Unifying Guarded and Unguarded Iteration

Sergey Goncharov[1(✉)], Lutz Schröder[1], Christoph Rauch[1], and Maciej Piróg[2]

[1] Friedrich-Alexander-Universität Erlangen-Nürnberg, Erlangen, Germany
sergey.goncharov@fau.de
[2] Department of Computer Science, KU Leuven, Leuven, Belgium

Abstract. Models of iterated computation, such as (completely) iterative monads, often depend on a notion of guardedness, which guarantees unique solvability of recursive equations and requires roughly that recursive calls happen only under certain guarding operations. On the other hand, many models of iteration do admit unguarded iteration. Solutions are then no longer unique, and in general not even determined as least or greatest fixpoints, being instead governed by quasi-equational axioms. Monads that support unguarded iteration in this sense are called (complete) Elgot monads. Here, we propose to equip monads with an abstract notion of guardedness and then require solvability of abstractly guarded recursive equations; examples of such *abstractly guarded pre-iterative monads* include both iterative monads and Elgot monads, the latter by deeming any recursive definition to be abstractly guarded. Our main result is then that Elgot monads are precisely the iteration-congruent retracts of abstractly guarded *iterative* monads, the latter being defined as admitting *unique* solutions of abstractly guarded recursive equations; in other words, models of unguarded iteration come about by quotienting models of guarded iteration.

1 Introduction

In recursion theory, notions of guardedness traditionally play a central role. Guardedness typically means that recursive calls must be in the scope of certain guarding operations, a condition aimed, among other things, at ensuring progress. The paradigmatic case are recursive definitions in process algebra, which are usually called guarded if recursive calls occur only under action prefixing [6]. A more abstract example are completely iterative theories [11] and monads [19], where, in the latter setting, a recursive definition is guarded if it factors through a given ideal of the monad. Guarded recursive definitions typically have unique solutions; e.g. the unique solution of the guarded recursive definition

$$X = a.\, X$$

is the process that keeps performing the action a.

For unguarded recursive definitions, the picture is, of course, different. E.g. to obtain the denotational semantics of an unproductive while loop $\mathsf{while\ true\ do\ skip}$ characterized by circular operational behavior

$$\mathsf{while\ true\ do\ skip} \to \mathsf{skip}; \mathsf{while\ true\ do\ skip} \to \mathsf{while\ true\ do\ skip}$$

J. Esparza and A.S. Murawski (Eds.): FOSSACS 2017, LNCS 10203, pp. 517–533, 2017.
DOI: 10.1007/978-3-662-54458-7_30

one will select one of many solutions of this trivial equation, e.g. the least solution in a domain-theoretic semantics.

Sometimes, however, one has a selection among non-unique solutions of unguarded recursive equations that is *not* determined order-theoretically, i.e. by picking least or greatest fixpoints. One example arises from *coinductive resumptions* [16,25,26]. In the paradigm of monad-based encapsulation of side-effects [21], coinductive resumptions over a base effect encapsulated by a monad T form a *coinductive resumption transform* monad T^ν given by

$$T^\nu X = \nu\gamma.\, T(X + \gamma) \tag{1}$$

– that is, a computation over X performs a step with effects from T, and then returns either a value from X or a resumption that, when resumed, proceeds similarly, possibly ad infinitum. We generally restrict to monads T for which (1) exists for all X (although many of our results do not depend on this assumption). Functors (or monads) T for which this holds are called *iteratable* [1]. Most computationally relevant monads are iteratable (notable exceptions in the category of sets are the powerset monad and the continuation monad). Notice that one has a natural *delay map* $T^\nu X \to T^\nu X$ that converts a computation into a resumption, i.e. prefixes it with a delay step. In fact, for $T = \mathsf{id}$, T^ν is precisely Capretta's *partiality monad* [8], also called the *delay monad*. It is not in general possible to equip $T^\nu X$ with a domain structure that would allow for selecting least or greatest solutions of unguarded recursive definitions over T^ν. However, one *can* select solutions in a coherent way, that is, such that a range of natural quasi-equational axioms is satisfied, making T^ν into a (complete) *Elgot monad* [2,15].

In the current work we aim to unify theories of guarded and unguarded iteration. To this end, we introduce a notion of *abstractly guarded monads*, that is, monads T equipped with a distinguished class of *abstractly guarded* equation morphisms satisfying some natural closure properties (Sect. 3). The notion of abstract guardedness can be instantiated in various ways, e.g. with the class of immediately terminating 'recursive' definitions, with the class of guarded morphisms in a completely iterative monad, or with the class of all equation morphisms. We call an abstractly guarded monad *pre-iterative* if all abstractly guarded equation morphisms have a solution, and *iterative* if these solutions are unique. Then completely iterative monads are iterative abstractly guarded monads in this sense, and (complete) Elgot monads are pre-iterative, where we deem every equation morphism to be abstractly guarded in the latter case.

The quasi-equational axioms of Elgot monads are easily seen to be satisfied when fixpoints are unique, i.e. in iterative abstractly guarded monads, and moreover stable under iteration-congruent retractions in a fairly obvious sense. Our first main result (Sect. 5, Theorem 22) states that the converse holds as well, i.e. *a monad T is a complete Elgot monad iff T is an iteration-congruent retract of an iterative abstractly guarded monad* – specifically of T^ν as in (1). As a slogan,

> monad-based models of unguarded recursion arise by quotienting models of guarded recursion.

Our second main result (Theorem 26) is an algebraic characterization of complete Elgot monads: We show that the construction $(-)^\nu$ mapping a monad T to T^ν as in (1) is a monad on the category of monads (modulo existence of T^ν), and *complete Elgot monads are precisely those* $(-)^\nu$*-algebras* T *that cancel the delay map on* T^ν, i.e. interpret the delay operation as identity.

As an illustration of these results, we show (Sect. 6) that sandwiching a complete Elgot monad between adjoint functors again yields a complete Elgot monad, in analogy to a corresponding result for completely iterative monads [26]. Specifically, we prove a sandwich theorem for iterative abstractly guarded monads and transfer it to complete Elgot monads using our first main result. For illustration, we then relate iteration in ultrametric spaces using Escardó's metric lifting monad [12] to iteration in pointed cpo's, by noting that the corresponding monads on sets obtained using our sandwich theorems are related by an iteration-congruent retraction in the sense of our first main result.

2 Preliminaries

We work in a category $\mathbf{C}$ with finite coproducts. We fix the notation $\mathsf{in}_i : X_i \to X_1 + \ldots + X_n$ for the i-th injection. A morphism $\sigma : Y \to X$ is a *summand* of X, which we denote $\sigma : Y \rightarrowtail X$, if there is $\sigma' : X' \to X$ such that X is a coproduct of Y and X' with σ and σ' being coproduct injections. The morphism σ' is called a *(coproduct) complement* of σ and by definition is also a summand. We are not assuming that $\mathbf{C}$ is *extensive*, and coproduct complements are not in general uniquely determined.

A *monad* $\mathbb{T}$ over $\mathbf{C}$ can be given in a form of a *Kleisli triple* $(T, \eta, -^\star)$ where T is an endomap over the objects $|\mathbf{C}|$ of $\mathbf{C}$, the unit η is a family of morphisms $(\eta_X : X \to TX)_{X \in |\mathbf{C}|}$, *Kleisli lifting* $(-)^\star$ is a family of maps : $\mathsf{Hom}(X, TY) \to \mathsf{Hom}(TX, TY)$, and the *monad laws* are satisfied:

$$\eta^\star = \mathsf{id}, \qquad f^\star \eta = f, \qquad (f^\star g)^\star = f^\star g^\star.$$

The standard (equivalent) categorical definition [18] of $\mathbb{T}$ as an endofunctor with natural transformation *unit* $\eta : \mathrm{Id} \to T$ and *multiplication* $\mu : TT \to T$ can be recovered by taking $Tf = (\eta\, f)^\star$, $\mu = \mathsf{id}^\star$. (We adopt the convention that monads and their functor parts are denoted by the same letter, with the former in blackboard bold.) We call morphisms $X \to TY$ *Kleisli morphisms* and view them as a high level abstraction of sequential programs with $\mathbb{T}$ encapsulating the underlying computational effect as proposed by Moggi [22], with X representing the input type and Y the output type. A more traditional use of monads in semantics is due to Lawvere [17], who identified finitary monads on **Set** with *algebraic theories*, hence objects TX can be viewed as sets of terms of the theory over free variables from X, the unit as the operation of casting a variable to a term, and Kleisli composition as substitution. We informally refer to this use of monads as *algebraic monads*.

A(n F*-)coalgebra* is a pair $(X, f : X \to FX)$ where $X \in |\mathbf{C}|$ and $F : \mathbf{C} \to \mathbf{C}$ is an endofunctor. Coalgebras form a category, with morphisms $(X, f) \to (Y, g)$

being $\mathbf{C}$-morphisms $h : X \to Y$ such that $(Fh)\, f = gh$. A final object of this category is called a *final coalgebra*, and we denote it by $(\nu F, \mathsf{out} : \nu F \to F\nu F)$ if it exists. For brevity, *we will be cavalier about existence of final coalgebras and silently assume they exist when we need them*; that is, we hide sanity conditions on the involved functors, such as accessibility. By definition, νF comes with *coiteration* as a definition principle (dual to the iteration principle for algebras): given a coalgebra $(X, f : X \to FX)$ there is a unique morphism $(\mathsf{coit}\ f) : X \to \nu F$ such that

$$\mathsf{out}\,(\mathsf{coit}\ f) = F(\mathsf{coit}\ f)\, f.$$

This implies that out is an isomorphism (*Lambek's lemma*) and that $\mathsf{coit}\ \mathsf{out} = \mathsf{id}$ (see [29] for more details about coalgebras for coiteration).

We generally drop sub- and superscripts, e.g. on natural transformations, whenever this improves readability.

3 Guarded Monads

The notion of guardedness is paramount in process algebra: typically one considers systems of mutually recursive process definitions of the form $x_i = t_i$, and a variable x_i is said to be guarded in t_j if it occurs in t_j only in subterms of the form $a.s$ where $a.(\text{–})$ is action prefixing. A standard categorical approach is to replace the set of terms over variables X by an object TX where $\mathbb{T}$ is a monad. We then can model separate variables by partitioning X into a sum $X_1 + \ldots + X_n$ and thus talk about guardedness of a morphism $f : X \to T(X_1 + \ldots + X_n)$ in any X_i, meaning that every variable from X_i is guarded in f. Since Kleisli morphisms can be thought of as abstract programs we can therefore speak about guardedness of a program in a certain portion of the output type, e.g. $X_i \hookrightarrow X_1 + \ldots + X_n$. One way to capture guardedness categorically is to identify the operations of $\mathbb{T}$ that serve as guards by distinguishing a suitable subobject of $\mathbb{T}$; e.g. the definition of completely iterative monad [19] follows this approach. For our purposes, we require a yet more general notion where we just distinguish some Kleisli morphisms as being guarded in certain output variables. This is formalized as follows.

Definition 1 ((Abstractly) guarded monad). We call a monad $\mathbb{T}$ *(abstractly) guarded* if it is equipped with a notion of (abstract) guardedness, i.e. with a relation between morphisms $f : X \to TY$ and summands of Y (by putting the word 'abstract' in brackets we mean that we will often omit it later). We call $f : X \to TY$ *(abstractly) σ-guarded* if f and $\sigma : Y' \hookrightarrow Y$ are in this relation, and then write $f : X \to_\sigma TY$. Abstract guardedness is required to be closed under the rules in Fig. 1. In rule **(wkn)**, σ and θ are composable summands.

Given guarded monads $\mathbb{T}$, $\mathbb{S}$, a monad morphism $\alpha : T \to S$ is *(abstractly) guarded* if whenever $f : X \to_\sigma TY$ then $\alpha f : X \to_\sigma SY$.

Intuitively, **(trv)** says that if a program does not output anything via a summand of the output type then it is guarded in that summand. Rule **(wkn)** is a

$$\textbf{(trv)}\quad \frac{f : X \to TY}{(T\,\mathsf{in}_1)\,f : X \to_{\mathsf{in}_2} T(Y+Z)} \qquad \textbf{(wkn)}\quad \frac{f : X \to_\sigma TY}{f : X \to_{\sigma\theta} TY}$$

$$\textbf{(cmp)}\quad \frac{f : X \to_{\mathsf{in}_2} T(Y+Z) \qquad g : Y \to_\sigma TV \qquad h : Z \to TV}{[g,h]^\star f : X \to_\sigma TV}$$

$$\textbf{(sum)}\quad \frac{f : X \to_\sigma TZ \qquad g : Y \to_\sigma TZ}{[f,g] : X + Y \to_\sigma TZ}$$

Fig. 1. Axioms of guardedness.

weakening principle: if a program is guarded in some summand then it is guarded in any subsummand of that summand. Rule **(cmp)** asserts that guardedness is preserved by composition: if the unguarded part of the output of a program is postcomposed with a σ-guarded program then the result is σ-guarded, no matter how the guarded part is transformed. Finally, rule **(sum)** says that putting two guarded equation systems side by side again produces a guarded system. The rules are designed so as to enable a reformulation of the classical laws of iteration w.r.t. abstract guardedness, as we shall see in Sect. 5.

We write $f : X \to_{i_1,\ldots,i_k} T(X_1 + \ldots + X_n)$ as a shorthand for $f : X \to_\sigma T(X_1 + \ldots + X_n)$ with $\sigma = [\mathsf{in}_{i_1}, \ldots, \mathsf{in}_{i_k}] : X_{i_1} + \ldots + X_{i_k} \hookrightarrow X_1 + \ldots + X_n$. More generally, we sometimes need to refer to components of some X_{i_j}. This amounts to replacing the corresponding i_j with a sequence of pairs $i_j n_{j,m}$, and in_{i_j} with $\mathsf{in}_{i_j}[\mathsf{in}_{n_{j,1}}, \ldots, \mathsf{in}_{n_{j,k_j}}]$, so, e.g. we write $f : X \to_{12,2} T((Y+Z)+Z)$ to mean that f is $[\mathsf{in}_1\,\mathsf{in}_2, \mathsf{in}_2]$-guarded. Where coproducts $Y + Z$ etc. appear in the rules, we mean any coproduct, not just some selected coproduct. We defined the notion of guardedness as a certain relation over Kleisli morphisms and summands. Clearly, the largest such relation is the one declaring all Kleisli morphisms to be σ-guarded for all σ. We call monads equipped with this notion of guardedness *totally guarded*. It turns out that for every monad we also have a least guardedness relation.

Definition 2. Let $\mathbb{T}$ be a monad. A morphism $f : X \to TY$ is *trivially σ-guarded* for $\sigma : Z \hookrightarrow Y$ if f factors through $T\sigma'$ for a coproduct complement σ' of σ.

Proposition 3. *Let $\mathbb{T}$ be a monad. Then taking the abstractly guarded morphisms to be the trivially guarded morphisms is the least guardedness relation making $\mathbb{T}$ into a guarded monad.*

We call a guarded monad *trivially guarded* if all its abstractly guarded morphisms are trivially guarded. As we see, the notion of abstract guardedness can vary on a large spectrum from *trivial guardedness* to *total guardedness*, thus somewhat detaching abstract guardedness from the original intuition. It is for this reason that we introduced the qualifier *abstract* into the terminology; for brevity, we will omit this qualifier in the sequel in contexts where no confusion is likely, speaking only of guarded monads, guarded morphisms etc.

Remark 4. Although by **(wkn)**, $f : X \to_{1,2} T(Y + Z)$ implies both $f : X \to_1 T(Y + Z)$ and $f : X \to_2 T(Y + Z)$, the converse is not required to be true, and in fact can fail even for trivial guardedness. This is witnessed by the following simple counterexample. Let $\mathbb{T}$ be the algebraic monad given by taking TX to be the free commutative semigroup over X satisfying the additional law $x + y = x$. Now the term $x + y \in T(X + Y)$ (seen as a morphism $1 \to T(X + Y)$) with $x \in X$ and $y \in Y$ is both in_1-guarded and in_2-guarded, being equivalent both to y and to x. But it is not id-guarded, because it does not factor through $T\emptyset = \emptyset$.

As usual, guardedness serves to identify systems of equations that admit solutions according to some global principle:

Definition 5 (Guarded (pre-)iterative monad). Given $f : X \to_2 T(Y+X)$, we say that $f^\dagger : X \to TY$ is a *solution* of f if $f^\dagger$ satisfies the *fixpoint identity* $f^\dagger = [\eta, f^\dagger]^\star f$. A guarded monad $\mathbb{T}$ is *guarded pre-iterative* if it is equipped with an *iteration operator* that assigns to every in_2-guarded morphism $f : X \to_2 T(Y + X)$ a solution $f^\dagger$ of f. If every such f has a unique solution, we call $\mathbb{T}$ *guarded iterative*.

We can readily check that the iteration operator preserves guardedness.

Proposition 6. *Let $\mathbb{T}$ be a guarded pre-iterative monad, let $\sigma : Y' \hookrightarrow Y$, and let $f : X \to_{\sigma+\mathsf{id}} T(Y + X)$. Then $f^\dagger : X \to_\sigma TY$.*

In trivially guarded monads, there is effectively nothing to iterate, so we have

Proposition 7. *Every trivially guarded monad is guarded iterative.*

Guardedness in Completely Iterative Monads. One instance of our notion of abstract guardedness is found in completely iterative monads [19], which are based on idealised monads. To make this precise, we need to recall some definitions. First, a *module* over a monad $\mathbb{T}$ on $\mathbf{C}$ is a pair $(M, -^\circ)$, where M is an endomap over the objects of $\mathbf{C}$, while the lifting $-^\circ$ is a map $\mathsf{Hom}(X, TY) \to \mathsf{Hom}(MX, MY)$ such that the following laws are satisfied:

$$\eta^\circ = \mathsf{id}, \qquad\qquad g^\circ f^\circ = (g^\star f)^\circ.$$

Note that M extends to an endofunctor by taking $Mf = (\eta f)^\circ$. Next, a *module-to-monad morphism* is a natural transformation $\xi : M \to T$ that satisfies $\xi f^\circ = f^\star \xi$. We call the triple $(\mathbb{T}, M, -^\circ, \xi)$ an *idealised monad*; when no confusion is likely, we refer to these data just as $\mathbb{T}$. Following [19], we can then define guardedness as follows:

Definition 8. Given an idealised monad $\mathbb{T}$ as above, a morphism $f : X \to T(Y + X)$ is *guarded* if it factors via $[\eta\,\mathsf{in}_1, \xi] : Y + M(Y + X) \to T(Y + X)$. Then, $\mathbb{T}$ is a *completely iterative monad* if every such guarded f has a unique solution.

It turns out that this notion of guardedness is not an instance of our notion of abstract guardedness. Fortunately, we can fix this by noticing that completely iterative monads actually support iteration for a wider class of morphisms:

Definition 9. Let $(\mathbb{T}, M, -^\circ, \xi)$ be an idealised monad. Given $\sigma : Z \hookrightarrow Y$, we say that a morphism $f : X \to TY$ is *weakly σ-guarded* if it factors through $[\eta\sigma', \xi]^\star : T(Y' + MY) \to TY$ for a complement $\sigma' : Y' \hookrightarrow Y$ of σ.

Since a morphism that factors as $[\eta\,\mathsf{in}_1, \xi]f$ can be rewritten as $[\eta\,\mathsf{in}_1, \xi]^\star \eta f$, every guarded morphism in an idealised monad is also weakly guarded.

Theorem 10. *Let $(\mathbb{T}, M, -^\circ, \xi)$ be an idealised monad. Then the following hold.*

1. *$\mathbb{T}$ becomes abstractly guarded when equipped with weak guardedness as the notion of abstract guardedness.*
2. *If $\mathbb{T}$ is completely iterative, then every weakly in_2-guarded morphism $f : X \to T(Y + X)$ has a unique solution.*

That is, completely iterative monads are abstractly guarded iterative monads w.r.t. weak guardedness.

4 Parametrizing Guardedness

Uustalu [28] defines a *parametrized monad* to be a functor from a category $\mathbf{C}$ to the category of monads over $\mathbf{C}$. We need a minor adaptation of this notion where we allow parameters from a different category than $\mathbf{C}$, and simultaneously introduce a guarded version of parametrized monads:

Definition 11 (Parametrized guarded monad). A *parametrized (guarded) monad* is a functor from a category $\mathbf{D}$ to the category of (guarded) monads and (guarded) monad morphisms over $\mathbf{C}$. Alternatively (by uncurrying), it is a bifunctor $\# : \mathbf{C} \times \mathbf{D} \to \mathbf{C}$ such that for any $X \in |\mathbf{D}|$, $- \# X : \mathbf{C} \to \mathbf{C}$ is a (guarded) monad, and for every $f : X \to Y$, $\mathsf{id} \# f : Z \# X \to Z \# Y$ is the Z-component of a (guarded) monad morphism $- \# f : - \# X \to - \# Y$.

A *parametrized (guarded) monad morphism* between guarded monads qua functors into the category of monads over $\mathbf{C}$ is a natural transformation that is componentwise a monad morphism. In uncurried notation, given parametrized monads $\#, \hat{\#} : \mathbf{C} \times \mathbf{D} \to \mathbf{C}$ a natural transformation $\alpha : \# \to \hat{\#}$ is a parametrized (guarded) monad morphism if for each $X \in |\mathbf{D}|$, $\alpha_{-,X} : - \# X \to - \hat{\#} X$ is a (guarded) monad morphism.

Guardedness of the monad morphisms $- \# f$ means explicitly that $g : Z \to_\sigma V \# X$ implies $(\mathsf{id} \# f)\, g : Z \to_\sigma V \# Y$.

Example 12. For the purposes of the present work, the most important example (taken from [28]) is $\# = T(- + \Sigma -) : \mathbf{C} \times \mathbf{C} \to \mathbf{C}$ where $\mathbb{T}$ is a (non-parametrized) monad on $\mathbf{C}$ and Σ is an endofunctor on $\mathbf{C}$. Informally, $\mathbb{T}$ captures a computational effect, e.g. nondeterminism for T being powerset, and Σ

captures a signature of actions, e.g. $\Sigma X = A \times X$, as in process algebra. Specifically, taking $A = 1$ we obtain $X \mathbin{\#} Y = T(X + Y)$; in this case, we have only one guard, which can be interpreted as a delay. The second argument of $\#$ can thus be thought of as designated for guarded recursion.

Incidentally, our modification of parametrized monads also covers Atkey's parametrized monads [5], which are certain functors $\mathbf{S} \times \mathbf{S}^{op} \times \mathbf{C} \to \mathbf{C}$ forming a monad in the third argument. The first and the second arguments serve, e.g., to parametrize the computational effect of interest with initial and final states of different types.

Theorem 13. *Let $\# : \mathbf{C} \times (\mathbf{C} \times \mathbf{D}) \to \mathbf{C}$ be a parametrized monad, with unit η and Kleisli lifting $(-)^\star$. Then*

$$X \mathbin{\#^\nu} Y = \nu\gamma.\, X \mathbin{\#} (\gamma, Y)$$

defines a parametrized monad $\#^\nu : \mathbf{C} \times \mathbf{D} \to \mathbf{C}$ whose unit and Kleisli lifting we denote η^ν and $-^{\bigstar}$, respectively. Moreover,

1. *If $\#$ is guarded then so is $\#^\nu$ with guardedness defined as follows: given $\sigma : Y' \hookrightarrow Y$, $f : X \to Y \mathbin{\#^\nu} Z$ is σ-guarded if $\mathsf{out} f : X \to Y \mathbin{\#} (Y \mathbin{\#^\nu} Z, Z)$ is σ-guarded.*
2. *If $\#$ is pre-iterative under an iteration operator $-^\dagger$ then so is $\#^\nu$ with the iteration operator $-^\ddagger$ defined as follows:*

$$\big(f : X \to_2 (Y + X) \mathbin{\#^\nu} Z\big)^\ddagger = \mathsf{coit}\ \big([\eta, (\mathsf{out} f)^\dagger]^\star \mathsf{out}\big)\ \eta^\nu\ \mathsf{in}_2$$

3. *If $\#$ is iterative then so is $\#^\nu$ under the definition from the previous clause.*

Example 14. We spell out one instance of Theorem 13 in case $\mathbf{D} = 1$ and $\# = T(- + -)$ where $\mathbb{T} = (T, \eta, -^\star)$ is a monad. Then $\#^\nu$ is isomorphically a monad $\mathbb{T}^\nu$ on $\mathbf{C}$ with $T^\nu X = \nu\gamma.\, T(X + \gamma)$, unit $\eta^\nu = \eta\, \mathsf{in}_1$ and Kleisli star $(f : X \to T^\nu Y)^{\bigstar}$ being uniquely determined by the equation

$$\mathsf{out} f^{\bigstar} = [\mathsf{out} f,\ \eta\, \mathsf{in}_2 f^{\bigstar}]^\star \mathsf{out}.$$

If $\mathbb{T}$ is pre-iterative then so is $\mathbb{T}^\nu$ with iteration

$$(f : X \to T^\nu(Y + X))^\ddagger = \mathsf{coit}\ \big([\eta\, \mathsf{in}_2, (T[\mathsf{in}_1 + \mathsf{id},\ \mathsf{in}_1\, \mathsf{in}_2] \mathsf{out}\, f)^\dagger]^\star \mathsf{out}\big)\ \eta^\nu\ \mathsf{in}_2.$$

Example 15. Theorem 13 shows that our notion of guardedness extends along the applications of the final coalgebra transformer $\# \mapsto \#^\nu$ on parametrized monads. This can be used to capture existing notions of guardedness as follows. Consider $X \mathbin{\#} Y = T(X + (1 + A) \times Y)$ where $\mathbb{T}$ is some monad. In $\mathbf{Set}$, $1 + A$ can be thought of as consisting of a set A of visible actions and a silent action τ. In process algebra we standardly consider a process definition to be guarded if every recursive call is preceded by a visible action from A. In our framework this can be reconstructed as follows. The obvious isomorphism

$$\nu\gamma.\, T(X + (1 + A) \times \gamma) \cong \nu\gamma'.\, \nu\gamma.\, T(X + \gamma + A \times \gamma')$$

involves two more parametrized monads: $T(- + - + A \times -) : \mathbf{C} \times \mathbf{C} \times \mathbf{C} \to \mathbf{C}$ and $\nu\gamma.\, T(- + \gamma + A \times -) : \mathbf{C} \times \mathbf{C} \to \mathbf{C}$. By taking the latter to be trivially guarded and then defining guardedness for $\nu\gamma'.\, \nu\gamma.\, T(X + \gamma + A \times \gamma')$ using Theorem 13, we arrive precisely at the notion we expected for the isomorphic monad $\#^\nu$.

5 Complete Elgot Monads and Iteration Congruences

Besides the fixpoint identity we are interested in the following classical properties of the iteration operator, which we refer to as the *iteration laws* [7,10,27]:

- *naturality:* $g^\star f^\dagger = ([(T\,\mathsf{in}_1)\, g, \eta\,\mathsf{in}_2]^\star\, f)^\dagger$ for $f : X \to_2 T(Y + X)$, $g : Y \to TZ$;
- *dinaturality:* $([\eta\,\mathsf{in}_1, h]^\star\, g)^\dagger = [\eta, ([\eta\,\mathsf{in}_1, g]^\star\, h)^\dagger]^\star\, g$ for $g : X \to_2 T(Y + Z)$ and $h : Z \to T(Y + X)$ or $g : X \to T(Y + Z)$ and $h : Z \to_2 T(Y + X)$;
- *codiagonal:* $(T[\mathsf{id}, \mathsf{in}_2]\, f)^\dagger = f^{\dagger\dagger}$ for $f : X \to_{12,2} T((Y + X) + X)$;
- *uniformity:* $f\, h = T(\mathsf{id} + h)\, g$ implies $f^\dagger\, h = g^\dagger$ for $f : X \to_2 T(Y + X)$, $g : Z \to_2 T(Y + Z)$ and $h : Z \to X$.

The axioms are summarized in graphical form in Fig. 2, and then become quite intuitive. The two versions of the dinaturality axiom correspond to the alternative sets of guardedness assumptions mentioned above. We indicate the scope of the iteration operator by a shaded box and guardedness by bullets at the outputs of a morphism.

A guarded pre-iterative monad is called a *complete Elgot monad* if it is totally guarded and satisfies all iteration laws. In the sequel we shorten 'complete Elgot monads' to 'Elgot monads' (to be distinguished from Elgot monads in the sense of [2], which have solutions only for morphisms with finitely presentable domain).

In general, the fact that the iteration laws are correctly formulated relies on the axioms for guardedness. E.g., in the dinaturality axiom it suffices to assume that $g : X \to T(Y + Z)$ is in_2-guarded and this implies that both $[\eta\,\mathsf{in}_1, h]^\star\, g$ and $[\eta\,\mathsf{in}_1, g]^\star\, h$ are in_2-guarded by **(cmp)** and **(trv)**, and additionally **(sum)** in the latter case. Symmetrically, it suffices to make the analogous assumption about h. In the codiagonal axiom, it follows from the assumption $f : X \to_{12,2} T((Y + X) + X)$ by **(cmp)** that $T[\mathsf{id}, \mathsf{in}_2]\, f$ is in_2-guarded and by Proposition 6 that $f^\dagger$ is in_2-guarded. Indeed, the axioms for guarded monads are designed precisely to enable the formulation of the iteration laws.

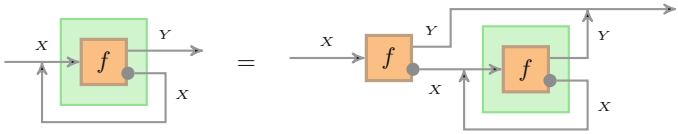
We show that for guarded iterative monads all iteration laws are automatic. Prior to that, we show that dinaturality follows from the others (thus generalizing the corresponding observation made recently [13,15]).

Proposition 16. *Any guarded pre-iterative monad satisfying naturality, codiagonal and uniformity also satisfies dinaturality, as well as the* Bekić identity

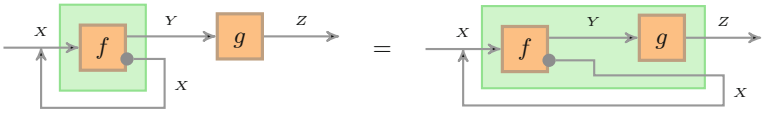
$$\left[T[\mathsf{id} + \mathsf{in}_1, \mathsf{in}_2]^\star\, f, T[\mathsf{id} + \mathsf{in}_1, \mathsf{in}_2]^\star\, g\right]^\dagger = [h^\dagger, [\eta, h^\dagger]^\star g^\dagger]$$

where $f : X \to_{12,2} T((Y + X) + Z)$, $g : Z \to_{12,2} T((Y + X) + Z)$, *and* $h = [\eta, g^\dagger]^\star f : X \to_2 T(Y + X)$.

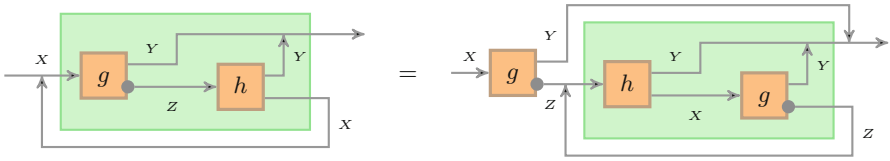
Fixpoint:



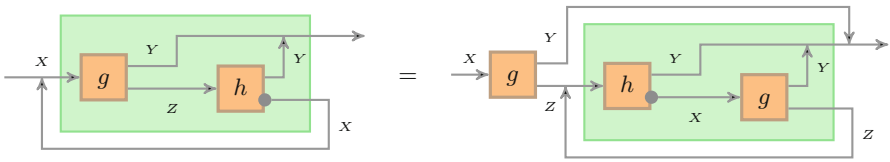
Naturality:



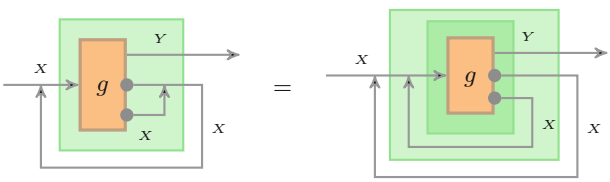
Dinaturality 1:



Dinaturality 2:



Codiagonal:



Uniformity:

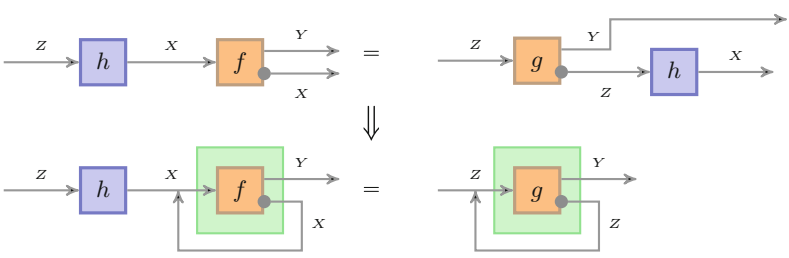


Fig. 2. Axioms of guarded iteration.

The proof of the following result runs in accordance with the original ideas of Elgot [10] for iterative theories, except that, by Proposition 16, dinaturality is now replaced by uniformity.

Theorem 17. *Every guarded iterative monad validates naturality, dinaturality, codiagonal and uniformity.*

We now proceed to introduce key properties of morphisms of guarded monads that allow for transferring pre-iterativity and the iteration laws, respectively.

Definition 18 (Guarded retraction). Let $\mathbb{T}$ and $\mathbb{S}$ be guarded monads. We call a monad morphism $\rho : \mathbb{T} \to \mathbb{S}$ a *guarded retraction* if there is a family of morphisms $(\upsilon_X : SX \to TX)_{X \in |\mathbf{C}|}$ (not necessarily natural in X!) such that

1. for every $f : X \to_\sigma SY$, we have $\upsilon_Y f : X \to_\sigma TY$,
2. $\rho_X \upsilon_X = \mathsf{id}$ for all $X \in |\mathbf{C}|$.

Theorem 19. *Let $\rho : \mathbb{T} \to \mathbb{S}$ be a guarded retraction, witnessed by $\upsilon : \mathbb{S} \to \mathbb{T}$, and suppose that $(\mathbb{T}, -^\dagger)$ is guarded pre-iterative. Then $\mathbb{S}$ is guarded pre-iterative with the iteration operator $(-)^\ddagger$ given by $f^\ddagger = \rho\,(\upsilon f)^\dagger$.*

Definition 20 (Iteration congruence). Let $\mathbb{T}$ be a guarded pre-iterative monad and let $\mathbb{S}$ be a monad. We call a monad morphism $\rho : \mathbb{T} \to \mathbb{S}$ an *iteration congruence* if for every pair of morphisms $f, g : X \to_2 T(Y + X)$,

$$\rho f = \rho g \implies \rho f^\dagger = \rho g^\dagger. \tag{2}$$

If ρ is moreover a guarded retraction, we call ρ an *iteration-congruent retraction.*

Theorem 21. *Under the premises of Theorem 19, assume moreover that ρ is an iteration-congruent retraction. Then any property out of naturality, dinaturality, codiagonal, and uniformity that is satisfied by $\mathbb{T}$ is also satisfied by $\mathbb{S}$.*

Proof (Sketch). The crucial observation is that under our assumptions, (2) is equivalent to the condition that for all $f : X \to_2 T(Y + X)$,

$$\rho\,(\upsilon\rho\, f)^\dagger = \rho f^\dagger. \tag{3}$$

Indeed, (2) $\implies$ (3), for $\rho\upsilon\rho\, f = \rho\, f$ and therefore $\rho(\upsilon\rho\, f)^\dagger = \rho\, f^\dagger$ and conversely, assuming (3) both for f and for g, and $\rho f = \rho g$, we obtain that $\rho f^\dagger = \rho(\upsilon\rho\, f)^\dagger = \rho(\upsilon\rho\, g)^\dagger = \rho g^\dagger$. Using (3), the claim is established routinely. □

Recall from the introduction that a monad $\mathbb{S}$ is *iteratable* if its coinductive resumption transform $\mathbb{S}^\nu$ exists. We make $\mathbb{S}^\nu$ into a guarded monad by applying Theorem 13 to $\mathbb{S}$ as a trivially guarded monad; explicitly: $f : X \to S^\nu(Y + X)$ is guarded iff $\mathsf{out}\, f = S(\mathsf{in}_1 + \mathsf{id})\, g$ for some $g : X \to S(Y + S^\nu(Y + X))$. We are now set to prove our first main result, which states that every iteratable Elgot monad can be obtained by quotienting a guarded iterative monad; that is, every choice of solutions that obeys the iteration laws arises by quotienting a more fine-grained model in which solutions are uniquely determined:

Theorem 22. *A totally guarded iteratable monad* $\mathbb{S}$ *is an Elgot monad iff there is a guarded iterative monad* $\mathbb{T}$ *and an iteration-congruent retraction* $\rho : \mathbb{T} \to \mathbb{S}$. *Specifically, every iteratable Elgot monad* $\mathbb{S}$ *is an iteration-congruent retract of its coinductive resumption transform* $\mathbb{S}^\nu$.

Proof (Sketch). Direction ($\Leftarrow$) immediately follows from Theorems 17 and 21.

In order to prove ($\Rightarrow$), we show that $\mathbb{S} = (S, \eta, -^\star, -^\dagger)$ is an iteration-congruent retract of $\mathbb{S}^\nu = (\nu\gamma.\, S(- + \gamma), \eta^\nu, -^\star, -^\ddagger)$. Let $\upsilon_X = \mathsf{out}^{-1}\, \eta\, \mathsf{in}_2\, \mathsf{out}^{-1}(S\, \mathsf{in}_1)$ and

$$\rho_X = \big(S^\nu X \xrightarrow{\mathsf{out}} S(X + TX)\big)^\dagger.$$

Clearly, υf is σ-guarded for every $f : X \to SY$ and it is easy to verify that υ is left inverse to ρ by using the fixpoint identity for $-^\dagger$ twice.

Naturality of ρ is proved straightforwardly from naturality of $-^\dagger$. The remaining calculations showing that ρ is a monad morphism and moreover an iteration congruence make heavy use of the Elgot monad laws. □

The notions of guarded retraction and iteration congruence straightforwardly extend to parametrized monads. We then can take the claims of Theorem 13 further.

Theorem 23. *Let* $\#, \hat{\#} : \mathbf{C} \times (\mathbf{C} \times \mathbf{D}) \to \mathbf{C}$ *be guarded parametrized monads and let* $\rho : \# \to \hat{\#}$ *be an iteration-congruent retraction. By Theorem 13,* $\#^\nu = - \#(\gamma, -)$ *and* $\hat{\#}^\nu = - \hat{\#}(\gamma, -)$ *are also parametrized guarded monads. Then* $\rho^\nu : \#^\nu \to \hat{\#}^\nu$, *with components*

$$\rho^\nu_{X,Y} = \mathsf{coit}\, \big(\nu\gamma.\, X \# (\gamma, Y) \xrightarrow{\rho\mathsf{out}} X \hat{\#} (\nu\gamma.\, X \# (\gamma, Y), Y)\big),$$

is again an iteration-congruent retraction.

Theorems 22 and 23 jointly provide a simple and structured way of showing that Elgotness extends along the parametrized monad transformer $\# \mapsto \hat{\#}$: If $- \#X$ is Elgot then by Theorem 22 there is an iteration-congruent retraction $\rho : \nu\gamma.\, - + \gamma \# X \to - \#X$; by Theorem 23, it gives rise to an iteration-congruent retraction

$$\rho^\nu : \nu\gamma'.\, \nu\gamma.\, - + \gamma \# (\gamma', X) \to \nu\gamma'.\, - \#(\gamma', X)$$

and by Theorem 22, the right-hand side is again Elgot. We have thus proved.

Corollary 24. *Given a parametrized monad* $\#$ *and* $X \in |\mathbf{C}|$, *if* $- \#X$ *is Elgot then so is* $- \#^\nu X = \nu\gamma.\, - \#(\gamma, X)$.

This yields a more general and simpler proof of one of the main results in [15].

Example 25. By instantiating $\#$ in Corollary 24 with $\mathcal{P}_\omega(- + A \times -) : \mathbf{Set} \times \mathbf{Set} \to \mathbf{Set}$ where $\mathcal{P}_\omega$ is the countable powerset monad, we obtain $\nu\gamma.\, \mathcal{P}_\omega(X + A \times \gamma)$, which can be viewed as a semantic domain for countably branching processes that possibly terminate with results in X and are taken modulo strong bisimilarity. The simple fact that $\mathcal{P}_\omega$ is Elgot [15] implies that so is $\nu\gamma.\, \mathcal{P}_\omega(X + A \times \gamma)$.

This justifies the use of systems of possibly unguarded recursive process algebra equations (as done, e.g., in [6]). It is worth noting that the iteration operator of the transformed monad is neither least nor unique. It arises by introducing an additional delay action that guards all recursive calls and then eliminating these delays from the unique solution of the new recursive definition; the delay elimination is the effect of ρ^ν in Theorem 23.

Theorem 22 characterizes iteratable Elgot monads as iteration-congruent retracts of their $(-)^\nu$-transforms. We take this perspective further as follows. Let us call $\mathbb{T}$ *strongly iteratable* if every $T^{\nu\cdots\nu}$ exists. Consider the functor $\mathbb{T} \mapsto \mathbb{T}^\nu$ on the category of strongly iteratable monads over $\mathbf{C}$. This is itself a monad: the unit $\boldsymbol{\eta}$ is the natural transformation with components $\boldsymbol{\eta}_X = \mathsf{out}^{-1}(T\,\mathsf{in}_1) : TX \to T^\nu X$ and the multiplication $\boldsymbol{\mu} : T^{\nu\nu} \to T^\nu$ has components

$$\boldsymbol{\mu}_X = \mathsf{coit}\,\big(T[\mathsf{id},\, \mathsf{in}_2\,\mathsf{out}^{-1}]\mathsf{out}\,\mathsf{out} : T^{\nu\nu}X \to T(X + T^{\nu\nu}X)\big).$$

For every T we define the *delay transformation* $\triangleright = \mathsf{out}^{-1}\,\eta\,\mathsf{in}_2 : T^\nu \to T^\nu$. This leads to our second main result:

Theorem 26. *The category of strongly iteratable Elgot monads over* $\mathbf{C}$ *is isomorphic to the full subcategory of the category of* $(-)^\nu$*-algebras for strongly iteratable* $\mathbb{S}$ *consisting of the* $(-)^\nu$*-algebras* $(S^\nu, \rho : S^\nu \to S)$ *satisfying* $\rho \triangleright = \rho$.

Proof (Sketch). To show that every strongly iteratable Elgot monad is a $(-)^\nu$-algebra, one has to check the equations $\rho\boldsymbol{\eta} = \mathsf{id}$ and $\rho\boldsymbol{\mu} = \rho\rho^\nu$ where $\rho^\nu = \mathsf{coit}\,(\rho\,\mathsf{out}) : S^{\nu\nu} \to S^\nu$. The first equation follows relatively easily. The second one is shown along the following lines:

$$\rho\boldsymbol{\mu} \overset{\text{(i)}}{=\!=} \rho[\eta,\, (\triangleright\,\mathsf{out})^\ddagger]^\star\mathsf{out} \overset{\text{(ii)}}{=\!=} \rho\big(\mathsf{out}^{-1}\,S(\,\mathsf{in}_1 + \eta^\nu\,\mathsf{in}_2)\rho\mathsf{out}\big)^\ddagger \overset{\text{(iii)}}{=\!=} \rho\rho^\nu.$$

Here, (i) and (iii) only amount to equivalent transformations of $\boldsymbol{\mu}$ and ρ^ν, respectively, while (ii) makes crucial use of the fact that ρ is an iteration congruence, as implied by Theorem 22.

For the converse implication, we start with a $(-)^\nu$-algebra and verify the Elgot monad laws for the iteration operator $f^\dagger = \rho(\mathsf{coit}\ f)$. □

Remark 27. The delay cancellation condition $\rho \triangleright = \rho$ is essential, as can be seen on a simple example. Let $\mathbf{Mon}(\mathbf{C})^\nu$ be the category of $(-)^\nu$-algebras and let $\mathbf{Mon}(\mathbf{C})^\nu_\triangleright$ be its subcategory figuring in Theorem 26. Since the identity functor is the initial monad, the initial object of $\mathbf{Mon}(\mathbf{C})^\nu$ is Capretta's delay monad [8] $D = \nu\gamma.(-+\gamma)$. On the other hand, the initial object of $\mathbf{Mon}(\mathbf{C})^\nu_\triangleright$ (if it exists) is the *initial Elgot monad* $\mathbb{L}$, which on $\mathbf{C} = \mathbf{Set}$ is the *maybe monad* $(-) + 1$.

If $\mathbf{C} = \mathbf{Set}$, then $DX = (X \times N + 1)$ does turn out to be Elgot [14] (but applying Theorem 26 to D qua Elgot monad yields a different $(-)^\nu$-algebra structure than the initial one), and $\mathbb{L}$ is, in this case, a retract of $\mathbb{D}$ in $\mathbf{Mon}(\mathbf{C})^\nu_\triangleright$. The situation is more intricate in categories with a non-classical internal logic, for which $\mathbb{D}$ is mainly intended. We believe that in such a setting, neither is $\mathbb{D}$ Elgot in general, nor is $\mathbb{L}$ the maybe monad. However, there will still be a unique $(-)^\nu$-algebra morphism $\mathbb{D} \to \mathbb{L}$ in $\mathbf{Mon}(\mathbf{C})^\nu$.

6 A Sandwich Theorem for Elgot Monads

As an application of Theorem 22, we show that sandwiching an Elgot monad between adjoint functors again yields an Elgot monad. A similar result has been shown for completely iterative monads [26]; this result generalizes straightforwardly to guarded iterative monads:

Theorem 28. *Let $F : \mathbf{C} \to \mathbf{D}$ and $U : \mathbf{D} \to \mathbf{C}$ be a pair of adjoint functors with the associated natural isomorphism $\Phi : \mathbf{D}(FX, Y) \to \mathbf{C}(X, UY)$, and let $\mathbb{T}$ be a guarded iterative monad on $\mathbf{D}$. Then the monad induced on the composite functor UTF is guarded iterative, with the guardedness relation defined by taking $f : X \to_\sigma UTFY$ if and only if $\Phi^{-1} f : FX \to_\sigma TFY$, and the unique solutions given by $f \mapsto \Phi((\Phi^{-1} f)^\dagger)$.*

Now, to obtain a similar result for Elgot monads, we can easily combine Theorems 22 and 28 without having to verify the equational properties by hand.

Theorem 29. *With an adjunction as in Theorem 28, let $\mathbb{S}$ be an Elgot monad on $\mathbf{D}$. Then, the monad induced on the composition USF is an Elgot monad.*

Proof (Sketch). By Theorem 22, there exists a guarded iterative monad $\mathbb{T}$ and an iteration-congruent retraction $\rho : \mathbb{T} \to \mathbb{S}$. By Theorem 28, the monad induced on UTF is guarded iterative. Again by Theorem 22, it suffices to show that $U\rho F : UTF \to USF$ is an iteration-congruent retraction, which is straightforward. □

Example 30 (From Metric to CPO-based Iteration). As an example exhibiting sandwiching as well as the setting of Theorem 22, we compare two iteration operators on **Set** that arise from different fixed-point theorems: Banach's, for complete metric spaces, and Kleene's, for complete partial orders, respectively. We obtain the first operator by sandwiching Escardo's *metric lifting monad* $\mathbb{S}$ [12] in the adjunction between sets and bounded complete ultrametric spaces (which forgets the metric in one direction and takes discrete spaces in the other), obtaining a monad $\bar{\mathbb{S}}$ on **Set**. Given a bounded complete metric space (X, d), $S(X, d)$ is a metric on the set $(X \times \mathbb{N}) \cup \{\bot\}$. As we show in the appendix, $\mathbb{S}$ is guarded iterative if we define $f : (X, d) \to S(Y, d')$ to be σ-guarded if $k > 0$ whenever $f(x) = (\sigma(y), k)$. By Theorem 28, $\bar{\mathbb{S}}$ is also guarded iterative (of course, this can also be shown directly). The second monad arises by sandwiching the identity monad on cpos with bottom in the adjunction between sets and cpos with bottom that forgets the ordering in one direction and adjoins bottom in the other, obtaining an Elgot monad $\mathbb{L}$ on Set according to Theorem 29. The latter is unsurprising, of course, as $\mathbb{L}$ is just the maybe monad $LX = X + 1$.

The monad $\bar{\mathbb{S}}$ keeps track of the number of steps needed to obtain the final result. We have an evident extensional collapse map $\rho : \bar{\mathbb{S}} \to \mathbb{L}$, which just forgets the number of steps. One can show that ρ is in fact an iteration-congruent retraction, so we obtain precisely the situation of Theorem 22.

7 Related Work

Alternatively to our guardedness relation on Kleisli morphisms, guardedness can be formalized using type constructors [23] or, categorically, functors, as in *guarded fixpoint categories* [20]; the latter cover also total guardedness, like we do. Our approach is slightly more fine-grained, and in particular natively supports the two variants of the dinaturality axiom (Fig. 2), which, e.g., in guarded fixpoint categories require additional assumptions [20, Proposition 3.15] akin to the one we discuss in Remark 4.

A result that resembles our Theorem 26, due to Adámek et al. [3], states roughly that if $\mathbf{C}$ is locally finitely presentable and hyperextensive (e.g. $\mathbf{C} = \mathbf{Set}$) then the finitary Elgot monads are the algebras for a monad on the category of endofunctors given by $H \mapsto L_H = \rho\gamma.(- + 1 + H\gamma)$ where ρ takes *rational fixpoints* (i.e. final coalgebras among those where every point generates a finite subcoalgebra). Besides Theorem 26 making fewer assumptions on $\mathbf{C}$, the key difference is that, precisely by dint of this result, L_H is already an Elgot monad; contrastingly, we characterize Elgot monads as quotients of *guarded iterative* monads, i.e. of monads where guarded recursive definitions have *unique* fixpoints.

8 Conclusions and Further Work

We have given a unified account of monad-based guarded and unguarded iteration by axiomatizing the notion of guardedness to cover standard definitions of guardedness, and additionally, as a corner case, what we call *total guardedness*, i.e. the situation when all morphisms are declared to be guarded. We thus obtain a common umbrella for *guarded iterative monads*, i.e. monads with unique iterates of guarded morphisms, and Elgot monads, i.e. totally guarded monads satisfying Elgot's classical laws of iteration. We reinforce the view that the latter constitute a canonical model for monad-based unguarded iteration by establishing the following equivalent characterizations: provided requisite final coalgebras exist, a monad $\mathbb{T}$ is Elgot iff

- it satisfies the quasi-equational theory of iteration [2,15] (definition);
- it is an iteration-congruent retract of a guarded iterative monad (Theorem 22);
- it is an algebra $(\mathbb{T}, \rho)$ of the monad $T \mapsto \nu\gamma.T(X + \gamma)$ in the category of monads satisfying a natural delay cancellation condition (Theorem 26).

In future work, we aim to investigate further applications of this machinery, in particular to examples which did not fit previous formalizations. One prospective target is suggested by the work of Nakata and Uustalu [24], who give a coinductive big-step trace semantics for a while-language. We conjecture that this work has an implicit guarded iterative monad $\mathbb{T}_{\mathbb{R}}$ under the hood, for which guardedness cannot be defined using the standard argument based on a final coalgebra structure of the monad because $\mathbb{T}_{\mathbb{R}}$ is not a final coalgebra.

In type theory, there is growing interest in forming an extensional quotient of the delay monad [4,9]. It is shown in [9] that under certain reasonable conditions, a suitable collapse of the delay monad by removing delays is again a

monad; however, the proof is already quite complex, and proving directly that the collapse is in fact an Elgot monad, as one would be inclined to expect, seems daunting. We expect that Theorem 26 may shed light on this issue. A natural question that arises in this regard is whether the subcategory of $(-)^\nu$-algebras figuring in the theorem is reflexive. A positive answer would provide a means of constructing canonical quotients of $(-)^\nu$-algebras (such as the delay monad) with the results automatically being Elgot monads.

References

1. Aczel, P., Adámek, J., Milius, S., Velebil, J.: Infinite trees and completely iterative theories: a coalgebraic view. Theoret. Comput. Sci. **300**(1–3), 1–45 (2003)
2. Adámek, J., Milius, S., Velebil, J.: Equational properties of iterative monads. Inf. Comput. **208**, 1306–1348 (2010)
3. Adámek, J., Milius, S., Velebil, J.: Elgot theories: a new perspective of the equational properties of iteration. Math. Struct. Comput. Sci. **21**, 417–480 (2011)
4. Altenkirch, T., Danielsson, N.: Partiality, revisited. In: Types for Proofs and Programs, TYPES 2016 (2016)
5. Atkey, R.: Parameterised notions of computation. J. Funct. Prog. **19**, 335 (2009)
6. Bergstra, J., Ponse, A., Smolka, S. (eds.): Handbook of Process Algebra. Elsevier, Amsterdam (2001)
7. Bloom, S., Ésik, Z.: Iteration Theories: The Equational Logic of Iterative Processes. Springer, Heidelberg (1993)
8. Capretta, V.: General recursion via coinductive types. Log. Meth. Comput. Sci. **1**(2), 1–28 (2005)
9. Chapman, J., Uustalu, T., Veltri, N.: Quotienting the delay monad by weak bisimilarity. In: Leucker, M., Rueda, C., Valencia, F.D. (eds.) ICTAC 2015. LNCS, vol. 9399, pp. 110–125. Springer, Cham (2015). doi:10.1007/978-3-319-25150-9_8
10. Elgot, C.: Monadic computation and iterative algebraic theories. In: Rose, H.E., Shepherdson, J.C. (eds.) Logic Colloquium 1973. Studies in Logic and the Foundations of Mathematics, vol. 80, pp. 175–230. Elsevier, Amsterdam (1975)
11. Elgot, C., Bloom, S., Tindell, R.: On the algebraic atructure of rooted trees. J. Comput. Syst. Sci. **16**, 362–399 (1978)
12. Escardó, M.H.: A metric model of PCF. In: Realizability Semantics and Applications (1999)
13. Ésik, Z., Goncharov, S.: Some remarks on Conway and iteration theories. CoRR, abs/1603.00838 (2016)
14. Goncharov, S., Milius, S., Rauch, C.: Complete Elgot monads and coalgebraic resumptions. In: Mathematical Foundations of Programming Semantics, MFPS 2016. ENTCS (2016)
15. Goncharov, S., Rauch, C., Schröder, L.: Unguarded recursion on coinductive resumptions. In: Mathematical Foundations of Programming Semantics, MFPS 2015. ENTCS (2015)
16. Goncharov, S., Schröder, L.: A coinductive calculus for asynchronous side-effecting processes. Inf. Comput. **231**, 204–232 (2013)
17. Lawvere, W.: Functorial semantics of algebraic theories. Proc. Natl. Acad. Sci. USA **50**, 869–872 (1963)
18. Mac Lane, S.: Categories for the Working Mathematician. Springer, Heidelberg (1971)

19. Milius, S.: Completely iterative algebras and completely iterative monads. Inf. Comput. **196**, 1–41 (2005)
20. Milius, S., Litak, T.: Guard your daggers and traces: properties of guarded (co-)recursion. Fund. Inform. **150**, 407–449 (2017)
21. Moggi, E.: A modular approach to denotational semantics. In: Pitt, D.H., Curien, P.-L., Abramsky, S., Pitts, A.M., Poigné, A., Rydeheard, D.E. (eds.) CTCS 1991. LNCS, vol. 530, pp. 138–139. Springer, Heidelberg (1991). doi:10.1007/BFb0013462
22. Moggi, E.: Notions of computation and monads. Inf. Comput. **93**, 55–92 (1991)
23. Nakano, H.: A modality for recursion. In: Logic in Computer Science, LICS 2000, pp. 255–266. IEEE Computer Society (2000)
24. Nakata, K., Uustalu, T.: A Hoare logic for the coinductive trace-based big-step semantics of while. Log. Meth. Comput. Sci. **11**(1), 1–32 (2015)
25. Piróg, M., Gibbons, J.: The coinductive resumption monad. In: Mathematical Foundations of Programming Semantics, MFPS 2014. ENTCS, vol. 308, pp. 273–288 (2014)
26. Piróg, M., Gibbons, J.: Monads for behaviour. In: Mathematical Foundations of Programming Semantics, MFPS 2013. ENTCS, vol. 298, pp. 309–324 (2015)
27. Simpson, A., Plotkin, G.: Complete axioms for categorical fixed-point operators. In: Logic in Computer Science, LICS 2000, pp. 30–41 (2000)
28. Uustalu, T.: Generalizing substitution. ITA **37**, 315–336 (2003)
29. Uustalu, T., Vene, V.: Primitive (co)recursion and course-of-value (co)iteration, categorically. Informatica **10**(1), 5–26 (1999). Lithuanian Academy of Sciences