Theoretical Informatics and Applications
Theoret. Informatics Appl. **37** (2003) 315–336
DOI: 10.1051/ita:2003022

GENERALIZING SUBSTITUTION *

Tarmo Uustalu [1, 2]

Abstract. It is well known that, given an endofunctor H on a category $\mathcal{C}$, the initial $(A + H-)$-algebras (if existing), *i.e.*, the algebras of (wellfounded) H-terms over different variable supplies A, give rise to a monad with substitution as the extension operation (the free monad induced by the functor H). Moss [17] and Aczel, Adámek, Milius and Velebil [2] have shown that a similar monad, which even enjoys the additional special property of having iterations for all guarded substitution rules (complete iterativeness), arises from the inverses of the final $(A + H-)$-coalgebras (if existing), *i.e.*, the algebras of non-wellfounded H-terms. We show that, upon an appropriate generalization of the notion of substitution, the same can more generally be said about the initial $T'(A, -)$-algebras resp. the inverses of the final $T'(A, -)$-coalgebras for any endobifunctor T' on any category $\mathcal{C}$ such that the functors $T'(-, X)$ uniformly carry a monad structure.

Mathematics Subject Classification. 08B20, 18C15, 18C50.

Keywords and phrases. Algebras of terms, non-wellfounded terms, substitution, iteration of guarded substitution rules, monads, hyperfunctions, finitely or possibly infinitely branching trees.

* *This work was supported by the Portuguese Foundation for Science and Technology under grant No. PRAXIS XXI/C/EEI/14172/98 and the Estonian Science Foundation under grant No. 4155. The author's visit to LMU München in Jan. 2002 was fully paid by the Graduiertenkolleg "Logik in der Informatik" of Deutsche Forschungsgemeinschaft; his participation in FICS'02 was made possible by a travel grant from the organizing committee of FLoC'02.*

[1] Inst. of Cybernetics, Tallinn Technical Univ., Akadeemia tee 21, EE-12618 Tallinn, Estonia; e-mail: tarmo@cs.ioc.ee

[2] The work was largely done during the author's postdoctoral leave to Dep. de Informática, Univ. do Minho, Braga, Portugal.

Introduction

It is well known that, given an endofunctor H on a category $\mathcal{C}$, the initial $(A + H-)$-algebras (if existing), *i.e.*, the algebras of (wellfounded) H-terms over different variable supplies A, give rise to a monad with substitution as the extension operation (the free monad generated by H). This is the starting point in the category-theoretic approach to universal algebra, *cf.* Manes [14]. In contrast, the comparably fundamental fact that a similar monad, which even enjoys the additional property of having iterations for all guarded substitution rules (complete iterativeness), arises from the inverses of the final $(A+H-)$-coalgebras (if existing), *i.e.*, the algebras of non-wellfounded H-terms over different A, was only recently pointed out by Moss [17] and Aczel, Adámek, Milius and Velebil [1, 2] (their substitution and solution theorems), although the "down-to-earth" universal-algebra case, where $\mathcal{C} = \mathbf{Set}$ and H is a polynomial endofunctor, was settled in the 1970s by Elgot and colleagues [9].

In this paper, we show that, upon an appropriate generalization of the notion of substitution, the same statements can more generally be made about the initial $T'(A, -)$-algebras (if existing) resp. the inverses of the final $T'(A, -)$-coalgebras (if existing) for any endobifunctor T' on any category $\mathcal{C}$ such that the functors $T'(-, X)$ uniformly carry a monad structure. The generalization gives simpler proofs, as all inessential detail pertaining solely to the special case $T'(A, X) = A + HX$ is removed from the main proofs into an auxiliary proof that the functors $T'(-, X)$ uniformly carry a monad structure. The generalization also gives new examples of structures carrying a substitution-like operation. The examples beyond wellfounded and non-wellfounded terms in a given signature include the inductive and coinductive versions of Krstić, Launchbury and Pavlović' hyperfunction spaces [12] and finitely or possibly infinitely branching wellfounded or non-wellfounded decorated trees. For the latter, which may be seen as Böhm tree notations of purely applicative terms (lambda-terms without lambdas), the notion of generalized substitution coincides with the substitution of Böhm trees. This suggests that the generalization is "right".

Some results of the present paper were preliminarily reported in the author's conference paper [19].

The paper is organized as follows. In Section 1, we recall some basics about monads, initial algebras and final coalgebras, mostly to fix the terminology and notation for the rest of the paper and to introduce the examples of monads and the disciplined recursion and corecursion schemes used later in the paper. Section 2 contains a recapitulation of the known facts that both the algebras of terms and those of non-wellfounded terms in a given signature are substitution-carrying and that the latter are moreover completely iterative. In the proof of the substitution theorem, we take advantage of the validity of primitive corecursion as a morphism definition principle, which significantly modularizes the argument. In Section 3, we proceed to generalized substitution and generalized versions of the results. We also discuss examples of families of algebras carrying generalized substitution. The related work is reviewed in Section 4. In Section 5, we conclude.

1. Monads, initial algebras, final coalgebras

We begin by recalling the concept of monad. We choose to employ the extension form (Kleisli triples), as completely iterative monads needed later on will be most intuitively definable from this basis.

Definition 1.1. A *monad* on a category $\mathcal{C}$ is a triple $(T, \eta, -^\star)$ formed of a mapping $T : |\mathcal{C}| \to |\mathcal{C}|$, a $|\mathcal{C}|$-indexed family η of $\mathcal{C}$-morphisms $\eta_A : A \to TA$ (*unit*), and an operation $-^\star$ taking every $\mathcal{C}$-morphism $f : A \to TB$ to a $\mathcal{C}$-morphism $f^\star : TA \to TB$ (*extension operation*) satisfying

$$\begin{array}{ll} \text{(i)} & f^\star \circ \eta_A = f \quad \text{for } f : A \to TB; \\ \text{(ii)} & {\eta_A}^\star = \mathsf{id}_{TA}; \\ \text{(iii)} & (g^\star \circ f)^\star = g^\star \circ f^\star \quad \text{for } f : A \to TB,\ g : B \to TC. \end{array}$$

The following examples standard in programming language semantics will be useful for us.

Example 1.2. Given an object E of a category $\mathcal{C}$ with finite coproducts. Then $(T, \eta, -^\star)$ defined by

$$\begin{array}{c} TA = A + E \\ \eta_A = \mathsf{inl}_{A,E} \\ f^\star = [\, f, \mathsf{inr}_{B,E} \,] \quad \text{for } f : A \to TB \end{array}$$

is a monad. In semantics, it is known as the exception monad. One can also think of it as the term algebras monad for a signature with E many 0-ary operators (constants) and no operators of higher arities.

Example 1.3. Given an object E of a cartesian closed category $\mathcal{C}$. Then $(T, \eta, -^\star)$ defined by

$$\begin{array}{c} TA = E \Rightarrow A \\ \eta_A = \mathsf{curry}\,(\mathsf{fst}_{A,E}) \\ f^\star = \mathsf{curry}\,(\mathsf{ev}_{E,B} \circ \langle\, f \circ \mathsf{ev}_{E,A}, \mathsf{snd}_{TA,E} \,\rangle) \quad \text{for } f : A \to TB \end{array}$$

is a monad, the storage reader monad.

Example 1.4. Given a monoid (E, e, m) in a category $\mathcal{C}$ with finite products. Then $(T, \eta, -^\star)$ defined by

$$\begin{array}{c} TA = A \times E \\ \eta_A = (\mathsf{id}_A \times e) \circ \mathsf{unit}^\times_A \\ f^\star = (\mathsf{id}_A \times m) \circ \mathsf{assoc}^\times_{B,E,E} \circ (f \times \mathsf{id}_E) \quad \text{for } f : A \to TB \end{array}$$

is a monad that models several notions of computation (output, complexity). Concretely, one could, *e.g.*, choose $(E, e, m) = (\mathsf{Nat}, 0, +)$, if $\mathcal{C}$ has a natural numbers object.

A monad's property of complete iterativeness in the sense of Aczel *et al.* [2] is, for a given, but unspecified notion of guardedness of morphisms, definable as follows.

Definition 1.5. Given a monad $(T, \eta, -^\star)$ on a category $\mathcal{C}$ with finite coproducts. If some of the morphisms $A \to T(A + B)$ have for all A, B been classified as guarded, the monad is said to be *completely iterative* with respect to this notion of guardedness, if, for any guarded morphism $f : A \to T(A+B)$, there is a unique morphism $g : A \to TB$, called its *iterate* and denoted $\bar{f}$, such that

$$g = [\, g, \eta_B \,]^\star \circ f.$$

It is useful to note that this is equivalent to the unique existence of a morphism $h : T(A + B) \to TB$, denoted $\tilde{f}$, such that

$$h = [\, h \circ f, \eta_B \,]^\star.$$

The conversions are: $\tilde{f} = \big[\, \bar{f}, \eta_B \,\big]^\star$, $\bar{f} = \tilde{f} \circ f = \tilde{f} \circ \eta_{A+B} \circ \mathsf{inl}_{A,B}$.

An example of a monad completely iterative with respect to a useful notion of guardedness will be discussed in the next section.

Let us also recall the concepts of (initial) algebra and (final) coalgebra of an endofunctor.

Definition 1.6. Given an endofunctor F on a category $\mathcal{C}$, an *F-algebra* is a pair (X, φ) consisting of an object X (*carrier*) and a morphism $\varphi : FX \to X$ (*algebra structure*). An *F-algebra map* from (X, φ) to (Y, ψ) is a morphism $h : X \to Y$ such that

$$\begin{array}{ccc} FX & \xrightarrow{\varphi} & X \\ {\scriptstyle Fh}\downarrow & & \downarrow{\scriptstyle h} \\ FY & \xrightarrow{\psi} & Y \end{array}$$

Dually, an *F-coalgebra* is a pair (A, φ) consisting of an object X (underlying object or *carrier*) and a morphism $\varphi : X \to FX$ (*coalgebra structure*). An *F-coalgebra map* from (X, φ) to (Y, ψ) is a morphism $h : X \to Y$ such that

$$\begin{array}{ccc} FX & \xleftarrow{\varphi} & X \\ {\scriptstyle Fh}\downarrow & & \downarrow{\scriptstyle h} \\ FY & \xleftarrow{\psi} & Y \end{array}$$

Both the F-algebras and the F-coalgebras form a category (with identity and composition inherited from $\mathcal{C}$). If $\mathcal{C}$ has an initial algebra for F (there can only be one up to isomorphism), we denote it $(\mu F, \mathsf{in}_F)$. For the final F-coalgebra, we write $(\nu F, \mathsf{out}_F)$, if it exists. In semantics, initial algebras and final coalgebras are employed to model inductive and coinductive types, *i.e.*, the types of wellfounded resp. non-wellfounded recursive data structures. Two important recursive schemes

of function definition are direct consequences from their initiality and finality: iteration (structural recursion) and coiteration. *Iteration* says that, for any morphism $\varphi : FX \to X$, there exists a unique morphism $h : \mu F \to X$, denoted $\mathsf{It}_F(\,\varphi\,)$, such that

$$\begin{array}{ccc} F(\mu F) & \xrightarrow{\mathsf{in}_F} & \mu F \\ {\scriptstyle Fh}\downarrow & & \downarrow{\scriptstyle h} \\ FX & \xrightarrow{\varphi} & X \end{array}$$

Coiteration, dually, asserts that, for any morphism $\varphi : X \to FX$, there is a unique morphism $h : X \to \nu F$, denoted $\mathsf{Coit}_F\,(\,\varphi\,)$, such that

$$\begin{array}{ccc} FX & \xleftarrow{\varphi} & X \\ {\scriptstyle Fh}\downarrow & & \downarrow{\scriptstyle h} \\ F(\nu F) & \xleftarrow{\mathsf{out}_F} & \nu F \end{array}$$

Often, however, it is practical to make use of more advanced recursion and corecursion schemes. We shall take advantage of *primitive corecursion*, see, *e.g.*, [20]: for any morphism $\varphi : X \to F(X + \nu F)$, there exists a unique morphism $h : X \to \nu F$, denoted $\mathsf{Corec}_F(\,\varphi\,)$, such that

$$\begin{array}{ccc} F(X + \nu F) & \xleftarrow{\varphi} & X \\ {\scriptstyle F[\,h,\mathsf{id}_{\nu F}\,]}\downarrow & & \downarrow{\scriptstyle h} \\ F(\nu F) & \xleftarrow{\mathsf{out}_F} & \nu F \end{array}$$

The one and only h with the requested property is

$$\mathsf{Corec}_F(\,\varphi\,) = \mathsf{Coit}_F\,(\,[\,\varphi, F\mathsf{inr}_{X,\nu F} \circ \mathsf{out}_F\,]\,) \circ \mathsf{inl}_{X,\nu F}.$$

Finally, we need to recall the well-known Lambek's lemma: in_F and out_F are both isomorphisms, with inverses $\mathsf{in}_F^{-1} = \mathsf{It}_F(\,F\mathsf{in}_F\,) = \mathsf{Rec}_F(\,F\mathsf{snd}_{F(\mu F),\mu F}\,)$, $\mathsf{out}_F^{-1} = \mathsf{Coit}_F\,(\,F\mathsf{out}_F\,) = \mathsf{Corec}_F(\,F\mathsf{inr}_{F(\nu F),\nu F}\,)$.

2. Substitution in wellfounded and non-wellfounded term algebras

We now proceed to discussing the properties of substitution in wellfounded and non-wellfounded term algebras. For the purposes of modular presentation, however, we first introduce what we call substitution-carrying families of algebras[1]. This concept is also useful as a basis for uniform treatment of not only wellfounded

[1]In a recent tutorial text [1], Aczel gave the same concept the name "substitution systems".

and non-wellfounded terms, but also term-equivalence-classes (wrt. a system of equations), term graphs, rational terms etc.

Definition 2.1. Given an endofunctor H on a category $\mathcal{C}$ with finite coproducts. Then any assignment (T, α) of some $(A + H-)$-algebra to every $\mathcal{C}$-object A, *i.e.*, any mapping $T : |\mathcal{C}| \to |\mathcal{C}|$ and $|\mathcal{C}|$-indexed family α of morphisms $\alpha_A : A + HTA \to TA$, yields two $|\mathcal{C}|$-indexed families η, τ of morphisms $\eta_A : A \to TA$, $\tau_A : HTA \to TA$ defined by

$$\eta_A = \alpha_A \circ \mathsf{inl}_{A,HTA}$$
$$\tau_A = \alpha_A \circ \mathsf{inr}_{A,HTA}$$

so that $\alpha_A = [\, \eta_A, \tau_A \,]$. We say that (T, α) is *substitution-carrying*, if, for every morphism $f : A \to TB$, there exists a unique morphism $h : TA \to TB$, denoted $f^\star$, satisfying

$$\begin{array}{ccc} A + HTA & \xrightarrow[(=[\,\eta_A,\tau_A\,])]{\alpha_A} & TA \\ {\scriptstyle A+Hh}\downarrow & & \downarrow{\scriptstyle h} \\ A + HTB & \xrightarrow{[\,f,\tau_B\,]} & TB \end{array} \qquad \textit{i.e.,} \qquad \begin{array}{ccccc} A & \xrightarrow{\eta_A} & TA & \xleftarrow{\tau_A} & HTA \\ & {\scriptstyle f}\searrow & \downarrow{\scriptstyle h} & & \downarrow{\scriptstyle Hh} \\ & & TB & \xleftarrow{\tau_B} & HTB \end{array}$$

Intuitively, if an assignment (T, α) of an $(A + H-)$-algebra to every object A is substitution-carrying, then TA is, in some (possibly quite metaphorical) sense of the word "term", the set of *H-terms* over *variables* from A, η_A is insertion of variables, τ_A is insertion of operator applications and $-^\star$ is substitution. For any morphism $f : A \to TB$ (a *substitution rule*), the morphism $f^\star$ (the corresponding *substitution function*) is by our definition required to be a unique morphism agreeing with f on variables and commuting with operator applications.

Carrying substitution always implies presence of a monad: the monad laws are valid properties of substitution.

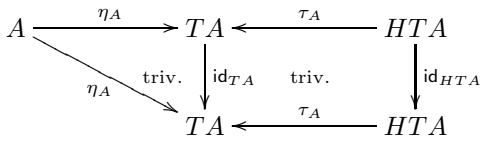
Theorem 2.2. *Given an endofunctor H on a category $\mathcal{C}$ with finite coproducts. If an assignment (T, α) of an $(A + H-)$-algebra to every $\mathcal{C}$-object A is substitution-carrying, then $(T, \eta, -^\star)$ is a monad.*

Proof. Assume (T, α) is substitution-carrying. The monad laws for $(T, \eta, -^\star)$ are verified as follows.

(i) The following triangle commutes for any $f : A \to TB$.

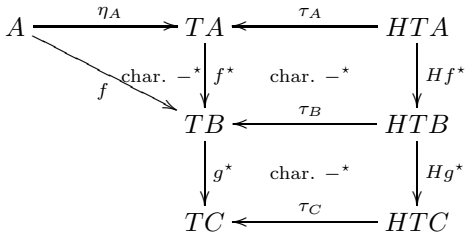
$$\begin{array}{ccc} A & \xrightarrow{\eta_A} & TA \\ & {\scriptstyle f}\searrow & \downarrow{\scriptstyle f^\star} \\ & & TB \end{array} \qquad \text{char. } -^\star$$

(ii) The triangle and square in the following diagram commute for any A.



Hence, by the characterization of substitution, $\eta_A{}^\star = \mathsf{id}_{TA}$.

(iii) The outer triangle and square in the following diagram commute for any $f : A \to TB$ and $g : B \to TC$.



Hence, by the characterization of substitution, $(g^\star \circ f)^\star = g^\star \circ f^\star$. □

We are interested in two examples: the initial $(A + H-)$-algebras for different objects A, *i.e.*, the algebras of wellfounded H-terms over different variable supplies, and the inverses of the final $(A+H-)$-algebras, *i.e.*, the algebras of non-wellfounded H-terms. Both structures carry substitution and are hence monads.

In the case of wellfounded H-terms, substitutionality and presence of a monad are nearly immediate.

Theorem 2.3. *If $\mathcal{C}$ has an initial $(A+H-)$-algebra for every object A, then (T, α) defined by*

$$(TA, \alpha_A) = (\mu(A + H-), \mathsf{in}_{A+H-})$$

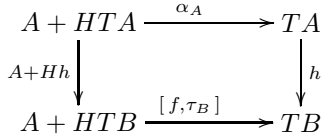
is substitution-carrying and hence $(T, \eta, -^\star)$ is a monad.

Proof. The proof is so trivial that it is a crime to speak of a theorem here.

(T, α) is substitution-carrying with

$$f^\star = \mathsf{It}_{A+H-}(\,[\,f, \tau_B\,]\,) \quad \text{for } f : A \to TB$$

since the right-hand side is, for a given f, by the characterization of iteration (initiality) the unique solution in h of the square



but that is also the square that $f^\star$ is supposed to be the unique solution in h of by the characterization of substitution. □

Remark 2.4. As is known since [5], the monad from the initial $(A+H-)$-algebras is actually the *free* monad generated by H.

The example of non-wellfounded H-terms, investigated in Moss [17] and Aczel *et al.* [2], is considerably more interesting. Substitution is definable also for non-wellfounded H-terms, but the definition is not as simple any more as for well-founded H-terms.

Theorem 2.5 (The substitution theorem of Moss [17] and Aczel *et al.* [2])**.** *If $\mathcal{C}$ has a final $(A + H-)$-coalgebra for every object A (which is what H being iteratable means in Aczel et al.'s terminology), then (T, α) defined by*

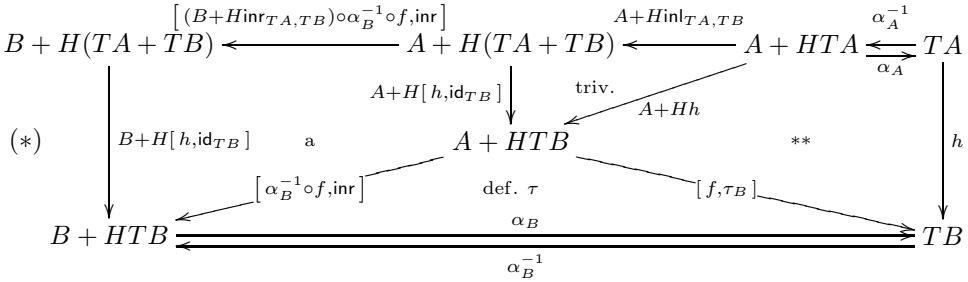
$$(TA, \alpha_A) = \left(\nu(A + H-), \mathsf{out}^{-1}_{A+H-}\right)$$

is substitution-carrying.

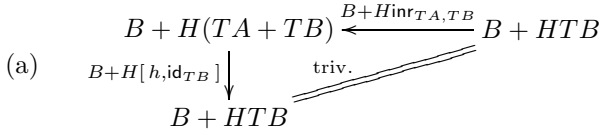
Proof. The substitution operation is conveniently definable with the help of primitive corecursion by

$$f^\star = \mathsf{Corec}_{B+H-}(\,[\,(B + H\mathsf{inr}_{TA,TB}) \circ \alpha_B^{-1} \circ f, \mathsf{inr}_{B,H(TA+TB)} \circ H\mathsf{inl}_{TA,TB}\,] \circ \alpha_A^{-1})\,) \quad \text{for } f : A \to TB.$$

The right-hand side is, for any given f, by the characterization of primitive corecursion, the unique solution in h of the outer square in the diagram



The left-hand side, *i.e.*, $f^\star$, must, at the same time, be the unique solution in h of the inner square marked (∗∗). But (∗∗) commutes for any h if and only if the outer square of (∗) does, as the triangle

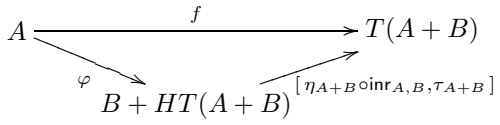


commutes. □

Remark 2.6. Substitution is, of course, also definable from the first principles (finality) without making use of primitive corecursion. Notably, however, the correctness proof is then considerably more complicated and the complications amount to nothing else than an implicit justification of an instance of primitive corecursion. Hence, it adds to the clarity and modularity of the proof, if primitive corecursion is justified first in its generality and only then used.

In addition to substitution, the algebra family of non-wellfounded H-terms also has iteration for all guarded substitution rules for a natural notion of guardedness. In Aczel *et al.*'s [2] terminology (which builds on that of Elgot *et al.* [8, 9]), their monad is completely iterative.

Definition 2.7. Given a substitution-carrying assignment (T, α) of an $(A + H-)$-algebra to every $\mathcal{C}$-object A, we say that a substitution rule $f : A \to T(A + B)$ is *guarded*, if it factors in the canonical way



through some morphism $\varphi : A \to B + HT(A + B)$.

If $\mathcal{C} = \mathbf{Set}$ and H is polynomial (which is the setting of universal algebra), then a rule $f : A \to T(A + B)$ for replacing variables from A with terms over the disjoint union of A and B is guarded, if every variable from A becomes either a variable from B or an operator application to terms over the disjoint union of A and B (and not a variable from A). For instance, if $A = \{x_1, x_2\}$, $B = \{y\}$, then $f_1 = [x_1 \mapsto \mathsf{f}(x_2), x_2 \mapsto \mathsf{g}(x_1)]$ and $f_2 = [x_1 \mapsto \mathsf{h}(x_1, x_2), x_2 \mapsto y]$ are guarded but $f_3 = [x_1 \mapsto \mathsf{f}(x_2), x_2 \mapsto x_2]$ is not.

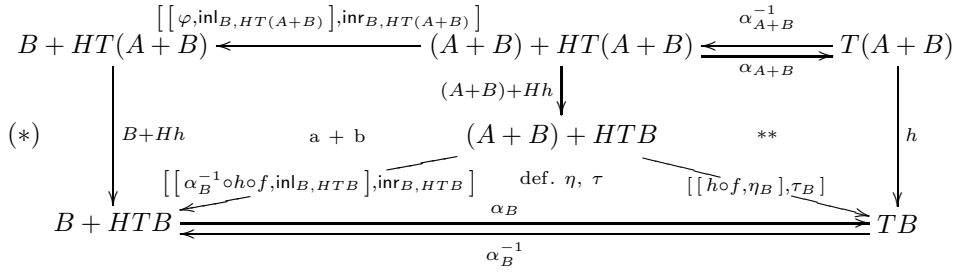
The iterate of f is its repetition as many times as are needed to get rid of all variables from A in the terms returned by the repetition. The iterates of f_1 and f_2 are $\bar{f}_1 = [x_1 \mapsto \mathsf{f}(\mathsf{g}(\mathsf{f}(\mathsf{g}(\ldots)))), x_2 \mapsto \mathsf{g}(\mathsf{f}(\mathsf{g}(\mathsf{f}(\ldots))))]$, $\bar{f}_2 = [x_1 \mapsto \mathsf{h}(\mathsf{h}(\mathsf{h}(\mathsf{h}(\ldots, y), y), y), y), x_2 \mapsto y]$, but f_3 cannot be iterated (the iterate is undefined). Intuitively, guarded substitution rules have iterates because guardedness is a simple sufficient condition for productivity of repetition.

Theorem 2.8 (The solution theorem of Moss [17] and Aczel *et al.* [2])**.** *Let* (T, α) *be defined as in Theorem* 2.5. *The monad* $(T, \eta, -^\star)$ *is completely iterative with respect to the guardedness notion of Definition* 2.7.

Proof. The operation $\tilde{-}$ is definable by

$$\tilde{f} = \mathsf{Coit}_{B+H-}\left(\left[\left[\varphi, \mathsf{inl}_{B,HT(A+B)}\right], \mathsf{inr}_{B,HT(A+B)}\right] \circ \alpha^{-1}_{A+B}\right) \quad \text{for } f : A \to T(A + B) \text{ guarded.}$$

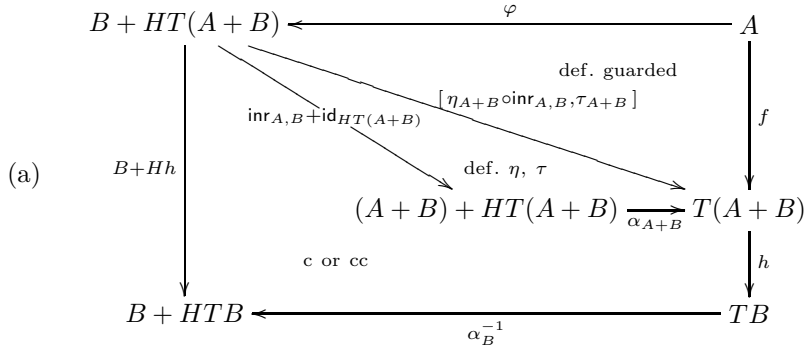
The right-hand side is, for any given f, by the characterization of coiteration (finality), the unique solution in h of the outer square of the diagram

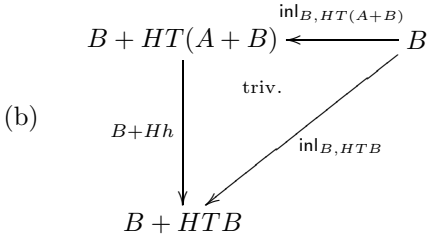


The left-hand side is, if f is guarded, supposed to be the unique solution in h of the equation

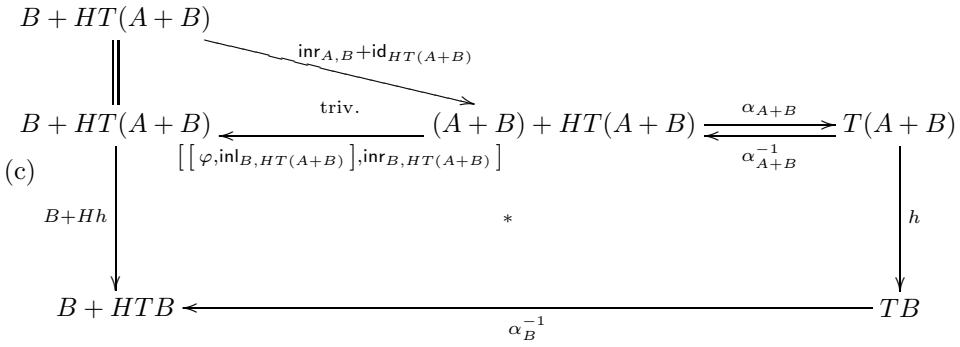
$$h = [\,h \circ f, \eta_B\,]^\star$$

which is equivalent, by the characterization of substitution, the square marked (**). Hence, it is enough to prove that, for any h, the outer square of (*) commutes if and only (**) commutes. Inspection of (*) shows that for this it suffices, if the outer square of diagram (a) and diagram (b) below commute both if the outer square of (*) commutes and if (**) commutes.

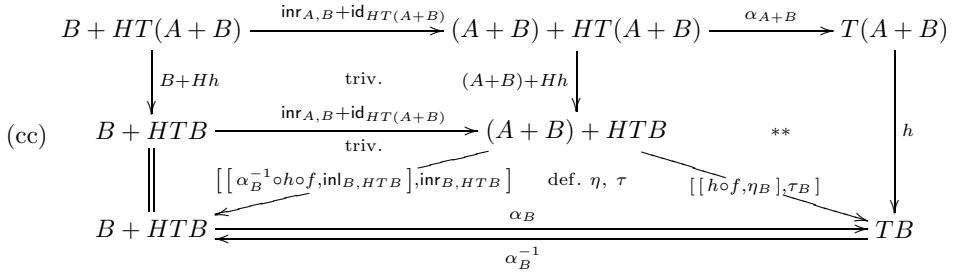




If the outer square of (*) commutes, then the outer square of (a) commutes because the outer square of



commutes. If (**) commutes, then the outer square of (a) commutes because the outer square of



commutes. (b) commutes always. □

Remark 2.9. Aczel *et al.* [2] have shown that the monad from the final $(A+H-)$-algebras is even the *free* completely iterative monad generated by H. In the present paper, this aspect will not be discussed.

3. Generalized substitution

The monads resulting from substitution-carrying assignments of an $(A+H-)$-algebra to every object A are, by the explicitly defining and characterizing equations of their constituent data, very similar to the monads that the object mappings $T'(-,X)$ given by $T'(A,X) = A+HX$ carry uniformly in X. Indeed, setting $\eta'_{A,X} = \mathsf{inl}_{A,HX}$, $\tau_{A,X} = \mathsf{inr}_{A,HX}$ and $f^\circ = [\,f,\tau_B\,]$ for $f : A \to T'(B,X)$, we get that $(T'(-,X),\eta'_{-,X},-^\circ)$ is a monad for every X. This observation leads to the questions: is this a hint about some "causality"? Can it be used to modularize the proofs of the statements above and to generalize them? The answer is: yes.

To present the generalization, we introduce the concept of parameterized monad.

Definition 3.1. A *parameterized monad* on category $\mathcal{C}$ is a triple $(T', \eta', -^\circ)$ formed of a mapping $T' : |\mathcal{C}| \times \mathcal{C} \to \mathcal{C}$ functorial in the 2nd argument, a $|\mathcal{C}| \times |\mathcal{C}|$-indexed family η' of morphisms $\eta'_{A,X} : A \to T'(A, X)$ and an operation $-^\circ$ taking every morphism $f : A \to T'(B, X)$ to a morphism $f^\circ : T'(A, X) \to T'(B, X)$ satisfying:

$$\begin{array}{ll}
(\mathrm{i}') & f^\circ \circ \eta'_{A,X} = f \quad \text{for } f : A \to T'(B, X); \\
(\mathrm{ii}') & {\eta'_{A,X}}^\circ = \mathsf{id}_{T'(A,X)}; \\
(\mathrm{iii}') & (g^\circ \circ f)^\circ = g^\circ \circ f^\circ \quad \text{for } f : A \to T'(B, X),\ g : B \to T'(C, X); \\
(\mathrm{iv}') & T'(A, \xi) \circ \eta'_{A,X} = \eta'_{A,Y} \quad \text{for } \xi : X \to Y; \\
(\mathrm{v}') & T'(B, \xi) \circ f^\circ = (T'(B, \xi) \circ f)^\circ \circ T'(A, \xi) \quad \text{for } f : A \to T'(B, X),\ \xi : X \to Y.
\end{array}$$

Remark 3.2. It is easy to verify that a parameterized monad is essentially just a functor from $\mathcal{C}$ to $\mathbf{Monad}(\mathcal{C})$. In principle, thus, the definition we just made is redundant. Practically, however, the "uncurried" format of a parameterized monad will be a lot handier to work with because the "curried" format of the equivalent monad-delivering functor fixes one of the two arguments of the uncurried format as the first and the other as the second in the "wrong" one of the two possible orders.

Each of the monad examples of Section 1 is easily modifiable to yield an example of a parameterized monad.

Example 3.3. Given an endofunctor H on a category $\mathcal{C}$ with finite coproducts. Then $(T', \eta', -^\circ)$ defined by

$$\begin{gathered}
T'(A, X) = A + HX \\
\eta'_{A,X} = \mathsf{inl}_{A,HX} \\
f^\circ = [\, f, \mathsf{inr}_{B,HX} \,] \quad \text{for } f : A \to T'(B, X)
\end{gathered}$$

is a parameterized monad.

Example 3.4. Given a contravariant endofunctor H on a cartesian closed category $\mathcal{C}$. Then $(T', \eta', -^\circ)$ defined by

$$\begin{gathered}
T'(A, X) = HX \Rightarrow A \\
\eta'_{A,X} = \mathsf{curry}\,(\mathsf{fst}_{A,HX}) \\
f^\circ = \mathsf{curry}\left(\mathsf{ev}_{HX,B} \circ \langle\, f \circ \mathsf{ev}_{HX,A}, \mathsf{snd}_{T'(A,X),HX} \,\rangle\right) \quad \text{for } f : A \to T'(B, X)
\end{gathered}$$

is a parameterized monad. A typical more specific example is obtained by choosing $HX = E \Rightarrow X$ for some fixed object E of $\mathcal{C}$.

Example 3.5. Given a functor $X \mapsto (HX, e_X, m_X)$ from a category $\mathcal{C}$ with finite products to $\mathbf{Monoid}(\mathcal{C})$. Then $(T', \eta', -^\circ)$ defined by

$$\begin{gathered}
T'(A, X) = A \times HX \\
\eta'_{A,X} = (\mathsf{id}_A \times e_X) \circ \mathsf{unit}^\times_A \\
f^\circ = (\mathsf{id}_A \times m_X) \circ \mathsf{assoc}^\times_{B,HX,HX} \circ (f \times \mathsf{id}_{HX}) \quad \text{for } f : A \to T'(B, X)
\end{gathered}$$

is a parameterized monad. A useful instance, if $\mathcal{C}$ has list objects, is $(HX, e_X, m_X) = (\mathsf{List}\ X, \mathsf{nil}_X, \mathsf{append}_X)$.

Parameterized monads give us a well-behaved generalization of the concept of substitution-carrying family of algebras.

Definition 3.6. Given any parameterized monad $(T', \eta', -^\circ)$ on any category $\mathcal{C}$. Then, given any assignment (T, α) of some $T'(A, -)$-algebra to every object A with $\alpha_A : T'(A, TA) \to TA$ iso for every A[2], we also have $|\mathcal{C}|$-indexed family η of morphisms $\eta_A : A \to TA$ and an operation $\{-\}$ taking every morphism $f : A \to TB$ to a morphism $\{f\} : T'(A, TB) \to TB$ defined by

$$\eta_A = \alpha_A \circ \eta'_{A,TA}$$
$$\{f\} = \alpha_B \circ (\alpha_B^{-1} \circ f)^\circ \quad \text{for } f : A \to TB$$

so that $\alpha_A = \{\eta_A\}$. We say that (T, α) is *substitution-carrying*, if, for every $f : A \to TB$, there is a unique $h : TA \to TB$, denoted $f^\star$, such that

$$\begin{array}{ccc} T'(A, TA) & \xrightarrow[(=\{\eta_A\})]{\alpha_A} & TA \\ {\scriptstyle T'(A,h)}\downarrow & & \downarrow{\scriptstyle h} \\ T'(A, TB) & \xrightarrow{\{f\}} & TB \end{array}$$

Clearly, the previous definition for the special case $T'(A, X) = A + HX$ is an instance of this new more general definition. But, pleasantly, everything we stated for the special concept generalizes. Moreover, the proofs become simpler, since the presence of a parameterized monad structure on T' is used consciously, not proven over and over again without noticing.

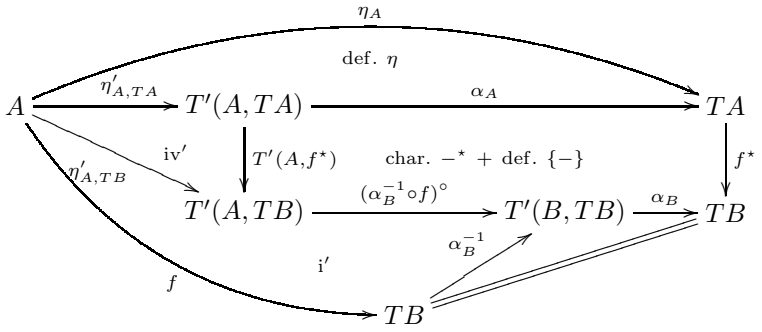
Firstly and foremostly, if an assignment of $T'(A, -)$-algebra to every object A is substitution-carrying, then we have a monad.

Theorem 3.7. *Given a parameterized monad $(T', \eta', -^\circ)$ on a category $\mathcal{C}$. If an assignment (T, α) of an $T'(A, -)$-algebra to every $\mathcal{C}$-object A is substitution-carrying, then $(T, \eta, -^\star)$ is a monad.*

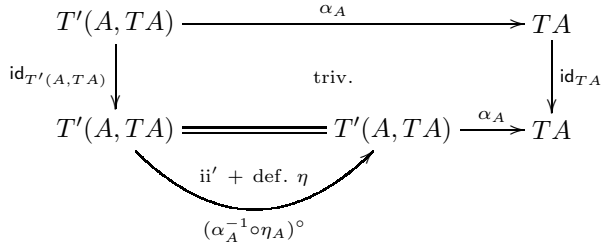
Proof. Assume that (T, α) is substitution-carrying. That $(T, \eta, -^\star)$ satisfies the monad laws is verified as follows.

[2]The iso requirement can be avoided at the cost of switching to a somewhat less intuitive setup of definitions and statements.

(i) The outer triangle of the following diagram commutes for any $f : A \to TB$, since the inner polygons commute.

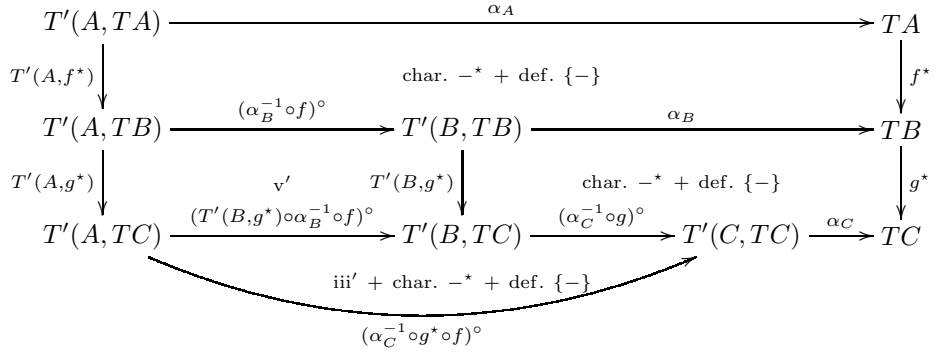


(ii) The outer square of the following diagram commutes for any A, as the inner polygons do.



Hence, by the characterization of $-^\star$ and definition of $\{-\}$, ${\eta_A}^\star = \mathsf{id}_{TA}$.

(iii) The outer square of the following diagram commutes for any $f : A \to TB$, $g : B \to TC$, as the inner polygons do.



Hence, by the characterization of $-^\star$ and definition of $\{-\}$, $(g^\star \circ f)^\star = g^\star \circ f^\star$. □

Further, the initial $T'(A,-)$-algebras give a monad.

Theorem 3.8. *If $\mathcal{C}$ has an initial $T'(A,-)$-algebra for every object A, then (T,α) defined by*

$$(TA,\alpha_A) = (\mu(T'(A,-)),\mathsf{in}_{T'(A,-)})$$

is substitution-carrying.

Proof. The substitution operation $-^\star$ is very straightforwardly defined by

$$f^\star = \mathsf{It}_{T'(A,-)}(\,\{f\}\,) \quad \text{for } f : A \to TB.$$

For a given f, the right hand side is by the definition of iteration (initiality) the unique solution in h of the square

$$\begin{array}{ccc} T'(A,TA) & \xrightarrow{\alpha_A} & TA \\ {\scriptstyle T'(A,h)}\downarrow & & \downarrow{\scriptstyle h} \\ T'(A,TB) & \xrightarrow{\{f\}} & TB \end{array}$$

which is also the square that $f^\star$ is supposed to be the unique solution of by the characterization of substitution. □

A monad is also given by the inverses of the final $T'(A,-)$-coalgebras.

Theorem 3.9 (Generalization of the substitution theorem)**.** *If $\mathcal{C}$ has a final $T'(A,-)$-coalgebra for every object A, then (T,α) defined by*

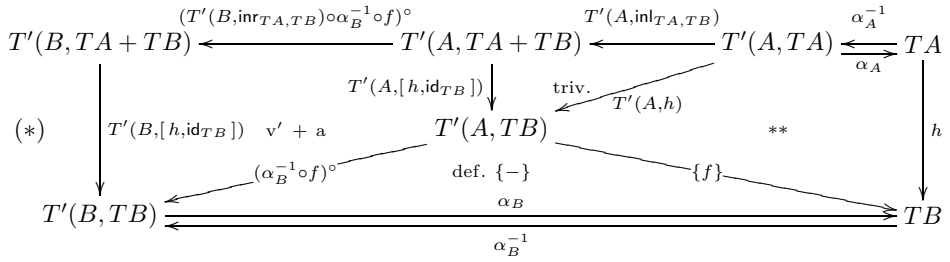
$$(TA,\alpha_A) = (\nu(T'(A,-)),\mathsf{out}^{-1}_{T'(A,-)})$$

is substitution-carrying.

Proof. The substitution operation $-^\star$ is defined by primitive corecursion by

$$f^\star = \mathsf{Corec}_{T'(B,-+TB)}(\,(T'(B,\mathsf{inr}_{TA,TB})\circ\alpha_B^{-1}\circ f)^\circ \circ T'(A,\mathsf{inl}_{TA,TB})\circ\alpha_A^{-1}\,) \quad \text{for } f : A \to TB.$$

The right-hand side is, for any given f, by the characterization of primitive corecursion, the unique solution in h of the outer square in the diagram



The left-hand side, *i.e.*, $f^\star$, must, by the characterization of substitution, be the unique solution in h of the inner square marked (**). The triangle

(a)
$$\begin{array}{ccc} T'(B,TA+TB) & \xleftarrow{T'(B,\mathsf{inr}_{TA,TB})} & T'(B,TB) \\ {\scriptstyle T'(B,[\,h,\mathsf{id}_{TB}\,])}\downarrow & \overset{\text{triv.}}{\diagdown\!\!\diagdown} & \\ T'(B,TB) & & \end{array}$$

commutes. Hence, for any h, (**) commutes if and only if the outer square of (*) does. □

Being a generalization of the monad of non-wellfounded H-terms, the monad from the inverses of the final $T'(A,-)$-coalgebras has the extra structure of being completely iterative with respect to an appropriate notion of guarded substitution rule.

Definition 3.10. Given a substitution-carrying assignment (T,α) of a $T'(A,-)$-algebra to every $\mathcal{C}$-object A, we say that a substitution rule $f : A \to T(A+B)$ is *guarded*, if it factors in the canonical way

$$\begin{array}{ccccc} A & & \xrightarrow{\qquad f \qquad} & & T(A+B) \\ & {\scriptstyle\varphi}\searrow & & \nearrow{\scriptstyle\{\eta_{A+B}\circ\mathsf{inr}_{A,B}\}} & \\ & & T'(B,T(A+B)) & & \end{array}$$

through some morphism $\varphi : A \to T'(B,T(A+B))$.

Theorem 3.11 (Generalization of the solution theorem)**.** *Let (T,α) be defined as in Theorem* 3.9*. The monad $(T,\eta,-^\star)$ is completely iterative with respect to the guardedness notion of Definition* 3.10*.*

Proof. The operation $\tilde{-}$ is definable by

$$\tilde{f} = \mathsf{Coit}_{T'(B,-)}\left(\left[\varphi,\eta'_{B,T(A+B)}\right]^\circ \circ \alpha^{-1}_{A+B}\right) \quad \text{for } f : A \to T(A+B) \text{ guarded.}$$

The right-hand side of this definition is, for any f, by the characterization of coiteration (finality), the unique solution h of the outer square of the diagram

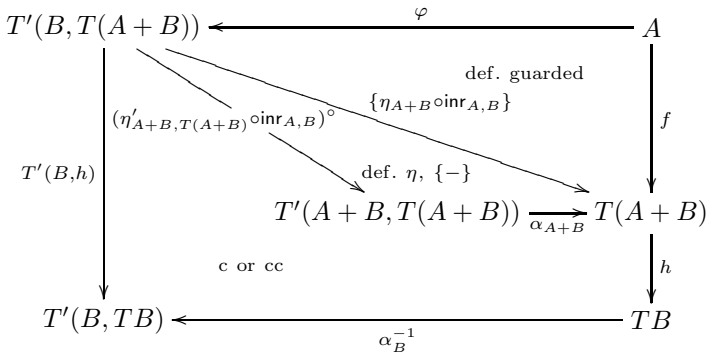
(∗)
$$\begin{array}{ccccc} T'(B,T(A+B)) & \xleftarrow{[\varphi,\eta'_{B,T(A+B)}]^\circ} & T'(A+B,T(A+B)) & \underset{\alpha_{A+B}}{\overset{\alpha^{-1}_{A+B}}{\rightleftarrows}} & T(A+B) \\ \downarrow{\scriptstyle T'(B,h)} & \text{v' + a + b} & \downarrow{\scriptstyle T'(A+B,h)} & ** & \downarrow{\scriptstyle h} \\ T'(B,TB) & \xleftarrow{[\alpha_B^{-1}\circ h\circ f,\eta'_{B,TB}]^\circ} & \text{def. } \eta,\ \{-\} & \xrightarrow{\{[h\circ f,\eta_B]\}} & TB \\ & & \underset{\alpha_B^{-1}}{\overset{\alpha_B}{\rightleftarrows}} & & \end{array}$$

The left-hand side is, if f is guarded, expected to be the unique solution in h of the equation

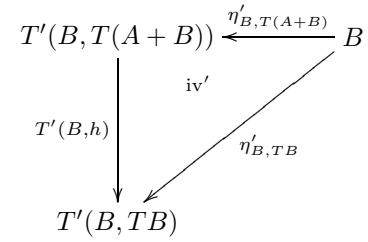
$$h = [\, h \circ f, \eta_B \,]^\star$$

which is equivalent, by the characterization of substitution, to the square marked (**). It therefore suffices to show that, for any h, the outer square of (*) commutes if and only (**) commutes. This will be the case if the outer square of diagram (a) and diagram (b) below commute both if the outer square of (*) commutes and if (**) commutes.

(a)

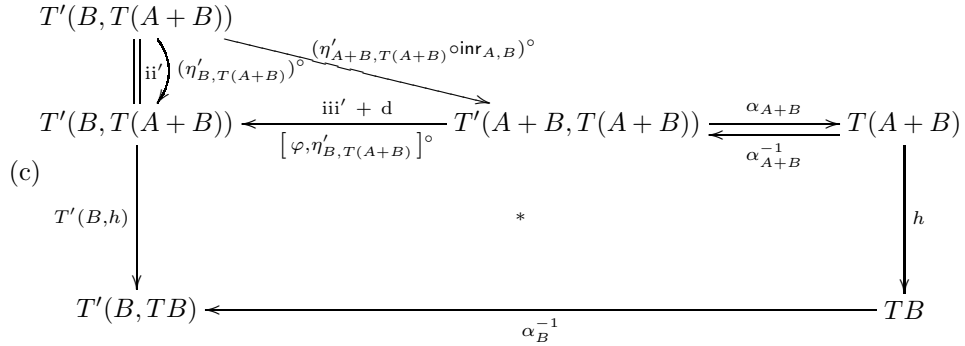


(b)

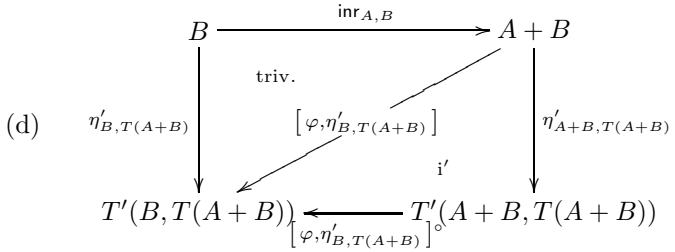


If the outer square of (*) commutes, then the outer square of (a) commutes because the outer square of

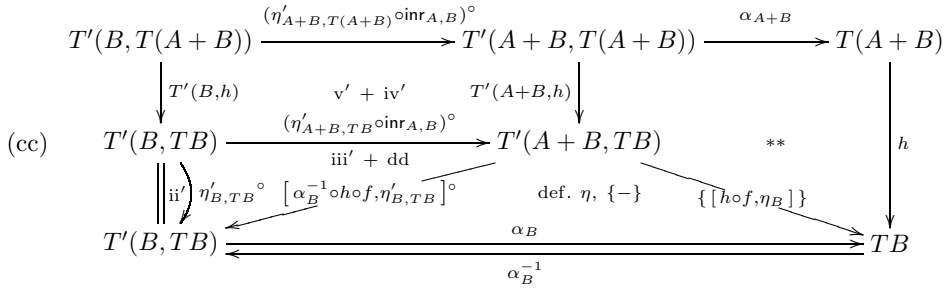
(c)



commutes, since so does the outer square of

(d)


If (**) commutes, then the outer square of (a) because of the commutation of the outer square of

(cc)


which, in turn, is a consequence of the commutation of the outer square of

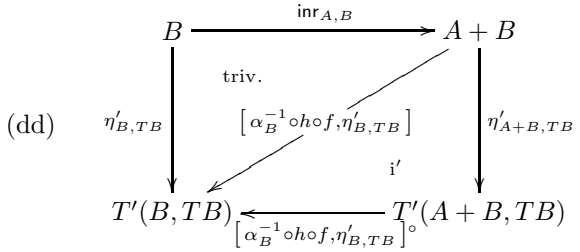
(dd)


Diagram (b) commutes always. □

With generalized substitution and generalized guardedness defined and two kinds of families of algebras shown to carry the structure of a monad resp. completely iterative monad with generalized substitution as the extension operation, we can now finish the section by looking at the examples we get from Examples 3.3–3.5 of parameterized monads.

Example 3.12. To start with, ordinary substitution in wellfounded and non-wellfounded terms is the first example of generalized substitution: if H is an endofunctor on a category $\mathcal{C}$ with finite coproducts, then the well-known monad

structure on T given by $TA = \mu(A + H-)$ (Th. 2.3) and the less well known completely iterative monad structure on T given by $TA = \nu(A + H-)$ (Th. 2.5 and 2.8) are both rooted in the parameterized monad structure on T' given by $T'(A, X) = A + HX$.

Example 3.13. If H is a contravariant endofunctor on a cartesian closed category $\mathcal{C}$, then the parameterized monad structure on T' given by $T'(A, X) = HX \Rightarrow A$ yields a monad structure on T given by $TA = \mu(H- \Rightarrow A)$ and a completely iterative monad structure on T given by $TA = \nu(H- \Rightarrow A)$.

In the special case of $HX = X \Rightarrow E$ for a fixed $\mathcal{C}$-object E, TA is, for a given A, the inductive resp. coinductive space of hyperfunctions from E to A in the sense of Krstić, Launchbury and Pavlović [12] (they wanted the inductive and the coinductive version to collapse into one assuming a restricted form of algebraic compactness, but this is not absolutely necessary). Hyperfunction spaces, with applications in the semantics of processes, have some properties remarkably similar to function spaces: denote the hyperfunctions from A to B by $[A, B]$, one can define identity and composition θ, $\#$ with the usual typing $\theta_A : 1 \to [A, A]$, $\# : [B, C] \times [A, B] \to [A, C]$ and the standard properties, but also an application-like operation $\cdot$ with the typing $\cdot : [A, B] \times [B, A] \to B$ that has the properties $(h\#g)\cdot f = h\cdot(g\#f)$ and $\theta_A \cdot f = f \cdot \theta_A$. There is also an operation lifting functions to hyperfunctions with sensible properties.

Example 3.14. If $X \mapsto (HX, e_X, m_X)$ is a functor from a category $\mathcal{C}$ with finite products to **Monoid**($\mathcal{C}$), then the parameterized monad structure on T' given by $T'(A, X) = A \times HX$ yields a monad structure on T given by $TA = \mu(A \times H-)$ and a completely iterative monad structure on $TA = \nu(A \times H-)$.

If $\mathcal{C}$ has list objects, then, in the special case of $HX = (\mathsf{List}X, \mathsf{nil}_X, \mathsf{append}_X)$, TA is, for a given A the object of wellfounded resp. non-wellfounded A-labeled finitely branching trees. These may be thought of as Böhm trees corresponding to lambda-terms without lambdas (purely applicative terms) over A. Remarkably, the generalized substitution operation delivered by our construction agrees with substitution in Böhm trees: the substitution function $f^\star : TA \to TB$ that corresponds to a given substitution rule $f : A \to TB$ is specified by the equation

$$f^\star(x(t_1, \ldots t_n)) = y(s_1, \ldots, s_m, f^\star(t_1), \ldots, f^\star(t_n)) \text{ where } f(x) = y(s_1, \ldots, s_m).$$

This is exactly how substitution is done in Böhm trees normally.

4. Related work

The study of the structure of the algebras of rational and non-wellfounded H-terms was started by Elgot and colleagues [8,9] in the universal-algebra setting, where $\mathcal{C} = \mathbf{Set}$ and H is polynomial.

Moss [17] and Aczel *et al.* [2] switched to an arbitrary $\mathcal{C}$ with finite coproducts and arbitrary H and proved that the algebras of non-wellfounded H-terms, defined as the inverses of the final $(A + H-)$-coalgebras (assuming they exist), yield

a completely iterative monad, *i.e.*, carry substitution and iteration for all guarded substitution rules (the substitution and solution theorems). Aczel *et al.* [2] also showed that this monad is, in fact, the free completely iterative monad generated by H, and more generally, for any monoidal category $(\mathcal{D}, I, \otimes)$ with finite coproducts and any $\mathcal{D}$-object E, the inverse of the final $(I + E \otimes -)$-coalgebra (assuming it exists) gives the free completely iterative monoid generated by E (the original statement pertains to the special case $\mathcal{D} = [\mathcal{C}, \mathcal{C}]$, $E = H$).

Milius [16] showed that, if H generates a free completely iterative monad, then it must be given by the inverses of the final $(A + H-)$-coalgebras, so they must then exist.

For $\mathcal{C}$ strongly locally finitely presentable and H finitary and monos-preserving, Adámek *et al.* [3, 4] showed that the algebras of rational H-terms exist and give the free iterative monad generated by H, *i.e.*, the free one among the monads from H-algebras that carry substitution and iteration for finitary guarded substitution rules.

Ghani *et al.* [11] considered even term graphs in addition to non-wellfounded and rational terms.

It was a surprise to the author to discover that the result of [19] and the present paper about the initial algebras and the inverses of the final coalgebras of the partial applications of a parameterized monad giving monads appears anticipated in unpublished notes by Paterson on monads for functional programming [18, Sect. 5], with reference to Meijer, personal communication. Unfortunately, however, Meijer and Paterson's statement is entirely unmathematically formulated, the assumptions made are not spelled out. Their construction uses type and term level general recursion, which are legitimate in **CPO**-like algebraically compact categories (categories where the initial algebra and the inverse of the final coalgebra of a functor always coincide). The generalized solution theorem is not in [18].

Advanced disciplined recursion and corecursion schemes as useful tools for constructing morphisms from carriers of initial algebras resp. to carriers of final coalgebras as necessarily uniquely existing solutions to certain equations have been promoted by Uustalu *et al.* [20, 22], Bartels [6], and Cancila *et al.* [7]. (Lenisa [13] required considerably less from such schemes.)

Uustalu and Vene [21] have pointed out that the duals of terms, non-wellfounded terms and substitution for variables, *i.e.*, of wellfounded and non-wellfounded trees with cuts and grafting into cuts, are non-wellfounded resp. wellfounded decorated trees and redecoration. In particular, while guarded substitution rules for non-wellfounded terms can be iterated, guarded redecoration rules for wellfounded decorated trees define new redecoration rules as uniquely existing solutions to a fixedpoint equation. In practice, this fact about guarded redecoration rules is routinely, but without noticing exploited in programming upwards accumulations. The issue of what "cosyntax" should be about in more closer-to-syntax terms has intrigued several authors, among them Ghani *et al.* [10].

5. Conclusions and future work

We showed that the notion of substitution as a unique operation satisfying certain equational conditions is naturally generizable from terms (incomplete trees of fixed branching factor) to hyperfunctions, finitely or possibly infinitely branching decorated trees, etc. For the generalized notion, the proofs of the substitution and solution theorems came out more transparent than those for the original notion, as the inessential specific details did not show up. In the proof of the substitution theorem, further economy was achieved by using primitive corecursion. This hints that a good proportion of the effort in the proof from the basics goes on justifying a specific application of primitive corecursion, unrecognized as such.

Future work includes extending the substitution and solution theorems to signatures with binding and explicit substitution operators (joint project of the author with Ralph Matthes, the substitution theorem appears in [15]), proving the free completely iterative monad theorem of [2] for generalized substitution (this will take generalizing their concept of ideal monad), a study of the pragmatics of generalized redecoration, and also a detailed comparison of the concept of substitution-carrying monad employed here to that of coalgebraic monad by Ghani *et al.* [11].

Acknowledgements. The author is thankful to Peter Hancock for a valuable insight that directed him to the present topic and to Falk Bartels, Neil Ghani, Ralph Matthes and Varmo Vene for technical discussions. He is also grateful to his referees for their helpful criticism and suggestions.

References

[1] P. Aczel, Algebras and coalgebras, in *Revised Lectures from Int. Summer School and Wksh. on Algebraic and Coalgebraic Methods in the Mathematics of Program Construction, ACMMPC 2000 (Oxford, April 2000)*, edited by R. Backhouse, R. Crole and J. Gibbons. Springer-Verlag, *Lecture Notes in Comput. Sci.* **2297** (2002) 79–88.

[2] P. Aczel, J. Adámek, S. Milius and J. Velebil, Infinite trees and completely iterative theories: a coalgebraic view. *Theor. Comput. Sci.* **300** (2003) 1–45.

[3] J. Adámek, S. Milius and J. Velebil, Free iterative theories: a coalgebraic view. *Math. Struct. Comput. Sci.* **13** (2003) 259–320.

[4] J. Adámek, S. Milius and J. Velebil, On rational monads and free iterative theories, in *Proc. of 9th Int. Conf. on Category Theory and Computer Science, CTCS'02 (Ottawa, Aug. 2002)*, edited by R. Blute and P. Selinger. Elsevier, *Electron. Notes Theor. Comput. Sci.* **69** (2003).

[5] M. Barr, Coequalizers and free triples. *Math. Z.* **116** (1970) 307–322.

[6] F. Bartels, Generalized coinduction. *Math. Struct. Comput. Sci.* **13** (2003) 321–348.

[7] D. Cancila, F. Honsell and M. Lenisa, Generalized coiteration schemata, in *Proc. of 6th Wksh. on Coalgebraic Methods in Computer Science, CMCS'03 (Warsaw, Apr. 2003)*, edited by H.P. Gumm. Elsevier, *Electron. Notes Theor. Comput. Sci.* **82** (2003).

[8] C.C. Elgot, Monadic computation and iterative algebraic theories, in *Proc. of Logic Colloquium '73 (Bristol, July 1973)*, edited by H.E. Rose and J.C. Shepherdson. North-Holland, *Stud. Logic Found Math.* **80** (1975) 175–230.

[9] C.C. Elgot, S.L. Bloom and R. Tindell, On the algebraic structure of rooted trees. *J. Comput. Syst. Sci.* **16** (1978) 362–399.

[10] N. Ghani, C. Lüth, F. de Marchi and J. Power, Dualising initial algebras. *Math. Struct. Comput. Sci.* **13** (2003) 349–370.
[11] N. Ghani, C. Lüth and F. de Marchi, Coalgebraic monads, in *Proc. of 5th Wksh. on Coalgebraic Methods in Computer Science, CMCS'02 (Grenoble, Apr. 2001)*, edited by L.S. Moss. Elsevier, *Electron. Notes Theor. Comput. Sci.* **65** (2002).
[12] S. Krstić, J. Launchbury and D. Pavlović, Categories of processes enriched in final coalgebras, in *Proc. of 4th Int. Conf. on Foundations of Software Science and Computation Structures, FoSSaCS'01 (Genova, Apr. 2001)*, edited by F. Honsell and M. Miculan. Springer-Verlag, *Lecture Notes in Comput. Sci.* **2030** (2001) 303–317.
[13] M. Lenisa, From set-theoretic coinduction to coalgebraic coinduction: some results, some problems, in *Proc. of 2nd Wksh. on Coalgebraic Methods in Computer Science, CMCS'99 (Amsterdam, March 1999)*, edited by B. Jacobs and J. Rutten. Elsevier, *Electron. Notes Theor. Comput. Sci.* **19** (1999).
[14] E.G. Manes, *Algebraic theories*, *Graduate Texts in Mathematics* **26**. Springer-Verlag, New York (1976).
[15] R. Matthes and T. Uustalu, Substitution in non-wellfounded syntax with variable binding, in *Proc. of 6th Wksh. on Coalgebraic Methods in Computer Science, CMCS'03 (Warsaw, Apr. 2003)*, edited by H.P. Gumm. Elsevier, *Electron. Notes Theor. Comput. Sci.* **82** (2003).
[16] S. Milius, On iteratable endofunctors, in *Proc. of 9th Int. Conf. on Category Theory and Computer Science, CTCS'02 (Ottawa, Aug. 2002)*, edited by R. Blute and P. Selinger. Elsevier, *Electron. Notes Theor. Comput. Science* **69** (2003).
[17] L.S. Moss, Parametric corecursion. *Theor. Comput. Sci.* **260** (2001) 139–163.
[18] R. Paterson, *Notes on monads for functional programming*, unpublished draft (1995).
[19] T. Uustalu, (Co)monads from inductive and coinductive types, in *Proc. of 2001 APPIA-GULP-PRODE Joint Conf. on Declarative Programming, AGP'01 (Évora, Sept. 2001)*, edited by L.M. Pereira and P. Quaresma. Dep. de Informática, Univ. do Évora (2001) 47–61.
[20] T. Uustalu and V. Vene, Primitive (co)recursion and course-of-value (co)iteration, categorically. *Informatica* **10** (1999) 5–26.
[21] T. Uustalu and V. Vene, The dual of substitution is redecoration, in *Trends in Functional Programming 3*, edited by K. Hammond and S. Curtis. Intellect, Bristol & Portland, OR (2002) 99–110.
[22] T. Uustalu, V. Vene and A. Pardo, Recursion schemes from comonads. *Nordic J. Comput.* **8** (2001) 366–390.

Received December 5, 2002. Accepted in final form August 13, 2003.