Temporal Logic with "Until", Functional Reactive Programming with Processes, and Concrete Process Categories

Wolfgang Jeltsch

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

Abstract

Recent research has revealed that the "always" and "eventually" operators from temporal logic correspond to the type constructors for behaviors and events from functional reactive programming (FRP). It is furthermore well-known that the "until" operators from LTL are generalizations of "always" and "eventually". In this paper, we show that behaviors and events can be generalized analogously. The result is a notion of process, which combines continuous and discrete aspects. We develop a common categorical semantics for an intuitionistic temporal logic with "until" and FRP with processes. This semantics reflects time-dependent trueness in temporal logic, time-dependent type inhabitance in FRP, and causality of FRP operations.

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, Causality

1. Introduction

Functional reactive programming (FRP) is a declarative approach to programming reactive systems. Its key constructs are behaviors and events. A behavior is a time-varying value, and an event is a value attached to a time. There is a Curry–Howard correspondence between FRP and an intuitionistic logic that has temporal operators "always" and "eventually" and a linear notion of time [4–6]. Thereby the type constructors for behaviors and events correspond to "always" and "eventually", respectively.

It is well-known that "always" and "eventually" can be defined using the more general "until" operators from linear-time temporal logic (LTL). This suggests that there are generalizations of behaviors and events that correspond to proofs of "until" propositions. We deal with this topic in Section 2 where we make the following contributions:

PLPV '13, January 22, 2013, Rome, Italy.

- We define an intuitionistic temporal logic with "until" operators by giving a syntax and a categorical semantics. Afterwards we

construct an FRP dialect whose syntax and semantics correspond to the syntax and semantics for that temporal logic. We show that proofs of "until" propositions correspond to FRP values that combine continuous and discrete aspects and cover behaviors and events as special cases. We call these values processes.

- We give several examples of how processes can be applied. These examples demonstrate that processes naturally represent constructs in real world applications, and that process types can be used to specify a variety of temporal protocols.

While the categorical semantics from Section 2 is appropriate to motivate processes, it does not exactly match real FRP. FRP programs are constructed using causal operations, so that they are able to produce output solely on the basis of already known input. The problem of the semantics from Section 2 is that it also models noncausal operations. In Section 3, we deal with this issue and thereby make the following contributions:

- We define concrete process categories (CPCs), which can serve as categorical models of FRP with processes. CPCs focus on the time-dependent knowledge about values, not on the values themselves. This makes it possible to express causality of FRP operations. On the logic side of the Curry–Howard correspondence, CPCs lead to a new temporal logic, which we call emergence logic (EL).
- We substantiate the claim that CPCs express causality by proving that certain noncausal operations related to events do not have meanings in arbitrary CPCs.

We discuss related work in Section 4 and give conclusions and an outlook on further work in Section 5.

2. From "Until" to Processes

In this section, we show how "until" operators inspired by linear-time temporal logic (LTL) [1] give rise to an FRP construct that generalizes behaviors and events. We first develop an intuitionistic temporal logic with "until" operators, and investigate its programming language analog afterwards.

2.1 Temporal Logic with "Until" Operators

We consider a temporal logic that is intuitionistic and treats time as linear, but not necessarily discrete.

If A denotes a set of atomic propositions, the syntax of our logic can be defined by the following BNF rule:

$$F ::= A \mid \top \mid \bot \mid F \wedge F \mid F \vee F \mid F \to F \mid \neg F \mid F \triangleright'' F \mid F \blacktriangleright'' F \mid$$
$$F \triangleright' F \mid F \blacktriangleright' F \mid F \triangleright F \mid F \blacktriangleright F \mid \Box' F \mid \Diamond' F \mid \Box F \mid \Diamond F$$

The operators that do not come from propositional logic are called temporal operators. They are further subdivided into strong "until" operators ($\triangleright''$, $\triangleright'$, and $\triangleright$), weak "until" operators ($\blacktriangleright''$, $\blacktriangleright'$, and $\blacktriangleright$), "always" operators ($\Box'$ and $\Box$), and "eventually" operators ($\Diamond'$ and $\Diamond$).[1] The unary operators bind stronger than the binary operators. Among the binary operators, the "until" operators have the highest precedence, followed by $\wedge$, $\vee$, and $\to$ in this order. We consider all binary

operators to be right-associative.

We now develop a categorical semantics for our logic. This semantics is strongly related to one that we used in an earlier work [7, Section 2] for a different temporal logic.

In our semantics, models are tuples $(T, \leqslant, \mathcal{B})$ where $(T, \leqslant)$ is a totally ordered set, and $\mathcal{B}$ is a cartesian closed category with coproducts (CCCC). We use $(T, \leqslant)$ to represent the time scale and $\mathcal{B}$ to model propositional logic in the well-known way [9]. Given a model $(T, \leqslant, \mathcal{B})$, we can regard T as a discrete category and form the functor category $\mathcal{B}^T$. The objects of $\mathcal{B}^T$ serve as meanings of temporal propositions. Since these objects are functions that map times to objects of $\mathcal{B}$, temporal propositions denote time-varying statements. A morphism $f : A \to B$ of $\mathcal{B}^T$ is a family $\{f_t\}_{t \in T}$ of morphisms with $f_t : A(t) \to B(t)$ for every $t \in T$; so if A and B model temporal propositions φ and ψ, f models a proof showing that φ implies ψ at every time.

The CCCC structure of $\mathcal{B}$ gives rise to a CCCC structure of $\mathcal{B}^T$, whose operations are pointwise applications of the respective operations of $\mathcal{B}$. The CCCC structure of $\mathcal{B}^T$ models the propositional fragment of our temporal logic. So operators $\top$, $\bot$, $\wedge$, $\vee$, $\to$, and $\neg$ work pointwise with respect to times; for example, a proposition $\varphi \wedge \psi$ holds at a certain time if φ and ψ individually hold at that time.

A proposition $\varphi \triangleright'' \psi$ holds if ψ will hold at some future time, and φ will hold until ψ holds. A proposition $\varphi \blacktriangleright'' \psi$ holds additionally if φ will hold forever, in which case ψ is not required to hold at any future time. We model the logical operators $\triangleright''$ and $\blacktriangleright''$ by two functors $\triangleright'', \blacktriangleright'' : \mathcal{B}^T \times \mathcal{B}^T \to \mathcal{B}^T$ that fulfill the following equations for all morphisms f and g and times t:[2]

$$(f \triangleright'' g)_t = \coprod_{t' \in (t,\infty)} \left(\left(\prod_{t'' \in (t,t')} f_{t''} \right) \times g_{t'} \right) \tag{1}$$

$$(f \blacktriangleright'' g)_t = (f \triangleright'' g)_t + \prod_{t' \in (t,\infty)} f_{t'} \tag{2}$$

From these equations, we can easily derive how the functors $\triangleright''$ and $\blacktriangleright''$ work on objects. This corresponds to the informal description of the meanings of the logical operators $\triangleright''$ and $\blacktriangleright''$ that we have given above.

For the definitions of the functors $\triangleright''$ and $\blacktriangleright''$ to work, all products and coproducts of families indexed by intervals (t, ∞) and all products of families indexed by intervals (t, t') must exist in $\mathcal{B}$. The latter requirement is covered by the former, since for any sets

[1] Namjoshi and Trefler [12] use $\triangleright$ for the "constrains" modality, while the typical notation for strong "until" is $\mathcal{U}$. We nevertheless decided to use $\triangleright$ for strong "until", since $\mathcal{U}$, being a letter, is not really suitable as an infix operator, and furthermore $\triangleright$, unlike most infix operators, has a filled variant $\blacktriangleright$, which we can use to denote weak "until".

[2] The notation (t, ∞) we use in these equations denotes the set $\{t' \in T \mid t < t'\}$. Note that such a set has a maximum if T has a maximum.

M and N with $M \subseteq N$ and any family $\{A_x\}_{x \in M}$ of objects, we have

$$\prod_{x \in M} A_x \cong \prod_{x \in N} \hat{A}_x \tag{3}$$

with $\left\{\hat{A}_x\right\}_{x \in N}$ defined as follows:

$$\hat{A}_x = \begin{cases} A_x & \text{if } x \in M \\ 1 & \text{otherwise} \end{cases} \tag{4}$$

Since the axioms of a CCCC only guarantee the existence of finite products and coproducts, and the intervals (t, ∞) may be infinite, we have to tighten our requirements on the category $\mathcal{B}$. This leads to the actual definition of our categorical models, which we call fan categories.

Definition 1 (Fan category)**.** *A fan category is a tuple* $(T, \leqslant, \mathcal{B})$ *where* $(T, \leqslant)$ *is a totally ordered set, and* $\mathcal{B}$ *is a CCCC that has all products and coproducts of families indexed by intervals* $(t, \infty) \subseteq T$*. The actual category that* $(T, \leqslant, \mathcal{B})$ *denotes is the functor category* $\mathcal{B}^T$*.*

We call the functors $\triangleright''$ and $\blacktriangleright''$ the basic temporal functors of the fan category $(T, \leqslant, \mathcal{B})$. In particular, we say that $\triangleright''$ is the strong and $\blacktriangleright''$ is the weak basic temporal functor of this fan category. We record this in the following definition.

Definition 2 (Basic temporal functors of a fan category)**.** *For any fan category* $(T, \leqslant, \mathcal{B})$*, the functors* $\triangleright'', \blacktriangleright'' : \mathcal{B}^T \times \mathcal{B}^T \to \mathcal{B}^T$ *that are defined according to* (1) *and* (2) *are called the strong and the weak basic temporal functor of* $(T, \leqslant, \mathcal{B})$*.*

Propositions $\varphi \triangleright' \psi$ and $\varphi \blacktriangleright' \psi$ are similar to $\varphi \triangleright'' \psi$ and $\varphi \blacktriangleright'' \psi$, but require φ to also hold at the present time. Propositions $\varphi \triangleright \psi$ and $\varphi \blacktriangleright \psi$ additionally hold if ψ holds at the present time, in which case φ is not required to hold at any time. We do not treat $\triangleright'$, $\blacktriangleright'$, $\triangleright$, and $\blacktriangleright$ as fundamental, since we can define them in terms of $\triangleright''$ and $\blacktriangleright''$:

$$\varphi \triangleright' \psi = \varphi \wedge \varphi \triangleright'' \psi \qquad \varphi \blacktriangleright' \psi = \varphi \wedge \varphi \blacktriangleright'' \psi \tag{5}$$

$$\varphi \triangleright \psi = \psi \vee \varphi \triangleright' \psi \qquad \varphi \blacktriangleright \psi = \psi \vee \varphi \blacktriangleright' \psi \tag{6}$$

A proposition $\Box'\varphi$ holds if φ will always hold, and a proposition $\Diamond'\varphi$ holds if φ will hold at some future time. A proposition $\Box\varphi$ requires φ to also hold at the present time, and a proposition $\Diamond\varphi$ additionally holds if φ holds at the present time. We define $\Box'$, $\Diamond'$, $\Box$, and $\Diamond$ in terms of "until" operators:

$$\Box'\varphi = \varphi \blacktriangleright'' \bot \qquad \Diamond'\varphi = \top \triangleright' \varphi \tag{7}$$

$$\Box\varphi = \varphi \blacktriangleright' \bot \qquad \Diamond\varphi = \top \triangleright \varphi \tag{8}$$

We can turn (5) through (8) into definitions of functors $\triangleright'$, $\blacktriangleright'$, $\triangleright$, $\blacktriangleright$, $\Box'$, $\Diamond'$, $\Box$, and $\Diamond$ that model the corresponding logical operators. All we have to do is replace propositions by morphisms and the logical operators $\top$, $\bot$, $\wedge$, and $\vee$ by the functors 1, 0, $\times$, and $+$.

Definition 3 (Derived temporal functors)**.** *If $\mathcal{C}$ is a CCCC equipped with functors $\triangleright'', \blacktriangleright'' : \mathcal{C} \times \mathcal{C} \to \mathcal{C}$, the derived temporal functors $\triangleright', \blacktriangleright', \triangleright, \blacktriangleright : \mathcal{C} \times \mathcal{C} \to \mathcal{C}$ and $\Box', \Diamond', \Box, \Diamond : \mathcal{C} \to \mathcal{C}$ are defined as follows:*

$$f \triangleright' g = f \times f \triangleright'' g \qquad f \blacktriangleright' g = f \times f \blacktriangleright'' g \tag{9}$$

$$f \triangleright g = g + f \triangleright' g \qquad f \blacktriangleright g = g + f \blacktriangleright' g \tag{10}$$

$$\Box' f = f \blacktriangleright'' 0 \qquad \Diamond' f = 1 \triangleright' f \tag{11}$$

$$\Box f = f \blacktriangleright' 0 \qquad \Diamond f = 1 \triangleright f \tag{12}$$

Corollary 1. *In every fan category, the following propositions hold:*

$$\Box' f \cong \prod_{t' \in (t,\infty)} f_{t'} \qquad \Diamond' f \cong \coprod_{t' \in (t,\infty)} f_{t'} \tag{13}$$

$$\Box f \cong \prod_{t' \in [t,\infty)} f_{t'} \qquad \Diamond f \cong \coprod_{t' \in [t,\infty)} f_{t'} \tag{14}$$

2.2 FRP with Processes

We now examine the FRP language that is connected to our temporal logic from the previous subsection via a Curry–Howard correspondence. Clearly, the type constructors of this FRP language correspond to the operators of our temporal logic. As usual, the analogs of the propositional logic operators are type constructors for forming finite products, finite sums, and function spaces. The analogs of temporal operators are temporal type constructors, which we denote by the same symbols as their logical counterparts. So if A denotes a set of atomic types that corresponds to the set of atomic temporal propositions, the syntax of our FRP language is defined as follows:[3]

$$T ::= A \mid 1 \mid 0 \mid T \times T \mid T + T \mid T \to T \mid T \triangleright'' T \mid T \blacktriangleright'' T \mid$$
$$T \triangleright' T \mid T \blacktriangleright' T \mid T \triangleright T \mid T \blacktriangleright T \mid \Box' T \mid \Diamond' T \mid \Box T \mid \Diamond T$$

We handle precedence and associativity analogously to our temporal logic.

Because of the Curry–Howard correspondence between our FRP language and our temporal logic, we can take the categorical semantics from the previous subsection as a semantics for FRP.

Let $(T, \leqslant, \mathcal{B})$ be a fan category. Since $\mathcal{B}$ is a CCCC, it models a system of ordinary types that comprises the type constructors for finite products, finite sums, and function spaces. The objects in the functor category $\mathcal{B}^T$ are the meanings of FRP types. Since they are functions that map times to objects of $\mathcal{B}$, FRP types can be regarded as functions from times to ordinary types. So it may depend on the time what values inhabit a certain FRP type. If objects A and B

model FRP types τ_1 and τ_2, a morphism $f : A \to B$ denotes an operation that turns any value that inhabits τ_1 at some time into a value that inhabits τ_2 at the same time.

Finite products, finite sums, and functions spaces in FRP are modeled by the CCCC structure of $\mathcal{B}^T$. So they are pointwise applications of the respective constructions on ordinary types.

We can see from (13) how the type constructors $\Box'$ and $\Diamond'$ work. A value b that inhabits a type $\Box'\tau$ at a time t corresponds to a function that maps each time $t' \in (t, \infty)$ to a value that inhabits τ at t'. So it denotes a time-varying value that persists forever once it has come into existence. Such a value b is called a behavior. A value e that inhabits a type $\Diamond'\tau$ at a time t corresponds to a pair of a time $t' \in (t, \infty)$ and a value x that inhabits τ at t'. So e denotes an event that occurs at t' and is further characterized by x.

Types $\Box'\tau$ contain behaviors that start immediately after the present time, and types $\Diamond'\tau$ contain events that fire in the future. We can see from (14) that the type constructors $\Box$ and $\Diamond$ are variants of $\Box'$ and $\Diamond'$ that also refer to the present. A behavior of a type $\Box\tau$ starts at the present time, and an event of a type $\Diamond\tau$ may also fire at the present time.

Equation (1) shows that each value p that inhabits a type $\tau_1 \triangleright'' \tau_2$ at a time t corresponds to a tuple with the following elements:

- a time $t' \in (t, \infty)$
- a function h that maps each time $t'' \in (t, t')$ to a value that inhabits τ_1 at t''
- a value z that inhabits τ_2 at t'

The function h denotes a time-varying value that does not persist forever, but vanishs immediately before t'. We call this time-varying

value the continuous part of p. The pair (t', z) denotes an event that immediately follows the continuous part. We call this event the terminal event of p. The value p itself is called a process in our terminology. We say that p emits the values $h(t'')$ and z at the times t'' and at the time t', respectively.

---

[3] We do not include a type constructor that corresponds to $\neg$, since such a type constructor is uncommon, and $\neg\varphi$ is just syntactic sugar for $\varphi \to \bot$.

According to (2), a value of a type $\tau_1 \blacktriangleright'' \tau_2$ covers all values of type $\tau_1 \triangleright'' \tau_2$ 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 definitions of the logical operators $\triangleright'$, $\blacktriangleright'$, $\triangleright$, and $\blacktriangleright$ in (5) and (6) give rise to the following definitions of the corresponding FRP type constructors:

$$\tau_1 \triangleright' \tau_2 = \tau_1 \times \tau_1 \triangleright'' \tau_2 \qquad \tau_1 \blacktriangleright' \tau_2 = \tau_1 \times \tau_1 \blacktriangleright'' \tau_2 \tag{15}$$

$$\tau_1 \triangleright \tau_2 = \tau_2 + \tau_1 \triangleright' \tau_2 \qquad \tau_1 \blacktriangleright \tau_2 = \tau_2 + \tau_1 \blacktriangleright' \tau_2 \tag{16}$$

Types $\tau_1 \triangleright' \tau_2$ and $\tau_2 \blacktriangleright' \tau_2$ contain processes whose continuous parts start at the present time instead of immediately after it. Types $\tau_1 \triangleright \tau_2$ and $\tau_1 \blacktriangleright \tau_2$ additionally contain processes that already stop at the present time, and whose continuous parts are therefore empty.

Temporal logic with "until" operators is more expressive than temporal logic with only $\Box'$, $\Diamond'$, $\Box$, and $\Diamond$. Likewise the introduction of processes expands the expressiveness of FRP. It turns out that the additional expressiveness can be used in practice to specify various temporal properties using types. Following we give examples of such uses:

Finite time-varying values. Behaviors can only represent time-

varying values that last forever. In practice, however, we often face situations where a time-varying value ceases to exist after a finite period of time. For example when a media player application plays back a song, it produces an audio signal of finite length, because the song is finite. We can represent finite time-varying values by inhabitants of types $\tau \triangleright 1$. In the case of the audio signal, we can use the type $(\mathbb{R} \times \mathbb{R}) \triangleright 1$, assuming that playback is in stereo. To represent time-varying values with no termination guarantee, we can take types $\tau \blacktriangleright 1$ instead of $\tau \triangleright 1$.

Termination-related information. Sometimes we want to describe information that arises when a time-varying value ends. For example, playback in a media player can terminate because the song has ended or because the user canceled the playback. We may want to state the reason for termination along with the audio signal itself. We can do this via a value of type $(\mathbb{R} \times \mathbb{R}) \triangleright (1 + 1)$.

Time-varying values whose type changes over time. There are situations where the type of a time-varying value changes at discrete points in time. For example, a media player might allow for switching between stereo and mono playback, which means that the audio signals it produces consist of intervals with values of type $\mathbb{R} \times \mathbb{R}$ and intervals with values of type $\mathbb{R}$. If we add least and greatest fixpoints to our FRP type system, we can represent such audio signals by FRP values. Several variants are possible, specifying different properties of signals:

- The type

$$\nu\alpha \,.\, (\mathbb{R} \times \mathbb{R}) \blacktriangleright' \mathbb{R} \blacktriangleright' \alpha$$

contains all stereo–mono signals that do not terminate.

- The type

$$\nu\alpha\ .\ (\mathbb{R} \times \mathbb{R}) \triangleright' \mathbb{R} \triangleright' \alpha$$

  additionally guarantees that a switch from stereo to mono or back will always occur after a finite amount of time.

- The type

$$\nu\alpha\ .\ 1 + (\mathbb{R} \times \mathbb{R}) \blacktriangleright' (1 + \mathbb{R} \blacktriangleright' \alpha)$$

  contains all stereo–mono signals, terminating and nonterminating.

- The type

$$\nu\alpha\ .\ 1 + (\mathbb{R} \times \mathbb{R}) \triangleright' (1 + \mathbb{R} \triangleright' \alpha)$$

is like the previous one, except that it provides an additional switch guarantee.

- Finally the type

$$\mu\alpha\,.\,1 + (\mathbb{R} \times \mathbb{R}) \triangleright' (1 + \mathbb{R} \triangleright' \alpha)$$

contains just the terminating stereo–mono signals.

Weak events. Values of types $\diamond'\tau$ represent events that will definitely occur. In practice, however, we usually do not have an occurrence guarantee. For example, we cannot describe the next key press on the keyboard by a value of type $\diamond'\mathrm{Key}$, because the user might not press a key anymore. However we can use types $1 \blacktriangleright' \tau$ to get rid of the occurrence requirement. For instance, we can represent the potential next key press by a value of type $1 \blacktriangleright' \mathrm{Key}$.

This list of examples gives only a first idea of what is possible with an FRP language that includes strong and weak process types. Temporal protocols more complex than the above ones can be specified with our FRP type system, in particular, by combining processes and fixpoints with sums to allow for alternatives.

3. Concrete Process Categories

Fan categories capture the notions of time-dependent trueness of propositions in temporal logic and time-dependent type inhabitance in FRP. However they do not express causality of FRP operations. In this section, we develop a novel categorical semantics for FRP with processes that overcomes this problem. This semantics gives rise to an intuitionistic temporal logic that comprises a notion of

time-dependent knowledge.

3.1 Causality

FRP values may contain information about the present and the future. However information about a certain time is not available before that time. So when an FRP program produces a value, we generally do not know this value completely, but only its present-related information. Its future-related information becomes increasingly available as time progresses.

Let us illustrate this with an example. We consider a program component P that gives the user some time to press a key on the keyboard. P yields an event e of type $\diamond'(\mathrm{Key} + 1)$. If the user presses a key k before P times out, e fires at the time of the key press and carries $\iota_1(k)$ as its value. Otherwise, e fires at the timeout and carries $\iota_2(\mathrm{tt})$ as its value.

P yields e when it starts to wait for a key press. However all the information that e provides refers to the time when e occurs. Until this time, we cannot obtain any information about e. This is sensible, because neither the occurrence time of e, nor the value it carries can be known before e fires. We cannot know whether the user will press a key before the timeout if neither a key has been pressed yet, nor the timeout has occurred so far. Furthermore we cannot know before a key press what key will be pressed at what time.

Let us now look at the type $\diamond'\mathrm{Key} + \diamond'1$. A value of this type is characterized by a value $i \in \{1, 2\}$, which denotes an alternative, and a value of type $\diamond'\mathrm{Key}$ or $\diamond'1$, depending on i. The information given by i is not tied to any future time, so it is available immediately. If there was a polymorphic operation d that turned values of types $\diamond'(\tau_1 + \tau_2)$ into values of types $\diamond'\tau_1 + \diamond'\tau_2$, we could use d to know

in the present what the alternative of a future sum type value is. In particular, we could apply d to the output of P to get a value of type $\Diamond'\mathrm{Key} + \Diamond' 1$ that tells us immediately whether the user will press a key early enough. An operation that enables us to transfer certain future knowledge to the present is a noncausal operation. Since looking into the future is not possible for FRP programs, noncausal operations like d cannot exist in FRP.

The absence of the operation d from FRP means that $\Diamond'(\varphi \vee \psi) \to \Diamond'\varphi \vee \Diamond'\psi$ cannot be a theorem in the logic that corresponds to FRP. This is unlike classical temporal logic, where $\Diamond'(\varphi \vee \psi)$ and $\Diamond'\varphi \vee \Diamond'\psi$ are equivalent.

The situation is analogous for nullary disjunction, which is $\bot$. In classical temporal logic, the propositions $\Diamond'\bot$ and $\bot$ are equivalent, but $\Diamond'\bot \to \bot$ cannot be a theorem in the logical counterpart of FRP. An FRP operation a from $\Diamond' 0$ to 0 would have to produce a value of type 0 immediately under the assumption that it will receive a value of type 0 in the future. Since there actually is no value of type 0, a could only generate its result by using the hypothetical value of type 0 it will receive later. However only a noncausal operation could do this.

Unfortunately fan categories do not only model causal, but also noncausal operations. In particular, they cover a natural isomorphism $\Diamond'(A + B) \cong \Diamond' A + \Diamond' B$ and an isomorphism $\Diamond' 0 \cong 0$. So the semantics from Section 2 is actually not a proper semantics for FRP. Therefore we develop a class of FRP models that take causality into account. We call these models concrete process categories (CPCs).

3.2 Objects and Morphisms

The key problem of the semantics from Section 2 is that it considers

FRP values to be always known completely. Remember that an FRP type τ is modeled by a function $A : T \to \mathrm{Ob}(\mathcal{B})$ where for each t, the object $A(t)$ denotes the type of all values that inhabit τ at t. So the semantics deals only with the inhabitants themselves, not with the knowledge about them, which is often only partial.

To get rid of this problem, we modify the semantics such that an object A that models an FRP type τ assigns objects of $\mathcal{B}$ to pairs (t, t_o) of times with $t \leqslant t_\mathrm{o}$. Thereby each object $A(t, t_\mathrm{o})$ 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.

We now extend the semantics such that every object A additionally maps each triple $(t, t_\mathrm{o}, t'_\mathrm{o})$ with $t \leqslant t_\mathrm{o} \leqslant t'_\mathrm{o}$ to a morphism

$$A\,(t, t_\mathrm{o}, t'_\mathrm{o}) : A\,(t, t'_\mathrm{o}) \to A(t, t_\mathrm{o})$$

that denotes a function that discards the information gathered after t_o. Certain restrictions apply to the choice of morphisms $A\,(t, t_\mathrm{o}, t'_\mathrm{o})$:

- If the source and the target observation time are the same, no information is actually discarded. So for all objects A and all times t and t_o with $t \leqslant t_\mathrm{o}$, the following equation must hold:

$$A(t, t_\mathrm{o}, t_\mathrm{o}) = \mathrm{id}_{A(t, t_\mathrm{o})} \tag{17}$$

- Two consecutive steps of information disposal lead to the same result as one corresponding single step. So for all objects A and all times t, t_o, t'_o, and t''_o with $t \leqslant t_\mathrm{o} \leqslant t'_\mathrm{o} \leqslant t''_\mathrm{o}$, the following equation must hold:

$$A\,(t, t_\mathrm{o}, t''_\mathrm{o}) = A\,(t, t_\mathrm{o}, t'_\mathrm{o})\,A\,(t, t'_\mathrm{o}, t''_\mathrm{o}) \tag{18}$$

If A and B are objects that model FRP types τ_1 and τ_2, we want the morphisms from A to B to model the causal transformations from τ_1 to τ_2. An operation is causal if the information about its output is uniquely determined by the information about its input for any observation time. Therefore we define a morphism $f : A \to B$ to be a family of morphisms

$$f_{(t,t_o)} : A(t, t_o) \to B(t, t_o)$$

with $t \leqslant t_o$ where each morphism $f_{(t,t_o)}$ models a function that transforms the respective input information into the corresponding output information. If two such functions refer to the same inhabitation time, but to different observation times, they must agree in how they generate common output information. This means that for all t, t_o,

and t'_{o} with $t \leqslant t_{\mathrm{o}} \leqslant t'_{\mathrm{o}}$, the following diagram must commute:

$$\begin{array}{ccc} A(t, t_{\mathrm{o}}) & \xleftarrow{A\,(t,\, t_{\mathrm{o}},\, t'_{\mathrm{o}})} & A\,(t, t'_{\mathrm{o}}) \\ \Big\downarrow{\scriptstyle f_{(t,t_{\mathrm{o}})}} & & \Big\downarrow{\scriptstyle f_{(t,t'_{\mathrm{o}})}} \\ B(t, t_{\mathrm{o}}) & \xleftarrow[B\,(t,\, t_{\mathrm{o}},\, t'_{\mathrm{o}})]{} & B\,(t, t'_{\mathrm{o}}) \end{array} \tag{19}$$

We can succinctly summarize the above development by saying that the category whose objects model FRP types and whose morphisms model causal operations is the functor category $\mathcal{B}^{\mathcal{I}}$ where $\mathcal{I}$ is the temporal index category of $(T, \leqslant)$ as defined by the following definition.

Definition 4 (Temporal index category)**.** *The temporal index category of a totally ordered set $(T, \leqslant)$ is the category $\mathcal{I}$ with*

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

and

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

Corollary 2. *Let $(T, \leqslant)$ be a totally ordered set and $\mathcal{T}$ be its category. Then the temporal index category of $(T, \leqslant)$ is isomorphic to the coproduct of the slice categories of $\mathcal{T}^{\mathrm{op}}$, that is, $\coprod_{t \in T}(\mathcal{T}^{\mathrm{op}} \downarrow t)$.*

3.3 Products and Coproducts

The cartesian and the cocartesian structure of $\mathcal{B}$ give rise to a cartesian and a cocartesian structure of $\mathcal{B}^{\mathcal{I}}$, where products and coproducts are formed pointwise. We use these structures of $\mathcal{B}^{\mathcal{I}}$ to model finite products and finite sums. This is analogous to fan categories, where we use the cartesian and the cocartesian structure of the functor category $\mathcal{B}^{\mathcal{T}}$ for the same purpose.

The cocartesian structure of $\mathcal{B}^{\mathcal{I}}$ is particularly interesting. Let us have a look at its implications on FRP:

- The coproduct of two objects of $\mathcal{B}^{\mathcal{I}}$ is given by the following equation:

$$(A + B)(t, t_o) = A(t, t_o) + B(t, t_o) \tag{20}$$

  So if a is an FRP value that inhabits a type $\tau_1 + \tau_2$ at a time t, we know the alternative that a belongs to at every time $t_o \geqslant t$, including t itself.

- The coproduct of zero objects of $\mathcal{B}^{\mathcal{I}}$ is given by the following equation:

$$0(t, t_o) = 0 \tag{21}$$

  So if we had an FRP value that inhabits 0 at a time t, we could derive a contradiction at every time $t_o \geqslant t$, including t itself.

3.4 Exponentials

For modeling function spaces, we need $\mathcal{B}^{\mathcal{I}}$ to have exponentials. However since the category $\mathcal{I}$ is typically not discrete, exponentials in $\mathcal{B}^{\mathcal{I}}$ cannot be constructed pointwise in general and are not even guaranteed to exist.

If $\mathcal{B} = \mathbf{Set}$, then $\mathcal{B}^{\mathcal{I}}$ is essentially a presheaf category. As a result, all exponentials exist, and they can be defined via the following

equation:

$$B^A(t, t_o) = \mathrm{Hom}_{\mathcal{B}^{\mathcal{I}}}\left(\mathrm{Hom}_{\mathcal{I}}((t, t_o), -) \times A, B\right) \tag{22}$$

This equation can be generalized as follows to cover cases other than $\mathcal{B} = \mathbf{Set}$:

$$B^A(t, t_o) = \int_{(t^*, t_o^*) \in \mathcal{I}} \prod_{\mathrm{Hom}_{\mathcal{I}}((t, t_o), (t^*, t_o^*))} B\left(t^*, t_o^*\right)^{A\left(t^*, t_o^*\right)} \tag{23}$$

If $t^* = t$ and $t_o^* \leqslant t_o$, then $\mathrm{Hom}_{\mathcal{I}}\Big((t, t_o), (t^*, t_o^*)\Big)$ is a terminal object in **Set**, otherwise it is the initial object $\emptyset$. This leads to a simplified version of (23):

$$B^A(t, t_o) = \int_{t_o^* \in [t, t_o]} B\left(t, t_o^*\right)^{A\left(t, t_o^*\right)} \tag{24}$$

Exponentials in $\mathcal{B}^{\mathcal{I}}$ exist if and only if the ends in (24) exist.

Let us see what the above definition of exponentials means for FRP. Say f inhabits a function type at a time t. The information about f at a time t_o is characterized by a family of functions $\left\{f_{t_o^*}\right\}_{t_o^* \in [t, t_o]}$ where for each observation time t_o^*, $f_{t_o^*}$ transforms information about arguments into information about results, and all $f_{t_o^*}$ agree in how they generate common output information. Morphisms $B^A\left(t, t_o, t_o'\right)$ denote FRP operations that discard information by dropping all functions $f_{t_o^*}$ with $t_o^* > t_o$.

3.5 Weak Basic Temporal Functor

We model the process type constructor $\blacktriangleright''$ by a functor $\blacktriangleright''$, like in the semantics from Section 2,

Let us see what information we have about a process p at an observation time t_o. In any case, we know whether p has already terminated or not. Our further knowledge about p depends on its termination status:

- If p has terminated, we know the time when it terminated. All values of its continuous part and the value of its terminal event have been emitted already. However these values may refer to the future; for example, they may be processes themselves. So we might not know them completely. Our information about them is limited by the observation time t_o.
- If p has not terminated (and will possibly never terminate at all), we do not have any information about a terminal event. Furthermore we do not know anything about the values of the continuous part of p that will be emitted after t_o. We only have information about the values of the continuous part up to and including t_o, and this information is itself limited by the observation time t_o.

To aid the definition of $\blacktriangleright''$, we introduce constructs that deal with a more refined notion of information about processes. Let τ_1 and τ_2 be FRP types, and t, $t_\mathrm{o}^\dagger$, and $t_\mathrm{o}^\ddagger$ be times with $t \leqslant t_\mathrm{o}^\dagger \leqslant t_\mathrm{o}^\ddagger$. For every p that inhabits $\tau_1 \blacktriangleright'' \tau_2$ at t, $p_{t_\mathrm{o}^\dagger, t_\mathrm{o}^\ddagger}$ shall denote the information we have about p at $t_\mathrm{o}^\dagger$ with the exception that all information about values emitted by p shall use $t_\mathrm{o}^\ddagger$ as the observation time. We introduce a family $\left\{K_{A,B,t,t_\mathrm{o}^\dagger,t_\mathrm{o}^\ddagger}\right\}$ of objects of $\mathcal{B}$ where $A, B \in \mathrm{Ob}\left(\mathcal{B}^I\right)$ and $t \leqslant t_\mathrm{o}^\dagger \leqslant t_\mathrm{o}^\ddagger$. If A and B model FRP types τ_1 and τ_2, then $K_{A,B,t,t_\mathrm{o}^\dagger,t_\mathrm{o}^\ddagger}$ models the type of all $p_{t_\mathrm{o}^\dagger,t_\mathrm{o}^\ddagger}$ where p inhabits $\tau_1 \blacktriangleright'' \tau_2$ at t. The definition of $K_{A,B,t,t_\mathrm{o}^\dagger,t_\mathrm{o}^\ddagger}$ is as follows:

$$K_{A,B,t,t_{\mathrm{o}}^{\dagger},t_{\mathrm{o}}^{\ddagger}} = \coprod_{t' \in \left(t, t_{\mathrm{o}}^{\dagger}\right]} \left(\left(\prod_{t'' \in (t,t')} A\left(t'', t_{\mathrm{o}}^{\ddagger}\right) \right) \times B\left(t', t_{\mathrm{o}}^{\ddagger}\right) \right) + \prod_{t' \in \left(t, t_{\mathrm{o}}^{\dagger}\right]} A\left(t', t_{\mathrm{o}}^{\ddagger}\right) \tag{25}$$

Applications of objects $A \blacktriangleright'' B$ to objects of $\mathcal{I}$ can be easily defined using the family $\left\{K_{A,B,t,t_{\mathrm{o}}^{\dagger},t_{\mathrm{o}}^{\ddagger}}\right\}$:

$$(A \blacktriangleright'' B)(t, t_{\mathrm{o}}) = K_{A,B,t,t_{\mathrm{o}},t_{\mathrm{o}}} \tag{26}$$

Now let us discuss information disposal for processes. Say A and B are objects that model FRP types τ_1 and τ_2, and t, t_{o}, and t'_{o} are times with $t \leqslant t_{\mathrm{o}} \leqslant t'_{\mathrm{o}}$. The morphism $(A \blacktriangleright'' B)\left(t, t_{\mathrm{o}}, t'_{\mathrm{o}}\right)$ models an FRP operation that transforms $p_{t'_{\mathrm{o}},t'_{\mathrm{o}}}$ into $p_{t_{\mathrm{o}},t_{\mathrm{o}}}$ for every p that

inhabits $\tau_1 \blacktriangleright'' \tau_2$ at t. This FRP operation can be defined as the composition of two suboperations, one that turns values $p_{t'_o,t'_o}$ into values p_{t_o,t'_o}, and another one that turns values p_{t_o,t'_o} into values p_{t_o,t_o}. We introduce two morphisms that model these two suboperations:

- The morphism σ_{A,B,t,t_o,t'_o} models the first step of information disposal. It is defined as follows:

$$\begin{aligned} \sigma_{A,B,t,t_o,t'_o} &: K_{A,B,t,t'_o,t'_o} \to K_{A,B,t,t_o,t'_o} \\ \sigma_{A,B,t,t_o,t'_o} &= \left[\left[\xi_{A,B,t,t',t_o,t'_o} \right]_{t' \in (t,t'_o]}, \iota_2 \langle \pi_{t'} \rangle_{t' \in (t,t_o]} \right] \end{aligned} \tag{27}$$

  This definition uses helper morphisms ξ_{A,B,t,t',t_o,t'_o} that are defined in the following way:

$$\begin{aligned} \xi_{A,B,t,t',t_o,t'_o} &: \left(\prod_{t'' \in (t,t')} A\,(t'', t'_o) \right) \times B\,(t', t'_o) \to K_{A,B,t,t_o,t'_o} \\ \xi_{A,B,t,t',t_o,t'_o} &= \begin{cases} \iota_1 \iota_{t'} & \text{if } t' \leqslant t_o \\ \iota_2 \langle \pi_{t''} \rangle_{t'' \in (t,t_o]} \pi_1 & \text{otherwise} \end{cases} \end{aligned} \tag{28}$$

- The morphism ρ_{A,B,t,t_o,t'_o} models the second of the abovementioned suboperations. Its definition is as follows:

$$\begin{aligned} \rho_{A,B,t,t_o,t'_o} &: K_{A,B,t,t_o,t'_o} \to K_{A,B,t,t_o,t_o} \\ \rho_{A,B,t,t_o,t'_o} &= \coprod_{t' \in (t,t_o]} \left(\left(\prod_{t'' \in (t,t')} A\,(t'', t_o, t'_o) \right) \times B\,(t', t_o, t'_o) \right) + \\ &\qquad \prod_{t' \in (t,t_o]} A\,(t', t_o, t'_o) \end{aligned} \tag{29}$$

The definition of $(A \blacktriangleright'' B)\,(t, t_\mathrm{o}, t'_\mathrm{o})$ is now simple:

$$(A \blacktriangleright'' B)\,(t, t_\mathrm{o}, t'_\mathrm{o}) = \left(\rho_{A,B,t,t_\mathrm{o},t'_\mathrm{o}}\right)\left(\sigma_{A,B,t,t_\mathrm{o},t'_\mathrm{o}}\right) \tag{30}$$

To complete the definition of the functor $\blacktriangleright''$, we have to specify how $\blacktriangleright''$ acts on morphisms of $\mathcal{B}^I$. If morphisms f and g model FRP operations u and v, the morphism $f \blacktriangleright'' g$ models the FRP operation that applies u to the values of the continuous part and v to the value of the terminal event of the respective process. This leads to the following definition:

$$(f \blacktriangleright'' g)_{(t,t_\mathrm{o})} = \coprod_{t' \in (t,t_\mathrm{o}]} \left(\left(\prod_{t'' \in (t,t')} f_{(t'',t_\mathrm{o})}\right) \times g_{(t',t_\mathrm{o})}\right) + \prod_{t' \in (t,t_\mathrm{o}]} f_{(t',t_\mathrm{o})} \tag{31}$$

3.6 Concrete Process Categories So Far

Before we discuss how to model the type constructor $\triangleright''$, let us compile what we have developed so far.

We have devised categorical models that are tuples $(T, \leqslant, \mathcal{B})$ where $(T, \leqslant)$ is a totally ordered set, and $\mathcal{B}$ is a CCCC. There are two things that constrain the choice of the category $\mathcal{B}$:

- We must be able to define the functor $\blacktriangleright''$. Therefore products of families indexed by intervals (t, t') as well as products and coproducts of families indexed by intervals $(t, t']$ must exist. Existence of products and coproducts for intervals $(t, t']$ implies existence of products and coproducts for intervals (t, t').[4] The converse is also true, since we have

$$\prod_{t'' \in (t,t']} A_{t''} = \left(\prod_{t'' \in (t,t')} A_{t''} \right) \times A_{t'}$$

[4] We discussed the product case in Subsection 2.1. The coproduct case is analogous.

and likewise for coproducts. So it is okay to require the existence of products and coproducts of families indexed by intervals (t, t').[5]

- The functor category $\mathcal{B}^{\mathcal{I}}$ must have exponentials. So the ends in (24) must exist.

Given these considerations, we arrive at the definition of CPCs.

Definition 5 (Concrete process category)**.** *Let $(T, \leqslant)$ be a totally ordered set, $\mathcal{I}$ be its temporal index category, and $\mathcal{B}$ be a CCCC 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 $(T, \leqslant, \mathcal{B})$ is a concrete process category (CPC). The actual category that $(T, \leqslant, \mathcal{B})$ denotes is the functor category $\mathcal{B}^{\mathcal{I}}$.*

We now give the definition of the functor $\blacktriangleright''$, which we call weak basic temporal functor, like we did in the case of a fan category.

Definition 6 (Weak basic temporal functor of a CPC)**.** *For any CPC $(T, \leqslant, \mathcal{B})$, the functor $\blacktriangleright'' : \mathcal{B}^{\mathcal{I}} \times \mathcal{B}^{\mathcal{I}} \to \mathcal{B}^{\mathcal{I}}$ that is defined according to* (26)*,* (30)*, and* (31) *is called the weak basic temporal functor of $(T, \leqslant, \mathcal{B})$.*

The only remaining bit of our categorical semantics is a functor

that models the type constructor $\triangleright''$. Functors that model the remaining temporal type constructors can be derived according to Definition 3, like for fan categories.

3.7 Strong Basic Temporal Functor

Processes of a type $\tau_1 \triangleright'' \tau_2$ are guaranteed to terminate. However it is not immediately clear whether such a constraint can be encoded using CPCs, and how it can be encoded if this is possible. The reason is that CPCs only deal with information we have at different times. This does not play well with global properties like the existence of termination times that can be arbitrarily far in the future. That said, it *is* possible to define a functor that models process type constructors that place an upper bound on the termination time. We try to define a meaning of the type constructor $\triangleright''$ based on this functor.

Let $\mathcal{T}$ be the category of $(T, \leqslant)$. We introduce the functor

$$\triangleright''_{\_} : \mathcal{T} \to \left(\mathcal{B}^I\right)^{\mathcal{B}^I \times \mathcal{B}^I}$$

such that for every t_b, the functor

$$\triangleright''_{t_\mathrm{b}} : \mathcal{B}^I \times \mathcal{B}^I \to \mathcal{B}^I$$

models a process type constructor $\triangleright''_{t_\mathrm{b}}$ that enforces a maximum t_b on termination times. So if A and B model FRP types τ_1 and τ_2, the object $A \triangleright''_{t_\mathrm{b}} B$ models the type $\tau_1 \triangleright''_{t_\mathrm{b}} \tau_2$ of all processes that are inhabitants of $\tau_1 \triangleright'' \tau_2$ and terminate at t_b or before. This leads to the following definition:

$$\left(A \triangleright''_{t_\mathrm{b}} B\right)(t, t_\mathrm{o}) =$$

$$\begin{cases} 0 & \text{if } t_\mathrm{b} < t \\ \coprod_{t' \in (t, t_\mathrm{b}]} \left(\left(\prod_{t'' \in (t,t')} A(t'', t_\mathrm{o}) \right) \times B(t', t_\mathrm{o}) \right) & \text{if } t \leqslant t_\mathrm{b} \leqslant t_\mathrm{o} \\ (A \blacktriangleright'' B)(t, t_\mathrm{o}) & \text{if } t_\mathrm{o} < t_\mathrm{b} \end{cases} \tag{32}$$

We leave the definition of morphisms $\left(A \rhd''_{t_\mathrm{b}} B\right)(t, t_\mathrm{o}, t'_\mathrm{o})$ as an exercise to the reader. We also do not show explicitly how to define applications of $\rhd''_{t_\mathrm{b}}$ to morphisms of $\mathcal{B}^{\mathcal{I}}$, since the respective equation can be easily derived from (32) by doing some simple replacements.

So far, we have dealt with applications of $\rhd''_{\_}$ to objects t_b of $\mathcal{T}$. We still have to discuss applications of $\rhd''_{\_}$ to morphisms of $\mathcal{T}$. Let t_b and t'_b be times with $t_\mathrm{b} \leqslant t'_\mathrm{b}$, and let A and B model FRP types τ_1

[5] Note that with a discrete time scale, this requirement is automatically fulfilled, since all intervals (t, t') are finite in this case.

and τ_2. We define $A \mathbin{\triangleright''_{(t_\mathrm{b}, t'_\mathrm{b})}} B$ such that it models the type conversion from $\tau_1 \mathbin{\triangleright''_{t_\mathrm{b}}} \tau_2$ to $\tau_1 \mathbin{\triangleright''_{t'_\mathrm{b}}} \tau_2$:[6]

$$\left(A \mathbin{\triangleright''_{(t_\mathrm{b}, t'_\mathrm{b})}} B\right)_{(t, t_\mathrm{o})} = \begin{cases} ? & \text{if } t_\mathrm{b} < t \\ [\iota_{t'}]_{t' \in (t, t_\mathrm{b}]} & \text{if } t \leqslant t_\mathrm{b} \leqslant t'_\mathrm{b} \leqslant t_\mathrm{o} \\ \iota_1[\iota_{t'}]_{t' \in (t, t_\mathrm{b}]} & \text{if } t \leqslant t_\mathrm{b} \leqslant t_\mathrm{o} < t'_\mathrm{b} \\ \mathrm{id} & \text{if } t_\mathrm{o} < t_\mathrm{b} \leqslant t'_\mathrm{b} \end{cases} \tag{33}$$

Definition 7 (Bounding temporal functor of a CPC)**.** *For any CPC $(T, \leqslant, \mathcal{B})$, the functor $\triangleright''_{\_}$ as defined above is called the bounding temporal functor of $(T, \leqslant, \mathcal{B})$.*

For any FRP types τ_1 and τ_2 and any time t, the inhabitants of $\tau_1 \mathbin{\triangleright''} \tau_2$ at t are the values that inhabit any type $\tau_1 \mathbin{\triangleright''_{t_\mathrm{b}}} \tau_2$ at t. So $\triangleright''$ is the union of all $\triangleright''_{t_\mathrm{b}}$, which means that it is the least upper bound of them. As a result, a functor $\triangleright''$ must be a colimit of the functor $\triangleright''_{\_}$.

Definition 8 (Strong basic temporal functor of a CPC)**.** *Let $(T, \leqslant, \mathcal{B})$ be a CPC and $\triangleright''_{\_}$ be its bounding temporal functor. If $\triangleright''_{\_}$ has a colimit $\triangleright''$, this colimit is called the strong basic temporal functor of $(T, \leqslant, \mathcal{B})$.*

Let us see if a strong basic temporal functor is guaranteed to exist and in case it is, whether it really models the type constructor $\triangleright''$. The situation is good if the time scale has a maximum t_max. In this case, the type constructor $\triangleright''$ is equivalent to $\triangleright''_{t_\mathrm{max}}$, and the categorical semantics is in line with this.

Theorem 3. *Let $(T, \leqslant, \mathcal{B})$ be a CPC where $(T, \leqslant)$ has a maximum t_max. Then $\triangleright''_{t_\mathrm{max}}$ is a colimit of $\triangleright''_{\_}$ with injections given by*

$\iota_{t_\mathrm{b}} = \triangleright''_{(t_\mathrm{b},t_\mathrm{max})}$.

Proof. The claim is true because t_max is a terminal object of $\mathcal{T}$ with $!_{t_\mathrm{b}} = (t_\mathrm{b}, t_\mathrm{max})$. □

Unfortunatly the situation is not so good if time will always progress. In this case, CPCs are not able to express the termination requirement at all.

Theorem 4. *Let $(T, \leqslant, \mathcal{B})$ be a CPC where for every $t \in T$, there is a $t' > t$. Then $\blacktriangleright''$ is a colimit of $\triangleright''_-$ with injections ι_{t_b} given by the following equation:*

$$\left(\iota_{t_\mathrm{b}}\right)_{(A,B),(t,t_\mathrm{o})} = \begin{cases} ? & \text{if } t_\mathrm{b} < t \\ \iota_1[\iota_{t'}]_{t' \in (t,t_\mathrm{b}]} & \text{if } t \leqslant t_\mathrm{b} \leqslant t_\mathrm{o} \\ \mathrm{id} & \text{if } t_\mathrm{o} < t_\mathrm{b} \end{cases} \tag{34}$$

Proof. We show that for all objects A and B of $\mathcal{B}^\mathcal{I}$ and all times t and t_o with $t \leqslant t_\mathrm{o}$, $(A \blacktriangleright'' B)(t, t_\mathrm{o})$ is a colimit of $(A \triangleright''_- B)(t, t_\mathrm{o})$ where the injection for a t_b is $\left(\iota_{t_\mathrm{b}}\right)_{(A,B),(t,t_\mathrm{o})}$. From this follows the claim of the theorem.

Let us fix some concrete A, B, t, and t_o. There exists a time t^*_b with $t^*_\mathrm{b} > t_\mathrm{o}$. We define a preorder $\preccurlyeq$ on T such that $t_\mathrm{b} \preccurlyeq t'_\mathrm{b}$ if and only if

$$(t_\mathrm{b} \leqslant t'_\mathrm{b}) \vee (t_\mathrm{b} > t'_\mathrm{b} \geqslant t^*_\mathrm{b}) \quad .$$

Let $\tilde{\mathcal{T}}$ be the category of $(T, \preccurlyeq)$, and let the functor $Q : \tilde{\mathcal{T}} \to \mathcal{B}$ be defined by the following equations:

$$Qt_\mathrm{b} = \left(A \triangleright''_{t_\mathrm{b}} B\right)(t, t_\mathrm{o}) \tag{35}$$

$$Q\left(t_{\mathrm{b}}, t'_{\mathrm{b}}\right) = \begin{cases} \left(A \rhd''_{\left(t_{\mathrm{b}}, t'_{\mathrm{b}}\right)} B\right)_{(t,t_{\mathrm{o}})} & \text{if } t_{\mathrm{b}} \leqslant t'_{\mathrm{b}} \\ \mathrm{id} & \text{if } t_{\mathrm{b}} > t'_{\mathrm{b}} \geqslant t^*_{\mathrm{b}} \end{cases} \tag{36}$$

[6] We use ? to denote the unique morphisms from 0 to the different objects of $\mathcal{B}$.

Since t^*_{b} is a terminal object of $\tilde{\mathcal{T}}$ with $!_{t_{\mathrm{b}}} = \left(t_{\mathrm{b}}, t^*_{\mathrm{b}}\right)$, the object

$$Qt^*_{\mathrm{b}} = \left(A \rhd''_{t^*_{\mathrm{b}}} B\right)(t, t_{\mathrm{o}}) = (A \blacktriangleright'' B)(t, t_{\mathrm{o}})$$

is a colimit of Q with injections $Q\left(t_{\mathrm{b}}, t^*_{\mathrm{b}}\right) = \left(\iota_{t_{\mathrm{b}}}\right)_{(A,B),(t,t_{\mathrm{o}})}$. The cocones over Q are exactly the cocones over $(A \rhd''_{-} B)(t, t_{\mathrm{o}})$. Therefore $(A \blacktriangleright'' B)(t, t_{\mathrm{o}})$ is also a colimit of $(A \rhd''_{-} B)(t, t_{\mathrm{o}})$ with injections $\left(\iota_{t_{\mathrm{b}}}\right)_{(A,B),(t,t_{\mathrm{o}})}$. □

Let us see what the reason for the mismatch between FRP and its categorical models in the case of an always progressing time is. We motivated our definition of $\rhd''$ with the observation that the type constructor $\rhd''$ is the least upper bound of the type constructors $\rhd''_{t_{\mathrm{b}}}$. Theorem 4 tells us that the type constructor $\blacktriangleright''$ is the least upper bound of the type constructors $\rhd''_{t_{\mathrm{b}}}$ if we only consider type constructors that can be modeled in CPCs. So the bottom line of Theorem 4 is that CPCs are unable to express the termination requirement of $\rhd''$. This is an unfortunate weakness of CPCs. In the future, we want to develop a more general categorical semantics that allows us to model $\rhd''$ precisely without requiring the time scale to have a maximum.

3.8 Causality Again

We developed CPCs, because we wanted a categorical semantics that expresses causality of FRP operations. In particular, we wanted a semantics that does not model polymorphic operations from $\diamond'(\tau_1 + \tau_2)$ to $\diamond'\tau_1 + \diamond'\tau_2$ and operations from $\diamond'0$ to 0, since such operations cannot be causal. In the following, we show that meanings of such operations do not exist in CPCs in general.

Theorem 5. *There are CPCs where no natural transformation τ with $\tau_{(A,B)} : \diamond'(A + B) \to \diamond'A + \diamond'B$ exists.*

Proof. Let $(T, \leqslant, \mathcal{B})$ be a CPC with the following properties:

- T contains exactly two times t_1 and t_2 with $t_1 < t_2$.
- The object $1 + 1$ in $\mathcal{B}$ is not a terminal object.

Let furthermore $\mathcal{I}$ be the temporal index category of $(T, \leqslant)$. We assume that there is a natural transformation $\tau : R \to S$, where $R(f, g) = \diamond'(f + g)$ and $S(f, g) = \diamond'f + \diamond'g$. From τ, we can construct the natural transformation

$$\tau(\Delta \times \Delta) : R(\Delta \times \Delta) \to S(\Delta \times \Delta) \ ,$$

where Δ denotes the diagonal functor from $\mathcal{B}$ to $\mathcal{B}^{\mathcal{I}}$. The functors $R(\Delta \times \Delta)$ and $S(\Delta \times \Delta)$ are objects of the category $\left(\mathcal{B}^{\mathcal{I}}\right)^{\mathcal{B}\times\mathcal{B}}$. There is a canonical isomorphism between this category and the category $\left(\mathcal{B}^{\mathcal{B}\times\mathcal{B}}\right)^{\mathcal{I}}$. Applying this isomorphism to $\tau(\Delta \times \Delta)$ yields a natural transformation σ with

$$\left(\sigma_{(t,t_0)}\right)_{(X,Y)} = \left(\tau_{(\Delta X,\Delta Y)}\right)_{(t,t_0)}$$

for all objects X and Y of $\mathcal{B}$ and all times t and t_{o} with $t \leqslant t_{\mathrm{o}}$. Naturality of σ implies that the following diagram commutes:

$$\begin{array}{ccc} R(\Delta X, \Delta Y)(t_1, t_1) & \xleftarrow{R(\Delta X, \Delta Y)(t_1, t_1, t_2)} & R(\Delta X, \Delta Y)(t_1, t_2) \\ \downarrow \left(\sigma_{(t_1,t_1)}\right)_{(X,Y)} & & \left(\sigma_{(t_1,t_2)}\right)_{(X,Y)} \downarrow \\ S(\Delta X, \Delta Y)(t_1, t_1) & \xleftarrow[S(\Delta X, \Delta Y)(t_1, t_1, t_2)]{} & S(\Delta X, \Delta Y)(t_1, t_2) \end{array} \tag{37}$$

According to (11), (9), and Theorem 3, we have

$$\diamondsuit' = 1 \triangleright' - = 1 \times 1 \triangleright'' - \cong 1 \triangleright'' - \cong 1 \triangleright''_{t_2} - \quad .$$

Using this fact as well as Definition 7, we can turn the above diagram into a simpler one, replacing objects by other objects that are isomorphic to them and adapting morphisms accordingly:

$$\begin{array}{ccc} 1 & \xleftarrow{!_{X+Y}} & X+Y \\ \Big\downarrow{\scriptstyle \left(\rho_{t_1}\right)_{(X,Y)}} & & \Big\downarrow{\scriptstyle \left(\rho_{t_2}\right)_{(X,Y)}} \\ 1+1 & \xleftarrow[!_X + !_Y]{} & X+Y \end{array} \tag{38}$$

Here ρ_{t_1} and ρ_{t_2} are natural transformations that work like $\sigma_{(t_1,t_1)}$ and $\sigma_{(t_1,t_2)}$ up to isomorphism. Since all natural transformations from the coproduct functor to itself are of the form $\alpha + \beta$ with $\alpha, \beta : \mathrm{Id} \to \mathrm{Id}$, we know that

$$\left(\left(\rho_{t_1}\right)_{(X,Y)}\right)(!_{X+Y}) = (!_X + !_Y)\left(\left(\rho_{t_2}\right)_{(X,Y)}\right) = !_X + !_Y$$

for all $X, Y \in \mathrm{Ob}(\mathcal{B})$. For $X = Y = 1$, this means that

$$\left(\left(\rho_{t_1}\right)_{(1,1)}\right)(!_{1+1}) = !_1 + !_1 = \mathrm{id}_1 + \mathrm{id}_1 = \mathrm{id}_{1+1} \quad .$$

On the other hand, we have

$$(!_{1+1})\left(\left(\rho_{t_1}\right)_{(1,1)}\right) = !_1 = \mathrm{id}_1 \quad .$$

So $1 + 1 \cong 1$, which contradicts our requirement that $1 + 1$ is not a terminal object. □

There is a proof of Theorem 5 that works with any time domain

that contains at least two elements and has a maximum, not just with time domains that contain exactly two elements. However this proof is more complicated, so we did not present it here.

Theorem 6. *There are CPCs where no morphism from* $\diamond' 0$ *to* 0 *exists.*

Proof. Let $(T, \leqslant, \mathcal{B})$ be a CPC with the following properties:

- T contains at least two times t_1 and t_2 with $t_1 < t_2$.
- The object 0 in $\mathcal{B}$ is not a terminal object.

We assume that there is a morphism from $\diamond' 0$ to 0. Since $\diamond' 0 \cong 1 \rhd'' 0$, there is also a morphism $f : 1 \rhd'' 0 \to 0$. Let $\iota_{t_2} : \rhd''_{t_2} \to \rhd''$ be the colimit injection for t_2, and let g be $f\left(\left(\iota_{t_2}\right)_{(1,0)}\right)$. Then we have $g : 1 \rhd''_{t_2} 0 \to 0$ and thus

$$g_{(t_1,t_1)} : \left(1 \rhd''_{t_2} 0\right)(t_1, t_1) \to 0 \quad .$$

However $\left(1 \rhd''_{t_2} 0\right)(t_1, t_1)$ is isomorphic to 1, so $g_{(t_1,t_1)}$ is a morphism from a terminal object to the initial object 0. As a result, 0 is also a terminal object, which contradicts our premises. □

3.9 Emergence Logic

As we have seen, fan categories are not proper models of FRP, since they do not express causality. As a result, the logic described in Subsection 2.1, which uses fan categories as its models, is not a precise Curry–Howard correspondent of FRP. However by using

CPCs instead of fan categories, we get a temporal logic that matches FRP exactly. We call this logic emergence logic (EL), since the notion of knowledge emerging over time is inherent in it.

Although many ideas behind EL come from LTL, EL and LTL differ in several ways. For a detailed comparison of both logics see Appendix A.

4. Related Work

Frob is a Haskell library for programming robots with FRP. It contains support for what the authors call processes [15] or tasks [2, 13, 14]. Processes and tasks in the Frob sense are similar to our processes, but differ in two important ways:

- Frob processes can cover effects like exception handling and interrupt monitoring, while our processes are pure values.
- Even Frob processes without effects generally carry more information than our processes. This is because the fundamental constructs in Frob are not processes, but behaviors and event streams. A Frob process p without effects can only be constructed as a pair of a behavior b and an event stream s. If s is empty, then p does not terminate, and b is the continuous part of p. Otherwise the first event in s is the terminal event of p, and the prefix of b that ends just before the terminal event is the continuous part of p. So the suffix of b that starts when the process terminates and all events in s that follow the terminal event are superfluous.

We think that our notion of process that comes out naturally as an

analog of an "until" proof is preferable, since it avoids the problem of superfluous information and can serve as the fundamental FRP concept from which the classical notions of behavior and event can be derived concisely.

Krishnaswami and Benton [8] use the category of complete 1-bounded ultrametric spaces and nonexpansive maps as a model of FRP. The FRP dialect they consider has a discrete notion of time and uses streams as its fundamental temporal construct. The authors prove that nonexpansiveness corresponds to causality, so that morphisms in their semantics only model causal operations. Streams are essentially behaviors in the case of discrete time. So it might be possible to generalize the approach of Krishnaswami and Benton to handle an FRP dialect that works with arbitrary linear time scales and uses behaviors as its only temporal core construct. However we cannot see at the moment, how we could extend this approach such that it covers FRP with processes.

Jeffrey [4] presents an implementation of FRP in the dependently typed programming language Agda. His approach to FRP is in the tradition of Yampa [3] 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 defines a category **RSet**, which is similar to our fan categories. However he interprets morphisms in **RSet** as signal functions that work pointwise. Jeffrey furthermore extends the **RSet** category using the "constrains" modality introduced by McMillan [11]. The morphisms of the resulting category model causal signal functions.

Our own work on the foundations of FRP covers the development of axiomatically defined categorical models of FRP without processes, called temporal categories [7]. Temporal categories are

a specialization of categorical models of intuitionistic S4. Fan categories are special cases of temporal categories. We strongly conjecture that also CPCs are temporal categories, but have not proved this yet.

There is an interesting connection between CPCs and a logic developed by Maier [10], which is based on LTL. The time scale of LTL is the totally ordered set of natural numbers. A typical semantics of LTL is based on ω-words over the alphabet $\mathcal{P}(\mathrm{AP})$ where AP is the set of atomic propositions. Each ω-word α denotes the situation where each atomic formula p is true at a time t if and only if $p \in \alpha_t$. The meaning of a proposition φ is the set of all ω-words that denote situations where φ holds at time 0.

Maier's logic differs from LTL in that its semantics also allow finite words. As a result, the law of excluded middle does not

hold anymore.[7] So Maier calls his logic intuitionistic LTL (ILTL). Maier also gives a new characterization of safety and liveness and proves that each ILTL proposition is equivalent to a conjunction of a safety and a liveness part. It turns out that for every proposition φ, the safety part of φ is characterized by the finite words in $[\![\varphi]\!]$, while the liveness part of φ is characterized by the infinite words in $[\![\varphi]\!]$. So if we change Maier's semantics to only allow finite words, we obtain a logic that can only express safety properties and therefore corresponds to the subset of ILTL that contains all ILTL operators except $\triangleright$. This is analogous to CPCs, which consider only knowledge about finite time intervals and cannot express $\triangleright$ in the case of an unlimited time scale.

5. Conclusions and Further Work

We have shown that proofs of "until" propositions in temporal logic correspond to values in FRP that combine continuous and discrete aspects. We have called these values processes and shown their usefulness. Furthermore we have introduced concrete process categories (CPCs). CPCs can serve as models of FRP with processes and its corresponding temporal logic. In particular, causality of FRP operations is captured by CPCs. We have shown that CPCs cannot express termination guarantees for processes if a time scale with no maximum time is used.

In the future, we want to develop an axiomatically defined categorical FRP semantics that covers both CPCs and temporal categories [7] as special cases. Furthermore we want to extend this semantics such that it covers additional computational concepts like, for example, recursion. We also want to use concepts from categorical FRP semantics in the interface design and possibly the

implementation of FRP systems.

A. Comparison of EL and LTL

LTL [1] is a classical logic that uses a discrete time scale. Its fundamental temporal operators are "next" ($\bigcirc$), "always" ($\Box$), "eventually" ($\Diamond$), and strong "until" ($\triangleright$). EL is an intuitionistic and causal version of LTL without the requirement of a discrete time scale. An immediate consequence of the latter is that we do not have an operator $\bigcirc$. The operators $\Box$ and $\Diamond$ are not fundamental in EL, because they can be derived, as shown in Subsection 2.1. It remains to discuss how and why EL and LTL differ in their treatment of the "until" operators.

LTL does not explicitly require a weak "until" operator to exist. This is because the $\blacktriangleright$-operator can be derived from other operators. One possible way of doing this is the following:

$$\varphi \blacktriangleright \psi = \neg(\neg\psi \triangleright \neg(\varphi \vee \psi)) \tag{39}$$

This definition is not possible in EL for three different reasons:

It does not work in an intuitionistic logic. This can already be seen by looking at the categorical semantics from Subsection 2.1. In this semantics, the isomorphism

$$f \blacktriangleright g \cong ((g \to 0) \triangleright ((f + g) \to 0)) \to 0$$

would have to hold, which is not the case in general.

It is not compatible with causality. Say there is a time t at which $\varphi \blacktriangleright \psi$ and $\neg\psi$ hold for some formulas φ and ψ. If $\varphi \blacktriangleright \psi$ would be equivalent to $\neg(\neg\psi \triangleright \neg(\varphi \vee \psi))$, then $(\neg\psi \triangleright \neg(\varphi \vee \psi)) \to \bot$

would hold at t. A proof of this would include a proof of $(\neg\psi \vee \neg(\varphi \vee \psi)) \rightarrow \bot$, referring to the knowledge we have at t. However such a proof does not exist, as $\neg\psi$ and therefore $\neg\psi \vee \neg(\varphi \vee \psi)$ holds at t.

[7] This is remarkable, since the semantics are essentially based on truth values.

It does not work with every time domain. Let us assume, for example, that our time domain is $(T, \leqslant)$ with $T = \{0\} \cup \left\{ n^{-1} \;\middle|\; n \in \mathbb{N} \right\}$ and $\leqslant$ being the ordinary order of the reals restricted to T. Let us pretend that our logic is classical and does not take causality into account. Say there are propositions φ and ψ such that $\varphi \wedge \neg\psi$ holds at time 0, and $\neg\varphi \wedge \psi$ holds at every other time. Then the formula $\neg(\neg\psi \rhd \neg(\varphi \vee \psi))$ is true at time 0, but $\varphi \blacktriangleright \psi$ is not true at time 0, because there is no least time at which $\neg\varphi \wedge \psi$.

Another way of defining $\blacktriangleright$ in LTL is as follows:[8]

$$\varphi \blacktriangleright \psi = \varphi \rhd \psi \vee \Box\varphi \tag{40}$$

This definition requires $\Box$ to be fundamental, but $\Box$ is not fundamental in EL. What is more, this definition is incompatible with causality: a proof of $\varphi \rhd \psi \vee \Box\varphi$ tells us immediately whether we have termination or not, while a proof of $\varphi \blacktriangleright \psi$ does not tell us about termination in advance.

Another difference between EL and LTL is that the fundamental "until" operators of EL are the future-only operators $\blacktriangleright''$ and $\rhd''$, while the fundamental "until" operator of LTL is $\rhd$ (from which $\blacktriangleright$ can be derived, as we have seen). We have shown in Subsection 2.1 that $\blacktriangleright$ and $\rhd$ can be defined in terms of $\blacktriangleright''$ and $\rhd''$. In LTL, it is also

possible to define $\blacktriangleright''$ and $\triangleright''$ in terms of $\blacktriangleright$ and $\triangleright$ as follows:

$$\varphi \blacktriangleright'' \psi \;=\; \bigcirc(\varphi \blacktriangleright \psi) \qquad\qquad \varphi \triangleright'' \psi \;=\; \bigcirc(\varphi \triangleright \psi) \tag{41}$$

We do not have this option in EL, because we do not have $\bigcirc$, which is why we start with $\blacktriangleright''$ and $\triangleright''$.

Acknowledgments

I thank Tarmo Uustalu for all the helpful discussions about the topics of this paper. Thanks go also to the anonymous reviewers, whose remarks were very helpful for improving the paper. I furthermore thank Mike Shulman and Todd Trimble for answering questions about category theory.

This work was supported by the ERDF through the Estonian Center of Excellence in Computer Science (EXCS).

References

[1] E. A. Emerson. Temporal and modal logic. In J. van Leeuwen, editor, *Formal Models and Semantics*, volume B of *Handbook of Theoretical Computer Science*, pages 995–1072. MIT Press, Cambridge, Massachusetts, Jan. 1994. ISBN 978-0262720151.

[2] G. Hager and J. Peterson. FROB: A transformational approach to the design of robot software. In J. M. Hollerbach and D. E. Koditschek, editors, *Robotics Research: The Ninth International Symposium*, pages 257–264. Springer, London, England, May 2000. ISBN 978-1852332921.

[3] 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.

[4] 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.

[5] W. Jeltsch. Programming in linear temporal logic. Slides of a talk given at the Computer Science Theory Seminar of the TTÜ Küberneetika Instituut, Feb. 2011. URL `http://cs.ioc.ee/~tarmo/tsem10/jeltsch.html`.

[8] This approach is closely related to the definition of the functor $\blacktriangleright''$ in the categorical semantics from Section 2, as shown in (2).

[6] W. Jeltsch. The Curry–Howard correspondence between temporal logic and functional reactive programming. Slides of a talk given at the Estonian Computer Science Theory Days at Nelijärve, Feb. 2011. URL `http://www.cs.ut.ee/~varmo/tday-nelijarve/ettekanded.html`.

[7] 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.

[8] 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.

[9] J. Lambek and P. J. Scott. *Introduction to Higher-Order Categorical Logic*. Number 7 in Cambridge Studies in Advanced Mathematics. Cambridge University Press, Cambridge, England, July 1988. ISBN 978-0521356534.

[10] P. Maier. Intuitionistic LTL and a new characterization

of safety and liveness. Technical Report MPI-I-2004-2-002, Max Planck Institut für Informatik, Saarbrücken, Germany, Aug. 2004. URL http://domino.mpi-inf.mpg.de/internet/reports.nsf/NumberView/2004-2-002.

[11] K. L. McMillan. Circular compositional reasoning about liveness. In L. Pierre and T. Kropf, editors, *Correct Hardware Design and Verification Methods*, volume 1703 of *Lecture Notes in Computer Science*, pages 342–346. Springer, London, England, 1999. ISBN 978-3-540-66559-5. doi: 10.1007/3-540-48153-2_30.

[12] K. S. Namjoshi and R. J. Trefler. On the completeness of compositional reasoning methods. *ACM Transactions on Computational Logic*, 11 (3):16:1–16:22, May 2010. ISSN 1529-3785. doi: 10.1145/1740582.1740584.

[13] J. Peterson and G. Hager. Monadic robotics. In *Proceedings of the 2nd Conference on Domain-Specific Languages (DSL '99)*, pages 95–108, New York, 1999. ACM. ISBN 1-58113-255-7. doi: 10.1145/331960.331976.

[14] J. Peterson, P. Hudak, and C. Elliott. Lambda in motion: Controlling robots with Haskell. In G. Gupta, editor, *Practical Aspects of Declarative Languages*, volume 1551 of *Lecture Notes in Computer Science*, pages 91–105. Springer, Berlin/Heidelberg, Germany, 1998. ISBN 978-3-540-65527-5. doi: 10.1007/3-540-49201-1_7.

[15] J. Peterson, G. Hager, and P. Hudak. A language for declarative robotic programming. In *Proceedings of the 1999 IEEE International Conference on Robotics and Automation (ICRA '99)*, pages 1144–1151, New York, June 1999. IEEE. ISBN 0-7803-5180-0. doi: 10.1109/ROBOT.1999.772516.