. We have to show that there exists a unique morphism $g^\dagger$ for which the following holds:

$$g^\dagger : C \to B + T'B$$
$$g^\dagger = \left(\mathrm{id}_B + \left(\mu'_B \circ T' \left[\iota_1, g^\dagger\right]\right)\right) \circ g \tag{28}$$

We define f as follows:

$$
\begin{aligned}
f &: T'(B+C) \to T'(B+T'(B+C)) \\
f &= T'[\iota_1, g]
\end{aligned}
\tag{29}
$$

Since (T', μ') is a completely iterative monad according to Definition 3, there exists a unique f^∞ for which the following holds:

$$
\begin{aligned}
f^\infty &: T'(B+C) \to T'B \\
f^\infty &= \mu'_B \circ T'(\mathrm{id}_B + f^\infty) \circ f
\end{aligned}
\tag{30}
$$

According to (29), we have

$$
\begin{aligned}
\mu'_B \circ T'(\mathrm{id}_B + f^\infty) \circ f &= \mu'_B \circ T'(\mathrm{id}_B + f^\infty) \circ T'[\iota_1, g] \\
&= \mu'_B \circ T'[\iota_1, (\mathrm{id}_B + f^\infty) \circ g] \ ,
\end{aligned}
\tag{31}
$$

so (30) is equivalent to the following equation:

$$
f^\infty = \mu'_B \circ T'[\iota_1, (\mathrm{id}_B + f^\infty) \circ g]
\tag{32}
$$

We define the functions q and r on morphisms as follows:

$$
\begin{aligned}
q &: \mathrm{Hom}(T'(B+C), T'B) \to \mathrm{Hom}(C, B+T'B) \\
q(a) &= (\mathrm{id}_B + a) \circ g
\end{aligned}
\tag{33}
$$

$$
\begin{aligned}
r &: \mathrm{Hom}(C, B+T'B) \to \mathrm{Hom}(T'(B+C), T'B) \\
r(b) &= \mu'_B \circ T'[\iota_1, b]
\end{aligned}
\tag{34}
$$

We know that there exists a unique f^∞ with

$$
f^\infty = r(q(f^\infty)) \ ,
\tag{35}
$$

and we have to prove that there exists a unique $g^\dagger$ with

$$
g^\dagger = q\left(r\left(g^\dagger\right)\right) \ .
\tag{36}
$$

From (35), it follows that

$$
q(f^\infty) = q(r(q(f^\infty))) \ ,
\tag{37}
$$

so $g^\dagger = q(f^\infty)$ fulfills (36). On the other hand, if we have a $g^\dagger$ that fulfills (36), we know that

$$
r\left(g^\dagger\right) = r\left(q\left(r\left(g^\dagger\right)\right)\right) \ .
\tag{38}
$$

Because there is only one f^∞ that fulfills (35), it follows that $f^\infty = r\left(g^\dagger\right)$. From this, we get

$$
g^\dagger = q\left(r\left(g^\dagger\right)\right) = q(f^\infty) \ .
\tag{39}
$$

So $g^\dagger = q(f^\infty)$ is the unique solution of (36). □

Theorem 4. *If (T', μ') is a Milius-style completely iterative monad on a category $\mathcal{C}$, then it is a completely iterative monad on $\mathcal{C}$ according to Definition 3.*

Proof. Let (T', μ') be a Milius-style completely iterative monad, and let f be an arbitrary morphism with $f : C \to T'(B + C)$. We have to show that there exists a unique morphism f^∞ for which the following holds:

$$f^\infty : C \to T'B$$
$$f^\infty = \mu'_B \circ T'[\iota_1, \iota_2 \circ f^\infty] \circ f \tag{40}$$

We define g as follows:

$$g : C \to B + T'(B + C)$$
$$g = \iota_2 \circ f \tag{41}$$

Since (T', μ') is a Milius-style completely iterative monad, there exists a unique $g^\dagger$ for which the following holds:

$$g^\dagger : C \to B + T'B$$
$$g^\dagger = \left(\mathrm{id}_B + \left(\mu'_B \circ T' \left[\iota_1, g^\dagger\right]\right)\right) \circ g \tag{42}$$

According to (41), we have

$$\begin{aligned} \left(\mathrm{id}_B + \left(\mu'_B \circ T' \left[\iota_1, g^\dagger\right]\right)\right) \circ g &= \left(\mathrm{id}_B + \left(\mu'_B \circ T' \left[\iota_1, g^\dagger\right]\right)\right) \circ \iota_2 \circ f \\ &= \iota_2 \circ \mu'_B \circ T' \left[\iota_1, g^\dagger\right] \circ f \quad , \end{aligned} \tag{43}$$

so (42) is equivalent to the following equation:

$$g^\dagger = \iota_2 \circ \mu'_B \circ T' \left[\iota_1, g^\dagger\right] \circ f \tag{44}$$

We define the functions q and r on morphisms as follows:

$$q : \mathrm{Hom}(C, B + T'B) \to \mathrm{Hom}(C, T'B)$$
$$q(a) = \mu'_B \circ T'[\iota_1, a] \circ f \tag{45}$$

$$r : \mathrm{Hom}(C, T'B) \to \mathrm{Hom}(C, B + T'B)$$
$$r(b) = \iota_2 \circ b \tag{46}$$

We know that there exists a unique $g^\dagger$ with

$$g^\dagger = r\left(q\left(g^\dagger\right)\right) \quad , \tag{47}$$

and we have to prove that there exists a unique f^∞ with

$$f^\infty = q(r(f^\infty)) \quad . \tag{48}$$

Using reasoning analogous to the one at the end of the previous proof, we can deduce that $f^\infty = q\left(g^\dagger\right)$ is the unique solution of (48). □

Acknowledgments

I thank Tarmo Uustalu for all the helpful discussions about the topics of this paper, and in particular, for introducing me to completely iterative monads and recursive comonads.

This work was supported by the target-financed research theme No. 0140007s12 of the Estonian Ministry of Education and Research and by the ERDF through the Estonian Center of Excellence in Computer Science (EXCS) and the national ICTP project *Coinduction for Semantics, Analysis, and Verification of Communicating and Concurrent Reactive Software*. I thank all tax payers in the European Union for funding the ERDF.

References

[1] Gianluigi Bellin, Valeria de Paiva & Eike Ritter (2001): *Extended Curry–Howard Correspondence for a Basic Constructive Modal Logic*. In: *Proceedings of the 2nd Workshop on Methods for Modalities (M4M-2)*.

[2] Gavin Bierman & Valeria de Paiva (2000): *On an Intuitionistic Modal Logic*. *Studia Logica* 65(3), pp. 383–416, doi:10.1023/A:1005291931660.

[3] Lars Birkedal, Rasmus Ejlers Møgelberg, Jan Schwinghammer & Kristian Støvring (2012): *First Steps in Synthetic Guarded Domain Theory: Step-Indexing in the Topos of Trees*. *Logical Methods in Computer Science* 8(4), pp. 1–45, doi:10.2168/LMCS-8(4:1)2012.

[4] Andrew Cave, Francisco Ferreira, Prakash Panangaden & Brigitte Pientka (2014): *Fair Reactive Programming*. In: *Proceedings of the 41st ACM SIGPLAN–SIGACT Symposium on Principles of Programming Languages (POPL '14)*, ACM, New York, pp. 361–372, doi:10.1145/2535838.2535881.

[5] Alan Jeffrey (2012): *LTL Types FRP: Linear-Time Temporal Logic Propositions as Types, Proofs as Functional Reactive Programs*. In: *Proceedings of the Sixth Workshop on Programming Languages Meets Program Verification (PLPV '12)*, ACM, New York, pp. 49–60, doi:10.1145/2103776.2103783.

[6] Wolfgang Jeltsch (2012): *Towards a Common Categorical Semantics for Linear-Time Temporal Logic and Functional Reactive Programming*. *Electronic Notes in Theoretical Computer Science* 286, pp. 229–242, doi:10.1016/j.entcs.2012.08.015.

[7] Wolfgang Jeltsch (2013): *Temporal Logic with "Until", Functional Reactive Programming with Processes, and Concrete Process Categories*. In: *Proceedings of the 7th Workshop on Programming Languages Meets Program Verification (PLPV '13)*, ACM, New York, pp. 69–78, doi:10.1145/2428116.2428128.

[8] Wolfgang Jeltsch (2014): *An Abstract Categorical Semantics for Functional Reactive Programming with Processes*. In: *Proceedings of the ACM SIGPLAN 2014 Workshop on Programming Languages Meets Program Verification (PLPV '14)*, ACM, New York, pp. 47–58, doi:10.1145/2541568.2541573.

[9] Satoshi Kobayashi (1997): *Monad as Modality*. *Theoretical Computer Science* 175(1), pp. 29–74, doi:10.1016/S0304-3975(96)00169-7.

[10] Neelakantan R. Krishnaswami & Nick Benton (2011): *Ultrametric Semantics of Reactive Programs*. In: *Proceedings of the 26th Annual IEEE Symposium on Logic in Computer Science (LICS '11)*, IEEE, New York, pp. 257–266, doi:10.1109/LICS.2011.38.

[11] Jon Pérez Laraudogoitia (2013): *Supertasks*. In Edward N. Zalta, editor: *The Stanford Encyclopedia of Philosophy*, spring 2013 edition, Stanford University, Stanford, California. Available at `http://plato.stanford.edu/archives/spr2013/entries/spacetime-supertasks/`.

[12] Stefan Milius (2005): *Completely Iterative Algebras and Completely Iterative Monads*. *Information and Computation* 196(1), pp. 1–41, doi:10.1016/j.ic.2004.05.003.