


Representing Guardedness in Call-By-Value

Sergey Goncharov 

Friedrich-Alexander-Universität Erlangen-Nürnberg, Germany

Abstract

Like the notion of computation via (strong) monads serves to classify various flavours of impurity, including exceptions, non-determinism, probability, local and global store, the notion of guardedness classifies well-behavedness of cycles in various settings. In its most general form, the guardedness discipline applies to general symmetric monoidal categories and further specializes to Cartesian and co-Cartesian categories, where it governs guarded recursion and guarded iteration respectively. Here, even more specifically, we deal with the semantics of call-by-value guarded iteration. It was shown by Levy, Power and Thielecke that call-by-value languages can be generally interpreted in Freyd categories, but in order to represent effectful function spaces, such a category must canonically arise from a strong monad. We generalize this fact by showing that representing *guarded* effectful function spaces calls for certain parametrized monads (in the sense of Uustalu). This provides a description of guardedness as an intrinsic categorical property of programs, complementing the existing description of guardedness as a predicate on a category.

2012 ACM Subject Classification Theory of computation → Categorical semantics; Theory of computation → Axiomatic semantics

Keywords and phrases Fine-grain call-by-value, Abstract guardedness, Freyd category, Kleisli category, Elgot iteration

Digital Object Identifier 10.4230/LIPIcs.FSCD.2023.34

Acknowledgements The author would like to thank anonymous reviewers of the present and previous editions of the paper for their diligence in their effort to improve it.

1 Introduction

A traditional way to model call-by-value languages is based on a clear cut separation between computations and values. A computation can be *suspended* and thus turned into a value, and a value can be *executed*, and thus again be turned into a computation. The paradigmatic example of these conversions are the application and abstraction mechanisms of λ -calculus. From the categorical modelling perspective, this view naturally requires two categories, suitably connected with each other. As essentially suggested by Moggi [28], a minimal modelling framework requires a Cartesian category (i.e. a category with finite products) as a category of values and a Kleisli category of a strong monad over it, as a category of (side-effecting) computations (also called *producers* [23]). A generic *computational metalanguage* thus arises as an internal language of strong monads. Levy, Power and Thielecke [25], designed a refinement of Moggi's computational metalanguage, called *fine-grain call-by-value (FGCBV)*, whose models are not necessarily strong monads, but are more general *Freyd categories*. They have shown that a strong monad in fact always emerges from a Freyd category if certain function spaces (needed to interpret higher-order functions), are *representable* as objects of the value category – thus strong monads arise from first principles.

Here we analyse an extension of the FGCBV paradigm with a notion of guardedness, which is a certain predicate on computations, certifying their well-behavedness, in particular that they can be iterated [19, 24]. A typical example is guardedness in process algebra, where guardedness is often used to ensure that recursive systems of process definitions have unique solutions [27]. FGCBV does not directly deal with fixpoints, since these are usually



© Sergey Goncharov;

licensed under Creative Commons License CC-BY 4.0

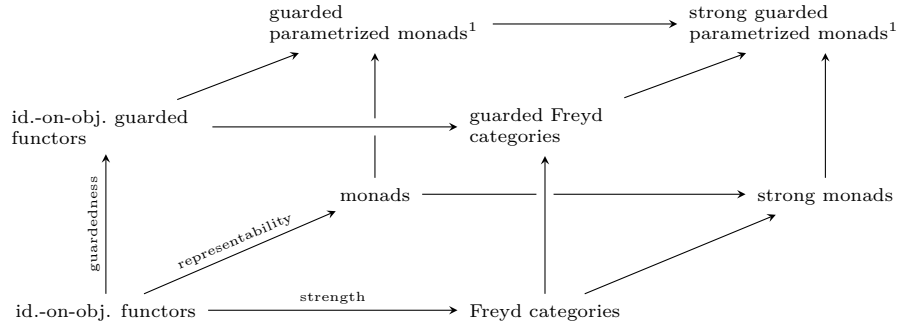
8th International Conference on Formal Structures for Computation and Deduction (FSCD 2023).

Editors: Marco Gaboardi and Femke van Raamsdonk; Article No. 34; pp. 34:1–34:21

Leibniz International Proceedings in Informatics



LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany



■ **Figure 1** Three dimensions within call-by-value.

considered to be features orthogonal to computational effects and evaluation strategies. Analogously, even though the notion of guardedness is motivated by fixpoints, here we do not consider (guarded) fixpoints as a core language feature. In fact, in practically relevant cases guardedness is meaningful on its own as a suitable notion of productivity of computation, and need not be justified via fixpoints, which may or may not exist. In FGCBV one typically regards general recursion to be supported by the category of values, and once the latter indeed supports it, it is obvious to add a corresponding fixpoint construct to the language. Nevertheless, general recursion entails partiality for programs, which means that even if we abstract away from recursion, the corresponding effect of *partiality* must be part of the computational effect abstraction (see e.g. [12]). Recursion and computational effects are thus intimately connected. This connection persists under the restriction from general recursion to iteration, which is subject to a much broader range of models, and triggers the partiality effect just as well. Arguably, the largest class of monads, supporting iteration are *Elgot monads* [3, 16]. These are monads T , equipped with *Elgot iteration*:

$$\frac{f: X \rightarrow T(Y + X)}{f^\dagger: X \rightarrow TY} \quad (1)$$

and subject to established equational laws of iteration [7, 35]. Intuitively, f^\dagger is obtained from f by iterating away the right summand in the output type $Y + X$. For example, the maybe-monad $(-) + 1$ is an Elgot monad over the category of *classical* sets, which yields a model for a while-language with non-termination as the only computational effect. Now, *guarded Elgot monads* [24] refine Elgot monads in that, the operator (1) needs only be defined w.r.t. a custom class of *guarded morphisms*, governed by simple laws. Proper partiality of the guardedness predicate is relevant for various reasons, including the following:

- Guarded fixpoints often uniquely satisfy the corresponding fixpoint equation, which greatly facilitates reasoning, which is extensively used in process algebra.
- In type-theoretic and constructive setting guarded iteration can often be defined natively and more generally, e.g. the “simplest” guarded Elgot monad is Capretta’s *delay monad* (initially called “partiality monad”) [8], rendered by final coalgebras $D = \nu\gamma. (- + \gamma)$, which yield an intensional counterpart of the maybe-monad; guardedness then means

¹ More precisely, representability yields parametrized guarded monads, subject to an additional monicity condition. This is treated in detail in Section 7.

productivity, i.e. that the computation signals as it evolves. Contrastingly, the “simplest” Elgot monad is much harder to construct and arguably requires additional principles to be available in the underlying metatheory [9, 4, 11, 14].

- Guardedness is a compositional type discipline, and hence it potentially helps to encapsulate additional information about productivity of programs in types, like monads encapsulate the information about potential side-effects.

As indicated above, strong monads can be regarded as structures, in a canonical way arising from FGCBV by adding the requirement of representability of certain function spaces in the category of values. This is behind the mechanism of representing computational effects via monads in type systems (e.g. in $F\omega$, by quantification over higher kinds) and hence in programming languages (e.g. in Haskell). Our goal is to provide an analogous mechanism for guardedness and for its combinations with computational effects and strength. That is, (strong) monads are an answer to the question: *what is the categorical/type-theoretic structure that faithfully represents computational effects within a higher-order universe?* Here, we answer the question: *what is the categorical/type-theoretic structure that faithfully represents guarded computational effects within a higher-order universe?* In other words, we seek a formulation of guardedness as an intrinsic structural property of morphisms, instead of additional data that (anonymously) identifies guarded morphisms among others. In doing so, we take inspiration from the view of monads as structures for representing effects, as summarized above. In fact, we show that strength, representability and guardedness can be naturally arranged within FGCBV as three orthogonal dimensions, as shown in Figure 1 (the arrows point from more general concepts to more specific ones). The bottom face of the cube features the above mentioned connection between Freyd categories and strong monads, and a corresponding connection between identity-on-object functors and (not necessarily strong) monads. We contribute with the top face, which combines guardedness with the other dimensions. The pivotal point is the combination of guardedness with representability, which produces a certain class of *parametrized monads* [37], which we dub *guarded parametrized monads*.

Related work. We benefit from the analysis of Power and Robinson [32] who introduced *pre-monoidal categories* as an abstraction of Kleisli categories. Freyd categories were subsequently defined by Power and Thielecke [33] as premonoidal categories with additional structure and also connected to strong monads. Levy [23] came up with an equivalent definition, which we use throughout. In the previous characterization [33, 25], strong monads were shown to arise jointly with Kleisli exponentials from *closed Freyd categories*. We refine this characterization (Corollary 9) by showing that strong monads in fact arise independently of exponentials (Proposition 8). *Distributive Freyd categories* were defined by Staton [36] – here we use them to extend the FGCBV language by coproducts and subsequently with guardedness predicates. Previous approaches to identifying structures for ensuring guardedness on monads involved *monad modules* [30, 2] – we make do with guarded parametrized monads instead, which combine monads with modules over them and arise universally.

Plan of the paper. After short technical preliminaries, we start off by introducing a restricted version of FGCBV in Section 3 and extensively discuss motivating examples, which (with a little effort) can already be encoded despite restrictions. We establish a very simple form of the representability scenario, producing monads, and meant to serve as a model for subsequent sections. In Section 4 we deal with full FGCBV, Freyd categories, modelling them and strong monads, representing Freyd categories. The guardedness dimension is added

34:4 Representing Guardedness in Call-By-Value

in Section 5 where we introduce *guarded Freyd categories* and in Section 6 we analyse the representability issue for them. Finally, in Section 7 we introduce an equational axiomatization of a categorical structure for representing guardedness, called *guarded parametrized monads*. As a crucial technical step, we establish a coherence property in the style of Mac Lane’s coherence theorem for monoidal categories [26]. Conclusions are drawn in Section 8.

2 Preliminaries

We assume familiarity with the basics of category theory [26, 5]. For a category \mathbf{V} , $|\mathbf{V}|$ will denote the class of objects and $\mathbf{V}(X, Y)$ will denote morphisms from X to Y . We tend to omit indexes at natural transformations for readability. A category with finite (co-)products is called (co-)Cartesian. In a co-Cartesian category with selected coproducts, we write $! : 0 \rightarrow A$ for the initial morphism, and $\text{inl} : A \rightarrow A + B$ and $\text{inr} : B \rightarrow A + B$ for the left and right coproduct injections correspondingly. A *distributive category* [10] is a Cartesian and co-Cartesian category, in which the natural transformation

$$X \times Y + X \times Z \xrightarrow{[\text{id} \times \text{inl}, \text{id} \times \text{inr}]} X \times (Y + Z)$$

is an isomorphism, whose inverse we denote $\text{dist}_{X,Y,Z}$ (a co-Cartesian and Cartesian closed category is always distributive). Let $\Delta = \langle \text{id}, \text{id} \rangle : X \rightarrow X \times X$ and $\nabla = [\text{id}, \text{id}] : X + X \rightarrow X$.

A monad \mathbf{T} on \mathbf{V} is determined by a *Kleisli triple* $(T, \eta, (-)^*)$, consisting of a map $T : |\mathbf{V}| \rightarrow |\mathbf{V}|$, a family of morphisms $(\eta_X : X \rightarrow TX)_{X \in |\mathbf{V}|}$ and *Kleisli lifting* sending each $f : X \rightarrow TY$ to $f^* : TX \rightarrow TY$ and obeying *monad laws*:

$$\eta^* = \text{id}, \quad f^* \circ \eta = f, \quad (f^* \circ g)^* = f^* \circ g^*.$$

It follows that T extends to a functor, η extends to a natural transformation – *unit*, $\mu = \text{id}^* : TTX \rightarrow TX$ extends to a natural transformation – *multiplication*, and that (T, η, μ) is a monad in the standard sense [26]. We will generally use blackboard capitals (such as \mathbf{T}) to refer to monads and the corresponding Roman letters (such as T) to refer to their functor parts. Morphisms of the form $f : X \rightarrow TY$ are called *Kleisli morphisms* and form the *Kleisli category* $\mathbf{V}_{\mathbf{T}}$ of \mathbf{T} under *Kleisli composition* $f, g \mapsto f^* \circ g$ with identity η .

A functor F is *strong* if it is equipped with a natural transformation *strength* $\tau : X \times FY \rightarrow F(X \times Y)$, such that the diagrams

$$\begin{array}{ccc} 1 \times FX & \xrightarrow{\text{snd}} & FX \\ \tau \downarrow & \nearrow F \text{snd} & \\ F(1 \times X) & & \end{array} \quad \begin{array}{ccc} (X \times Y) \times FZ & \xrightarrow{\tau} & F((X \times Y) \times Z) \\ \parallel & & \parallel \\ X \times (Y \times FY) & \xrightarrow{X \times \tau} & X \times F(Y \times Z) \xrightarrow{\tau} F(X \times (Y \times Z)) \end{array}$$

commute. A natural transformation, between two strong functors is strong if it preserves strength in the obvious sense, and a monad \mathbf{T} is strong if T is strong with some strength $\tau : X \times TY \rightarrow T(X \times Y)$ and η and μ are strong with id being the strength of Id and $T\tau \circ \tau : X \times TTY \rightarrow TT(X \times Y)$ being the strength of TT .

3 Simple FGCBV with Coproducts

We start off with a restricted – single-variable – fragment of FGCBV, but extended with coproduct types. Since we will not deal with operational semantics, we simplify the language slightly (e.g. we do not include let-expressions for values). We also stick to a Haskell-style

$$\begin{array}{c}
\frac{}{x: A \vdash_v x: A} \quad \frac{f: A \rightarrow B \in \Sigma_v \quad \Gamma \vdash_v v: A}{\Gamma \vdash_v f(v): B} \quad \frac{f: A \rightarrow B \in \Sigma_c \quad \Gamma \vdash_v v: A}{\Gamma \vdash_c f(v): B} \\
\frac{\Gamma \vdash_v v: A}{\Gamma \vdash_c \text{return } v: A} \quad \frac{\Gamma \vdash_c p: A \quad x: A \vdash_c q: B}{\Gamma \vdash_c \text{do } x \leftarrow p; q: B} \quad \frac{\Gamma \vdash_v v: 0}{\Gamma \vdash_v \text{init } v: A} \\
\frac{\Gamma \vdash_v v: A}{\Gamma \vdash_v \text{inl } v: A + B} \quad \frac{\Gamma \vdash_v v: B}{\Gamma \vdash_v \text{inr } v: A + B} \quad \frac{\Gamma \vdash_v v: A + B \quad x: A \vdash_c p: C \quad y: B \vdash_c q: C}{\Gamma \vdash_c \text{case } v \text{ of inl } x \mapsto p; \text{inr } y \mapsto q: C}
\end{array}$$

■ **Figure 2** Simple FGCBV with coproducts.

syntax with do-notation and case-expressions. We fix a collection of sorts S_1, S_2, \dots , a signature Σ_v of pure programs $f: A \rightarrow B$, and a signature Σ_c of effectful programs $f: A \rightarrow B$ (also called *generic effects* [31]) where A and B are types, generated with the grammar

$$A, B ::= S_1, S_2, \dots \mid 0 \mid A + B. \quad (2)$$

We then define terms in context of the form $x: A \vdash_v v: B$ and $x: A \vdash_c p: B$ for value terms and computation terms inductively by the rules given in Figure 2. (where we chose to stick to the syntax of the familiar Haskell’s do-notation): This language is essentially a refinement of Moggi’s *simple (!) computational metalanguage*, which only has one-variable contexts, instead of the fully fledged multi-variable contexts. In terms of monads, the present language corresponds to not necessarily strong ones. Such monads are not very useful in traditional programming languages semantics, however we dwell on this case for several reasons. We aim to explore the interaction between guardedness and monads from a foundational perspective and stay as general as possible to cover the cases where strength does not exist or is not relevant. We also would like to identify the basic representation scenario, to be extended later in more sophisticated cases.

An obvious extension of the presented language would be the iteration operator:

$$\frac{\Gamma \vdash_c p: A \quad x: A \vdash_c q: B + A}{\Gamma \vdash_c \text{iter } x \leftarrow p; q: B} \quad (3)$$

meant to satisfy the fixpoint equality $\text{iter } x \leftarrow p; q = \text{iter } x \leftarrow (\text{do } x \leftarrow p; q); q$. This syntax serves its technical purpose of adding expressivity to the language, but can be criticized from a pragmatic perspective – an arguably more convenient, equivalent syntax of “*labelled iteration*” can be used instead [13], and carried over to guarded setting [17]. Presently, we focus on representing guardedness as such and do not include iteration in the language.

We present three examples, which can be interpreted w.r.t. the single-variable case, to demonstrate the unifying power of FGCBV and to illustrate various flavours of guardedness.

► **Example 1** (Basic Process Algebra [6]). *Basic process algebra (BPA)* over a set of actions \mathcal{A} is defined by the grammar: $P, Q ::= (a \in \mathcal{A}) \mid P + Q \mid P \cdot Q$. One typically considers BPA-terms over free variables (seen as process names) to solve systems of recursive process equations w.r.t. these variables. E.g. we can specify a 2-bit FIFO buffer as a solution to

$$B_0 = \text{in}_0 \cdot B_1^0 + \text{in}_1 \cdot B_1^1, \quad B_1^i = \text{in}_0 \cdot B_2^{0,i} + \text{in}_1 \cdot B_2^{1,i} + \text{out}_i \cdot B_0, \quad B_2^{i,j} = \text{out}_j \cdot B_1^i,$$

where $i, j \in \{0, 1\}$. We view B_0 as an empty FIFO, B_1^i as a FIFO carrying only i and $B_2^{i,j}$ as a FIFO, carrying i and j . For example, the trace $B_0 \xrightarrow{\text{in}_0} B_1^0 \xrightarrow{\text{in}_1} B_2^{1,0} \xrightarrow{\text{out}_0} B_1^1 \xrightarrow{\text{out}_1} B_0$ is valid and represents pushing 0 and 1 to an empty FIFO and then popping them out in

34:6 Representing Guardedness in Call-By-Value

the same order. We can model such systems of equations in FGCBV as follows. Let us fix a single sort 1 and identify an n -fold sum $(\dots(1 + \dots)\dots) + 1$ with the natural number n . The injections $\text{inj}_i: 1 \rightarrow n$ are defined inductively in the obvious way. Let $\Sigma_v = \emptyset$ and $\Sigma_c = \{a: 1 \rightarrow 1 \mid a \in \mathcal{A}\} \cup \{\text{toss}: 1 \rightarrow 2\}$. A BPA-term over process names $\{N_1, \dots, N_n\}$ can be translated to FGCBV recursively, with the following rules where \rightsquigarrow reads as “translates”:

$$\begin{array}{c} \overline{N_i \rightsquigarrow x: 1 \vdash_c \text{return}(\text{inl}(\text{inj}_i x)): n + 1} \quad \overline{a \rightsquigarrow x: 1 \vdash_c \text{do } x \leftarrow a(x); \text{return}(\text{inr } x): n + 1} \\ \\ \frac{P \rightsquigarrow x: 1 \vdash_c p: n + 1 \quad Q \rightsquigarrow x: 1 \vdash_c q: n + 1}{P + Q \rightsquigarrow x: 1 \vdash_c \text{do } x \leftarrow \text{toss}(x); \text{case } x \text{ of } \text{inl } x \mapsto p; \text{inr } x \mapsto q: n + 1} \\ \\ \frac{P \rightsquigarrow x: 1 \vdash_c p: n + 1 \quad Q \rightsquigarrow x: 1 \vdash_c q: n + 1}{P \cdot Q \rightsquigarrow x: 1 \vdash_c \text{do } x \leftarrow p; \text{case } x \text{ of } \text{inl } x \mapsto \text{return}(\text{inl } x); \text{inr } x \mapsto q(x): n + 1} \end{array}$$

Intuitively, the terms $x: 1 \vdash_c p: n + 1$ represent processes with $n + 1$ exit points: every process name N_i identifies an exit i , in addition to the global anonymous exit, as e.g. in an action, regarded as a process. The generic effect `toss` induces binary nondeterminism as a coin-tossing act. Every tuple $(x: 1 \vdash_c p_i: m)_{i < n}$ can be represented by a single term $x: n \vdash_c \hat{p}_n: m$, inductively defined as follows:

$$\hat{p}_0 = \text{return}(\text{init } x), \quad \hat{p}_{n+1} = \text{case } x \text{ of } \text{inl } x \mapsto \hat{p}_n; \text{inr } x \mapsto p_{n+1}.$$

Every system of n equations with $m + n$ variables is thus represented by a term $x: n \vdash_c p: m + n$. The iteration $x: n \vdash_c \text{iter } x \leftarrow \text{return } x; p: m$ computes a solution of this system, sending every i -th variable to a term over the remaining m free variables. Guarded systems are those, where recursive calls are preceded by actions. Such systems have a unique solution (under bisimilarity). The simplest unguarded example $P = P$ has arbitrary solutions, and translates to $x: 1 \vdash_c \text{iter } x \leftarrow \text{return } x; \text{return}(\text{inr } x): 0$.

► **Example 2 (Imperative Traces).** We adapt the semantic framework of Nakata and Uustalu [29] for imperative coinductive traces to our setting. Let us fix a set P of *predicates*, a set T of *state transformers*, and let the corresponding pure and effectful signatures be $\Sigma_v = \{p: S \rightarrow S + S \mid p \in P\} \cup \{t: S \rightarrow S \mid t \in T\}$ and $\Sigma_c = \{\text{put}: S \rightarrow 1, \text{get}: 1 \rightarrow S\}$ over the set of sorts $\{S, 1\}$. The intended interpretation of this data is as follows:

- S is a set of memory states, e.g. the set of finitely supported partial functions $\mathbb{N} \hookrightarrow 2$;
- T are state transformers, e.g. functions, updating precisely one specified memory bit;
- $p \in P$ encode predicates: $p(s) = \text{inl } s$ if the predicate is satisfied and $p(s) = \text{inr } s$ otherwise, e.g. p can capture functions that give a Boolean answer to the questions “is the specified bit 0?” and “is the specified bit 1?”.

For example, the following program negates the i -th memory bit (if it is present)

$$x: 1 \vdash_c \text{do } s \leftarrow \text{get}(x); \text{case } (s[i] = 0) \text{ of } \text{inl } s \mapsto \text{put}(s[i := 1]); \text{inr } s \mapsto \text{put}(s[i := 0]): 1$$

where $(-[i] = 0)$, $(-[i := 0])$ and $(-[i := 1])$ are the obvious predicate and state transformers. Nakata and Uustalu [29] argued in favour of (infinite) traces as a particularly suitable semantics for reasoning about imperative programs. This means that store updates must contribute to the semantics, which can be ensured by a judicious choice of syntax, e.g. by using `skip = do s ← get(x); put(s)`, but not `return`. In FGCBV, however, iterating $x: 1 \vdash_c \text{return}(\text{inr } x): 1$ would not yield any trace. By restricting to guarded iteration with guardedness meaning writing to the store, we can indeed prevent iterating such programs, by defining guardedness in such a way that recursive calls are preceded by `put`.

► **Example 3** (Hybrid Programs). Hybrid programs combine discrete and continuous capabilities and thus can be used to describe behaviours of cyber-physical systems. For simplicity we consider time delays as the only hybrid facility – more sophisticated scenarios are treated elsewhere [15] (more sophisticated scenarios can be modelled in a similar way [15]). Let $\mathbb{R}_{\geq 0}$ be the sort of non-negative real numbers and let Σ_v contain all unary operations on non-negative reals and additionally $\text{is}_0: \mathbb{R}_{\geq 0} \rightarrow \mathbb{R}_{\geq 0} + \mathbb{R}_{\geq 0}$, which sends $n = 0$ to $\text{inl } n$ and $n > 0$ to $\text{inr } n$. Let $\Sigma_c = \{\text{wait}: \mathbb{R}_{\geq 0} \rightarrow \mathbb{R}_{\geq 0}\}$. With $\text{wait}(r)$ we can introduce a time delay of length r and return r . With iteration we can write programs like

$$\begin{aligned} x: \mathbb{R}_{\geq 0} \vdash_c \text{iter } x \leftarrow \text{return } x; \text{ case } \text{is}_0(x) \text{ of} \\ \quad \text{inl } x \mapsto \text{return}(\text{inl } x); \\ \quad \text{inr } x \mapsto (\text{do } x \leftarrow \text{wait}(x); \text{return}(\text{inr } f(x))) : \mathbb{R}_{\geq 0}, \end{aligned}$$

which terminate successfully in finite time ($f(x) = x \dot{-} 1^2$), run infinitely ($f(x) = 1$), or exhibit *Zeno behaviour* ($f(x) = x/2$), i.e. consume finite time, but never terminate. In all these examples, every iteration consumes non-zero time. This is also often considered to be a well-behavedness condition, which can be interpreted as guardedness.

In order to interpret the language from Figure 2, let us fix two co-Cartesian categories \mathbf{V} and \mathbf{C} , and an identity-on-objects functor $J: \mathbf{V} \rightarrow \mathbf{C}$ (hence $|\mathbf{V}| = |\mathbf{C}|$), strictly preserving coproducts. A semantics of (Σ_v, Σ_c) over J assigns

- an object $\llbracket A \rrbracket \in |\mathbf{V}|$ to each sort A ;
- a morphism $\llbracket f \rrbracket \in \mathbf{V}(\llbracket A \rrbracket, \llbracket B \rrbracket)$ to each $f: A \rightarrow B \in \Sigma_v$;
- a morphism $\llbracket f \rrbracket \in \mathbf{C}(\llbracket A \rrbracket, \llbracket B \rrbracket)$ to each $f: A \rightarrow B \in \Sigma_c$,

which extends to types and terms as follows: $\llbracket 0 \rrbracket = 0$, $\llbracket A + B \rrbracket = \llbracket A \rrbracket + \llbracket B \rrbracket$,

- $\llbracket x: A \vdash_v x: A \rrbracket = \text{id}$;
- $\llbracket \Gamma \vdash_v f(v): B \rrbracket = \llbracket f \rrbracket \circ \llbracket \Gamma \vdash_v v: A \rrbracket$;
- $\llbracket \Gamma \vdash_c f(v): B \rrbracket = \llbracket f \rrbracket \circ J[\llbracket \Gamma \vdash_v v: A \rrbracket]$;
- $\llbracket \Gamma \vdash_c \text{return } v: A \rrbracket = J[\llbracket \Gamma \vdash_v v: A \rrbracket]$;
- $\llbracket \Gamma \vdash_c \text{do } x \leftarrow p; q: B \rrbracket = \llbracket x: A \vdash_c q: B \rrbracket \circ \llbracket \Gamma \vdash_c p: A \rrbracket$;
- $\llbracket \Gamma \vdash_v \text{init } v: A \rrbracket = !$;
- $\llbracket \Gamma \vdash_v \text{inl } v: A + B \rrbracket = \text{inl} \circ \llbracket \Gamma \vdash_v v: A \rrbracket$;
- $\llbracket \Gamma \vdash_v \text{inr } v: A + B \rrbracket = \text{inr} \circ \llbracket \Gamma \vdash_v v: B \rrbracket$;
- $\llbracket \Gamma \vdash_c \text{case } v \text{ of } \text{inl } x \mapsto p; \text{inr } y \mapsto q: C \rrbracket \\ = \llbracket [x: A \vdash_c p: C], [y: B \vdash_c q: C] \rrbracket \circ J[\llbracket \Gamma \vdash_v v: A + B \rrbracket]$.

As observed by Power and Robinson [32] (cf. [34, 0.1]), monads arise from the requirement that J is a left adjoint, thus simple FGCBV can be interpreted w.r.t. a monad on \mathbf{V} :

► **Proposition 4.** *Let $J: \mathbf{V} \rightarrow \mathbf{C}$ be an identity-on-objects functor. Then J is a left adjoint iff \mathbf{C} is isomorphic to a Kleisli category of some monad \mathbf{T} on \mathbf{V} and $Jf = H(\eta \circ f)$ for all $f \in \mathbf{V}(X, Y)$ where $H: \mathbf{V}_{\mathbf{T}} \cong \mathbf{C}$ is the relevant isomorphism.*

Moreover, in this situation, finite coproducts in \mathbf{C} are inherited from those in \mathbf{V} , i.e. $J!$ is the initial morphism in \mathbf{C} and $(A + B, J \text{inl}, J \text{inr})$ is a binary coproduct in \mathbf{C} .

² $\dot{-}$ refers to truncated subtraction: $x \dot{-} y = x - y$ if $x \geq y$, and $x \dot{-} y = 0$ otherwise.

$$\begin{array}{c}
 \frac{x: A \text{ in } \Gamma}{\Gamma \vdash_v x: A} \quad \frac{f: A \rightarrow B \in \Sigma_v \quad \Gamma \vdash_v v: A}{\Gamma \vdash_v f(v): B} \quad \frac{f: A \rightarrow B \in \Sigma_c \quad \Gamma \vdash_v v: A}{\Gamma \vdash_c f(v): B} \\
 \\
 \frac{\Gamma \vdash_v v: A}{\Gamma \vdash_c \text{return } v: A} \quad \frac{\Gamma \vdash_c p: A \quad \Gamma, x: A \vdash_c q: B}{\Gamma \vdash_c \text{do } x \leftarrow p; q: B} \quad \frac{\Gamma \vdash_v v: 0}{\Gamma \vdash_v \text{init } v: A} \\
 \\
 \frac{\Gamma \vdash_v v: A}{\Gamma \vdash_v \text{inl } v: A + B} \quad \frac{\Gamma \vdash_v v: B}{\Gamma \vdash_v \text{inr } v: A + B} \quad \frac{\Gamma \vdash_v v: A + B \quad x: A \vdash_c p: C \quad y: B \vdash_c q: C}{\Gamma \vdash_c \text{case } v \text{ of inl } x \mapsto p; \text{inr } y \mapsto q: C} \\
 \\
 \frac{\Gamma \vdash_v v: A \quad \Gamma \vdash_v w: B}{\Gamma \vdash_v \langle v, w \rangle: A \times B} \quad \frac{\Gamma \vdash_v p: A \times B \quad \Gamma, x: A, y: B \vdash_c q: C}{\Gamma \vdash_c \text{case } p \text{ of } \langle x, y \rangle \mapsto q: C}
 \end{array}$$

■ **Figure 3** FGCBV with coproducts.

- **Example 5 (Monads).** Let us recall relevant monads on $\mathbf{V} = \mathbf{Set}$ for further reference.
1. $TX = \nu\gamma. \mathcal{P}_\omega((X + 1) + A \times \gamma)$ where \mathcal{P}_ω is the finite powerset functor and $\nu\gamma. F\gamma$ denotes a final F -coalgebra. This monad provides a standard strong bisimulation semantics for BPA (Example 1). The denotations in TX are finitely branching trees with edges labelled by actions and with terminal nodes labelled in X (free variables) or in 1 (successful termination). This monad is an instance of the *coinductive resumption monad* [30].
 2. $TX = \mathcal{P}(A^* \times (X + 1) + A^*)$ is the monad of finite traces (terminating successfully $A^* \times (X + 1)$ and divergent A^*), which can again be used as a semantics of Example 1.
 3. $TX = \mathcal{P}(A^* \times (X + 1) + (A^* + A^\omega))$ is a refinement of **2.** collecting not only finite, but also infinite traces. If we extend BPA with countable non-determinism, we obtain a semantics properly between strong bisimilarity finite trace equivalence. For example, the equation $P = a \cdot P$ produces the infinite trace a^ω and $P' = \sum_{i \in \mathbb{N}} P_i$ with $P_0 = a$ and $P_{i+1} = a \cdot P_i$ does not, and P is finite trace equivalent to $P + P'$, but not infinite trace equivalent.
 4. $TX = (\nu\gamma. X \times S + \gamma \times S)^S$ can be for Example 2. In \mathbf{Set} , $TX \cong (X \times S^+ + S^\omega)^S$, i.e. an element TX is isomorphic to a function that takes an initial state in S and returns either a finite trace in $X \times S^+$ or an infinite trace in S^ω . We can use Proposition 4 to argue that T indeed extends to a monad. Let \mathbf{C} be the category with $\mathbf{C}(X, Y) = \mathbf{Set}(X \times S, \nu\gamma. Y \times S + \gamma \times S)$, which is a full subcategory of the Kleisli category of the coinductive resumption monad $\nu\gamma. (- + \gamma \times S)$. Now, the obvious identity-on-objects functor $J: \mathbf{Set} \rightarrow \mathbf{C}$ is a left adjoint, which yields the original T .
 5. $TX = \mathbb{R}_{\geq 0} \times X + \bar{\mathbb{R}}_{\geq 0}$ is a monad, which can be used for Example 3. Here $\mathbb{R}_{\geq 0} \times X$ refers to terminating behaviours and $\bar{\mathbb{R}}_{\geq 0} = \mathbb{R}_{\geq 0} \cup \{\infty\}$ to Zeno and infinite behaviours.

4 Freyd Categories and Strong Monads

The full FGCBV (with coproducts) is obtained by extending the type syntax (2) with products $A \times B$, and by replacing the rules in Figure 2 with the rules in Figure 3. We now assume that variable contexts Γ are (possibly empty) lists $(x_1: A_1, \dots, x_n: A_n)$ with non-repetitive x_1, \dots, x_n . To interpret the resulting language, again, we need an identity-on-objects functor $J: \mathbf{V} \rightarrow \mathbf{C}$, an action of \mathbf{V} on \mathbf{C} , and J to preserve this action.

► **Definition 6 (Actegory [20]).** Let (\mathbf{V}, \otimes, I) be a monoidal category. Then an action of \mathbf{V} on a category \mathbf{C} is a functor $\odot: \mathbf{V} \times \mathbf{C} \rightarrow \mathbf{C}$ together with the unitor and the actor natural isomorphisms $v: 1 \otimes X \cong X$, $\alpha: X \odot (Y \odot Z) \cong (X \otimes Y) \odot Z$, satisfying expected coherence conditions. Then \mathbf{C} is called an (\mathbf{V}) -actegory.

Note that every monoidal category trivially acts on itself via $\otimes = \otimes$. In the sequel, we will only consider *Cartesian* categories, i.e. actegories w.r.t. $(\mathbf{V}, \times, 1)$.

► **Definition 7** ((Distributive) Freyd Category [23, 36]). A Freyd category $(\mathbf{V}, \mathbf{C}, J(-), \otimes)$ consists of the following data:

- a Cartesian category \mathbf{V} ;
- a category \mathbf{C} with $|\mathbf{V}| = |\mathbf{C}|$;
- an identity-on-objects functor $J: \mathbf{V} \rightarrow \mathbf{C}$;
- a monoidal action of \mathbf{V} on \mathbf{C} , such that J preserves the \mathbf{V} -action, i.e. $J(f \times g) = f \otimes Jg$ for all $f \in \mathbf{V}(X, X')$, $g \in \mathbf{V}(Y, Y')$, $v = J \text{snd}$ and $\alpha = J(\text{id} \times \text{fst}, \text{snd} \circ \text{snd})$.

A Freyd category $(\mathbf{V}, \mathbf{C}, J(-), \otimes)$ is distributive if \mathbf{V} is distributive, \mathbf{C} is co-Cartesian and J strictly preserves coproducts (this is equivalent to the requirement that the action preserves binary coproducts in the second argument coherently with dist).

Given a distributive Freyd category $(\mathbf{V}, \mathbf{C}, J(-), \otimes)$, we update the semantics from Section 3 as follows, where $\llbracket A \times B \rrbracket = \llbracket A \rrbracket \times \llbracket B \rrbracket$, $\llbracket x_1: A_1, \dots, x_n: A_n \rrbracket = \llbracket A_1 \rrbracket \times \dots \times \llbracket A_n \rrbracket$:

- $\llbracket x_1: A_1, \dots, x_n: A_n \vdash_v x_i: A_i \rrbracket = \text{proj}_i$;
- $\llbracket \Gamma \vdash_c \text{do } x \leftarrow p; q: B \rrbracket = \llbracket \Gamma, x: A \vdash_c q: B \rrbracket \circ (\text{id} \otimes \llbracket \Gamma \vdash_c p: A \rrbracket) \circ \Delta$;
- $\llbracket \Gamma \vdash_c \text{case } v \text{ of } \text{inl } x \mapsto p; \text{inr } y \mapsto q: C \rrbracket$
 $= \llbracket \llbracket \Gamma, x: A \vdash_c p: C \rrbracket, \llbracket \Gamma, y: B \vdash_c q: C \rrbracket \rrbracket \circ J \text{dist} \circ (\text{id} \otimes J \llbracket \Gamma \vdash_v v: A + B \rrbracket) \circ J \Delta$;
- $\llbracket \Gamma \vdash_v \langle v, w \rangle: A \times B \rrbracket = \langle \llbracket \Gamma \vdash_v v: A \rrbracket, \llbracket \Gamma \vdash_v w: B \rrbracket \rangle$.

Freyd categories are to strong monads as identity-on-objects functors to monads.

► **Proposition 8.** Let $(\mathbf{V}, \mathbf{C}, J(-), \otimes)$ be a Freyd category. Then J is a left adjoint iff \mathbf{C} is isomorphic to a Kleisli category of some strong monad \mathbf{T} on \mathbf{V} and $Jf = H(\eta \circ f)$ for all $f \in \mathbf{V}(X, Y)$ where $H: \mathbf{V}_{\mathbf{T}} \cong \mathbf{C}$ is the relevant isomorphism.

Proposition 8 allows us to refactor the existing characterization of *closed Freyd categories* [25, Theorem 7.3] along the following lines. In order to include higher-order types to the language, we would need to add $A \rightarrow B$ as a new type former and the following term formation rules:

$$\frac{\Gamma, x: A \vdash_c p: B}{\Gamma \vdash_v \lambda x. p: A \rightarrow B} \qquad \frac{\Gamma \vdash_v w: A \quad \Gamma \vdash_v v: A \rightarrow B}{\Gamma \vdash_c vw: B}$$

We then need to provide the following additional semantic clauses:

- $\llbracket \Gamma \vdash_v \lambda x. p: A \rightarrow B \rrbracket = \text{curry} \llbracket \Gamma, x: A \vdash_c p: B \rrbracket$;
- $\llbracket \Gamma \vdash_c vw: B \rrbracket = \text{curry}^{-1} \llbracket \Gamma \vdash_v v: A \rightarrow B \rrbracket \circ (\text{id} \otimes J \llbracket \Gamma \vdash_v w: A \rrbracket) \circ J \Delta$,

where $\llbracket A \rightarrow B \rrbracket = \llbracket A \rrbracket \multimap \llbracket B \rrbracket$, $\multimap: |\mathbf{V}| \times |\mathbf{C}| \rightarrow |\mathbf{C}|$, and curry is an isomorphism

$$\text{curry}: \mathbf{C}(J(X \times A), B) \cong \mathbf{V}(X, A \multimap B) \tag{4}$$

natural in X . In particular, this says that J is left adjoint to $1 \multimap (-)$, which, as we have seen in Proposition 4, means that \mathbf{C} is isomorphic to the Kleisli category of a strong monad \mathbf{T} , and hence (4) amounts to $\mathbf{V}(X \times A, TB) \cong \mathbf{V}(X, A \multimap B)$, i.e. to the existence of *Kleisli exponentials*, which are exponentials of the form $(TB)^A$. We thus obtain the following

► **Corollary 9.** Let $(\mathbf{V}, \mathbf{C}, J(-), \otimes)$ be a Freyd category. The following are equivalent:

- an isomorphism (4) natural in X exists;
- for all $A \in |\mathbf{V}|$, $J(- \times A): \mathbf{V} \rightarrow \mathbf{C}$ is a left adjoint;
- \mathbf{C} is isomorphic to a Kleisli category of a strong monad, and Kleisli exponentials exist.

A yet another way to express (4) is to state that the presheaves $\mathbf{C}(J(- \times A), B): \mathbf{V}^{\text{op}} \rightarrow \mathbf{Set}$ are representable. We will use this formulation in our subsequent analysis of guardedness.

5 Guarded Freyd Categories

We proceed to recall the formal notion of guardedness [19, 24].

► **Definition 10** (Guardedness). A guardedness predicate on a co-Cartesian category \mathbf{C} provides for all $X, Y, Z \in |\mathbf{C}|$ a subset $\mathbf{C}_\bullet(X, Y, Z) \subseteq \mathbf{C}(X, Y + Z)$, whose elements we write as $f: X \rightarrow Y \rangle Z$ and call guarded (in Z), such that

$$(\text{trv.}) \quad \frac{f: X \rightarrow Y}{\text{inl} \circ f: X \rightarrow Y \rangle Z} \quad (\text{par.}) \quad \frac{f: X \rightarrow V \rangle W \quad g: Y \rightarrow V \rangle W}{[f, g]: X + Y \rightarrow V \rangle W}$$

$$(\text{cmp.}) \quad \frac{f: X \rightarrow Y \rangle Z \quad g: Y \rightarrow V \rangle W \quad h: Z \rightarrow V + W}{[g, h] \circ f: X \rightarrow V \rangle W}$$

A guarded category is a category, equipped with a guardedness predicate. A guarded functor between two guarded categories is a functor $F: \mathbf{C} \rightarrow \mathbf{D}$ that strictly preserves coproducts, and preserves guardedness in the following sense: $f \in \mathbf{C}_\bullet(X, Y, Z)$ entails $f \in \mathbf{D}_\bullet(FX, FY, FZ)$.

Intuitively, $\mathbf{C}_\bullet(X, Y, Z)$ axiomatically and compositionally distinguishes those morphisms $X \rightarrow Y + Z$ for which the program flow from X to Z is guarded, in particular, if $X = Z$ then the corresponding guarded loop can be safely closed. Note that the standard (total) iteration is an instance with $\mathbf{C}_\bullet(X, Y, Z) = \mathbf{C}(X, Y + Z)$. Consider other instances.

► **Example 11** (Vacuous Guardedness [18]). The least guardedness predicate is as follows: $\mathbf{C}_\bullet(X, Y, Z) = \{\text{inl} \circ f: X \rightarrow Y + Z \mid f \in \mathbf{C}(X, Y)\}$. Such \mathbf{C} is called *vacuously guarded*.

► **Example 12** (Coalgebraic Resumptions). Let \mathbf{T} be a monad on a co-Cartesian category \mathbf{V} and let $H: \mathbf{V} \rightarrow \mathbf{V}$ be an endofunctor such that all fixpoints $T_H X = \nu \gamma. T(X + H\gamma)$ exist. These extend to a monad \mathbf{T}_H , called the (*generalized*) *coalgebraic resumption monad (transform)* of \mathbf{T} [30, 19]. The Kleisli category of \mathbf{T}_H is guarded with $f: X \rightarrow Y \rangle Z$ if

$$\begin{array}{ccc} X & \xrightarrow{g} & T(Y + HT_H(Y + Z)) \\ f \downarrow & & \downarrow T(\text{inl} + \text{id}) \\ T_H(Y + Z) & \xrightarrow{\text{out}} & T((Y + Z) + HT_H(Y + Z)) \end{array}$$

for some $g: X \rightarrow T(Y + HT_H(Y + Z))$. Guarded iteration operators canonically extend from \mathbf{T} to \mathbf{T}_H [24].

► **Example 13** (Algebraic Resumptions). A simple variation of the previous example involves least fixpoints $T^H X = \mu \gamma. T(X + H\gamma)$ instead of the greatest ones and in^{-1} instead of out where $\text{in}: T(X + HT^H X) \rightarrow T^H X$ is the initial algebra structure of $T^H X$, which is an isomorphism by Lambek's lemma. However, we can no longer generally induce non-trivial (guarded) iteration operators for \mathbf{T}^H .

► **Example 14.** Let us describe natural guardedness predicates on the Kleisli categories of monads from Example 5.

1. $TX = \nu \gamma. \mathcal{P}_\omega((X + 1) + A \times \gamma)$ is a special case of Example 12.
2. For $TX = \mathcal{P}(A^* \times (X + 1) + A^*)$, let $f: X \rightarrow Y \rangle Z$ if for every $x \in X$, $\text{inl}(w, \text{inl} \text{ inr } y) \in f(x)$ entails $w \neq \epsilon$.
3. For $TX = \mathcal{P}(A^* \times (X + 1) + (A^* + A^\omega))$ guardedness is defined as in clause 2.
4. For $TX = (\nu \gamma. X \times S + \gamma \times S)^S$, recall that $\mathbf{Set}_\mathbf{T}$ is isomorphic to a full subcategory of the Kleisli category of $\nu \gamma. (- + \gamma \times S)$, which is again an instance of Example 12 with $TX = X$ and $HX = X \times S$. The guardedness predicate for \mathbf{T} thus restricts accordingly.
5. For $TX = (\mathbb{R}_{\geq 0} \times X) + \bar{\mathbb{R}}_{\geq 0}$ let $f: X \rightarrow Y \rangle Z$ if $f(x) = \text{inl}(r, \text{inr } z)$ implies $r > 0$.

$$\begin{array}{c}
\frac{x: A \text{ in } \Gamma}{\Gamma \vdash_v x: A} \quad \frac{f: A \rightarrow B \in \Sigma_v \quad \Gamma \vdash_v v: A}{\Gamma \vdash_v f(v): B} \quad \frac{f: A \rightarrow B \rangle C \in \Sigma_c \quad \Gamma \vdash_v v: A}{\Gamma \vdash_c f(v): B \rangle C} \\
\\
\frac{\Gamma \vdash_v v: A}{\Gamma \vdash_c \text{return } v: A \rangle B} \quad \frac{\Gamma \vdash_c p: A \rangle B \quad \Gamma, x: A \vdash_c q: C \rangle D \quad \Gamma, y: B \vdash_c r: C + D \rangle 0}{\Gamma \vdash_c \text{docase } p \text{ of } \text{inl } x \mapsto q; \text{inr } y \mapsto r: C \rangle D} \\
\\
\frac{\Gamma \vdash_v v: 0}{\Gamma \vdash_v \text{init } v: A} \quad \frac{\Gamma \vdash_v v: A}{\Gamma \vdash_v \text{inl } v: A + B} \quad \frac{\Gamma \vdash_v v: B}{\Gamma \vdash_v \text{inr } v: A + B} \\
\\
\frac{\Gamma \vdash_v v: A + B \quad \Gamma, x: A \vdash_c p: C \rangle D \quad \Gamma, y: B \vdash_c q: C \rangle D}{\Gamma \vdash_c \text{case } v \text{ of } \text{inl } x \mapsto p; \text{inr } y \mapsto q: C \rangle D} \\
\\
\frac{\Gamma \vdash_v v: A \quad \Gamma \vdash_v w: B}{\Gamma \vdash_v \langle v, w \rangle: A \times B} \quad \frac{\Gamma \vdash_v p: A \times B \quad \Gamma, x: A, y: B \vdash_c q: C \rangle D}{\Gamma \vdash_c \text{case } p \text{ of } \langle x, y \rangle \mapsto q: C \rangle D}
\end{array}$$

■ **Figure 4** Term formation rules of guarded FGCBV.

We proceed to extend the language of Figure 3 with guardedness data. As before, Σ_v consists of constructs of the form $f: A \rightarrow B$, while Σ_c consists of constructs of the form $f: A \rightarrow B \rangle C$. In Figure 4 we display the new formation rules that replace their counterparts from Figure 3. The rule for **return** corresponds to the **(trv_•)** rule. The rule for **do** now handles the guarded and unguarded branches, as prescribed by the **(cmp_•)** rule, that is,

$$\text{docase } p \text{ of } \text{inl } x \mapsto q; \text{inr } y \mapsto r$$

is meant to have the same semantics as $\text{do } z \leftarrow p; \text{case } z \text{ of } \text{inl } x \mapsto q; \text{inr } y \mapsto r$, modulo guardedness information. The rule **(par_•)** corresponds to the rule for **case**, which is essentially unchanged w.r.t. Figure 3. Note that a guarded iteration operator could be added with the following rule:

$$\frac{\Gamma \vdash_c p: A \rangle 0 \quad \Gamma, x: A \vdash_c q: B \rangle C + A}{\Gamma \vdash_c \text{iter } x \leftarrow p; q: B \rangle C}$$

The rule **(par_•)** corresponds to the rule for **case**. For every $\Gamma \vdash_c p: A \rangle B + C$ we can construct

$$\begin{aligned}
&\Gamma \vdash_c \text{docase } p \text{ of } \text{inl } x \mapsto \text{return}(\text{inl } x); \\
&\quad \text{inr } z \mapsto \text{case } z \text{ of } \text{inl } x \mapsto \text{return}(\text{inl } \text{inr } x); \\
&\quad \text{inr } y \mapsto \text{return}(\text{inr } y): A + B \rangle C,
\end{aligned}$$

which means that weakening the guardedness guarantee is expressible.

► **Example 15.** The updated effectful signature of Example 1 now involves $a: 1 \rightarrow 0 \rangle 1$ and $\text{toss}: 1 \rightarrow 2 \rangle 0$, indicating that actions guard everything, while nondeterminism guards nothing. The signature Σ_c from Example 2 can be refined to $\{\text{put}: S \rightarrow 0 \rangle 1, \text{get}: 1 \rightarrow S \rangle 0\}$, meaning again that **put** guards everything and **get** guards nothing. Example 3 is more subtle since $\text{wait}: \mathbb{R}_{\geq 0} \rightarrow \mathbb{R}_{\geq 0}$ is meant to be guarded only for non-zero inputs. We thus can embed the involved case distinction into **wait** by redefining it as $\text{wait}: \mathbb{R}_{\geq 0} \rightarrow \mathbb{R}_{\geq 0} \rangle \mathbb{R}_{\geq 0}$.

► **Definition 16** (Guarded Freyd Category). *A distributive Freyd category $(\mathbf{V}, \mathbf{C}, J(-), \otimes)$ is guarded if \mathbf{C} is guarded and the action of \mathbf{V} on \mathbf{C} preserves guardedness in the following sense: Given $f \in \mathbf{V}(A, B)$, $g \in \mathbf{C}_\bullet(X, Y, Z)$, $J \text{ dist } \circ (f \otimes g) \in \mathbf{C}_\bullet(A \times X, B \times Y, B \times Z)$.*

34:12 Representing Guardedness in Call-By-Value

The semantics of (Σ_v, Σ_c) over a guarded Freyd category $(\mathbf{V}, \mathbf{C}, J(-), \otimes)$ interprets types and operations from Σ_v as before and sends each $f: A \rightarrow B \rangle C \in \Sigma_c$ to $\llbracket f \rrbracket \in \mathbf{C}_\bullet(\llbracket A \rrbracket, \llbracket B \rrbracket, \llbracket C \rrbracket)$. Terms in context are interpreted as $\llbracket \Gamma \vdash_v v: B \rrbracket \in \mathbf{V}(\llbracket A \rrbracket, \llbracket B \rrbracket)$ and $\llbracket \Gamma \vdash_c p: B \rangle C \rrbracket \in \mathbf{C}(\llbracket A \rrbracket, \llbracket B \rrbracket + \llbracket C \rrbracket)$ as follows:

- $\llbracket x_1: A_1, \dots, x_n: A_n \vdash_v x_i: A_i \rrbracket = \text{inj}_i$;
- $\llbracket \Gamma \vdash_v f(v): B \rrbracket = \llbracket f \rrbracket \circ \llbracket \Gamma \vdash_v v: A \rrbracket$;
- $\llbracket \Gamma \vdash_c f(v): B \rrbracket = \llbracket f \rrbracket \circ J\llbracket \Gamma \vdash_v v: A \rrbracket$;
- $\llbracket \Gamma \vdash_c \text{return } v: A \rangle B \rrbracket = J \text{inl} \circ J\llbracket \Gamma \vdash_v v: A \rrbracket$;
- $\llbracket \Gamma \vdash_c \text{docase } p \text{ of inl } x \mapsto q; \text{ inr } y \mapsto r: C \rangle D \rrbracket$
 $= \llbracket \llbracket \Gamma, x: A \vdash_c q: C \rangle D \rrbracket, J[\text{id}, !] \circ \llbracket \Gamma, y: B \vdash_c r: C + D \rangle 0 \rrbracket$
 $\circ J \text{dist} \circ (\text{id} \otimes \llbracket \Gamma \vdash_c p: A \rangle B \rrbracket) \circ \Delta$;
- $\llbracket \Gamma \vdash_v \text{init } v: A \rrbracket = !$;
- $\llbracket \Gamma \vdash_v \text{inl } v: A + B \rrbracket = \text{inl} \circ \llbracket \Gamma \vdash_v v: A \rrbracket$;
- $\llbracket \Gamma \vdash_v \text{inr } v: A + B \rrbracket = \text{inr} \circ \llbracket \Gamma \vdash_v v: B \rrbracket$;
- $\llbracket \Gamma \vdash_c \text{case } v \text{ of inl } x \mapsto p; \text{ inr } y \mapsto q: C \rangle D \rrbracket$
 $= \llbracket \llbracket \Gamma, x: A \vdash_c p: C \rangle D \rrbracket, \llbracket \Gamma, y: B \vdash_c q: C \rangle D \rrbracket$
 $\circ J \text{dist} \circ (\text{id} \otimes J\llbracket \Gamma \vdash_v v: A + B \rrbracket) \circ J\Delta$.
- $\llbracket \Gamma \vdash_v \langle v, w \rangle: A \times B \rrbracket = \langle \llbracket \Gamma \vdash_v v: A \rrbracket, \llbracket \Gamma \vdash_v w: B \rrbracket \rangle$.

This is well-defined, which can be easily shown by structural induction:

► **Proposition 17.** *For any derivable $\Gamma \vdash_c p: A \rangle B$, $\llbracket \Gamma \vdash_c p: A \rangle B \rrbracket \in \mathbf{C}_\bullet(\llbracket A \rrbracket, \llbracket B \rrbracket, \llbracket C \rrbracket)$.*

6 Representing Guardedness

In Section 4 we explored the combination of strength (i.e. multivariable contexts) and representability of presheaves $\mathbf{C}(J(-), X): \mathbf{V}^{\text{op}} \rightarrow \mathbf{Set}$, sticking to the bottom face of the cube in Figure 1. Our plan is to obtain further concepts via representability of $\mathbf{C}_\bullet(J(-), X, Y): \mathbf{V}^{\text{op}} \rightarrow \mathbf{Set}$. Note that representability of guardedness jointly with function spaces amounts to representability of $\mathbf{C}_\bullet(J(- \times X), Y, Z): \mathbf{V}^{\text{op}} \rightarrow \mathbf{Set}$, i.e. existence of an endofunctor $\dashv: \mathbf{V}^{\text{op}} \times \mathbf{C} \times \mathbf{C} \rightarrow \mathbf{C}$, such that $\mathbf{C}_\bullet(J(- \times X), Y, Z) \cong \mathbf{V}(-, X \dashv_Z Y)$. This is exactly the structure, one would need to extend Figure 4 with function spaces as follows:

$$\frac{\Gamma, x: A \vdash_c p: B \rangle C}{\Gamma \vdash_v \lambda x. p: A \rightarrow_C B} \qquad \frac{\Gamma \vdash_v w: A \quad \Gamma \vdash_v v: A \rightarrow_C B}{\Gamma \vdash_c vw: B \rangle C}$$

The decorated function spaces $A \rightarrow_C B$ then can be interpreted as $\llbracket A \rrbracket \dashv_{\llbracket C \rrbracket} \llbracket B \rrbracket$, which is a subobject of the Kleisli exponential $\llbracket A \rrbracket \rightarrow T(\llbracket B \rrbracket + \llbracket C \rrbracket)$, consisting of guarded morphisms.

► **Definition 18.** *Given $J: \mathbf{V} \rightarrow \mathbf{C}$, where \mathbf{C} is guarded co-Cartesian, we call the guardedness predicate \mathbf{C}_\bullet J -representable if for all $X, Y \in |\mathbf{C}|$ the presheaf $\mathbf{C}_\bullet(J(-), X, Y): \mathbf{V}^{\text{op}} \rightarrow \mathbf{Set}$ is representable; \mathbf{C}_\bullet J -guarded if it is equipped with a J -representable guardedness predicate.*

J -representability means that for all $X, Y \in |\mathbf{C}|$ there is $U(X, Y) \in |\mathbf{V}|$ such that $\mathbf{C}_\bullet(JZ, X, Y) \cong \mathbf{V}(Z, U(X, Y))$ naturally in Z . Hence, J -representability of guardedness entails that J is a left adjoint. This can be formulated in terms of *free objects* [1].

► **Lemma 19.** *Given an identity-on-objects guarded functor $J: \mathbf{V} \rightarrow \mathbf{C}$ (with \mathbf{V} regarded as vacuously guarded), \mathbf{C}_\bullet is J -representable iff*

- *there is a family of objects $(U(X, Y) \in |\mathbf{V}|)_{X, Y \in |\mathbf{C}|}$;*
- *there is a family of guarded morphisms $(\epsilon_{X, Y}: U(X, Y) \rightarrow X \rangle Y)_{X, Y \in |\mathbf{C}|}$;*

- there is an operator $(-)^{\sharp}: \mathbf{C}_{\bullet}(Z, X, Y) \rightarrow \mathbf{V}(Z, U(X, Y))$ sending each $f: Z \rightarrow X \wr Y$ to the unique morphism f^{\sharp} for which the diagram

$$\begin{array}{ccc} & & U(X, Y) \\ & \nearrow Jf^{\sharp} & \downarrow \epsilon_{X, Y} \\ X & \xrightarrow{f} & X + Y \end{array}$$

commutes.

These conditions entail that U is a bifunctor and that $\epsilon_{X, Y}$ is natural in X and Y .

- **Lemma 20.** If \mathbf{C} is J -guarded co-Cartesian then $J \dashv U(-, 0)$ with U as in Lemma 19.

By Lemma 20, representability fails already if J fails to be adjoint. Instructive examples of non-representability are thus only those, where J does have a right adjoint.

- **Proposition 21.** Let \mathbf{T} be a monad over the category of sets \mathbf{Set} with the axiom of choice. If $\mathbf{Set}_{\mathbf{T}}$ is guarded, the guardedness predicate is representable iff every $f: X \rightarrow T(Y + Z)$ is guarded whenever all the compositions $1 \hookrightarrow X \xrightarrow{f} T(Y + Z)$ are guarded.

- **Example 22 (Failure of Representability).** In \mathbf{Set} , let $f: X \rightarrow Y + Z$ be guarded in Z if $\{z \in Z \mid f^{-1}(\text{inr } z) \neq \emptyset\}$ is finite. The axioms of guardedness are easy to verify. By Proposition 21, this predicate is not Id-representable, as any $1 \hookrightarrow X \xrightarrow{\text{inr}} 0 + X$ is guarded, but inr is not if X is infinite.

In what follows, we will be using $\#$ as a binary operation that binds stronger than monoidal products $(\otimes, +, \dots)$, so e.g. $X \otimes Y \# Z$ will read as $X \otimes (Y \# Z)$.

- **Theorem 23.** Given an identity-on-objects guarded $J: \mathbf{V} \rightarrow \mathbf{C}$, \mathbf{C}_{\bullet} is J -representable iff
 - there is a bifunctor $\#: \mathbf{V} \times \mathbf{V} \rightarrow \mathbf{V}$, such that $- \# 0$ is a monad and $\mathbf{C} \cong \mathbf{V}_{- \# 0}$;
 - there is a family of guarded morphisms (w.r.t. the guardedness predicate, induced by $\mathbf{C} \cong \mathbf{V}_{- \# 0}$) $(\epsilon_{X, Y}: X \# Y \rightarrow X \wr Y)_{X, Y \in |\mathbf{V}|}$, natural in X and Y ;
 - for every guarded $f: X \rightarrow Y \wr Z$ there is unique $f^{\sharp}: X \rightarrow Y \# Z$, such that the diagram

$$\begin{array}{ccc} & & Y \# Z \\ & \nearrow f^{\sharp} & \downarrow \epsilon_{Y, Z} \\ X & \xrightarrow{f} & (Y + Z) \# 0 \end{array}$$

commutes.

Theorem 23 provides a bijective correspondence between morphisms $f: X \rightarrow Y \wr Z$ in \mathbf{C} and the morphisms $f^{\sharp}: X \rightarrow Y \# Z$ in \mathbf{V} , representing them. Uniqueness of the f^{\sharp} is easily seen to be equivalent to the monicity of the $\epsilon_{X, Z}$.

7 Guarded Parametrized Monads

Theorem 23 describes guardedness as a certain bifunctor $\#: \mathbf{V} \times \mathbf{V} \rightarrow \mathbf{V}$ and a family of morphisms $(\epsilon_{X, Y})_{X, Y \in |\mathbf{V}|}$, so that the guardedness predicate is derivable. However, the guardedness laws are still formulated in terms of this predicate, and not in terms of $\#$ and ϵ . To resolve this issue, we must identify a collection of canonical morphisms, and a complete set of equations relating them, in the sense that the guardedness laws for all derived guarded morphisms follow from them. For example, by applying $(-)^{\sharp}$ to the composition

$$A \# (B + C) \xrightarrow{\epsilon_{A, B+C}} (A + (B + C)) \# 0 \cong ((A + B) + C) \# 0$$

34:14 Representing Guardedness in Call-By-Value

we obtain a morphism $v_{A,B,C}: A\#(B+C) \rightarrow (A+B)\#C$, which represents weakening of the guardedness guarantee: in $A\#(B+C)$ the guarded part is $B+C$, while in $(A+B)\#C$ the guarded part is only C . It should not make a difference though if starting from $A\#(B+(C+D))$ we apply v twice or rearrange $B+(C+D)$ by associativity and subsequently apply v only once – the results must be canonically isomorphic, which is indeed provable. Similarly, we will introduce further morphisms like v and derive laws, relating them. We then prove that the resulting axiomatization enjoys a coherence property (Theorem 25) in the style of Mac Lane’s coherence theorem for (symmetric) monoidal categories [26],

► **Definition 24 (Guarded Parametrized Monad).** A guarded parametrized monad on a symmetric monoidal category (\mathbf{V}, \otimes, I) consists of a bifunctor $\#: \mathbf{V} \times \mathbf{V} \rightarrow \mathbf{V}$ and natural transformations

$$\begin{aligned} \eta: A &\rightarrow A\#I, \\ v: A\#(B \otimes C) &\rightarrow (A \otimes B)\#C, & \xi: (A\#B)\#C &\rightarrow A\#(B \otimes C), \\ \chi: A\#B \otimes C\#D &\rightarrow (A \otimes C)\#(B \otimes D), & \zeta: A\#(B\#C) &\rightarrow A\#(B \otimes C). \end{aligned}$$

such that the following diagrams commute, where \cong refers to obvious canonical isomorphisms

$$\begin{array}{ccc} A\#(I \otimes C) & \xrightarrow{v} & (A \otimes I)\#C \\ \cong \downarrow & & \cong \downarrow \\ & & A\#C \end{array} \qquad \begin{array}{ccc} A\#(B \otimes (C \otimes D)) & \cong & A\#((B \otimes C) \otimes D) \\ v \downarrow & & \downarrow v \\ (A \otimes B)\#(C \otimes D) & & \\ v \downarrow & & \\ ((A \otimes B) \otimes C)\#D & \cong & (A \otimes (B \otimes C))\#D \end{array}$$

$$\begin{array}{ccc} A \otimes B & \xrightarrow{\eta \otimes \eta} & A\#I \otimes B\#I \\ \eta \downarrow & & \downarrow \chi \\ (A \otimes B)\#I & \cong & (A \otimes B)\#(I \otimes I) \end{array} \qquad \begin{array}{ccc} A\#B \otimes C\#D & \cong & C\#D \otimes A\#B \\ \chi \downarrow & & \downarrow \chi \\ (A \otimes C)\#(B \otimes D) & \cong & (C \otimes A)\#(D \otimes B) \end{array}$$

$$\begin{array}{ccc} A\#B \otimes (C\#D \otimes E\#F) & \cong & (A\#B \otimes C\#D) \otimes E\#F \\ \text{id}\#\chi \downarrow & & \downarrow \chi\#\text{id} \\ A\#B \otimes (C \otimes E)\#(D \otimes F) & & (A \otimes C)\#(B \otimes D) \otimes E\#F \\ \chi \downarrow & & \downarrow \chi \\ (A \otimes (C \otimes E))\#(B \otimes (D \otimes F)) & \cong & ((A \otimes C) \otimes E)\#((B \otimes D) \otimes F) \end{array}$$

$$\begin{array}{ccc} (A\#I)\#B & \xrightarrow{\xi} & A\#(I \otimes B) \\ \eta\#\text{id} \swarrow & & \cong \downarrow \\ & & A\#B \end{array} \qquad \begin{array}{ccc} A\#(B\#I) & \xrightarrow{\zeta} & A\#(B \otimes I) \\ \text{id}\#\eta \swarrow & & \cong \downarrow \\ & & A\#B \end{array}$$

$$\begin{array}{ccc} & & A\#((B\#C)\#(D\#E)) \\ \zeta \swarrow & & \searrow \text{id}\#\zeta \\ A\#((B\#C) \otimes (D\#E)) & & A\#((B\#C)\#(D \otimes E)) \\ \text{id}\#\chi \downarrow & & \downarrow \text{id}\#\zeta \\ A\#((B \otimes D)\#(C \otimes E)) & & A\#(B\#(C \otimes (D \otimes E))) \\ \zeta \downarrow & & \downarrow \zeta \\ A\#((B \otimes D) \otimes (C \otimes E)) & \cong & A\#(B \otimes (C \otimes (D \otimes E))) \end{array}$$

$$\begin{array}{c}
(A \# (B \# C)) \# (D \# E) \\
\swarrow \xi \quad \searrow \zeta \# \text{id} \\
A \# ((B \# C) \otimes (D \# E)) \qquad (A \# (B \otimes C)) \# (D \# E) \\
\downarrow \text{id} \# \chi \quad \downarrow \text{id} \# \zeta \\
A \# ((B \# C) \otimes (D \# E)) \qquad (A \# (B \otimes C)) \# (D \otimes E) \\
\downarrow \zeta \quad \downarrow \xi \\
A \# ((B \otimes D) \otimes (C \otimes E)) \cong A \# ((B \otimes C) \otimes (D \otimes E)) \\
\\
((A \# B) \# C) \otimes ((D \# E) \# F) \xrightarrow{x} (A \# B \otimes D \# E) \# (C \otimes F) \\
\downarrow \xi \# \epsilon \quad \downarrow \chi \# \text{id} \\
A \# (B \otimes C) \otimes D \# (E \otimes F) \qquad ((A \otimes D) \# (B \otimes E)) \# (C \otimes F) \\
\downarrow x \quad \downarrow \xi \\
(A \otimes D) \# ((B \otimes C) \otimes (E \otimes F)) \cong (A \otimes D) \# ((B \otimes E) \otimes (C \otimes F)) \\
\\
(A \# (B \# C)) \otimes (D \# (E \# F)) \xrightarrow{x} (A \otimes D) \# (B \# C \otimes E \otimes F) \\
\downarrow \zeta \# \chi \quad \downarrow \text{id} \# \chi \\
A \# (B \otimes C) \otimes D \# (E \otimes F) \qquad (A \otimes D) \# ((B \otimes E) \# (C \otimes F)) \\
\downarrow x \quad \downarrow \zeta \\
(A \otimes D) \# ((B \otimes C) \otimes (E \otimes F)) \cong (A \otimes D) \# ((B \otimes E) \otimes (C \otimes F)) \\
\\
(A \# B) \# (C \# D \otimes E \# F) \xrightarrow{v} (A \# B \otimes C \# D) \# (E \# F) \\
\downarrow \text{id} \# \chi \quad \downarrow \chi \# \text{id} \\
(A \# B) \# ((C \otimes E) \# (D \otimes F)) \qquad ((A \otimes C) \# (B \otimes D)) \# (E \# F) \\
\downarrow \zeta \quad \downarrow \zeta \\
(A \# B) \# ((C \otimes E) \otimes (D \otimes F)) \qquad ((A \otimes C) \# (B \otimes D)) \otimes (E \# F) \\
\downarrow \xi \quad \downarrow \xi \\
A \# (B \otimes ((C \otimes D) \otimes (E \otimes F))) \qquad (A \otimes C) \# ((B \otimes D) \otimes (E \otimes F)) \\
\cong \quad \cong \\
A \# (C \otimes ((B \otimes D) \otimes (E \otimes F))) \xrightarrow{v} (A \otimes C) \# ((B \otimes D) \otimes (E \otimes F)) \\
\\
A \# B \otimes C \# (D \otimes E) \xrightarrow{\text{id} \otimes v} A \# B \otimes (C \otimes D) \# E \\
\downarrow x \quad \downarrow x \\
(A \otimes C) \# (B \otimes (D \otimes E)) \qquad (A \otimes (C \otimes D)) \# (B \otimes E) \\
\cong \quad \cong \\
(A \otimes C) \# (D \otimes (B \otimes E)) \xrightarrow{v} ((A \otimes C) \otimes D) \# (B \otimes E) \\
\\
A \# (B \# (C \otimes D)) \xrightarrow{\text{id} \# v} A \# ((B \otimes C) \# D) \qquad (A \# (B \otimes C)) \# D \xrightarrow{v \# \text{id}} ((A \otimes B) \# C) \# D \\
\downarrow \zeta \quad \downarrow \zeta \quad \downarrow \xi \quad \downarrow \xi \\
A \# (B \otimes (C \otimes D)) \cong A \# ((B \otimes C) \otimes D) \qquad A \# ((B \otimes C) \otimes D) \qquad (A \otimes B) \# (C \otimes D) \\
\cong \quad \cong \quad \cong \\
A \# (B \otimes (C \otimes D)) \xrightarrow{v} (A \otimes B) \# (C \otimes D)
\end{array}$$

34:16 Representing Guardedness in Call-By-Value

The value of the presented axiomatization is attested by the following

► **Theorem 25 (Coherence).** *Let $\mathcal{E}_1, \mathcal{E}_2$ and $\mathcal{E}_2 \# \mathcal{E}'_2$ be expressions, built from $\otimes, \#$ and I over a set of letters, in such a way that \mathcal{E}_1 and $\mathcal{E}_2 \# \mathcal{E}'_2$ contain each letter at most once and neither \mathcal{E}_2 nor \mathcal{E}'_2 contain $\#$. Given two expressions f and g built from \otimes and $\#$ over identities, associators, unitors, braidings, $\eta, \nu, \xi, \zeta, \chi$, in such a way that formally $f, g: \mathcal{E}_1 \rightarrow \mathcal{E}_2 \# \mathcal{E}'_2$, then $f = g$ follows from the axioms of guarded parametrized monads.*

Proof Sketch. Let us refer to the described expressions \mathcal{E}_1 as *object expressions* and to f formed as described as *morphism expressions*. For two morphism expressions $f, g: E \rightarrow E'$, let $f \equiv g$ denote ‘ $f = g$ follows from the axioms of guarded parametrized monads’. An object expression is *normal* if it is of the form $\mathcal{E} \# \mathcal{E}'$ and \mathcal{E} and \mathcal{E}' do not contain $\#$. For an object expression \mathcal{E} , we define object expressions $\text{nf}_1(\mathcal{E})$ and $\text{nf}_2(\mathcal{E})$ recursively with the clauses:

- $\text{nf}_1(\mathcal{E}) = \mathcal{E}, \text{nf}_2(\mathcal{E}) = I$ if $\mathcal{E} = I$ or \mathcal{E} is a letter;
- $\text{nf}_1(\mathcal{E} \otimes \mathcal{E}') = \text{nf}_1(\mathcal{E}) \otimes \text{nf}_1(\mathcal{E}'), \text{nf}_2(\mathcal{E} \otimes \mathcal{E}') = \text{nf}_2(\mathcal{E}) \otimes \text{nf}_2(\mathcal{E}')$;
- $\text{nf}_2(\mathcal{E} \# \mathcal{E}') = \text{nf}_1(\mathcal{E}), \text{nf}_2(\mathcal{E} \otimes \mathcal{E}') = \text{nf}_1(\mathcal{E}') \otimes (\text{nf}_2(\mathcal{E}) \otimes \text{nf}_2(\mathcal{E}'))$.

Let $\text{nf}(\mathcal{E}) = \text{nf}_1(\mathcal{E}) \# \text{nf}_2(\mathcal{E})$, so $\text{nf}(\mathcal{E})$ is normal. For any object expression \mathcal{E} we also define a *normalization morphism expression* $\text{nm}(\mathcal{E}): \mathcal{E} \rightarrow \text{nf}(\mathcal{E})$, by induction as follows:

- $\text{nm}(\mathcal{E}) = \eta$ if $\mathcal{E} = I$ or \mathcal{E} is a letter;
- $\text{nm}(\mathcal{E} \otimes \mathcal{E}') = \chi \circ (\text{nm}(\mathcal{E}) \otimes \text{nm}(\mathcal{E}'))$;
- $\text{nm}(\mathcal{E} \# \mathcal{E}') = \xi \circ \zeta \circ (\text{nm}(\mathcal{E}) \# \text{nm}(\mathcal{E}'))$.

The goal will follow from the following subgoals.

1. If a morphism expression $f: \mathcal{E} \rightarrow \mathcal{E}'$ does not contain ν then $\text{nm}(\mathcal{E}') \circ f \equiv \text{nm}(\mathcal{E}) \circ g$ for a suitable isomorphism g , constructed from $\otimes, \#$ and the coherent isomorphisms of the monoidal structure.
2. If a morphism expression $f: \mathcal{E} \rightarrow \mathcal{E}'$ does not contain η, ξ, ζ, χ then there exists $g: \text{nf}(\mathcal{E}) \rightarrow \text{nf}(\mathcal{E}')$ that also does not contain η, ξ, ζ, χ and such that $\text{nm}(\mathcal{E}') \circ f \equiv g \circ \text{nm}(\mathcal{E})$.
3. If $f, g: \mathcal{E} \rightarrow \mathcal{E}'$, \mathcal{E} is a normal object expression and g and f do not contain η, ξ, ζ and χ then $f \equiv g$.

Indeed, given $f, g: \mathcal{E} \rightarrow \mathcal{E}'$ with normal \mathcal{E}' , to prove $f \equiv g$, it suffices to prove that f is equal to $\mathcal{E} \xrightarrow{\text{nm}(\mathcal{E})} \text{nf}(\mathcal{E}) \xrightarrow{f'} \mathcal{E}'$ for a suitable f' – the analogous statement would be true for g , and we would be done by **3**. Let us represent f as a composition $f_n \circ \dots \circ f_1$ where every f_i with even i contains precisely one occurrence of ν and every f_i with odd i contains no occurrences of ν . We obtain

$$\begin{array}{ccccccc}
 \mathcal{E} & \xrightarrow{f_1} & \mathcal{E}_1 & \xrightarrow{f_2} & \mathcal{E}_2 & \cdots & \mathcal{E}_n \\
 \text{nm}(\mathcal{E}) \downarrow & & \text{nm}(\mathcal{E}_1) \downarrow & & \text{nm}(\mathcal{E}_2) \downarrow & & \downarrow \text{nf}(\mathcal{E}_n) \\
 \text{nf}(\mathcal{E}) & \cong & \text{nf}(\mathcal{E}_1) & \longrightarrow & \text{nf}(\mathcal{E}_2) & \cdots & \text{nf}(\mathcal{E}_n)
 \end{array}$$

where every odd diagram commutes by **1** and every even diagram commutes by **2**. Note that $\text{nf}(\mathcal{E}_n)$ is an isomorphism, since $\mathcal{E}_n = \mathcal{E}'$ is normal, and therefore we obtain the desired presentation for f . Let us show the subgoals.

1. W.l.o.g. assume that f contains precisely one letter from the list $\eta, \xi, \zeta, \chi, \rho, \lambda, \alpha, \gamma$ where ρ, λ are the right and left unitors of \otimes , α is the associator and γ is braiding. The general case will follow by induction. Furthermore, by structural induction over \mathcal{E} , we restrict to the situation that $f \in \{\eta, \xi, \zeta, \chi, \rho, \lambda, \alpha, \gamma\}$. The rest follows by case distinction.
2. Again, w.l.o.g. f contains precisely one occurrence of ν . The reduction to $f = \nu$, runs by structural induction over \mathcal{E} and in contrast to the previous clause relies on the diagrams, combining ν with ξ, ζ and χ correspondingly.

3. Observe that f is a composition of morphisms of the form $\beta \circ h \circ \alpha$ where α and β are coherent isomorphisms and h contains one occurrence of v . By induction, using the properties of v we can reduce to the case $f = \alpha \circ h \circ \beta$ and analogously, we can reduce to $g = \alpha' \circ h' \circ \beta'$. It is then easy to see that h and h' are equal up to a coherent isomorphism, and the desired equality $f \equiv g$ follows by coherence for symmetric monoidal categories. \blacktriangleleft

The present version is sufficient for our purposes. It is an open question if a stronger version with general $f, g: \mathcal{E}_1 \rightarrow \mathcal{E}_2$ can be proven. In the sequel, we will only deal with guarded parametrized monads over $(\mathbf{V}, +, 0)$. Recall that a parametrized monad [37] is a bifunctor $T: \mathbf{V} \times \mathbf{V} \rightarrow \mathbf{V}$, such that each $T(-, X)$ is a monad and each $T(-, f)$ is a monad morphism.

► **Proposition 26.** *Every guarded parametrized monad is a parametrized monad with $A \xrightarrow{\eta} A \# 0 \xrightarrow{\text{id} \# !} A \# B$ as unit and $(A \# B) \# B \xrightarrow{\xi} A \# (B + B) \xrightarrow{\text{id} \# \nabla} A \# B$ as multiplication.*

Proof Sketch. The proof essentially follows from coherence. For example, consider the associativity monad law $\mu_{A,B} \circ \mu_{A \# B, B} = \mu_{A,B} \circ (\mu_{A,B} \# B)$. On the one hand (by naturality),

$$\begin{aligned} \mu_{A,B} \circ \mu_{A \# B, B} &= (A \# \nabla) \circ \xi_{A,B,B} \circ ((A \# B) \# \nabla) \circ \xi_{A \# B, B, B} \\ &= (A \# \nabla) \circ (A \# (B + \nabla)) \circ \xi_{A,B,B+B} \circ \xi_{A \# B, B, B} \\ &= (A \# (\nabla \circ (B + \nabla))) \circ \xi_{A,B,B+B} \circ \xi_{A \# B, B, B}. \end{aligned}$$

On the other hand,

$$\begin{aligned} \mu_{A,B} \circ (\mu_{A,B} \# B) &= (A \# \nabla) \circ \xi_{A,B,B} \circ ((A \# \nabla) \circ \xi_{A,B,B} \# B) \\ &= (A \# \nabla) \circ (A \# (\nabla + B)) \circ \xi_{A,B+B,B} \circ (\xi_{A,B,B} \# B) \\ &= (A \# (\nabla \circ (\nabla + B))) \circ \xi_{A,B+B,B} \circ (\xi_{A,B,B} \# B). \end{aligned}$$

By coherence, $\xi_{A,B,B+B} \circ \xi_{A \# B, B, B}$ and $\xi_{A,B+B,B} \circ (\xi_{A,B,B} \# B)$ are equal up to the canonical isomorphism $(B + B) + B \cong B + (B + B)$ and hence the expressions we computed above are equal as well. \blacktriangleleft

► **Theorem 27.** *Given co-Cartesian \mathbf{V} and an identity-on-object functor $J: \mathbf{V} \rightarrow \mathbf{C}$ strictly preserving coproducts, \mathbf{C} is guarded and \mathbf{C}_\bullet is representable iff $\mathbf{C} \cong \mathbf{V}_{-\#0}$ for a guarded parametrized monad $(\#, \eta, v, \chi, \xi, \zeta)$, the compositions $v_{X,Y,0} \circ (\text{id} \# \text{inl})$ are all monic and $f: X \rightarrow Y \rceil Z$ iff f factors through $Y \# (Z + 0) \xrightarrow{v} (Y + Z) \# 0$.*

Vacuous guardedness is clearly representable and by Theorem 27 corresponds to those guarded parametrized monads $\#$, which do not depend on the parameter, i.e. to monads.

► **Example 28.** Let us revisit Example 12. Let $X \# Y = T(X + HT_H(X + Y))$, and note that $-\#0$ is isomorphic to T_H . Assuming existence of some morphism $p: 1 \rightarrow H1$, for every X , we obtain the final map $\hat{p}: 1 \rightarrow T_H X$, induced by the coalgebra map $1 \xrightarrow{\eta \circ \text{inr} \circ p} T(X + H1)$. Now, $T(\text{inl} + \text{id})$ is a section, since $T[\text{id} + H\hat{p} \circ p \circ !, \text{inr}] \circ T(\text{inl} + \text{id})$ is the identity. By Theorem 23, $\#$ is a guarded parametrized monad.

► **Example 29.** Let us revisit Example 14. Let $X \# Y = \mathbb{R}_{\geq 0} \times X + \mathbb{R}_{> 0} \times Y + \bar{\mathbb{R}}_{\geq 0}$. Then $X \# 0 \cong \mathbb{R}_{\geq 0} \times X + \bar{\mathbb{R}}_{\geq 0}$ and there is an obvious injection $\epsilon_{X,Y}$ from $X \# Y$ to $(X + Y) \# 0$. By definition, every guarded $f: X \rightarrow Y \# Z$ uniquely factors through $\epsilon_{Y,Z}$, and hence $\#$ is a guarded parametrized monad.

34:18 Representing Guardedness in Call-By-Value

► **Definition 30** (Strong Guarded Parametrized Monad). *A guarded parametrized monad $(\#, \eta, v, \chi, \xi, \zeta)$ is strong, if $\#$ is strong as a monad in the first argument and as a functor in the second argument, and the diagram*

$$\begin{array}{ccc} X \times (Y \# Z) & \xrightarrow{\text{id} \times \epsilon} & X \times (Y + Z) \# 0 \xrightarrow{\tau} (X \times (Y + Z)) \# 0 \xrightarrow{\text{dist} \# 0} (X \times Y + X \times Z) \# 0 \\ \tau \downarrow & & \downarrow (\text{id} + \text{snd}) \# 0 \\ (X \times Y) \# Z & \xrightarrow{\epsilon} & (X \times Y + Z) \# 0 \end{array}$$

commutes, where $\epsilon_{X,Y} = v_{X,Y,0} \circ (\text{id} \# \text{inl})$ and τ is the monadic strength of $\#$.

► **Remark 31.** Strength is commonly referred to as a “technical condition”. This is justified by the fact that in self-enriched categories strength is equivalent to enrichment of the corresponding functor or a monad [21], and in foundational categories, like **Set**, every functor and every natural transformation are canonically enriched w.r.t. Cartesian closeness as the self-enrichment structure. Then canonical strength $\rho_{X,Y}: X \times FY \rightarrow F(X \times Y)$ for a functor F is defined by the expression $\rho_{X,Y} = \lambda(x, z). F(\lambda y. (x, y))(z)$. We conjecture that strengths involved in Definition 30 are technical in the same sense, in particular the requested commutative diagram is entailed by enrichment of ϵ .

Finally, let us establish the analogue of Theorem 27 for Freyd categories.

► **Theorem 32.** *A Freyd category $(\mathbf{V}, \mathbf{C}, J(-), \mathcal{O})$ is guarded and \mathbf{C}_\bullet is representable iff $\mathbf{C} \cong \mathbf{V}_{-\#0}$ for a strong guarded parametrized monad $(\#, \eta, v, \chi, \xi, \zeta)$, the compositions $v_{X,Y,0} \circ (\text{id} \# \text{inl})$ are all monic and $f: X \rightarrow Y \succ Z$ iff f factors through $Y \# (Z + 0) \xrightarrow{v} (Y + Z) \# 0$.*

For a strong guarded parametrized monad $\#$, let $\tilde{\tau}$ be the composition

$$\begin{aligned} X \times (Y \# Z) & \xrightarrow{\Delta \times \text{id}} (X \times X) \times (Y \# Z) \cong X \times (X \times (Y \# Z)) \\ & \xrightarrow{\text{id} \times \rho} X \times (Y \# (X \times Z)) \xrightarrow{\tau} (X \times Y) \# (X \times Z) \end{aligned}$$

where τ is the monadic strength of $\#$ and ρ is the functorial strength of $\#$. It is easy to check that τ and ρ are derivable from $\tilde{\tau}$, and in the sequel, we will include it as the last element in a tuple $(\#, \eta, v, \chi, \xi, \zeta, \tilde{\tau})$, defining a strong guarded parametrized monad.

Finally, we can interpret the guarded version of FGCBV over a strong guarded parametrized monad $(\#, \eta, v, \chi, \xi, \zeta, \tilde{\tau})$ on \mathbf{V} . Let sorts and function symbols from Σ_v be interpreted as usual and let $\llbracket f \rrbracket \in \mathbf{V}(\llbracket A \rrbracket, \llbracket B \rrbracket \# \llbracket C \rrbracket)$ for $f: A \rightarrow B \succ C \in \Sigma_c$. Then $\llbracket \Gamma \vdash_v v: B \rrbracket \in \mathbf{V}(\llbracket A \rrbracket, \llbracket B \rrbracket)$ and $\llbracket \Gamma \vdash_c p: B \succ C \rrbracket \in \mathbf{V}(\llbracket A \rrbracket, \llbracket B \rrbracket \# \llbracket C \rrbracket)$ are defined with the following clauses:

- $\llbracket x_1: A_1, \dots, x_n: A_n \vdash_v x_i: A_i \rrbracket = \text{inj}_i$;
- $\llbracket \Gamma \vdash_v f(v): B \rrbracket = \llbracket f \rrbracket \circ \llbracket \Gamma \vdash_v v: A \rrbracket$;
- $\llbracket \Gamma \vdash_c f(v): B \rrbracket = \llbracket f \rrbracket \circ \llbracket \Gamma \vdash_v v: A \rrbracket$;
- $\llbracket \Gamma \vdash_c \text{return } v: A \succ B \rrbracket = \eta_{\llbracket A \rrbracket, \llbracket B \rrbracket} \circ \llbracket \Gamma \vdash_v v: A \rrbracket$;
- $\llbracket \Gamma \vdash_c \text{docase } p \text{ of inl } x \mapsto q; \text{ inr } y \mapsto r: C \succ D \rrbracket = (\nabla \# \nabla) \circ \xi_{\llbracket C \rrbracket, \llbracket D \rrbracket, \llbracket C \rrbracket + \llbracket D \rrbracket}$
 $\circ \zeta_{\llbracket C \rrbracket \# \llbracket D \rrbracket, \llbracket C \rrbracket, \llbracket D \rrbracket} \circ (\llbracket \Gamma, x: A \vdash_c q: C \succ D \rrbracket \# \llbracket \Gamma, y: B \vdash_c r: C \succ D \rrbracket)$
 $\circ \tilde{\tau}_{\llbracket \Gamma \rrbracket, \llbracket A \rrbracket, \llbracket B \rrbracket} \circ \langle \text{id}, \llbracket \Gamma \vdash_c p: A \succ B \rrbracket \rangle$;
- $\llbracket \Gamma \vdash_v \text{init } v: A \rrbracket = !$;
- $\llbracket \Gamma \vdash_v \text{inl } v: A + B \rrbracket = \text{inl} \circ \llbracket \Gamma \vdash_v v: A \rrbracket$;
- $\llbracket \Gamma \vdash_v \text{inr } v: A + B \rrbracket = \text{inr} \circ \llbracket \Gamma \vdash_v v: B \rrbracket$;
- $\llbracket \Gamma \vdash_c \text{case } v \text{ of inl } x \mapsto p; \text{ inr } y \mapsto q: C \succ D \rrbracket = (\nabla \# \nabla) \circ \chi_{\llbracket C \rrbracket, \llbracket D \rrbracket, \llbracket C \rrbracket, \llbracket D \rrbracket}$
 $\circ (\llbracket \Gamma, x: A \vdash_c p: C \succ D \rrbracket + \llbracket \Gamma, y: B \vdash_c q: C \succ D \rrbracket) \circ \text{dist} \circ \langle \text{id}, \llbracket \Gamma \vdash_v v: A + B \rrbracket \rangle$.
- $\llbracket \Gamma \vdash_v \langle v, w \rangle: A \times B \rrbracket = \langle \llbracket \Gamma \vdash_v v: A \rrbracket, \llbracket \Gamma \vdash_v w: B \rrbracket \rangle$.

Note that what allows us to sidestep the monicity condition of the representability criterion (Theorem 27) is that we gave up on the assumption that the space of guarded morphisms $X \rightarrow Y \# Z$ injectively embeds to the space of all morphisms $X \rightarrow (Y + Z) \# 0$, in particular, the entire notion of guardedness predicate is eliminated.

8 Conclusions and Further Work

We investigated a combination of FGCBV and guardedness by taking inspiration from the previous work on relating Freyd categories with strong monads via a natural requirement of representability of certain presheaves. An abstract notion of guardedness naturally fits the FGCBV paradigm and gives rise to more general formats of presheaves, which must be representable e.g. in order to be able to interpret higher-order (guarded) functions. In our case, the representability requirement gave rise to a novel categorical structure, we dub (strong) guarded parametrized monad, which encapsulate computational effects under consideration, simultaneously with guardedness guarantees.

We regard our present results as a prerequisite step for implementing guarded programs in existing higher-order languages, such as Haskell, and in proof assistants with strict support of the propositions-as-types discipline, such as Coq and Agda, where unproductive recursive definitions cannot be implemented directly, and thus the importance of guarded iteration is particularly high. It would be interesting to further refine guarded parametrized monads so as to include further quantitative information on how productive a computation is, or how unproductive it is, so that this relative unproductivity could possibly be cancelled out by composition with something very productive. Another strand for future work comes from the observation that guarded iteration is a formal dual of guarded recursion [18]. A good deal of the present theory can be easily dualized, which will presumably lead to guarded parametrized comonads and comonadic recursion – we are planning to investigate these structures from the perspective of comonadic notion of computation [38]. In terms of syntax, a natural extension of fine-grain call-by-value is call-by-push-value [22]. We expect it to be a natural environment for analysing the above mentioned aspects in the style of the presented approach.

References

- 1 Jiří Adámek, Horst Herrlich, and George Strecker. *Abstract and concrete categories*. John Wiley & Sons Inc., New York, 1990.
- 2 Jiří Adámek, Stefan Milius, and Jiří Velebil. On rational monads and free iterative theories. In *Proc. Category Theory and Computer Science (CTCS'02)*, volume 69 of *Electron. Notes Theor. Comput. Sci.*, pages 23–46, 2002.
- 3 Jiří Adámek, Stefan Milius, and Jiří Velebil. Equational properties of iterative monads. *Inf. Comput.*, 208(12):1306–1348, 2010.
- 4 Thorsten Altenkirch, Nils Danielsson, and Nicolai Kraus. Partiality, revisited - the partiality monad as a quotient inductive-inductive type. In Javier Esparza and Andrzej Murawski, editors, *Foundations of Software Science and Computation Structures, FOSSACS 2017*, volume 10203 of *LNCS*, pages 534–549, 2017.
- 5 Steve Awodey. *Category Theory*. Oxford University Press, Inc., New York, NY, USA, 2nd edition, 2010.
- 6 J. Bergstra, A. Ponse, and Scott Smolka, editors. *Handbook of Process Algebra*. Elsevier, 2001.
- 7 Stephen Bloom and Zoltán Ésik. *Iteration theories: the equational logic of iterative processes*. Springer, 1993.

- 8 Venanzio Capretta. General recursion via coinductive types. *Log. Meth. Comput. Sci.*, 1(2), 2005.
- 9 James Chapman, Tarmo Uustalu, and Niccolò Veltri. Quotienting the delay monad by weak bisimilarity. In *Theoretical Aspects of Computing, ICTAC 2015*, volume 9399 of *LNCS*, pages 110–125. Springer, 2015.
- 10 J. Robin B. Cockett. Introduction to distributive categories. *Mathematical Structures in Computer Science*, 3(3):277–307, 1993.
- 11 Martín H. Escardó and Cory M. Knapp. Partial Elements and Recursion via Dominances in Univalent Type Theory. In Valentin Goranko and Mads Dam, editors, *26th EACSL Annual Conference on Computer Science Logic (CSL 2017)*, volume 82 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 21:1–21:16, Dagstuhl, Germany, 2017. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.
- 12 M.P. Fiore. *Axiomatic Domain Theory in Categories of Partial Maps*. Distinguished Dissertations in Computer Science. Cambridge University Press, 2004.
- 13 Bram Geron and Paul Blain Levy. Iteration and labelled iteration. In *Proc. 32nd Conference on the Mathematical Foundations of Programming Semantics (MFPS 2016)*, volume 325 of *ENTCS*, pages 127–146. Elsevier, 2016.
- 14 Sergey Goncharov. Uniform Elgot Iteration in Foundations. In Nikhil Bansal, Emanuela Merelli, and James Worrell, editors, *48th International Colloquium on Automata, Languages, and Programming (ICALP 2021)*, volume 198 of *LIPIcs*, pages 131:1–131:16. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2021.
- 15 Sergey Goncharov, Renato Neves, and José Proença. Implementing hybrid semantics: From functional to imperative. In Volker Stolz Violet Ka I Pun, Adenildo da Silva Simão, editor, *17th International Colloquium on Theoretical Aspects of Computing (ICTAC 2020)*, 2020.
- 16 Sergey Goncharov, Christoph Rauch, and Lutz Schröder. Unguarded recursion on coinductive resumptions. In *Mathematical Foundations of Programming Semantics, MFPS 2015*, volume 319 of *ENTCS*, pages 183–198. Elsevier, 2015.
- 17 Sergey Goncharov, Christoph Rauch, and Lutz Schröder. A metalanguage for guarded iteration. *Theoretical Computer Science*, 880:111–137, 2021.
- 18 Sergey Goncharov and Lutz Schröder. Guarded traced categories. In Christel Baier and Ugo Dal Lago, editors, *Proc. 21th International Conference on Foundations of Software Science and Computation Structures (FoSSaCS 2018)*, LNCS. Springer, 2018.
- 19 Sergey Goncharov, Lutz Schröder, Christoph Rauch, and Maciej Piróg. Unifying guarded and unguarded iteration. In Javier Esparza and Andrzej Murawski, editors, *Foundations of Software Science and Computation Structures, FoSSaCS 2017*, volume 10203 of *LNCS*, pages 517–533. Springer, 2017.
- 20 George Janelidze and Gregory M Kelly. A note on actions of a monoidal category. *Theory Appl. Categ.*, 9(61-91):02, 2001.
- 21 Anders Kock. Strong functors and monoidal monads. *Archiv der Mathematik*, 23(1):113–120, 1972.
- 22 Paul Blain Levy. Call-by-push-value: A subsuming paradigm. In Jean-Yves Girard, editor, *TLCA*, volume 1581 of *Lecture Notes in Computer Science*, pages 228–242. Springer, 1999.
- 23 Paul Blain Levy. *Call-By-Push-Value: A Functional/Imperative Synthesis (Semantics Structures in Computation, V. 2)*. Kluwer Academic Publishers, USA, 2004.
- 24 Paul Blain Levy and Sergey Goncharov. Coinductive resumption monads: Guarded iterative and guarded elgot. In *Proc. 8rd international conference on Algebra and coalgebra in computer science (CALCO 2019)*, LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019.
- 25 Paul Blain Levy, John Power, and Hayo Thielecke. Modelling environments in call-by-value programming languages. *Inf. & Comp.*, 185:2003, 2002.
- 26 Saunders Mac Lane. *Categories for the Working Mathematician*. Springer, 1971.
- 27 R. Milner. *Communication and concurrency*. Prentice-Hall, 1989.

- 28 Eugenio Moggi. A modular approach to denotational semantics. In *Category Theory and Computer Science, CTCS 1991*, volume 530 of *LNCS*, pages 138–139. Springer, 1991.
- 29 Keiko Nakata and Tarmo Uustalu. A Hoare logic for the coinductive trace-based big-step semantics of While. *Log. Meth. Comput. Sci.*, 11(1), 2015.
- 30 Maciej Piróg and Jeremy Gibbons. The coinductive resumption monad. In *Mathematical Foundations of Programming Semantics, MFPS 2014*, volume 308 of *ENTCS*, pages 273–288, 2014.
- 31 Gordon Plotkin and John Power. Adequacy for algebraic effects. In *FoSSaCS'01*, volume 2030 of *LNCS*, pages 1–24, 2001.
- 32 A. J. Power and E. P. Robinson. Premonoidal categories and notions of computation. *Mathematical Structures in Computer Science*, 7(5):453–468, October 1997.
- 33 A. John Power and Hayo Thielecke. Closed Freyd- and kappa-categories. In *Proceedings of the 26th International Colloquium on Automata, Languages and Programming, ICAL '99*, pages 625–634, Berlin, Heidelberg, 1999. Springer-Verlag.
- 34 Dietmar Schumacher. Minimale und maximale tripelerzeugende und eine bemerkung zur tripelbarkeit. *Archiv der Mathematik*, 20(4):356–364, September 1969.
- 35 Alex Simpson and Gordon Plotkin. Complete axioms for categorical fixed-point operators. In *Logic in Computer Science, LICS 2000*, pages 30–41, 2000.
- 36 Sam Staton. Freyd categories are enriched Lawvere theories. *Electron. Notes Theor. Comput. Sci.*, 303:197–206, March 2014.
- 37 Tarmo Uustalu. Generalizing substitution. *ITA*, 37(4):315–336, 2003.
- 38 Tarmo Uustalu and Varmo Vene. Comonadic notions of computation. *Electron. Notes Theor. Comput. Sci.*, 203(5):263–284, 2008.