An Abstract Categorical Semantics for Functional Reactive Programming with Processes

Wolfgang Jeltsch

TTÜ Küberneetika Instituut
Akadeemia tee 21, 12618 Tallinn, Estonia
http://wolfgang.jeltsch.info/

Abstract

Linear-time temporal logic and functional reactive programming (FRP) are related via a Curry–Howard correspondence. Thereby proofs of "always," "eventually," and "until" propositions correspond to behaviors, events, and processes, respectively. Processes in the FRP sense combine continuous and discrete aspects and generalize behaviors and events. In this paper, we develop a class of axiomatically defined categorical models of FRP with processes. We call these models abstract process categories (APCs). We relate APCs to other categorical models of FRP, namely temporal categories and concrete process categories.

Categories and Subject Descriptors F.3.2 [*Logics and Meanings of Programs*]: Semantics of Programming Languages; D.1.1 [*Programming Techniques*]: Applicative (Functional) Programming; F.4.1 [*Mathematical Logic and Formal Languages*]: Mathematical Logic—Temporal logic

Keywords Functional Reactive Programming, Curry–Howard Correspondence, Category Theory, Categorical Semantics

1. Introduction

Functional reactive programming (FRP) allows programs to deal with temporal aspects in a declarative way. 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 FRP and intuitionistic temporal logic (Jeltsch 2012; Jeffrey 2012). Thereby the type constructors for behaviors and events correspond to the "always" and "eventually" modalities. We have shown that the proofs of "until" propositions in temporal logic correspond to constructs that combine aspects of behaviors and events (Jeltsch 2013, Section 2). We call these constructs processes.

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

Temporal categories (Jeltsch 2012) model FRP with behaviors and events, but without processes. They are based on categorical models of intuitionistic S4 developed by Kobayashi (1997) and Bierman and de Paiva (2000). Temporal categories are defined in a purely axiomatic way.

Concrete process categories (Jeltsch 2013) model FRP with processes. They are not purely axiomatically definied, but use concrete constructions to express time-dependence of type inhabitance and causality of FRP operations.

In this paper, we develop an axiomatically defined categorical semantics for FRP with processes. We make the following contributions:

- In Section 2, we give a revised definition of temporal categories that fixes some issues of the original definition. We discuss the differences between the old and the new definition in Appendix A.
- In Section 3, we define abstract process categories (APCs), which are models of FRP with processes. Like temporal categories, APCs are axiomatically defined, but they contain more structure than temporal categories.
- In Section 4, we present several results about APCs. In particular, we show that every APC gives rise to a temporal category and that APCs generalize concrete process categories.

We discuss related work in Section 5 and give conclusions and an outlook on further work in Section 6.

2. Temporal Categories

Temporal categories model a variant of FRP that supports behaviors and events, but not processes. This FRP variant 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 $\to$, there are two unary type constructors $\Box'$ and $\Diamond'$ for behaviors and events. A value that inhabits a type $\Box'\tau$ at a time t corresponds to a function that maps each time $t' > t$ to a value that inhabits τ at t'. Likewise a value that inhabits a type $\Diamond'\tau$ at a time t corresponds to a pair of a time $t' > t$ and a value that inhabits τ at t'.

The FRP variant 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 $\Box'$ and $\Diamond'$ correspond to an "always" operator $\Box'$ and an "eventually" operator $\Diamond'$, respectively. A proposition $\Box'\varphi$ states that φ will hold at every future time, and a proposition $\Diamond'\varphi$ states that φ will hold at some future time.

In the remainder of this section, we describe how type constructors and operations of FRP can be modeled by categorical constructs, which ultimately leads to the definition of temporal categories.

PLPV '14, January 21, 2014, San Diego, CA, USA.

ACM 978-1-4503-2567-7/14/01…$15.00.
http://dx.doi.org/10.1145/10.1145/2541568.2541573

2.1 The Basics

A temporal category is a category C with some additional structure. FRP types are modeled by objects of 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 C to be a cartesian closed category (CCC) with finite coproducts.

2.2 Behaviors

We model the type constructor $\Box'$ by an endofunctor $\Box'$ on the category C. If a morphism $f : A \to B$ models an FRP operation q, the morphism $\Box' f : \Box' A \to \Box' B$ models an FRP operation that transforms a behavior by applying q to all values the behavior takes during its lifetime.

Inhabitants of types $\Box'\tau$ assign values only to future times. We can represent time-varying values that start at the present time by pairs of an initial value and an ordinary behavior. So a type constructor $\Box$ for present-including time-varying values can be defined by the equation $\Box\tau = \tau \times \Box'\tau$. The type constructor $\Box$ is modeled by the endofunctor $\Box$ that is defined by $\Box = \mathrm{Id} \times \Box'$.

A pair of behaviors corresponds to a behavior of pairs. Furthermore there is only one value of type $\Box' 1$, namely the behavior that assigns the sole value of type 1 to every future time. Therefore we require the existence of two natural transformations φ' and ψ' that together with $\Box'$ form a cartesian functor.

Definition 1 (Cartesian functor)**.** Let C and $\mathcal{D}$ be categories with finite products. A cartesian functor from C to $\mathcal{D}$ is a monoidal functor[1] from $(C, \times, 1)$ to $(\mathcal{D}, \times, 1)$.

The cartesian functor $(\Box', \varphi', \psi')$ gives rise to a cartesian functor structure for $\Box$ in a straightforward way.

There is an important lemma for cartesian functors, which we want to mention here without a proof.

Lemma 1. *If C and $\mathcal{D}$ are categories with finite products, and (F, φ, ψ) is a cartesian functor from C to $\mathcal{D}$, then (F, φ, ψ) is a symmetric monoidal functor, and the equations $\varphi^{-1} = \langle F\pi_1, F\pi_2 \rangle$ and $\psi^{-1} = !_{F1}$ hold.*

From a behavior b that inhabits a type $\Box'\tau$ at a time t, we can construct a behavior of type $\Box'\Box\tau$ whose value at a time $t' > t$ is the suffix of b that starts at t'. We model the operation that constructs such behaviors of suffixes by a natural transformation $\delta' : \Box' \to \Box'\Box$. The endofunctor $\Box'$ and the natural transformation δ' together have to form an ideal comonad.

Definition 2 (Ideal comonad)**.** Let C be a category with binary products, let U' be an endofunctor on C, and let $\delta' : U' \to U'(\mathrm{Id} \times U')$ be a natural transformation. We define U, ε, and δ as follows:

$$U : C \to C \qquad \varepsilon : U \to \mathrm{Id} \qquad \delta : U \to UU$$
$$U = \mathrm{Id} \times U' \qquad \varepsilon = \pi_1 \qquad \delta = \langle \mathrm{id}_U, \delta' \circ \pi_2 \rangle$$

The pair (U', δ') is an ideal comonad on C if and only if the diagrams in Figure 1 commute.

Corollary 2. *If C is a category with binary products, (U', δ') is an ideal comonad on C, and U, ε, and δ are defined as in Definition 2, then (U, ε, δ) is a comonad.*

[1] Note that our use of the term "monoidal functor" implies that the coherence maps are isomorphisms.

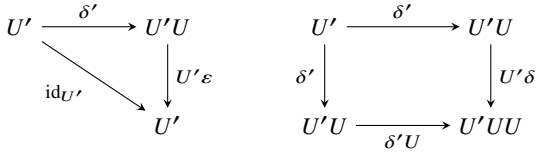

Figure 1. Coherence of an ideal comonad

We want the cartesian functor $(\Box', \varphi', \psi')$ and the ideal comonad $(\Box', \delta')$ to cohere, that is, we want $(\Box', \varphi', \psi', \delta')$ to be a symmetric monoidal ideal comonad.

Definition 3 (Symmetric monoidal ideal comonad)**.** Let $(C, \otimes, I)$ be a symmetric monoidal category where C has binary products. A symmetric monoidal ideal comonad on $(C, \otimes, I)$ is a tuple $(U', \varphi', \psi', \delta')$ where (U', φ', ψ') is a symmetric monoidal endofunctor on $(C, \otimes, I)$, (U', δ') is an ideal comonad on C, and δ' is a monoidal transformation.

We do not have to require the coherence between the cartesian functor and the ideal comonad explicitly, since it holds automatically according to the following lemma.

Lemma 3. *If C and $\mathcal{D}$ are categories with finite products, (F, φ, ψ) and (F', φ', ψ') are cartesian functors from C to $\mathcal{D}$, and τ is a natural transformation from F to F', then τ is a monoidal transformation from (F, φ, ψ) to (F', φ', ψ').*

This lemma can be proved using the equations from Lemma 1.

2.3 Events

The meaning of the type constructor $\Diamond'$ is an endofunctor $\Diamond'$ on C. If a morphism $f : A \to B$ models an FRP operation q, the morphism $\Diamond' f : \Diamond' A \to \Diamond' B$ models an FRP operation that transforms an event by applying q to the value it carries.

Types $\Diamond'\tau$ only allow for event occurrences in the future. We can represent events that may also occur immediately by values of types $\tau + \Diamond'\tau$. Thereby a value $\iota_1(x)$ denotes an event that occurs at the present time and carries the value x, while a value $\iota_2(e)$ denotes an ordinary event e, which occurs in the future. We can define a type constructor $\Diamond$ by $\Diamond\tau = \tau + \Diamond'\tau$ and model it by the endofunctor $\Diamond$ that is defined by $\Diamond = \mathrm{Id} + \Diamond'$.

An event e that inhabits a type $\Diamond'\Diamond\tau$ at a time t occurs at a time $t' > t$ and carries as its value an event that occurs at a time $t'' \geq t'$ and carries a value x of type τ. We can turn e into an event e' that inhabits $\Diamond'\tau$ at t, occurs at t'', and carries x. We model the operation that turns e into e' by a natural transformation $\mu' : \Diamond'\Diamond \to \Diamond'$. The endofunctor $\Diamond'$ and the natural transformation μ' together have to form an ideal monad.

Definition 4 (Ideal monad)**.** Let C be a category with binary coproducts. A pair (T', μ') is an ideal monad on C if and only if it is an ideal comonad on C^{op}.

Ideal monads give rise to monads analogously to the way ideal comonads give rise to comonads shown in Corollary 2.

We can merge two events e_1 and e_2 of types $\Diamond'\tau_1$ and $\Diamond'\tau_2$ to become a single event e' of type $\Diamond'(\tau_1 \odot \tau_2)$ where the type constructor $\odot$ is defined by

$$\tau_1 \odot \tau_2 = \tau_1 \times \tau_2 + \tau_1 \times \Diamond'\tau_2 + \Diamond'\tau_1 \times \tau_2 \quad .$$

If e_1 and e_2 occur at times t_1 and t_2 and carry values x_1 and x_2, e' occurs at $\min(t_1, t_2)$ and carries a value $\iota_1(s)$, $\iota_2(s)$, or $\iota_3(s)$ depending on whether $t_1 = t_2$, $t_1 < t_2$, or $t_1 > t_2$. In the first case, s is (x_1, x_2). In the second case, it is $\left(x_1, e_2'\right)$ where e_2' occurs at t_2 and carries the value x_2, but unlike e_2, inhabits its type at t_1. The third case is analogous to the second one. We define a functor

$\odot : C \times C \to C$ analogously to the type constructor $\odot$ and model merging of events by a natural transformation ϱ' with

$$\varrho'_{A,B} : \diamond' A \times \diamond' B \to \diamond'(A \odot B) \quad .$$

An event e' that results from merging two events e_1 and e_2 conveys the same information as the pair (e_1, e_2). So ϱ' must be an isomorphism. Either event e_i with $i \in \{1,2\}$ can be recovered from e' by applying the operation that is modeled by the natural transformation ζ'_i defined as follows:

$$\zeta'_i \ : \ \diamond'(C_1 \odot C_2) \to \diamond' C_i$$
$$\zeta'_i = \mu'_{C_i} \circ \diamond'[\iota_1 \circ \pi_i, \iota_i \circ \pi_i, \iota_{3-i} \circ \pi_i]$$

So $\left\langle \zeta'_1, \zeta'_2 \right\rangle$ must be the inverse of ϱ'. Therefore we do not directly require ϱ' to exist. Instead we require $\left\langle \zeta'_1, \zeta'_2 \right\rangle$ to be an isomorphism. We can then define ϱ' as $\left\langle \zeta'_1, \zeta'_2 \right\rangle^{-1}$.

For every $i \in \{1,2\}$, we can define a natural transformation $\zeta_i : \diamond(C_1 \odot C_2) \to \diamond C_i$ by taking the definition of ζ'_i and replacing μ' and $\diamond'$ with μ and $\diamond$. The reader is invited to check that $\left\langle \zeta'_1, \zeta'_2 \right\rangle$ being an isomorphism implies that $\langle \zeta_1, \zeta_2 \rangle$ is an isomorphism. The inverse of $\langle \zeta_1, \zeta_2 \rangle$ models an event merging operation that works with $\diamond$ instead of $\diamond'$.

2.4 Interplay between Behaviors and Events

Behaviors and events can interact through the sampling operation. This operation turns a behavior b and an event e that occurs at a time t into an event that also occurs at t, but carries a pair that consists of the value of b at t and the value that e carries. We model the sampling operation by a natural transformation σ' with

$$\sigma'_{A,B} : \square' A \times \diamond' B \to \diamond'(A \times B) \quad .$$

To ensure coherence between the cartesian functor $(\square', \varphi', \psi')$ and σ', we require $(\diamond', \sigma')$ to be a $\square'$-strong functor.

Definition 5 (Strong functor)**.** Let $(C, \otimes, I)$ and $(\mathcal{D}, \oplus, J)$ be symmetric monoidal categories, and let (F, φ, ψ) be a symmetric monoidal functor from $(C, \otimes, I)$ to $(\mathcal{D}, \oplus, J)$. An F-strong functor is a pair (G, σ) where G is a functor from C to $\mathcal{D}$, and σ is a natural transformation with

$$\sigma_{A,B} : FA \oplus GB \to G(A \otimes B)$$

for which the diagrams in Figure 2 commute. The natural transformation σ is called tensorial strength.

Corollary 4. *If $(C, \otimes, I)$, $(\mathcal{D}, \oplus, J)$, and $(\mathcal{E}, \ominus, K)$ are symmetric monoidal categories, $(F_1, \varphi_1, \psi_1) : (\mathcal{D}, \oplus, J) \to (\mathcal{E}, \ominus, K)$ and $(F_2, \varphi_2, \psi_2) : (C, \otimes, I) \to (\mathcal{D}, \oplus, J)$ are symmetric monoidal functors, (G_1, σ_1) is an F_1-strong functor, and (G_2, σ_2) is an F_2-strong functor, then $(G_1 G_2, G_1 \sigma_2 \circ \sigma_1)$ is an $F_1 F_2$-strong functor.*

We also need coherence between the ideal monad $(\diamond', \mu')$ and σ'. Therefore we require $(\diamond', \sigma', \mu')$ to be a strong ideal monad, which means that we require μ' to be a strong transformation.

Definition 6 (Strong transformation)**.** Let $(C, \otimes, I)$ and $(\mathcal{D}, \oplus, J)$ be symmetric monoidal categories, let (F, φ, ψ) be a symmetric monoidal functor from $(C, \otimes, I)$ to $(\mathcal{D}, \oplus, J)$, and let (G, σ) and (G', σ') be F-strong functors. A strong transformation from (G, σ) to (G', σ') is a natural transformation $\tau : G \to G'$ for which the diagram in Figure 3 commutes.

Definition 7 (Strong ideal monad)**.** Let C be a distributive category, let (U', φ', ψ') be a cartesian endofunctor on C, and let (U', δ') be an ideal comonad on C. A U'-strong ideal monad is a tuple (T', σ', μ') where (T', σ') is a U'-strong functor, (T', μ') is an ideal monad on C, and μ' is a strong transformation.

The requirement that μ' is a strong transformation only makes sense if $\diamond'\diamond$, the domain of μ', has an accompanying tensorial strength. The following lemma shows that this is indeed the case.

Lemma 5. *If C is a distributive category, (U', φ', ψ') is a cartesian endofunctor on C, (U', δ') is an ideal comonad on C, and (G', σ') is a U'-strong functor, then there is a canonical natural transformation $\hat{\sigma}$ such that $(G'(\mathrm{Id} + G'), \hat{\sigma})$ is a U'-strong functor.*

Proof. Let U and G be defined by $U = \mathrm{Id} \times U'$ and $G = \mathrm{Id} + G'$. Let furthermore ς denote the inverse of the canonical distributivity transformation:

$$\varsigma_{A,B,C} \ : \ A \times (B + C) \to A \times B + A \times C$$
$$\varsigma_{A,B,C} = [\mathrm{id}_A \times \iota_1, \mathrm{id}_A \times \iota_2]^{-1}$$

We define the natural transformation σ as follows:

$$\sigma_{A,B} \ : \ UA \times GB \to G(A \times B)$$
$$\sigma_{A,B} = ((\pi_1 \times \mathrm{id}_B) + \sigma'_{A,B} \circ (\pi_2 \times \mathrm{id}_{G'B})) \circ \varsigma_{UA,B,G'B}$$

The reader is invited to check that (G, σ) is a U-strong functor. According to Corollary 4, it follows that $(G'G, G'\sigma \circ \sigma')$ is a $U'U$-strong functor. Using the fact that δ' is a monoidal transformation, we can deduce that $(G'G, \hat{\sigma})$ with $\hat{\sigma} = G'\sigma \circ \sigma' \circ (\delta' \times \mathrm{id})$ is a U'-strong functor. □

2.5 Summary

We formulate the definition of temporal categories based on the above explanations.

Definition 8 (Temporal category)**.** A temporal category is a tuple $(C, \square', \varphi', \psi', \delta', \diamond', \sigma', \mu')$ with the following properties:

- C is a CCC with finite coproducts.
- $(\square', \varphi', \psi')$ is a cartesian endofunctor on C.
- $(\square', \delta')$ is an ideal comonad on C.
- $(\diamond', \sigma', \mu')$ is a $\square'$-strong ideal monad.
- The natural transformation $\left\langle \zeta'_1, \zeta'_2 \right\rangle$ with

  $$\zeta'_i \ : \ \diamond'(C_1 \times C_2 + C_1 \times \diamond' C_2 + \diamond' C_1 \times C_2) \to \diamond' C_i$$
  $$\zeta'_i = \mu'_{C_i} \circ \diamond'[\iota_1 \circ \pi_i, \iota_i \circ \pi_i, \iota_{3-i} \circ \pi_i]$$

  is an isomorphism.

3. Abstract Process Categories

Let us now consider a dialect of FRP with support for processes. Compared to the FRP variant from the previous section, this FRP dialect additionally contains two binary type constructors $\triangleright''_0$ and $\triangleright''_1$, which we call the strong and the weak basic process type constructor, respectively.[2]

A value that inhabits a type $\tau_1 \triangleright''_0 \tau_2$ 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 x that inhabits τ_2 at t'

The function h denotes a time-varying value that does not persist forever, but vanishes immediately before t'. We call this time-varying value the continuous part of the process. The pair (t', x) denotes an event that immediately follows the continuous part. We call this event the terminal event of the process.

[2] In an earlier work (Jeltsch 2013), we named these type constructors $\triangleright''$ and $\blacktriangleright''$ instead.

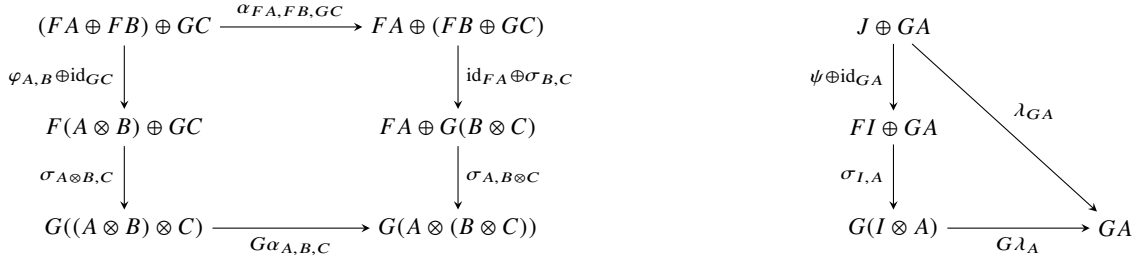

Figure 2. Coherence of a strong functor

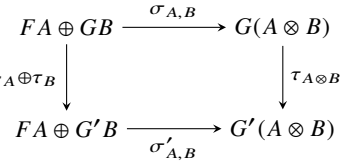

Figure 3. Coherence of a strong transformation

A value of a type $\tau_1 \triangleright_1'' \tau_2$ covers all values of type $\tau_1 \triangleright_0'' \tau_2$ and furthermore all behaviors over τ_1. We regard the latter as special processes that never terminate and thus have an infinite continuous part and no terminal event.

The counterparts of $\triangleright_0''$ and $\triangleright_1''$ in temporal logic are future-only variants of the strong and weak "until" operators $\mathcal{U}$ and $\mathcal{W}$ from linear-time temporal logic (LTL).

From $\triangleright_0''$ and $\triangleright_1''$, we derive the type constructors $\triangleright_0'$ and $\triangleright_1'$ whose inhabitants are processes whose continuous parts already start at the present time:[3]

$$\tau_1 \triangleright_0' \tau_2 = \tau_1 \times \tau_1 \triangleright_0'' \tau_2 \qquad \tau_1 \triangleright_1' \tau_2 = \tau_1 \times \tau_1 \triangleright_1'' \tau_2$$

From $\triangleright_0'$ and $\triangleright_1'$ in turn, we derive the type constructors $\triangleright_0$ and $\triangleright_1$ whose inhabitants are processes that may terminate immediately:

$$\tau_1 \triangleright_0 \tau_2 = \tau_2 + \tau_1 \triangleright_0' \tau_2 \qquad \tau_1 \triangleright_1 \tau_2 = \tau_2 + \tau_1 \triangleright_1' \tau_2$$

Behaviors and events are just special kinds of processes, since we can define the type constructors $\square'$ and $\Diamond'$ in terms of process type constructors:

$$\square'\tau = \tau \triangleright_1'' 0 \qquad \Diamond'\tau = 1 \triangleright_0' \tau$$

Therefore we use temporal categories as the starting point for the development of APCs. We extend the categorical structure that is related to behaviors and events in order to get a categorical structure that models process types and process operations.

3.1 Temporal Functors

Let **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 $\mathbf{2} \times C \times C$. We require the existence of a functor $\triangleright'' : \mathcal{Q} \to C$, which we call the basic temporal functor. We use the notation $A \triangleright_W'' B$ for $\triangleright''(W, A, B)$. We model the process type constructors $\triangleright_0''$ and $\triangleright_1''$ by the partial functor applications $\triangleright_0''$ and $\triangleright_1''$, which are themselves functors from $C \times C$ to C.

Every inhabitant of a type $\tau_1 \triangleright_0'' \tau_2$ corresponds to an inhabitant of $\tau_1 \triangleright_1'' \tau_2$. So there should be an operation that performs a type conversion from $\triangleright_0''$ to $\triangleright_1''$. We use the natural transformation $\triangleright_w'' : \triangleright_0'' \to \triangleright_1''$ to model this operation.

[3] For reading the following equations, note that all process type constructors have higher precedence than all other binary type constructors.

If W is an object of **2**, $f : A_1 \to A_2$ is a morphism that models an FRP operation q, and $g : B_1 \to B_2$ is a morphism that models an FRP operation r, the morphism $f \triangleright_{\mathrm{id}_W}'' g : A_1 \triangleright_W'' B_1 \to A_2 \triangleright_W'' B_2$ models an FRP operation that transforms a process by applying q to all values of its continuous part and r to the value that its terminal event carries.

From the functor $\triangleright''$, we can derive functors $\triangleright', \triangleright : \mathcal{Q} \to C$ for modeling the type constructors $\triangleright_0'$, $\triangleright_1'$, $\triangleright_0$, and $\triangleright_1$, as well as functors $\square'$, $\square$, $\Diamond'$, and $\Diamond$ that model type constructors for behaviors and events:

$$A \triangleright_W' B = A \times A \triangleright_W'' B \qquad \square' A = A \triangleright_1'' 0 \qquad \Diamond' B = 1 \triangleright_0' B$$
$$A \triangleright_W B = B + A \triangleright_W' B \qquad \square A = A \triangleright_1' 0 \qquad \Diamond B = 1 \triangleright_0 B$$

We call these functors derived temporal functors.

3.2 Process Expansion

We saw in Subsection 2.2 that a temporal category comprises an ideal comonad whose functor part is $\square'$. We want the same to hold for an APC, since we want an APC to contain all the structure of a temporal category. As we saw at the end of the previous section, $\square'$ is defined as $- \triangleright_1'' 0$ for APCs. So we require an APC to contain a natural transformation δ' with

$$\delta'_A : A \triangleright_1'' 0 \to \left(A \triangleright_1' 0\right) \triangleright_1'' 0$$

such that $\left(- \triangleright_1'' 0, \delta'\right)$ is an ideal comonad.

We extend this structure in a straightforward way by allowing arbitrary arguments $W \in \mathrm{Ob}(\mathbf{2})$ and $B \in \mathrm{Ob}(C)$ for the functors $\triangleright''$ and $\triangleright'$ in place of the specific arguments 1 and 0. That is, we require that every functor $- \triangleright_W'' B$ has an associated natural transformation with whom it forms an ideal comonad. We want the family of all these natural transformations to be natural in W and B. Therefore we require that there exists a single natural transformation θ'' with

$$\theta''_{W,A,B} : A \triangleright_W'' B \to \left(A \triangleright_W' B\right) \triangleright_W'' B$$

such that for all $W \in \mathrm{Ob}(\mathbf{2})$ and $B \in \mathrm{Ob}(C)$, $\left(- \triangleright_W'' B, \theta''_{W,-,B}\right)$ is an ideal comonad.

The natural transformation θ'' 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.

In Corollary 2, we showed how a comonad can be derived from an ideal comonad, and in particular, how the comultiplication δ of such a comonad is derived from the natural transformation δ' of the corresponding ideal comonad. We analogously derive a variant

of θ'' that works with $\triangleright'$ instead of $\triangleright''$:

$$\theta'_{W,A,B} \;:\; A \triangleright'_W B \to \left(A \triangleright'_W B\right) \triangleright'_W B$$
$$\theta'_{W,A,B} = \left\langle \mathrm{id}_{A \triangleright'_W B}, \theta''_{W,A,B} \circ \pi_2 \right\rangle$$

Finally we derive a natural transformation θ that is analogous to θ'' and θ', but works with $\triangleright$:

$$\theta_{W,A,B} \;:\; A \triangleright_W B \to \left(A \triangleright'_W B\right) \triangleright_W B$$
$$\theta_{W,A,B} = \mathrm{id}_B + \theta'_{W,A,B}$$

3.3 Process Joining

In Subsection 2.3, we saw that a temporal category also comprises an ideal monad whose functor part is $\diamond'$. Therefore we require an APC to contain a natural transformation μ' with

$$\mu'_B : 1 \triangleright'_0 (1 \triangleright_0 B) \to 1 \triangleright'_0 B$$

such that $\left(1 \triangleright'_0 -, \mu'\right)$ is an ideal monad.

We extend this structure such that it works with arbitrary functor arguments W and A in place of the specific arguments 0 and 1. This means that we require that there exists a natural transformation ϑ' with

$$\vartheta'_{W,A,B} : A \triangleright'_W (A \triangleright_W B) \to A \triangleright'_W B$$

such that for all $W \in \mathrm{Ob}(\mathbf{2})$ and $A \in \mathrm{Ob}(C)$, $\left(A \triangleright'_W -, \vartheta'_{W,A,-}\right)$ is an ideal monad.

The natural transformation ϑ' 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 a variant of ϑ' that works with $\triangleright$ instead of $\triangleright'$. We do this dually to deriving θ' from θ'':

$$\vartheta_{W,A,B} \;:\; A \triangleright_W (A \triangleright_W B) \to A \triangleright_W B$$
$$\vartheta_{W,A,B} = \left[\mathrm{id}_{A \triangleright_W B}, \iota_2 \circ \vartheta'_{W,A,B}\right]$$

We also want to have a variant of ϑ' and ϑ that works with $\triangleright''$. However we cannot derive such a variant from ϑ' and ϑ. Therefore we explicitly require that there exists a natural transformation ϑ'' with

$$\vartheta''_{W,A,B} : A \triangleright''_W (A \triangleright_W B) \to A \triangleright''_W B$$

such that the diagrams in Figure 4 commute. The coherence conditions expressed by these diagrams are analogous to the coherence conditions of ideal monads, which in turn are the duals of the coherence conditions of ideal comonads expressed by Figure 1.

Now that the existence of ϑ'' is ensured, we do not have to explicitly require the existence of ϑ' anymore. Instead we can derive ϑ' from ϑ'' dually to deriving θ from θ':

$$\vartheta'_{W,A,B} \;:\; A \triangleright'_W (A \triangleright_W B) \to A \triangleright'_W B$$
$$\vartheta'_{W,A,B} = \mathrm{id}_A \times \vartheta''_{W,A,B}$$

The coherence conditions from Figure 4 ensure that every pair $\left(A \triangleright'_W -, \vartheta'_{W,A,-}\right)$ is an ideal monad.

3.4 Coherence between Expansion and Joining

Let p be a value of a type $\tau_1 \triangleright''_W (\tau_1 \triangleright_W \tau_2)$, and let p' be the process that results from applying process joining to p. If p does not terminate, p and p' have identical structure. Otherwise the structure of p and p' is almost identical; the only difference is that p' contains a single continuous part, while p contains two adjacent continuous parts: the one that belongs to p itself, and the one that belongs to the process that the terminal event of p carries.

We can define an alternative expansion operation for values of types $\tau_1 \triangleright''_W (\tau_1 \triangleright_W \tau_2)$ that works like process expansion, except that every split between two continuous parts in an argument carries over to the result, including processes that the result contains. This alternative expansion operation is modeled by a natural transformation $\hat{\theta}''$ that is defined as follows:

$$\hat{\theta}''_{W,A,B} \;:\; A \triangleright''_W \left(A \triangleright_W B\right)$$
$$\to \left(A \triangleright'_W \left(A \triangleright_W B\right)\right) \triangleright''_W \left(\left(A \triangleright'_W B\right) \triangleright_W B\right)$$
$$\hat{\theta}''_{W,A,B} = \left(\mathrm{id} \triangleright''_{\mathrm{id}} \theta_{W,A,B}\right) \circ \theta''_{W,A,A \triangleright_W B}$$

We can furthermore define an alternative joining operation that is analogous to the alternative expansion operation we have just discussed. This joining operation is modeled by a natural transformation $\hat{\vartheta}''$ that is defined as follows:

$$\hat{\vartheta}''_{W,A,B} \;:\; \left(A \triangleright'_W \left(A \triangleright_W B\right)\right) \triangleright''_W \left(\left(A \triangleright'_W B\right) \triangleright_W B\right)$$
$$\to \left(A \triangleright'_W B\right) \triangleright''_W B$$
$$\hat{\vartheta}''_{W,A,B} = \vartheta''_{W,A \triangleright'_W B,B} \circ \left(\vartheta'_{W,A,B} \triangleright''_{\mathrm{id}} \mathrm{id}\right)$$

If we perform alternative expansion followed by alternative joining, we should get the same result as when we perform process joining followed by process expansion. Therefore we require that $\hat{\vartheta}'' \circ \hat{\theta}'' = \theta'' \circ \vartheta''$, or equivalently that the diagram in Figure 5 commutes.

3.5 Process Merging

In Subsection 2.3, we saw how we can merge two events. Since processes generalize events, it is reasonable to ask for a generalization of event merging that works with processes. In this subsection, we present such a process merging operation and show how we model it in APCs. Compared to event merging, process merging additionally deals with two things: the presence of continuous parts and the possibility of nontermination.

Let p_1 and p_2 be two processes, and let p' be the process that results from merging them. For every $i \in \{1,2\}$, let t_i be the termination time of p_i or ∞ if p_i does not terminate. If both p_1 and p_2 are nonterminating, p' does not terminate as well. Otherwise p' terminates at $\min(t_1,t_2)$. In this case, its terminal event carries a value $\iota_1(s)$, $\iota_2(s)$, or $\iota_3(s)$ depending on whether $t_1 = t_2$, $t_1 < t_2$, or $t_1 > t_2$. In the first case, s is (x_1,x_2) where x_i is the value carried by the terminal event of p_i. In the second case, s is $\left(x_1, p'_2\right)$ where p'_2 is the suffix of p_2 that starts at t_1. The third case is analogous to the second one. The continuous part h' of p' is formed from the continuous parts h_1 and h_2 of p_1 and p_2 by pointwise pairing, which means that $h'(t) = (h_1(t), h_2(t))$.

We model process merging by a natural transformation ω'' that consists of morphisms $\omega''_{(W_1,A_1,B_1),(W_2,A_2,B_2)}$ whose types are of the form

$$A_1 \triangleright''_{W_1} B_1 \times A_2 \triangleright''_{W_2} B_2 \to A \triangleright''_W B \quad .$$

It is immediate that A must be $A_1 \times A_2$, and B must be

$$B_1 \times B_2 + B_1 \times A_2 \triangleright'_{W_2} B_2 + A_1 \triangleright'_{W_1} B_1 \times B_2 \quad .$$

A process that results from merging two processes p_1 and p_2 is guaranteed to terminate if at least one of the processes p_1 and p_2 is guaranteed to terminate. Therefore we choose W to be the minimum of W_1 and W_2. A minimum in a totally ordered set is a product in the corresponding category, so $W = W_1 \times W_2$.

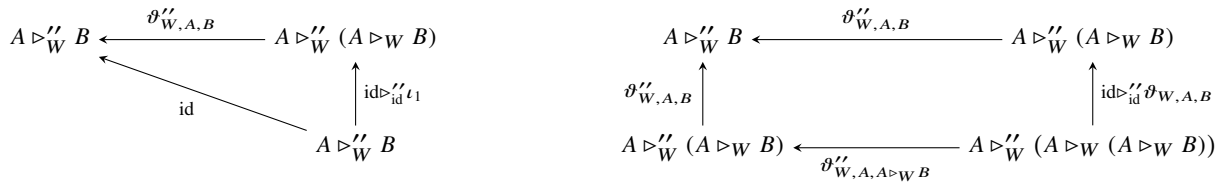

Figure 4. Coherence of process joining

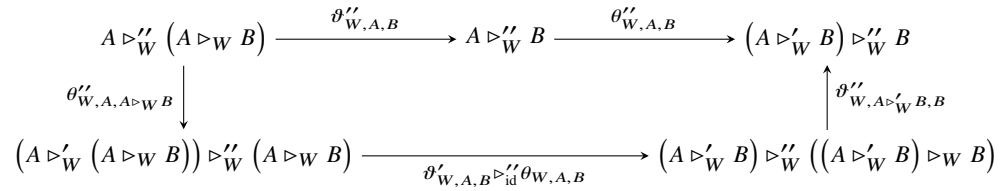

Figure 5. Coherence between process expansion and joining

We define a functor $\circledast : \mathcal{Q} \times \mathcal{Q} \to \mathcal{Q}$ such that $(W_1, A_1, B_1) \circledast (W_2, A_2, B_2)$ is (W, A, B) with W, A, and B being defined as above. Then we have

$$\omega''_{Q_1,Q_2} : \triangleright'' Q_1 \times \triangleright'' Q_2 \to \triangleright''(Q_1 \circledast Q_2)$$

for all $Q_1, Q_2 \in \mathrm{Ob}(\mathcal{Q})$.

A process p' that results from merging two processes p_1 and p_2 conveys the same information as the pair (p_1, p_2). This is analogous to event merging. As a result, ω'' must be an isomorphism. The operation that recovers a process p_i from p' is modeled by the natural transformation χ''_i whose definition is similar to the definition of ζ_i from Subsection 2.3:

$$\chi''_i : \triangleright''(Q_1 \circledast Q_2) \to \triangleright'' Q_i$$
$$\chi''_i = \vartheta''_{Q_i} \circ \left(\pi_i \triangleright''_{\pi_i} [\iota_1 \circ \pi_i, \iota_i \circ \pi_i, \iota_{3-i} \circ \pi_i]\right)$$

Now $\left\langle \chi''_1, \chi''_2 \right\rangle$ must be the inverse of ω''. We do not directly require ω'' to exist, but require $\left\langle \chi''_1, \chi''_2 \right\rangle$ to be an isomorphism.

For every $i \in \{1, 2\}$, we can define natural transformations χ'_i and χ_i with

$$\chi'_i : \triangleright'(Q_1 \circledast Q_2) \to \triangleright' Q_i$$
$$\chi_i : \triangleright(Q_1 \circledast Q_2) \to \triangleright Q_i$$

by taking the definition of χ''_i and replacing ϑ'' and $\triangleright''$ with ϑ' and $\triangleright'$, or ϑ and $\triangleright$, respectively. The reader is invited to check that $\left\langle \chi''_1, \chi''_2 \right\rangle$ being an isomorphism implies that $\left\langle \chi'_1, \chi'_2 \right\rangle$ and $\langle \chi_1, \chi_2 \rangle$ are isomorphisms. The inverses of $\left\langle \chi'_1, \chi'_2 \right\rangle$ and $\langle \chi_1, \chi_2 \rangle$ model process merging operations that work with $\triangleright'$ and $\triangleright$, respectively.

3.6 The Canonical Nonterminating Process

It turns out that there is a nullary version of process merging. In this subsection, we present this "nullary process merging operation" and show how we model it in APCs.

The nullary operation works analogously to binary process merging. We have the following situation:

- In the binary case, we construct a process from a pair of processes. In the nullary case, we construct a process from a 0-tuple of processes, that is, from the sole value tt of type 1.
- In the binary case, the values of the continuous part of the constructed process are pairs. In the nullary case, the continuous part of the constructed process has the value tt at every time.
- In the binary case, the constructed process terminates as soon as one of the original processes terminates. In the nullary case, there is no original process that could terminate. Therefore the constructed process never terminates.

So "nullary process merging" turns tt into a process with a continuous part that emits tt forever. We call this resulting process the canonical nonterminating process (CNP).

We model the construction of the CNP by a morphism ξ'' whose type is of the form $1 \to A \triangleright''_W B$. It is immediate that W must be 1, and A must be 1. We choose B to be 0. This guarantees that the CNP is in fact nonterminating, as there is no value of type 0, which could be carried by a terminal event. We define $N \in \mathrm{Ob}(\mathcal{Q})$ to be $(1, 1, 0)$. Then we have $\xi'' : 1 \to \triangleright'' N$.

The CNP conveys the same information as the value tt it is constructed from, namely no information. This is because there is no other process that has the same type as the CNP. As a result, ξ'' must be an isomorphism, and $!_{\triangleright'' N}$ must be its inverse. This is analogous to the binary case, since ! is the nullary version of $\langle -, - \rangle$. We do not directly require ξ'' to exist, but require $!_{\triangleright'' N}$ to be an isomorphism.

The reader is invited to check that $!_{\triangleright'' N}$ being an isomorphism implies that $!_{\triangleright' N}$ and $!_{\triangleright N}$ are isomorphisms. The inverses of $!_{\triangleright' N}$ and $!_{\triangleright N}$ model operations whose results are variants of the CNP that are inhabitants of $\triangleright'$ and $\triangleright$, respectively.

While process merging generalizes event merging, there is no event counterpart of the CNP. The reason is that the CNP is nonterminating, while events are always guaranteed to occur.

3.7 Generalized Notion of Weakness

So far, we have used the interval category **2** as the category of weaknesses, with object 0 meaning "strong" and object 1 meaning "weak." However the APC constructs we have developed do not require the category of weaknesses to be **2**. Merely the structure for modeling process merging requires binary products to exist, and the structure for modeling CNP construction requires a terminal object to exist. Therefore we allow any category $\mathcal{W}$ with finite products to serve as a category of weaknesses. In particular, we allow 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. Let $(T, \leq)$ be a totally ordered set that denotes the time scale. From $(T, \leq)$, we can derive the totally

ordered set $(T_\infty, \leq_\infty)$ with $T_\infty = T \cup \{\infty\}$ and

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

Now we can use the category of $(T_\infty, \leq_\infty)$ as the category of weaknesses. Thereby any object $t_\mathrm{b} \in T$ stands for termination at or before the time t_b, and the object ∞ stands for the absence of any termination guarantee. Upper bounds on the termination time only apply to processes that do not terminate immediately. This is necessary, because objects $A \rhd_{t_\mathrm{b}} B$ are defined as $B + A \rhd'_{t_\mathrm{b}} B$, so that the types they model always contain the immediately terminating processes $\iota_1(x)$, independently of t_b.

3.8 Summary

We have developed the structure of APCs by extending constructs from temporal categories. The reader might have noticed that we did not explicitly extend the cartesian endofunctor structure and tensorial strength. We will see in Subsection 4.2 that the structure that models process merging also covers tensorial strength, and that the process merging structure together with the structure that models CNP construction covers the cartesian endofunctor structure. So we are ready to formulate the definition of APCs.

Definition 9 (Abstract process category)**.** Say we have the following situation:

- $\mathcal{W}$ is a category with finite products.
- C is a CCC with finite coproducts.
- $\mathcal{Q}$ is the category $\mathcal{W} \times C \times C$.
- $\rhd''$ is a functor from $\mathcal{Q}$ to C.
- The functors $\rhd', \rhd : \mathcal{Q} \to C$ are defined as follows:

$$A \rhd'_W B = A \times A \rhd''_W B \qquad A \rhd_W B = B + A \rhd'_W B$$

- The functor $\circledast : \mathcal{Q} \times \mathcal{Q} \to \mathcal{Q}$ is defined by

$$(W_1, A_1, B_1) \circledast (W_2, A_2, B_2) = (W, A, B)$$

  where W, A, and B are given as follows:

$$\begin{aligned} W &= W_1 \times W_2 \\ A &= A_1 \times A_2 \\ B &= B_1 \times B_2 + B_1 \times A_2 \rhd'_{W_2} B_2 + A_1 \rhd'_{W_1} B_1 \times B_2 \end{aligned}$$

- N is the object $(1, 1, 0)$.
- θ'' and ϑ'' are natural transformations with the following typings:

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

- The natural transformations θ', θ, ϑ', and ϑ are defined as follows:

$$\begin{aligned} \theta'_{W,A,B} &: A \rhd'_W B \to \left(A \rhd'_W B\right) \rhd'_W B \\ \theta'_{W,A,B} &= \left\langle \mathrm{id}_{A \rhd'_W B}, \theta''_{W,A,B} \circ \pi_2 \right\rangle \\ \theta_{W,A,B} &: A \rhd_W B \to \left(A \rhd'_W B\right) \rhd_W B \\ \theta_{W,A,B} &= \mathrm{id}_B + \theta'_{W,A,B} \\ \vartheta'_{W,A,B} &: A \rhd'_W (A \rhd_W B) \to A \rhd'_W B \\ \vartheta'_{W,A,B} &= \mathrm{id}_A \times \vartheta''_{W,A,B} \\ \vartheta_{W,A,B} &: A \rhd_W (A \rhd_W B) \to A \rhd_W B \\ \vartheta_{W,A,B} &= \left[\mathrm{id}_{A \rhd_W B}, \iota_2 \circ \vartheta'_{W,A,B}\right] \end{aligned}$$

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

- For all $W \in \mathrm{Ob}(\mathcal{W})$ and $B \in \mathrm{Ob}(C)$, $\left(- \rhd''_W B, \theta''_{W,-,B}\right)$ is an ideal comonad.
- For all $W \in \mathrm{Ob}(\mathcal{W})$ and $A, B \in \mathrm{Ob}(C)$, the diagrams in Figures 4 and 5 commute.
- The natural transformation $\left\langle \chi''_1, \chi''_2 \right\rangle$ with

$$\begin{aligned} \chi''_i &: \rhd''(Q_1 \circledast Q_2) \to \rhd'' Q_i \\ \chi''_i &= \vartheta''_{Q_i} \circ \left(\pi_i \rhd''_{\pi_i} [\iota_1 \circ \pi_i, \iota_i \circ \pi_i, \iota_{3-i} \circ \pi_i]\right) \end{aligned}$$

  is an isomorphism.
- The morphism $!_{\rhd'' N}$ is an isomorphism.

4. Results about Abstract Process Categories

We now state and prove several results about APCs, one regarding the structure that models process merging and CNP construction (Subsection 4.1), one regarding the relationship between APCs and temporal categories (Subsection 4.2), and one regarding the relationship between APCs and concrete process categories (Subsection 4.3).

4.1 Process Merging and CNP Construction

It turns out that a functor that models a process type, the corresponding natural transformation that models process merging, and the corresponding morphism that models CNP construction form a symmetric monoidal functor. For this statement to make sense, we first have to prove that $(\mathcal{Q}, \circledast, N)$ is a symmetric monoidal category.

Theorem 6. *Let $(\mathcal{W}, C, \rhd'', \theta'', \vartheta'')$ be an APC, and let $\mathcal{Q}$, $\circledast$, and N be defined as in Definition 9. Then $(\mathcal{Q}, \circledast, N)$ is a symmetric monoidal category.*

Proof. We construct the coherence maps of the symmetric monoidal category as follows:

Associator α. Let Q_1, Q_2, and Q_3 be objects of $\mathcal{Q}$. Then we have

$$\alpha_{Q_1,Q_2,Q_3} : (Q_1 \circledast Q_2) \circledast Q_3 \to Q_1 \circledast (Q_2 \circledast Q_3) \quad .$$

Since α_{Q_1,Q_2,Q_3} is a morphism of $\mathcal{Q}$, it is a triple $(\alpha_1, \alpha_2, \alpha_3)$ where $\alpha_1 \in \mathrm{Mor}(\mathcal{W})$ and $\alpha_2, \alpha_3 \in \mathrm{Mor}(C)$.
Let us look at α_1 and α_2 first. For every $i \in \{1, 2, 3\}$, let W_i be the first element of the triple Q_i, and let A_i be the second. Then the typings of α_1 and α_2 are as follows:

$$\begin{aligned} \alpha_1 &: (W_1 \times W_2) \times W_3 \to W_1 \times (W_2 \times W_3) \\ \alpha_2 &: (A_1 \times A_2) \times A_3 \to A_1 \times (A_2 \times A_3) \end{aligned}$$

Let α_c denote the associator of any cartesian structure, which means that

$$\alpha_\mathrm{c} = \langle \pi_1 \circ \pi_1, \langle \pi_2 \circ \pi_1, \pi_2 \rangle \rangle \quad .$$

We set $\alpha_1 = \alpha_\mathrm{c}$ and $\alpha_2 = \alpha_\mathrm{c}$.
Now let us construct α_3. For every $i \in \{1, 2, 3\}$, let B_i be the third element of the triple Q_i, and for every $i, j \in \{1, 2, 3\}$, let

$$C_{i,j} = B_i \times B_j + B_i \times \rhd' Q_j + \rhd' Q_i \times B_j \quad .$$

Then the typing of α_3 is as follows:

$$\begin{aligned} \alpha_3 : C_{1,2} \times B_3 + C_{1,2} \times \rhd' Q_3 + \rhd'(Q_1 \circledast Q_2) \times B_3 \\ \to B_1 \times C_{2,3} + B_1 \times \rhd'(Q_2 \circledast Q_3) + \rhd' Q_1 \times C_{2,3} \end{aligned}$$

Let κ and $\varkappa$ be defined by the following equations:[4]

$$\begin{aligned} \kappa &= [\iota_1 \times \mathrm{id}, \iota_2 \times \mathrm{id}, \iota_3 \times \mathrm{id}] \\ \varkappa &= [\mathrm{id} \times \iota_1, \mathrm{id} \times \iota_2, \mathrm{id} \times \iota_3] \end{aligned}$$

We construct α_3 by composing isomorphisms as shown in Figure 6.

[4] Both κ and $\varkappa$ are ternary distributivity transformations.

$$
\begin{aligned}
& C_{1,2} \times B_3 + C_{1,2} \times \triangleright' Q_3 + \triangleright'(Q_1 \circledast Q_2) \times B_3 \\
\xrightarrow{\kappa^{-1}+\kappa^{-1}+\langle \chi_1', \chi_2' \rangle \times \mathrm{id}_{B_3}} & ((B_1 \times B_2) \times B_3 + (B_1 \times \triangleright' Q_2) \times B_3 + (\triangleright' Q_1 \times B_2) \times B_3) \\
& \quad + ((B_1 \times B_2) \times \triangleright' Q_3 + (B_1 \times \triangleright' Q_2) \times \triangleright' Q_3 + (\triangleright' Q_1 \times B_2) \times \triangleright' Q_3) \\
& \quad + (\triangleright' Q_1 \times \triangleright' Q_2) \times B_3 \\
\xrightarrow{(\alpha_c+\alpha_c+\alpha_c)+(\alpha_c+\alpha_c+\alpha_c)+\alpha_c} & (B_1 \times (B_2 \times B_3) + B_1 \times (\triangleright' Q_2 \times B_3) + \triangleright' Q_1 \times (B_2 \times B_3)) \\
& \quad + (B_1 \times (B_2 \times \triangleright' Q_3) + B_1 \times (\triangleright' Q_2 \times \triangleright' Q_3) + \triangleright' Q_1 \times (B_2 \times \triangleright' Q_3)) \\
& \quad + \triangleright' Q_1 \times (\triangleright' Q_2 \times B_3) \\
\xrightarrow{[[\iota_1 \circ \iota_1, \iota_1 \circ \iota_3, \iota_3 \circ \iota_1], [\iota_1 \circ \iota_2, \iota_2, \iota_3 \circ \iota_2], \iota_3 \circ \iota_3]} & (B_1 \times (B_2 \times B_3) + B_1 \times (B_2 \times \triangleright' Q_3) + B_1 \times (\triangleright' Q_2 \times B_3)) \\
& \quad + B_1 \times (\triangleright' Q_2 \times \triangleright' Q_3) \\
& \quad + (\triangleright' Q_1 \times (B_2 \times B_3) + \triangleright' Q_1 \times (B_2 \times \triangleright' Q_3) + \triangleright' Q_1 \times (\triangleright' Q_2 \times B_3)) \\
\xrightarrow{\varkappa + \mathrm{id}_{B_1} \times \langle \chi_1', \chi_2' \rangle^{-1} + \varkappa} & B_1 \times C_{2,3} + B_1 \times \triangleright'(Q_2 \circledast Q_3) + \triangleright' Q_1 \times C_{2,3}
\end{aligned}
$$

Figure 6. Third element of the associator of $(Q, \circledast, N)$

$$
\begin{aligned}
& 0 \times B + 0 \times A \triangleright'_W B + 1 \triangleright'_1 0 \times B \\
\xrightarrow{\pi_1+\pi_1+!_{1 \triangleright'_1 0} \times \mathrm{id}_B} & 0 + 0 + 1 \times B \\
\xrightarrow{[?_B, ?_B, \lambda_c]} & B
\end{aligned}
$$

Figure 7. Third element of the left unitor of $(Q, \circledast, N)$

$$
\begin{aligned}
& B_1 \times B_2 + B_1 \times \triangleright' Q_2 + \triangleright' Q_1 \times B_2 \\
\xrightarrow{\gamma_c+\gamma_c+\gamma_c} & B_2 \times B_1 + \triangleright' Q_2 \times B_1 + B_2 \times \triangleright' Q_1 \\
\xrightarrow{[\iota_1, \iota_3, \iota_2]} & B_2 \times B_1 + B_2 \times \triangleright' Q_1 + \triangleright' Q_2 \times B_1
\end{aligned}
$$

Figure 8. Third element of the braiding of $(Q, \circledast, N)$

Left unitor λ. Let W be an object of $\mathcal{W}$, and let A and B be objects of C. Then we have

$$\lambda_{W,A,B} : (1,1,0) \circledast (W,A,B) \to (W,A,B) \quad .$$

The morphism $\lambda_{W,A,B}$ is a triple $(\lambda_1, \lambda_2, \lambda_3)$ where $\lambda_1 \in \mathrm{Mor}(\mathcal{W})$ and $\lambda_2, \lambda_3 \in \mathrm{Mor}(C)$.
The typings of λ_1 and λ_2 are as follows:

$$\lambda_1 : 1 \times W \to W \qquad \lambda_2 : 1 \times A \to A$$

Let λ_c denote the left unitor of any cartesian structure, which means that $\lambda_c = \pi_2$. We set $\lambda_1 = \lambda_c$ and $\lambda_2 = \lambda_c$.
The typing of λ_3 is as follows:

$$\lambda_3 : 0 \times B + 0 \times A \triangleright'_W B + 1 \triangleright'_1 0 \times B \to B$$

We construct λ_3 by composing isomorphisms as shown in Figure 7.[5]

Right unitor ρ. The right unitor can be constructed analogously to the left unitor. It can equivalently be defined as $\lambda \circ \gamma$, using the braiding γ discussed below.

Braiding γ. Let Q_1 and Q_2 be objects of Q. Then we have

$$\gamma_{Q_1,Q_2} : Q_1 \circledast Q_2 \to Q_2 \circledast Q_1 \quad .$$

The morphism γ_{Q_1,Q_2} is a triple $(\gamma_1, \gamma_2, \gamma_3)$.
For every $i \in \{1,2\}$, let W_i be the first element of Q_i, and let A_i be the second. Then the typings of γ_1 and γ_2 are as follows:

$$\gamma_1 : W_1 \times W_2 \to W_2 \times W_1$$
$$\gamma_2 : A_1 \times A_2 \to A_2 \times A_1$$

Let γ_c denote the braiding of any cartesian structure, which means that $\gamma_c = \langle \pi_2, \pi_1 \rangle$. We set $\gamma_1 = \gamma_c$ and $\gamma_2 = \gamma_c$.

[5] Thereby we write $?_C$ for the unique morphism from 0 to C.

For every $i \in \{1,2\}$, let B_i be the third element of the triple Q_i. Then the typing of γ_3 is as follows:

$$\gamma_3 : B_1 \times B_2 + B_1 \times \triangleright' Q_2 + \triangleright' Q_1 \times B_2 \to B_2 \times B_1 + B_2 \times \triangleright' Q_1 + \triangleright' Q_2 \times B_1$$

We construct γ_3 by composing isomorphisms as shown in Figure 8.

Note that we construct the coherence maps from isomorphisms in such a way that the property of being an isomorphism carries over to the coherence maps themselves. We leave it to the reader to check that the coherence maps fulfill the necessary equations. □

We are now ready to state the actual result of this subsection for the functor $\triangleright''$.

Theorem 7. *Let $(\mathcal{W}, C, \triangleright'', \theta'', \vartheta'')$ be an APC, and let Q, $\circledast$, N, χ_1'', and χ_2'' be defined as in Definition 9. Then the triple $\left(\triangleright'', \left\langle \chi_1'', \chi_2'' \right\rangle^{-1}, !^{-1}_{\triangleright'' N}\right)$ is a symmetric monoidal functor from $(Q, \circledast, N)$ to $(C, \times, 1)$.*

Proof. The coherence maps $\left\langle \chi_1'', \chi_2'' \right\rangle^{-1}$ and $!^{-1}_{\triangleright'' N}$ are isomorphisms, since they are inverses of isomorphisms. It only remains to show that the coherence maps fulfill the equations of a symmetric monoidal functor. Since this is a mostly mechanical task, we leave this out here. □

The statement of Theorem 7 holds analogously for the functors $\triangleright'$ and $\triangleright$. We just state this proposition as a theorem and do not give a proof, since this theorem can be proved fairly easily by using Theorem 7.

Theorem 8. *Let $(\mathcal{W}, C, \triangleright'', \theta'', \vartheta'')$ be an APC, and let Q, $\circledast$, N, χ_i', and χ_i be defined as in Definition 9. Then the triples*

$\left(\triangleright', \left\langle \chi_1', \chi_2' \right\rangle^{-1}, !_{\triangleright' N}^{-1}\right)$ *and* $\left(\triangleright, \langle \chi_1, \chi_2 \rangle^{-1}, !_{\triangleright N}^{-1}\right)$ *are symmetric monoidal functors from* $(\mathcal{Q}, \circledast, N)$ *to* $(C, \times, 1)$.

4.2 Relationship to Temporal Categories

In this subsection, we prove that an APC gives rise to a temporal category in a canonical way. This shows that the structure of an APC is indeed an extension of the temporal category structure.

Theorem 9. *Let* $(\mathcal{W}, C, \triangleright'', \theta'', \vartheta'')$ *be an APC, let* W *be an object of* $\mathcal{W}$*, and let* $\Box'$ *and* $\Diamond'$ *be defined as follows:*

$$\Box' A = A \triangleright_1'' 0 \qquad\qquad \Diamond' B = 1 \triangleright_W' B$$

Then there exist natural transformations φ'*,* ψ'*,* δ'*,* σ'*, and* μ' *such that* $(C, \Box', \varphi', \psi', \delta', \Diamond', \sigma', \mu')$ *is a temporal category.*

Proof. We construct the required natural transformations as shown in Figure 9. The check that $(C, \Box', \varphi', \psi', \delta', \Diamond', \sigma', \mu')$ is indeed a temporal category is left to the reader. □

So APCs are an extension of temporal categories. However we conjecture that they are not a conservative extension. Typical FRP systems provide a switching operator that turns a pair (b, e) of a type $\Box'\tau \times \Diamond'\Box\tau$ into a behavior b' of type $\Box'\tau$. The behavior b' first behaves like b, but as soon as e occurs, b' continues to behave like the behavior that e carries. We suppose that in general, temporal categories do not contain a natural transformation that models this switching operator. However APCs do contain such a natural transformation, namely the transformation υ defined in Figure 10.

4.3 Relationship to 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 us (Jeltsch 2013, Section 3). Here we only give a short introduction to CPCs and then prove that APCs generalize CPCs.

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 10 (Temporal index category)**.** The temporal index category of a totally ordered set $(T, \leq)$ is the category $\mathcal{I}$ with

$$\mathrm{Ob}(\mathcal{I}) = \{\, (t, t_\mathrm{o}) \in T \times T \mid t \leq t_\mathrm{o} \,\}$$

and

$$\hom_{\mathcal{I}}\left((t', t_\mathrm{o}'), (t, t_\mathrm{o})\right) = \begin{cases} \{(t, t_\mathrm{o}, t_\mathrm{o}')\} & \text{if } t = t' \text{ and } t_\mathrm{o} \leq t_\mathrm{o}' \\ \emptyset & \text{otherwise} \end{cases}.$$

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 $A(t, t_\mathrm{o})$ 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 $(t, t_\mathrm{o}, t_\mathrm{o}')$ of $\mathcal{B}^{\mathcal{I}}$, the morphism $A\,(t, t_\mathrm{o}, t_\mathrm{o}')$ models a 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 11 (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 $\mathcal{B}^{\mathcal{I}}$.

Our earlier work (Jeltsch 2013) defines the following temporal functors for each CPC:

Weak basic temporal functor, for modeling types of nonterminating processes.

Strong basic temporal functor, for modeling types of terminating processes.

Bounding temporal functor, for modeling types of terminating processes with upper bounds on termination times.

We have shown (Jeltsch 2013, Subsection 3.7) that the strong basic temporal functor of a CPC with an infinite time scale cannot model the termination requirement and is actually identical to the weak basic temporal functor. Therefore we do not consider it here. Instead we introduce the basic temporal functor of a CPC, which combines the weak basic temporal functor and the bounding temporal functor.

The basic temporal functor is a functor

$$\triangleright'' : \mathcal{W} \times \mathcal{B}^{\mathcal{I}} \times \mathcal{B}^{\mathcal{I}} \to \mathcal{B}^{\mathcal{I}}$$

where $\mathcal{W}$ is the category of the totally ordered set $(T_\infty, \leq_\infty)$ that is defined as described in Subsection 3.7. We can completely define $\triangleright''$ by fixing the following constructions:

1. For all $t_\mathrm{b} \in \mathrm{Ob}(\mathcal{W})$, $A, B \in \mathrm{Ob}\left(\mathcal{B}^{\mathcal{I}}\right)$, and $(t, t_\mathrm{o}) \in \mathrm{Ob}(\mathcal{I})$, the object
$$\left(A \triangleright_{t_\mathrm{b}}'' B\right)(t, t_\mathrm{o}) \quad .$$
Such an object models a type whose inhabitants denote information about processes.
2. For all $t_\mathrm{b} \in \mathrm{Ob}(\mathcal{W})$, $A, B \in \mathrm{Ob}\left(\mathcal{B}^{\mathcal{I}}\right)$, and $(t, t_\mathrm{o}, t_\mathrm{o}') \in \mathrm{Mor}(\mathcal{I})$, the morphism
$$\left(A \triangleright_{t_\mathrm{b}}'' B\right)(t, t_\mathrm{o}, t_\mathrm{o}') \quad .$$
Such a morphism models an information disposal function.
3. For all $t_\mathrm{b} \in \mathrm{Ob}(\mathcal{W})$, $f, g \in \mathrm{Mor}\left(\mathcal{B}^{\mathcal{I}}\right)$, and $(t, t_\mathrm{o}) \in \mathrm{Ob}(\mathcal{I})$, the morphism
$$\left(f \triangleright''_{(t_\mathrm{b}, t_\mathrm{b})} g\right)_{(t, t_\mathrm{o})} \quad .$$
Such a morphism models a function that transforms the values of continuous parts and the values that terminal events carry.
4. For all $\left(t_\mathrm{b}, t_\mathrm{b}'\right) \in \mathrm{Mor}(\mathcal{W})$, $A, B \in \mathrm{Ob}\left(\mathcal{B}^{\mathcal{I}}\right)$, and $(t, t_\mathrm{o}) \in \mathrm{Ob}(\mathcal{I})$, the morphism
$$\left(\mathrm{id}_A \triangleright''_{\left(t_\mathrm{b}, t_\mathrm{b}'\right)} \mathrm{id}_B\right)_{(t, t_\mathrm{o})} \quad .$$
Such a morphism models a function that converts between process types by relaxing the termination constraint.

We define the objects from 1. in Figure 11 and the morphisms from 4. in Figure 12. We do not show how to define the morphisms from 2. and 3., since their definitions are straightforward.

Definition 12 (Basic temporal functor of a CPC)**.** The basic temporal functor $\triangleright''$ of a CPC $(T, \leq, \mathcal{B})$ is defined as described above.

We are now ready to state and prove the theorem about the relationship between APCs and CPCs.

Theorem 10. *Let* $(T, \leq, \mathcal{B})$ *be a CPC, let* $\mathcal{I}$ *be the temporal index category of* $(T, \leq)$*, let* $\triangleright''$ *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 3.7. Then there exist natural*

$$\varphi'_{A_1,A_2} \;:\; A_1 \rhd''_1 0 \times A_2 \rhd''_1 0 \to (A_1 \times A_2) \rhd''_1 0 \qquad \delta'_A \;:\; A \rhd''_1 0 \to \left(A \rhd'_1 0\right) \rhd''_1 0$$
$$\varphi'_{A_1,A_2} = \left(\mathrm{id}_{A_1\times A_2} \rhd''_{!_{1\times 1}} [\pi_1,\pi_1,\pi_2]\right) \circ \left\langle \chi''_1, \chi''_2 \right\rangle^{-1} \qquad \delta'_A = \theta''_{1,A,0}$$
$$\psi' \;:\; 1 \to 1 \rhd''_1 0 \qquad \mu'_B \;:\; 1 \rhd'_W (1 \rhd_W B) \to 1 \rhd'_W B$$
$$\psi' = !_{\rhd'' N} \qquad \mu'_B = \vartheta'_{W,1,B}$$
$$\sigma'_{A,B} \;:\; A \rhd''_1 0 \times 1 \rhd'_W B \to 1 \rhd'_W (A \times B)$$
$$\sigma'_{A,B} = \left\langle !_{1 \rhd''_W (A\times B)}, \mathrm{id}_{1 \rhd''_W (A\times B)} \right\rangle \circ \left(\pi_2 \rhd''_{\pi_2} [?_{A\times B} \circ \pi_1, ?_{A\times B} \circ \pi_1, \pi_1 \times \mathrm{id}_B]\right) \circ \left\langle \chi''_1, \chi''_2 \right\rangle^{-1} \circ \left(\mathrm{id}_{A \rhd''_1 0} \times \pi_2\right)$$

Figure 9. Natural transformations of a temporal category that is induced by an APC

$$\upsilon \;:\; A \rhd''_1 0 \times 1 \rhd'_W \left(A \rhd'_1 0\right) \to A \rhd''_1 0$$
$$\upsilon = \vartheta''_{1,A,0} \circ \left(\pi_1 \rhd''_{\pi_1} \left[?_{A \rhd'_1 0} \circ \pi_1, ?_{A \rhd'_1 0} \circ \pi_1, \pi_2\right]\right) \circ \left\langle \chi''_1, \chi''_2 \right\rangle^{-1} \circ \left(\mathrm{id}_{A \rhd''_1 0} \times \pi_2\right)$$

Figure 10. Meaning of the switching operator

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

Figure 11. Semantics of process information types

$$\left(\mathrm{id}_A \rhd''_{(t_\mathrm{b},t'_\mathrm{b})} \mathrm{id}_B\right)_{(t,t_\mathrm{o})} = \begin{cases} ? & \text{if } t_\mathrm{b} < t \\ [\iota_{t'}]_{t' \in (t,t_\mathrm{b}]} & \text{if } t \le t_\mathrm{b} \le t'_\mathrm{b} \le t_\mathrm{o} \\ \iota_1 \circ [\iota_{t'}]_{t' \in (t,t_\mathrm{b}]} & \text{if } t \le t_\mathrm{b} \le t_\mathrm{o} <_\infty t'_\mathrm{b} \\ \mathrm{id} & \text{if } t_\mathrm{o} <_\infty t_\mathrm{b} \le_\infty t'_\mathrm{b} \end{cases}$$

Figure 12. Semantics of process type conversion

transformations θ'' and ϑ'' such that $\left(\mathcal{W}, \mathcal{B}^{\mathcal{I}}, \rhd'', \theta'', \vartheta''\right)$ is an APC.

Proof. Let ς^* denote the inverse of the canonical distributivity transformation for coproducts of arbitary arity:

$$\varsigma^*_{A,\{B_i\}_{i\in I}} \;:\; A \times \coprod_{i\in I} B_i \to \coprod_{i\in I} (A \times B_i)$$
$$\varsigma^*_{A,\{B_i\}_{i\in I}} = [\mathrm{id}_A \times \iota_i]^{-1}_{i\in I}$$

We construct θ'' as shown in Figure 13, and ϑ'' as shown in Figure 14. The check that $\left(\mathcal{W}, \mathcal{B}^{\mathcal{I}}, \rhd'', \theta'', \vartheta''\right)$ is indeed a temporal category is left to the reader. □

5. Related Work

FRP with processes corresponds to an intuitionistic linear-time temporal logic with "until" operators. We can build this logic by starting with the intuitionistic modal logic IK (Bellin et al. 2001) and successively adding logical operators and axioms. Possible intermediate logics are intuitionistic S4 and intuitionistic linear-time temporal logic with the temporal operators $\Box'$ and $\Diamond'$.

APCs can be built in an analogous way. We start with the categorical semantics of IK given by Bellin et al. (2001). By adding more structure, we can get from there to categorical semantics of different intuitionistic S4 variants. These include the semantics by Kobayashi (1997) and the semantics by Bierman and de Paiva (2000) as well as intuitionistic S4 categories (Jeltsch 2012, Section 4). Subsequent additions to intuitionistic S4 categories lead to temporal categories (Jeltsch 2012, Sections 5 and 6) and from there to APCs, as we showed in Theorem 9.

We do not know of any purely axiomatically defined categorical FRP semantics besides temporal categories and APCs. However there are several categorical FRP semantics that are based on concrete constructions. These include the semantics based on 1-bounded ultrametric spaces by Krishnaswami and Benton (2011), the categories **RSet**, ⊟**RSet**, and ⊵**RSet** by Jeffrey (2012, Section 2), and CPCs (Jeltsch 2013, Section 3).

The semantics by Krishnaswami and Benton ensure that morphisms only model causal functions, which is also a key property of CPCs. However they require the time scale to be discrete and model only behaviors, not events or even processes.

Jeffrey's semantics use an FRP approach that is in the tradition of Yampa (Hudak et al. 2003) in that it uses signal functions as the core construct of FRP. A signal is a time-varying value that spans the whole time scale, and a signal function is a function that turns one source signal into one target signal. Jeffrey interprets morphisms in **RSet** as signal functions that work pointwise. The morphisms of ⊟**RSet** model causal signal functions, and the morphisms of ⊵**RSet** model causal signal functions which produce output values based only on a finite history.

6. Conclusions and Further Work

We have developed abstract process categories (APCs), which are axiomatically defined categorical models of FRP with processes. We have shown that we can canonically construct temporal cate-

$$\left(\theta''_{t_\mathrm{b},A,B}\right)_{(t,t_\mathrm{o})} : \left(A \rhd''_{t_\mathrm{b}} B\right)(t,t_\mathrm{o}) \to \left(\left(A \rhd'_{t_\mathrm{b}} B\right) \rhd''_{t_\mathrm{b}} B\right)(t,t_\mathrm{o})$$

$$\left(\theta''_{t_\mathrm{b},A,B}\right)_{(t,t_\mathrm{o})} = \begin{cases} \mathrm{id}_0 & \text{if } t_\mathrm{b} < t \\ \phi_{t_\mathrm{b}} & \text{if } t \le t_\mathrm{b} \le t_\mathrm{o} \\ \phi_{t_\mathrm{o}} + \left\langle \left\langle \pi_{t'}, \iota_2 \circ \langle \pi_{t''} \rangle_{t'' \in (t',t_\mathrm{o}]} \right\rangle \right\rangle_{t' \in (t,t_\mathrm{o}]} & \text{if } t_\mathrm{o} <_\infty t_\mathrm{b} \end{cases}$$

$$\phi_{t^*} : \coprod_{t' \in (t,t^*]} \left(\left(\prod_{t'' \in (t,t')} A(t'',t_\mathrm{o}) \right) \times B(t',t_\mathrm{o}) \right)$$
$$\to \coprod_{t' \in (t,t^*]} \left(\left(\prod_{t'' \in (t,t')} \left(A(t'',t_\mathrm{o}) \times \coprod_{t' \in (t'',t^*]} \left(\left(\prod_{t''' \in (t'',t')} A(t''',t_\mathrm{o}) \right) \times B(t',t_\mathrm{o}) \right) \right) \right) \times B(t',t_\mathrm{o}) \right)$$

$$\phi_{t^*} = \coprod_{t' \in (t,t^*]} \left\langle \left\langle \pi_{t''} \circ \pi_1, \iota_{t'} \circ \left(\langle \pi_{t'''} \rangle_{t''' \in (t'',t')} \times \mathrm{id}_{B(t',t_\mathrm{o})} \right) \right\rangle \right\rangle_{t'' \in (t,t')}, \pi_2 \Bigg\rangle$$

Figure 13. Meaning of process expansion that is induced by a CPC

gories (Jeltsch 2012) from APCs and APCs from concrete process categories (Jeltsch 2013).

In the future, we want to extend APCs such that they cover more computation concepts. In particular, we are interested in recursion and corecursion on processes, and in the integration of FRP and mutable state. Another goal is to implement FRP in mainstream functional programming languages such that the interface directly reflects the abstract categorical semantics.

A. Changes in the Temporal Category Definition

Our original definition of temporal categories (Jeltsch 2012) has some deficiencies, which are fixed in the definition given in Section 2. In the following, we discuss the differences between the original and the current definition.

A.1 Ideal Comonads and Ideal Monads

Our original definition of temporal categories uses a different definition of ideal comonads (Jeltsch 2012, Definition 5.1). According to this definition, a pair (U', δ') with $U' : C \to C$ and $\delta' : U' \to U'U$ is an ideal comonad if and only if (U, ε, δ) is a comonad where U, ε, and δ are defined like in Definition 2. So the property from Corollary 2 is used as the defining property of ideal comonads. With the original definition, the diagrams in Figure 1 are not guaranteed to commute. They commute however if $\pi_2 : U \to U'$ is an epimorphism.

The original definition of temporal categories furthermore uses a definition of ideal monads that is dual to the abovementioned definition of ideal comonads (Jeltsch 2012, Definition 5.2). So its notion of ideal monad is generally weaker than the notion used in this paper. However it is equivalent to the ideal monad notion of Ghani and Uustalu (2004, Definition 3.1).

There is a third definition of ideal monads, used by Milius (2003, Definition 2.10). This definition is equivalent to our original definition plus the additional requirement that (T', ι_2) is a subfunctor of T where T' is the endofunctor of the ideal monad, and T is $\mathrm{Id} + T'$. This additional requirement implies that $\iota_2 : T' \to T$ is a monomorphism. So ideal monads according to Milius are also ideal monads according to our current definition, but Milius requires an additional property for the injection ι_2, which we do not demand.

A.2 Strong Ideal Monads

Our original definition of temporal categories uses a definition of strong ideal monads (Jeltsch 2012, Definition 5.5) that differs from Definition 7 in this paper in several ways:

1. The original definition specifies the properties of the tensorial strength σ' only indirectly, analogously to the original definition of ideal comonads and ideal monads. It derives a natural transformation σ from σ' and requires that σ and the monad derived from the ideal monad together form a strong monad. This does not guarantee in general that σ' has all the properties that our current definition guarantees. On the other hand, the current definition implies that σ has all the properties that the original definition requires.
2. The original definition does not use the notions of strong functor and strong transformation. Instead it incorporates the properties of strong functors and strong transformations into the definition of strong monads, which makes the structure of strong monads and strong ideal monads less clear.
3. The original definition uses a tensorial strength τ' with
$$\tau'_{A,B} : U'A \times T'B \to T'(UA \times B)$$
where U', U, and T' are defined as in Subsection A.1. If appropriate properties are required for such a tensorial strength, this approach is equivalent to using a tensorial strength σ' with
$$\sigma'_{A,B} : U'A \times T'B \to T'(A \times B) \ ,$$
as we do in this paper. We can derive τ' from σ' and σ' from τ' as follows:
$$\tau' = \sigma' \circ (\delta' \times \mathrm{id}) \qquad \sigma' = T'(\pi_1 \times \mathrm{id}) \circ \tau'$$
An analogous result holds in the non-ideal case (Cockett and Seely 1999, Lemma 11).

   The current approach of using σ' has several advantages over the original approach of using τ':

   - With the original approach, strong functors must be endofunctors, because T' and U are nested in the codomain of τ'. The current approach does not have this restriction.[6]
   - In the original definition, we have to explicitly require that the tensorial strength of a strong monad and the comultiplication of the corresponding comonad cohere (Jeltsch 2012, Figure 2). This is not the case with the current approach.
   - The current approach corresponds to the notion of tensorial strength used in categorical models of IK by Bellin et al. (2001, Definition 5). In the context of these models, the two

[6] Of course, a strong functor must be an endofunctor if it is the functor of an ideal monad.

$$\left(\vartheta''_{t_\mathrm{b},A,B}\right)_{(t,t_\mathrm{o})} \;:\; \left(A \rhd''_{t_\mathrm{b}} \left(A \rhd_{t_\mathrm{b}} B\right)\right)(t,t_\mathrm{o}) \to \left(A \rhd''_{t_\mathrm{b}} B\right)(t,t_\mathrm{o})$$

$$\left(\vartheta''_{t_\mathrm{b},A,B}\right)_{(t,t_\mathrm{o})} = \begin{cases} \mathrm{id}_0 & \text{if } t_\mathrm{b} < t \\ \left[\left[\iota_{t'}, [\iota_{t''} \circ \nu]_{t'' \in (t', t_\mathrm{b}]} \circ \varsigma^* \circ \alpha_\mathrm{c}^{-1}\right] \circ \varsigma\right]_{t' \in (t, t_\mathrm{b})} & \text{if } t \le t_\mathrm{b} \le t_\mathrm{o} \\ \left[\left[\left[\iota_1 \circ \iota_{t'}, \left[[\iota_1 \circ \iota_{t''} \circ \nu]_{t'' \in (t', t_\mathrm{o}]} \circ \varsigma^*, \iota_2 \circ \left\langle \beta_{t'', (t, t_\mathrm{o})} \right\rangle_{t'' \in (t, t_\mathrm{o}]}\right] \circ \varsigma \circ \alpha_\mathrm{c}^{-1}\right] \circ \varsigma\right]_{t' \in (t, t_\mathrm{o})}, \iota_2\right] & \text{if } t_\mathrm{o} <_\infty t_\mathrm{b} \end{cases}$$

$$\nu \;:\; \left(\left(\prod_{t''' \in (t,t')} A(t''', t_\mathrm{o})\right) \times A(t', t_\mathrm{o})\right) \times \left(\left(\prod_{t''' \in (t', t'')} A(t''', t_\mathrm{o})\right) \times B(t'', t_\mathrm{o})\right) \to \left(\prod_{t''' \in (t, t'')} A(t''', t_\mathrm{o})\right) \times B(t'', t_\mathrm{o})$$

$$\nu = \left(\left\langle \beta_{t''', (t,t'')} \right\rangle_{t''' \in (t,t'')} \times \mathrm{id}_{B(t'', t_\mathrm{o})}\right) \circ \alpha_\mathrm{c}^{-1}$$

$$\beta_{t^\dagger, I} \;:\; \left(\left(\prod_{t^\dagger \in (t,t')} A(t'', t_\mathrm{o})\right) \times A(t', t_\mathrm{o})\right) \times \prod_{t^\dagger \in (t', \infty) \cap I} A(t'', t_\mathrm{o}) \to A(t^\dagger, t_\mathrm{o})$$

$$\beta_{t^\dagger, I} = \begin{cases} \pi_{t^\dagger} \circ \pi_1 \circ \pi_1 & \text{if } t'' < t' \\ \pi_2 \circ \pi_1 & \text{if } t^\dagger = t' \\ \pi_{t^\dagger} \circ \pi_2 & \text{if } t'' > t' \end{cases}$$

Figure 14. Meaning of process joining that is induced by a CPC

approaches to defining tensorial strength are not equivalent, because the endofunctor that models the $\Box$-modality lacks a comonad structure.

Improvements 2. and 3. were suggested to the author by Tarmo Uustalu, who also pointed out some of their advantages.

A.3 Categorical Structure for Event Merging

The original definition of temporal categories has a different way of modeling event merging. It requires that each object $A \odot B$ is a product of A and B in the Kleisli category of $(\diamond, \eta, \mu)$ with projections that correspond to the following morphisms ϖ_i in the original category C:

$$\varpi_i \;:\; C_1 \odot C_2 \to \diamond C_i$$
$$\varpi_i = [\iota_1 \circ \pi_i, \iota_i \circ \pi_i, \iota_{3-i} \circ \pi_i]$$

This is equivalent to the requirement that the natural transformation $\langle \zeta_1, \zeta_2 \rangle$ is an isomorphism.[7] However the current definition of temporal categories requires that the natural transformation $\left\langle \zeta'_1, \zeta'_2 \right\rangle$ is an isomorphism, which is generally stronger.

Acknowledgments

I thank Tarmo Uustalu for all the helpful discussions about the topics of this paper, and in particular, for his suggestions on how to improve the temporal category definition.

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

G. Bellin, V. de Paiva, and E. Ritter. Extended Curry–Howard correspondence for a basic constructive modal logic. In *Proceedings of the 2nd Workshop on Methods for Modalities (M4M-2)*, 2001.

G. Bierman and V. de Paiva. On an intuitionistic modal logic. *Studia Logica*, 65(3):383–416, Aug. 2000. ISSN 0039-3215. DOI: 10.1023/A:1005291931660.

J. R. B. Cockett and R. A. G. Seely. Linearly distributive functors. *Journal of Pure and Applied Algebra*, 143(1–3):155–203, Nov. 1999. ISSN 0022-4049. DOI: 10.1016/S0022-4049(98)00110-8.

N. Ghani and T. Uustalu. Coproducts of ideal monads. *RAIRO Theoretical Informatics and Applications*, 38(4):321–342, Oct. 2004. ISSN 0988-3754. DOI: 10.1051/ita:2004016.

P. Hudak, A. Courtney, H. Nilsson, and J. Peterson. Arrows, robots, and functional reactive programming. In J. Jeuring and S. Peyton Jones, editors, *Advanced Functional Programming*, volume 2638 of *Lecture Notes in Computer Science*, pages 159–187. Springer, Berlin/Heidelberg, Germany, 2003. ISBN 978-3-540-40132-2. DOI: 10.1007/978-3-540-44833-4_6.

A. Jeffrey. 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)*, pages 49–60, New York, 2012. ACM. ISBN 978-1-4503-1125-0. DOI: 10.1145/2103776.2103783.

W. Jeltsch. Towards a common categorical semantics for linear-time temporal logic and functional reactive programming. *Electronic Notes in Theoretical Computer Science*, 286:229–242, Sept. 2012. ISSN 1571-0661. DOI: 10.1016/j.entcs.2012.08.015.

W. Jeltsch. 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)*, pages 69–78, New York, 2013. ACM. ISBN 978-1-4503-1860-0. DOI: 10.1145/2428116.2428128.

S. Kobayashi. Monad as modality. *Theoretical Computer Science*, 175(1):29–74, Mar. 1997. ISSN 0304-3975. DOI: 10.1016/S0304-3975(96)00169-7.

N. R. Krishnaswami and N. Benton. Ultrametric semantics of reactive programs. In *Proceedings of the 26th Annual IEEE Symposium on Logic in Computer Science (LICS '11)*, pages 257–266, New York, June 2011. IEEE. ISBN 978-1-4577-0451-2. DOI: 10.1109/LICS.2011.38.

S. Milius. On iteratable [sic] endofunctors. *Electronic Notes in Theoretical Computer Science*, 69:287–304, Feb. 2003. ISSN 1571-0661. DOI: 10.1016/S1571-0661(04)80570-X.

[7] Note that $\zeta_i = \mu_{C_i} \circ \diamond \varpi_i$.