Substitution in Non-wellfounded Syntax with Variable Binding

Ralph Matthes [1]

Institut für Informatik der Ludwig-Maximilians-Universität München
Oettingenstraße 67, D-80538 München, Germany

Tarmo Uustalu [2,3]

Institute of Cybernetics at Tallinn Technical University
Akadeemia tee 21, EE-12618 Tallinn, Estonia

Abstract

Inspired from the recent developments in theories of non-wellfounded syntax (coinductively defined languages) and of syntax with binding operators, the structure of algebras of wellfounded and non-wellfounded terms is studied for a very general notion of signature permitting both simple variable binding operators as well as operators of explicit substitution. This is done in an extensional mathematical setting of initial algebras and final coalgebras of endofunctors on a functor category. In the non-wellfounded case, the fundamental operation of substitution is more beneficially defined in terms of primitive corecursion than coiteration.

Keywords: substitution, non-wellfounded syntax, variable binding, monad, functor category, final coalgebra, primitive corecursion

1 Introduction

Moss [26], Aczel, Adámek et al. [5,4], and Ghani et al. [17,16] have recently given a rather complete categorical analysis of *non-wellfounded* syntax, i.e.,

[1] matthes@informatik.uni-muenchen.de

[2] Supported by the Portuguese Foundation for Science and Technology under grant No. PRAXIS/C/EEI/14172/98 and by the Estonian Science Foundation under grant No. 4155. The work was largely done while the author was with Dep. de Informática, Universidade do Minho, Braga, Portugal. His visit to LMU München in Jan. 2002 was financed by the Graduiertenkolleg "Logik in der Informatik" of the Deutsche Forschungsgemeinschaft. Attending ETAPS 2003 was made possible by a travel grant from the Estonian Information Technology Foundation.

[3] tarmo@cs.ioc.ee

languages coinductively determined by universal-algebraic signatures. Fiore, Plotkin and Turi [15], at the same time, have provided a categorical account of syntax with *variable binding* à la de Bruijn [13] building on a suitable generalization of universal algebra—the theory of binding algebras [3,29,31]. In both lines of work, one of the first things done is checking that all is well with *substitution*. The language induced by a signature has to carry a unique operation exhibiting what are, in the setting studied, considered to be the characteristic properties of substitution. These properties have to guarantee that the operation, whenever uniquely existing, verifies what are known as the syntactic substitution lemmata or, in categorical terms, the laws of a monad.

In the present article, we take a step towards combining these two directions of categorical analysis of syntax. In the setting of initial algebras and final coalgebras of endofunctors on a functor category, we look at substitution in both wellfounded and non-wellfounded syntax for a very general notion of signature allowing, in addition to simple variable binding operators, also explicit substitution operators (as an example, we consider what we call the explicit flattening operator). The technical contribution of the work consists of definitions of heterogeneous signature and substitution system for a heterogeneous signature, and proofs that a substitution system for a heterogeneous signature always gives a monad and both the wellfounded and non-wellfounded syntax given by a heterogeneous signature form a substitution system. The latter proofs are made short by appealing to "generalized iteration" (a version of the generalized folds scheme of [11]) and primitive corecursion as pre-justified principles for constructing unique morphisms from and to carriers of initial algebras and final coalgebras.

The article is largely motivated by our interest in the design of useful typed lambda calculi supporting inductive and coinductive constructors of higher kinds. Any reasonable such calculus should certainly be good for representing and manipulating syntax with variable binding; this is one of the obvious applications to try. We are therefore specifically interested in constructions working well also in type-theoretical systems where the primary concern is intensional reductions rather than extensional equality. There, a program employing an advanced recursion or corecursion scheme often exhibits a reduction behavior very different from a version relying on an extensionally valid reduction to iteration or coiteration (a well-known example is programming the number-theoretic predecessor function: one has to use primitive recursion to achieve the desirable reduction behavior, iteration is not enough). These issues will be discussed in detail elsewhere.

As related work, we mention the following. The rank 2 inductive constructor representation of the lambda calculus syntax in the de Bruijn version in either a functional language or a typed lambda calculus is implicit in [9] and appears explicated in [12,7]. In the functional programming community, also the general theory of heterogeneous, non-uniform or nested datatypes (recursive constructors of rank 2) is currently on the research agenda, see

[10,11,20,27]. Typed lambda calculi featuring heterogeneous and higher kind inductive and coinductive constructors are the topic of [23,1,2]. A very accessible extended presentation of aspects of Fiore, Plotkin and Turi's work [15] appears in [14]. Hofmann [21] has given a category-theoretic explanation of the higher-order abstract syntax approach to variable binding [19,28].

To defend our engagement with non-wellfounded syntax, we also refer to some uses of this flavor of syntax. In proof theory, Mints' work [24] from the 1970s on the normalization of infinite derivations (continuous normalization) has recently been revived by him and others [25,6]. In rewriting, ongoing work on infinitary or coinductive lambda calculus [22,30] explains aspects of the model theory of the ordinary lambda calculus. Finally, also the syntax of Girard's Ludics [18], the language of designs, is non-wellfounded (think of FAX).

The article is structured as follows. We begin in Sect. 2 by reviewing the necessary preliminaries: generalized iteration, primitive corecursion, and some specifics about initial algebras and final coalgebras of endofunctors on functor categories. In Sect. 3, we recapitulate the known facts that a substitution system for a universal-algebraic signature gives a monad and that both the wellfounded and non-wellfounded syntax given by a universal-algebraic signature form a substitution system. In Sect. 4 (the central section), we define heterogeneous signatures and substitution systems for heterogeneous signatures and reprove the statements of Sect. 3 for these concepts. In Sect. 5, we list some conclusions and goals for future work.

2 Preliminaries

We begin by reviewing generalized iteration, primitive corecursion, and some facts about initial algebras and final coalgebras of endofunctors on functor categories.

2.1 *Generalized iteration, primitive corecursion*

For an endofunctor F on a category $\mathcal{C}$, we let $(\mu F, \mathsf{in}_F)$ denote its initial algebra (if it exists) and $(\nu F, \mathsf{out}_F)$ denote its final coalgebra (if it exists). Iteration and coiteration, the basic principles for constructing unique morphisms from μF and to νF, are immediate consequences from the initiality resp. finality of $(\mu F, \mathsf{in}_F)$ and $(\nu F, \mathsf{out}_F)$. *Iteration* says that, for any $\mathcal{C}$-morphism $\varphi : FX \to X$, there exists a unique $\mathcal{C}$-morphism $h : \mu F \to X$, denoted $\mathsf{It}_F(\varphi)$, such that

$$\begin{tikzcd} F(\mu F) \arrow[r, "\mathsf{in}_F"] \arrow[d, "Fh"'] & \mu F \arrow[d, "h"] \\ FX \arrow[r, "\varphi"'] & X \end{tikzcd}$$

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

$$\begin{tikzcd} FX \arrow[d, "Fh"'] & X \arrow[l, "\varphi"'] \arrow[d, "h"] \\ F(\nu F) & \nu F \arrow[l, "\mathsf{out}_F"] \end{tikzcd}$$

Often, however, it is practical to make use of more advanced recursion and corecursion schemes whose validity is not entirely immediate. We shall need *"generalized iteration"*, which states the following. Given an endofunctor G on a category $\mathcal{C}'$, a functor $L : \mathcal{C} \to \mathcal{C}'$ with a right adjoint R and a natural transformation $\theta : L \cdot F \to G \cdot L$ between functors from $\mathcal{C}$ to $\mathcal{C}'$. Then, for any $\mathcal{C}'$-morphism $\varphi : GX \to X$, there exists a unique $\mathcal{C}'$-morphism $h : L(\mu F) \to X$, denoted $\mathsf{It}^{L,\theta}_{F,G}(\,\varphi\,)$, such that

$$\begin{tikzcd} L(F(\mu F)) \arrow[r, "L\mathsf{in}_F"] \arrow[d, "\theta_{\mu F}"'] & L(\mu F) \arrow[dd, "h"] \\ G(L(\mu F)) \arrow[d, "Gh"'] & \\ GX \arrow[r, "\varphi"'] & X \end{tikzcd}$$

The h characterized by the property above is

$$\mathsf{It}^{L,\theta}_{F,G}(\,\varphi\,) = \varepsilon_X \circ L\ \mathsf{It}_F(\,R(\varphi \circ G\varepsilon_X \circ \theta_{RX}) \circ \eta_{F(RX)}\,)$$

where η is the unit and ε the counit of the adjunction.

In [11], section 6.2, a slightly more general result is shown where G, θ and φ are replaced by a natural transformation $\Psi : \mathcal{C}'(L-, X) \to \mathcal{C}'(L(F-), X)$ between functors $\mathcal{C}^{\mathrm{op}} \to \mathbf{Set}$ (the particular Ψ corresponding to our G, θ, φ is defined by $\Psi(f) = \varphi \circ Gf \circ \theta_A$ for $f : LA \to X$). On the other hand, our decomposition hints at how to find examples and resembles more the traditional-style iteration in that it refers to a G-algebra structure φ for some functor G (where in most examples, $G \neq F$).

We shall also make use of *primitive corecursion*, see, e.g., [33]. If $\mathcal{C}$ has binary sums, then for any $\mathcal{C}$-morphism $\varphi : X \to F(X + \nu F)$, there exists a unique $\mathcal{C}$-morphism $h : X \to \nu F$, denoted $\mathsf{Corec}_F(\,\varphi\,)$, such that

$$\begin{tikzcd} F(X + \nu F) \arrow[d, "{F[\,h, \mathsf{id}_{\nu F}\,]}"'] & X \arrow[l, "\varphi"'] \arrow[d, "h"] \\ F(\nu F) & \nu F \arrow[l, "\mathsf{out}_F"] \end{tikzcd}$$

The one and only h with the requested property is

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

2.2 Initial algebras, final coalgebras of partial applications of bifunctors vs. of functors on functor categories

Given a functor $F : \mathcal{C} \times \mathcal{D} \to \mathcal{D}$, every $\mathcal{C}$-object A determines an endofunctor $F|A$ on $\mathcal{D}$ given by $(F|A)X = F(A, X)$. It is easy to observe that, if initial algebras exist for all of them, then, setting $(\hat{\mu}F)A = \mu(F|A)$, $(\hat{\mathsf{in}}_F)_A = \mathsf{in}_{F|A}$, we get an algebra $(\hat{\mu}F, \hat{\mathsf{in}}_F)$ of an endofunctor $[F]$ on the functor category $[\mathcal{C}, \mathcal{D}]$ given by $([F]X)A = F(A, XA)$. But this is not all: $(\hat{\mu}F, \hat{\mathsf{in}}_F)$ is, in fact, an initial algebra of $[F]$, with the A-component of the iterative extension of any given $[F]$-algebra (X, φ) given as the iterative extension of the $(F|A)$-algebra (XA, φ_A).

Under a reasonable additional condition, this existence result also holds in the opposite direction. Provided that $\mathcal{C}$ is locally small and $\mathcal{D}$ has powers indexed by homsets of $\mathcal{C}$, if an initial $[F]$-algebra exists, then $((\mu[F])A, (\mathsf{in}_{[F]})_A)$ are initial $(F|A)$-algebras; the iterative extension of an $(F|A)$-algebra (X, φ) is constructed as $\pi_X(\mathsf{id}_A) \circ (\mathsf{It}_{[F]}(\,\bar{\varphi}\,))_A$ where $\bar{\varphi}$ is an $[F]$-algebra structure on $\prod_{f \in \mathcal{C}(-,A)} X$ defined by $(\bar{\varphi})_{A'} = \langle \varphi \circ F(f, \pi_X(f)) \rangle_{f \in \mathcal{C}(A',A)}$.

Dual statements may be made about final coalgebras for $F|A$ vs. $[F]$. The existence of final $(F|A)$-coalgebras follows from the existence of a final $[F]$-coalgebra in case $\mathcal{C}$ is locally small and $\mathcal{D}$ has copowers indexed by homsets of $\mathcal{C}$.

2.3 A special case of generalized iteration

For an endofunctor F on a functor category $[\mathcal{C}, \mathcal{D}]$ with an initial algebra, the following recursion scheme is a special case of generalized iteration. Given an endofunctor G on a functor category $[\mathcal{C}', \mathcal{D}]$, a functor $Z : \mathcal{C}' \to \mathcal{C}$ such that the reduction functor $- \cdot Z : [\mathcal{C}, \mathcal{D}] \to [\mathcal{C}', \mathcal{D}]$ has a right adjoint (the right Kan extension $\mathit{Ran}_Z\ Y$ along Z exists for any functor $Y : \mathcal{C}' \to \mathcal{D}$), and a natural transformation $\theta : (F-) \cdot Z \to G(- \cdot Z)$ between functors $[\mathcal{C}, \mathcal{D}] \to [\mathcal{C}', \mathcal{D}]$. Then, for any $[\mathcal{C}', \mathcal{D}]$-morphism $\varphi : GX \to X$, there exists a unique $[\mathcal{C}', \mathcal{D}]$-morphism $h : \mu F \cdot Z \to X$

$$\begin{tikzcd}
F(\mu F) \cdot Z \arrow[r, "\mathsf{in}_F \cdot Z"] \arrow[d, "\theta_{\mu F}"'] & \mu F \cdot Z \arrow[dd, "h"] \\
G(\mu F \cdot Z) \arrow[d, "Gh"'] & \\
GX \arrow[r, "\varphi"'] & X
\end{tikzcd}$$

The same kind of specialization is carried out in [11], yielding "generalized folds" which, again, are potentially more general but, as a rule scheme, less informative.

3 Wellfounded and non-wellfounded term algebras

We now proceed to discussing the properties of substitution in wellfounded and non-wellfounded term algebras. Categorically, these are the initial $(A+H-)$-algebras resp. inverses of the final $(A+H-)$-coalgebras for different objects A for an endofunctor H on a category $\mathcal{C}$ (where, in universal algebra, $\mathcal{C} = \mathbf{Set}$ and H is polynomial). For the purposes of modular presentation, however, we first introduce the concept of substitution system, cf. [4]. This concept is usable as a basis for a uniform treatment of not only wellfounded and non-wellfounded terms, but also term equivalence classes (wrt. a system of equations), term graphs, rational terms etc.

Definition 3.1 Given an endofunctor H on a category $\mathcal{C}$ with finite coproducts. For any assignment $A \mapsto (TA, \alpha_A)$ of some $(A+H-)$-algebra to every $\mathcal{C}$-object A, the $|\mathcal{C}|$-indexed family α of morphisms $\alpha_A : A + H(TA) \to TA$ decomposes into two $|\mathcal{C}|$-indexed families η, τ of morphisms $\eta_A : A \to TA$, $\tau_A : H(TA) \to TA$ defined by

$$\eta_A = \alpha_A \circ \mathsf{inl}_{A,H(TA)} \qquad \text{and} \qquad \tau_A = \alpha_A \circ \mathsf{inr}_{A,H(TA)}$$

We say that (T, α) is a *substitution system* for H, if, for every morphism $f : A \to TB$, there exists a unique morphism $h : TA \to TB$, denoted $f^\star$, satisfying

$$\begin{tikzcd} A + H(TA) \arrow[r, "\alpha_A", "(=[\eta_A, \tau_A])"'] \arrow[d, "\mathsf{id}_A + Hh"'] & TA \arrow[d, "h"] \\ A + H(TB) \arrow[r, "{[f, \tau_B]}"] & TB \end{tikzcd} \quad \text{i.e.,} \quad \begin{tikzcd} A \arrow[r, "\eta_A"] \arrow[dr, "f"'] & TA \arrow[d, "h"] & H(TA) \arrow[l, "\tau_A"'] \arrow[d, "Hh"] \\ & TB & H(TB) \arrow[l, "\tau_B"] \end{tikzcd}$$

Intuitively, if an assignment (T, α) of an $(A+H-)$-algebra to every object A is a substitution system, 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.

Having a substitution system implies having a monad: the monad laws are valid properties of substitution.

Theorem 3.2 *If an assignment (T, α) of an $(A+H-)$-algebra to every $\mathcal{C}$-object A forms a substitution system, then $(T, \eta, -^\star)$ is a monad (in Kleisli form).*

Proof. The monad law (i) $f^\star \circ \eta_A = f$ $(f : A \to TB)$ is immediate; the laws (ii) $\eta_A{}^\star = \mathsf{id}_{TA}$, (iii) $(g^\star \circ f)^\star = g^\star \circ f^\star$ $(f : A \to TB,\ g : B \to TC)$ are verified straightforwardly. $\square$

Note that the second and third law are actually the well-known syntactic lemmata of substitution.

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. In the case of wellfounded H-terms, the presence of a substitution system and a monad is immediate.

Theorem 3.3 *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 a substitution system and hence $(T,\eta,-^\star)$ is a monad (as is known since [8], it is even the free monad generated by H).

Proof. (Trivial, but useful to record for comparison with Thms. 3.4, 4.3.) (T,α) is a substitution system 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

$$\begin{tikzcd} A+H(TA) \arrow[r, "\alpha_A"] \arrow[d, "\mathsf{id}_A+Hh"'] & TA \arrow[d, "h"] \\ A+H(TB) \arrow[r, "{[\,f,\tau_B\,]}"] & TB \end{tikzcd}$$

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

The example of non-wellfounded H-terms, investigated in Moss [26] and Aczel et al. [5], is considerably more interesting. Substitution is definable also for non-wellfounded H-terms, but the definition is not as simple any more as for wellfounded H-terms.

Theorem 3.4 (The substitution theorem of [26,5]) *If $\mathcal{C}$ has a final $(A+H-)$-coalgebra for every object A, then (T,α) defined by*

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

is a substitution system.

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

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

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

$$
\begin{tikzcd}
B + H(TA+TB) \arrow[dd, "\mathsf{id}_B + H[h,\mathsf{id}_{TB}]"] & & A + H(TA+TB) \arrow[ll, "{[(\mathsf{id}_B + H\mathsf{inr}_{TA,TB}) \circ \alpha_B^{-1} \circ f, \mathsf{inr}]}"'] \arrow[d, "\mathsf{id}_A + H[h,\mathsf{id}_{TB}]"'] & & A + H(TA) \arrow[ll, "\mathsf{id}_A + H\mathsf{inl}_{TA,TB}"'] \arrow[dll, "\mathsf{id}_A + Hh"] \arrow[r, shift right, "\alpha_A"'] & TA \arrow[l, shift right, "\alpha_A^{-1}"'] \arrow[dd, "h"] \\
(*) & & A + H(TB) \arrow[dll, "{[\alpha_B^{-1} \circ f, \mathsf{inr}]}"] \arrow[drrr, "{[f, \tau_B]}"] & & ** & \\
B + H(TB) \arrow[rrrrr, shift left, "\alpha_B"] & & & & & TB \arrow[lllll, shift left, "\alpha_B^{-1}"]
\end{tikzcd}
$$

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

Substitution is, of course, also definable from the first principles (finality) without making use of primitive corecursion. Notably, however, the correctness proof is then more involved and the complications amount to nothing else than an implicit justification of an instance of primitive corecursion. This means that it adds to the clarity and modularity of the proof, if primitive corecursion is justified first in its generality and only then used. In a type-theoretic system, moreover, only the definition using primitive corecursion gives the right reduction behavior.

4 A generalization for variable binding

While the terms of a universal-algebra signature are definable for all possible supplies of variables independently, this is no longer so in the presence of variable binding operators. If a lambda term over a given supply of variables A is a lambda abstraction, then the body is a lambda term over $1 + A$, not over A: it has (potentially) one free variable more. Hence the lambda terms have to be defined as an inductive family for all possible supplies of variables simultaneously; they are an example of what is called a *heterogeneous* datatype. In category-theoretic terms, this means a shift from a $|\mathcal{C}|$-indexed (functorial) family of initial algebras of endofunctors on some base category $\mathcal{C}$ (normally **Set**) to an initial algebra of an endofunctor on $[\mathcal{C}, \mathcal{C}]$. For lambda calculus, this has been worked out in [12,7] and of course also in [15]. The lambda calculus syntax is the initial algebra of the endofunctor F on $[\mathcal{C}, \mathcal{C}]$ given by $(FX)A = A + XA \times XA + X(1+A)$ or, equivalently, $FX = \mathsf{Id} + X \times X + X \cdot \Delta$ where $\Delta A = 1 + A$: set $(T, \alpha) = (\mu F, \mathsf{in}_F)$, then TA represents the lambda terms with free variables taken from A and $\alpha_A : A + TA \times TA + T(1+A) \to TA$ gives the constructions for variables, application and lambda abstraction. The non-wellfounded version is given by the inverse of the final coalgebra of the same functor F. In Sect. 2.2, we saw that the initial $F(A, -)$-algebras for different A's for $F : \mathcal{C} \times \mathcal{C} \to \mathcal{C}$ given by $F(A, X) = A + KX$ with K an endofunctor on $\mathcal{C}$ are essentially the same as the initial algebra of $F' : [\mathcal{C},\mathcal{C}] \to [\mathcal{C},\mathcal{C}]$ given by $(F'X)A = A + K(XA)$, i.e., $F'X = \mathsf{Id} + K \cdot X$. Hence, wellfounded syntax without variable binding admits both the view discussed in the previous section and an alternative view extensible to treat also the lambda calculus syntax; a similar statement applies to non-wellfounded syntax.

Given these considerations, it is a natural goal to look for a notion of signature based on endofunctors on functor categories which accounts for variable binding and is as general as possible so that the project of the previous section can still be carried out. While a signature in Sec. 3 was any endofunctor K on $\mathcal{C}$ and a substitution system was (essentially) an $(\mathsf{Id} + K \cdot -)$-algebra, here we would rather like to see that a substitution system is an $(\mathsf{Id} + H-)$-algebra where H is some endofunctor on $[\mathcal{C},\mathcal{C}]$ (most likely not every H would be acceptable, but, e.g., H given by $HX = K \cdot X$ should be in order to cover the classical results). One possible "heterogeneous" notion of signature—the proposal of this article—is that of an endofunctor on $[\mathcal{C},\mathcal{C}]$ together with a strength-like additional datum. We give first the definition and then some justification. Recall that a pointed endofunctor on $\mathcal{C}$ is an endofunctor Z on $\mathcal{C}$ together with a natural transformation $e : \mathsf{Id} \to Z$; a pointed functor morphism from (Z, e) to (Z', e') is a natural transformation $f : Z \to Z'$ between endofunctors on $\mathcal{C}$ such that $f \circ e = e'$. We write $\mathbf{Ptd}(\mathcal{C})$ for the category of pointed endofunctors on $\mathcal{C}$ and U for the forgetful functor $\mathbf{Ptd}(\mathcal{C}) \to [\mathcal{C},\mathcal{C}]$.

Definition 4.1 Assume a functor category $[\mathcal{C},\mathcal{C}]$ has finite coproducts and a right adjoint for the functor $- \cdot Z : [\mathcal{C},\mathcal{C}] \to [\mathcal{C},\mathcal{C}]$ for every $[\mathcal{C},\mathcal{C}]$-object Z. Given an endofunctor H on $[\mathcal{C},\mathcal{C}]$ together with a natural transformation $\theta : (H-) \cdot U{\sim} \to H(- \cdot U{\sim})$ between functors $[\mathcal{C},\mathcal{C}] \times \mathbf{Ptd}(\mathcal{C}) \to [\mathcal{C},\mathcal{C}]$ such that

$$\theta_{X,(\mathsf{Id},\mathsf{id}_{\mathsf{Id}})} = \mathsf{id}_{HX}$$

$$\theta_{X,(Z'\cdot Z, e'\cdot e)} = \theta_{X\cdot Z',(Z,e)} \circ (\theta_{X,(Z',e')} \cdot Z)$$

For any $(\mathsf{Id} + H-)$-algebra (T, α), the $[\mathcal{C},\mathcal{C}]$-morphism α decomposes into two $[\mathcal{C},\mathcal{C}]$-morphisms $\eta : \mathsf{Id} \to T$, $\tau : HT \to T$ defined by

$$\eta = \alpha \circ \mathsf{inl}_{\mathsf{Id},HT} \qquad \text{and} \qquad \tau = \alpha \circ \mathsf{inr}_{\mathsf{Id},HT}$$

We call (T, α) a *heterogeneous substitution system* for (H, θ), if, for every $\mathbf{Ptd}(\mathcal{C})$-morphism $f : (Z, e) \to (T, \eta)$, there exists a unique $[\mathcal{C},\mathcal{C}]$-morphism $h : T \cdot Z \to T$, denoted $\{f\}_{(Z,e)}$, satisfying

$$\begin{tikzcd} Z + (HT)\cdot Z \arrow[r, "\alpha\cdot Z"] \arrow[d, "\mathsf{id}_Z + \theta_{T,(Z,e)}"'] & T\cdot Z \arrow[dd, "h"] \\ Z + H(T\cdot Z) \arrow[d, "\mathsf{id}_Z + Hh"'] & \\ Z + HT \arrow[r, "{[f,\tau]}"] & T \end{tikzcd} \quad \text{i.e.,} \quad \begin{tikzcd} Z \arrow[r, "\eta\cdot Z"] \arrow[ddr, "f"'] & T\cdot Z \arrow[dd, "h"] & (HT)\cdot Z \arrow[l, "\tau\cdot Z"'] \arrow[d, "\theta_{T,(Z,e)}"] \\ & & H(T\cdot Z) \arrow[d, "Hh"] \\ & T & HT \arrow[l, "\tau"] \end{tikzcd}$$

This definition appears satisfactory in several ways. First of all, it covers our examples.

- For an endofunctor K on $\mathcal{C}$ seen as a homogeneous signature, the equivalent heterogeneous signature is (H, θ) where $(HX)A = K(XA)$ and $(\theta_{X,(Z,e)})_A = \mathsf{id}_{K(X(ZA))}$.
- The lambda calculus signature is captured in (H, θ) where $(HX)A = XA \times XA + X(1 + A)$ and $(\theta_{X,(Z,e)})_A = \mathsf{id}_{X(ZA)\times X(ZA)} + X[\, e_{1+A} \circ \mathsf{inl}_{1,A}, Z\mathsf{inr}_{1,A}\,]$.

Moreover, we can also capture, e.g., the signature with one "normal" binary operator and an operator of "explicit flattening" (formal flattening) which takes terms whose variables are terms over A to terms over A. For this, one sets $(HX)A = XA \times XA + X(XA)$, $(\theta_{X,(Z,e)})_A = \mathsf{id}_{X(ZA)\times X(ZA)} + Xe_{X(ZA)}$. Both in this and the previous example, it is crucial for the definition of θ that θ is parametrized in a pointed functor (Z, e), not just a functor Z. Checking that the characteristic property of substitution is what one would expect is straightforward in both cases.

By replacing $X(XA)$ by $\int^B \coprod_{f\in\mathcal{C}(B,XA)} XB$, the "explicit flattening" operator can be changed into an "explicit substitution" operator taking a term over some supply of variables B and an assignment of a term over A to every element of B (a substitution rule) to a term over A.

Combination of lambda abstraction and "explicit flattening" is possible as the requirements for θ are modular in the sense that these natural transformations can be provided separately for every summand of H.

There is also an instructive non-example adequately filtered out by Def. 4.1: the example of powerlists (also discussed in [2]). Powerlists, that is, all lists whose length is 2^n for some n, are represented the initial $(\mathsf{Id} + H-)$-algebra for H given by $(HX)A = X(A \times A)$. There are two candidates for θ defined by $(\theta_{X,(Z,e)})_A = X(Z\langle \mathsf{id}_A, \mathsf{id}_A\rangle \circ \pi) : X(ZA \times ZA) \to X(Z(A \times A))$ with π either the left or the right projection, but both fail to meet the first condition on θ. This is, however, how things should be, since what is substitution for lists does not qualify as substitution for powerlists: it does not maintain the constraint that the length of a list may only be a power of 2.

The definition is also good in that every substitution system implies a monad and both the wellfounded and the non-wellfounded syntax generated by a signature are substitution systems. The proof that a substitution system implies a monad reveals why it makes sense to require $\theta_{X,(Z,e)}$ to be defined for any pointed functor (Z, e) instead of just for (T, η) and why it then has to be natural in (Z, e) and satisfy the two conditions.

Theorem 4.2 *If an* $(\mathsf{Id} + H-)$*-algebra* (T, α) *forms a heterogeneous substitution system, then* $(T, \eta, \{\mathsf{id}_T\}_{(T,\eta)})$ *is a monad (in triple form).*

Proof. The law (i) $\mu \circ (\eta \cdot T) = \mathsf{id}_T$ is trivial from $\mu = \{\mathsf{id}_T\}_{(T,\eta)}$ being a solution in h of the diagram above for $(Z, e) = (T, \eta)$, $f = \mathsf{id}_T$. The law (ii) $\mu \circ (T \cdot \eta) = \mathsf{id}_T$ follows from (ii-a) $\mu^{(1)} = \mathsf{id}_T$ and (ii-b) $\mu^{(1)} = \mu \circ (T \cdot \eta)$

where $\mu^{(1)} = \{\eta\}_{(\mathsf{Id},\mathsf{id}_{\mathsf{Id}})}$, which are proved from the diagram having at most one solution in h for $(Z, e) = (\mathsf{Id}, \mathsf{id}_{\mathsf{Id}})$, $f = \eta$. The law (iii) $\mu \circ (\mu \cdot T) = \mu \circ (T \cdot \mu)$ follows from (iii-a) $\mu^{(3)} = \mu \circ (\mu \cdot T)$ and (iii-b) $\mu^{(3)} = \mu \circ (T \cdot \mu)$ where $\mu^{(3)} = \{\mu\}_{(T \cdot T, \eta \cdot \eta)}$. These are proved from the diagram not having more than one solution in h for $(Z, e) = (T \cdot T, \eta \cdot \eta)$, $f = \mu$ (this f is a pointed functor morphism from (Z, e) to (T, η) by (i)).

Importantly, in the proof of (ii-b), one needs $\theta_{T,(T,\eta)} \circ (HT \cdot \eta) = H(T \cdot \eta)$, which follows from the naturality of θ in the second argument and the first of the coherence conditions on θ. In the proof of (iii-b), one needs $\theta_{T,(T,\eta)} \circ (HT \cdot \mu) = H(T \cdot \mu) \circ \theta_{T \cdot T,(T,\eta)} \circ (\theta_{T,(T,\eta)} \cdot T)$, which follows from the naturality of θ in the second argument and the second of the coherence conditions on θ. □

Once generalized iteration in the sense of Sect. 2.3 is accepted, the proof that the wellfounded syntax defined by a heterogeneous signature is a substitution system is as trivial as that of Thm. 3.3. This also motivates the introduction of θ in the first place: in the wellfounded case, we want the the desideratum for substitution expressed in the notion of heterogeneous substitution system to be fulfilled "automatically".

Theorem 4.3 *If $[\mathcal{C}, \mathcal{C}]$ has an initial $(\mathsf{Id} + H-)$-algebra, then (T, α) defined by*

$$(T, \alpha) = (\mu(\mathsf{Id} + H-), \mathsf{in}_{\mathsf{Id}+H-})$$

is a heterogeneous substitution system.

Proof. (T, α) is a heterogeneous substitution system with $\{-\}$ definable by

$$\{f\}_{(Z,e)} = \mathsf{It}^{- \cdot Z, \mathsf{id}_Z + \theta_{-,(Z,e)}}_{\mathsf{Id}+H-, Z+H-}(\,[\,f, \tau\,]\,)$$

since the right-hand side is, for any given f, by the characterization of generalized iteration the unique solution in h of the square

$$\begin{array}{ccc}
Z + (HT) \cdot Z & \xrightarrow{\alpha \cdot Z} & T \cdot Z \\
\downarrow{\scriptstyle \mathsf{id}_Z + \theta_{T,(Z,e)}} & & \\
Z + H(T \cdot Z) & & \downarrow{\scriptstyle h} \\
\downarrow{\scriptstyle \mathsf{id}_Z + Hh} & & \\
Z + HT & \xrightarrow{[f,\tau]} & T
\end{array}$$

and this is the same square that $\{f\}_{(Z,e)}$ is supposed to be the unique solution of in h. □

The proof in the case of non-wellfounded syntax follows that of the corresponding Thm. 3.4 in the previous section very closely.

Theorem 4.4 *If $[\mathcal{C}, \mathcal{C}]$ has a final $(\mathsf{Id} + H-)$-coalgebra, then (T, α) defined by*

$$(T, \alpha) = (\nu(\mathsf{Id} + H-), \mathsf{out}^{-1}_{\mathsf{Id}+H-})$$

is a heterogeneous substitution system.

Proof. $\{-\}$ is definable by

$$\{f\}_{(Z,e)} = \mathsf{Corec}_{\mathsf{Id}+H-}(\,[\,(\mathsf{id}_{\mathsf{Id}} + H\mathsf{inr}_{T\cdot Z,T}) \circ \alpha^{-1} \circ f, \mathsf{inr} \circ H\mathsf{inl}_{T\cdot Z,T} \circ \theta_{T,(Z,e)}\,] \circ(\alpha^{-1} \cdot Z)\,)$$

The right-hand side is, for any given (Z,e) and f, by the characteristic property of primitive corecursion, the unique solution in h of the outer square in the diagram

$$\begin{tikzcd}
 & \mathsf{Id}+H(T\cdot Z+T) \arrow[dd, "{\mathsf{id}_{\mathsf{Id}}+H[h,\mathsf{id}_T]}"] & Z+H(T\cdot Z+T) \arrow[l, "{[(\mathsf{id}_{\mathsf{Id}}+H\mathsf{inr}_{T\cdot Z,T})\circ\alpha^{-1}\circ f,\mathsf{inr}]}"'] \arrow[d, "{\mathsf{id}_Z+H[h,\mathsf{id}_T]}"'] & Z+H(T\cdot Z) \arrow[l, "\mathsf{id}_Z+H\mathsf{inl}_{T\cdot Z,T}"'] \arrow[dl, "\mathsf{id}_Z+Hh"] & T\cdot Z \arrow[l, "(\mathsf{id}_Z+\theta_{T,(Z,e)})\circ(\alpha^{-1}\cdot Z)"'] \arrow[dd, "h"] \\
(*) & & Z+HT \arrow[dl, "{[\alpha^{-1}\circ f,\mathsf{inr}]}"] \arrow[drr, "{[f,\tau]}"] & ** & \\
 & \mathsf{Id}+HT \arrow[rrr, shift left, "\alpha"] & & & T \arrow[lll, shift left, "\alpha^{-1}"]
\end{tikzcd}$$

The left-hand side, i.e., $\{f\}_{(Z,e)}$, must, at the same time, be the unique solution of the inner square marked (**). But (**) commutes for any h if and only if the outer square of (*) does. $\square$

5 Conclusions and future work

We have demonstrated that substitution in wellfounded and non-wellfounded syntax with variable binding does not need much more work than in first-order syntax. We still get substitution monads. Apart from the use of generalized iteration à la [11] and primitive corecursion (as opposed to coiteration), the essential ingredient is a kind of distributivity parametrized in a pointed functor. With a functor parameter, neither lambda calculus nor explicit substitution would have been instances. Had we used a monad instead of a pointed functor, the proofs would have become circular. Hence, the current definition seems to be reasonable.

Certainly, one of the next aims is the study of the benefits of our representation of substitution lambda calculi with monotone inductive and coinductive constructors of rank 2 introduced in [23,1,2]. Another is to prove the solution theorem for non-wellfounded syntax with variable binding.

Acknowledgements

We are grateful to Felix Joachimski for discussions and encouragement and to Andreas Abel for hinting at the anomaly with the powerlists.

References

[1] Abel, A. and R. Matthes, *(Co-)iteration for higher-order nested datatypes*, to appear in: H. Geuvers and F. Wiedijk, eds., Post-Conf. Proc. of IST WG TYPES 2nd Ann. Meeting, TYPES'02 (Berg en Dal near Nijmegen, Apr. 2002), Lect. Notes in Comput. Sci., Springer-Verlag.

[2] Abel, A., R. Matthes and T. Uustalu, *Generalized iteration and coiteration for higher-order nested datatypes*, to appear in: A. Gordon, ed., Proc. of 6th Int. Conf. on Foundations of Software Science and Computation Structures, FoSSaCS'03 (Warsaw, Apr. 2003), Lect. Notes in Comput. Sci., Springer-Verlag.

[3] Aczel, P., *Frege structures and the notions of proposition, truth and set*, in: J. Barwise, H. J. Keisler and K. Kunen, eds., *Proc. of The Kleene Symp. (Madison, WI, June 1978)*, Studies in Logic and Found. of Math. **101**, North Holland, Amsterdam, 1980 pp. 31–59.

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

[5] Aczel, P., J. Adámek and J. Velebil, *A coalgebraic view of infinite trees and iteration*, in: A. Corradini, M. Lenisa and U. Montanari, eds., *Proc. of 4th Wksh. on Coalgebraic Methods in Computer Science, CMCS'01 (Genova, Apr. 2001)*, Electr. Notes in Theoret. Comput. Sci. **44(1)**, Elsevier, Amsterdam, 2001. Journal version with S. Milius as an additional coauthor to appear in Theoret. Comput. Sci.

[6] Aehlig, K. and F. Joachimski, *On continuous normalization*, in: J. C. Bradfield, ed., *Proc. of 16th Int. Wksh. on Computer Science Logic, CSL 2002 (Edinburgh, Sept. 2002)*, Lect. Notes in Comput. Sci. **2471**, Springer-Verlag, Berlin, 2002 pp. 59–73.

[7] Altenkirch, T. and B. Reus, *Monadic presentations of lambda terms using generalized inductive types*, in: J. Flum and M. Rodríguez-Artalejo, eds., *Proc. of 13th Int. Wksh. on Computer Science Logic, CSL'99 (Madrid, Sept. 1999)*, Lect. Notes in Comput. Sci. **1683**, Springer-Verlag, Berlin, 1999 pp. 453–468.

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

[9] Bellegarde, F. and J. Hook, *Substitution: A formal methods case study using monads and transformations*, Sci. of Comput. Program. **23** (1994), pp. 287–311.

[10] Bird, R. and L. Meertens, *Nested datatypes*, in: J. Jeuring, ed., *Proc. of 4th Int. Conf. on Mathematics of Program Construction, MPC'98 (Marstrand, June 1998)*, Lect. Notes in Comput. Sci. **1422**, Springer-Verlag, Berlin, 1998 pp. 52–67.

[11] Bird, R. and R. Paterson, *Generalised folds for nested datatypes*, Formal Aspects of Computing **11** (1999), pp. 200–222.

[12] Bird, R. S. and R. Paterson, *De Bruijn notation as a nested datatype*, J. of Functional Programming **9** (1999), pp. 77–91.

[13] de Bruijn, N. G., *Lambda calculus notation with nameless dummies, a tool for automatic formula manipulation, with applications to the Church-Rosser theorem*, Indagationes Math. **34** (1972), pp. 381–392.

[14] Crole, R. L., *Basic category theory for models of syntax*, Course notes for Summer School on Generic Programming, SSGP'02 (Oxford, Aug. 2002), to appear.

[15] Fiore, M., G. D. Plotkin and D. Turi, *Abstract syntax and variable binding (extended abstract)*, in: *Proc. of 14th Ann. IEEE Symp. on Logic in Computer Science, LICS'99 (Trento, July 1999)*, IEEE CS Press, Los Alamitos, CA, 1999 pp. 193–202.

[16] Ghani, N., C. Lüth and F. de Marchi, *Coalgebraic monads*, in: L. S. Moss, ed., *Proc. of 5th Wksh. on Coalgebraic Methods in Computer Science, CMCS'02 (Grenoble, Apr. 2002)*, Electr. Notes in Theoret. Comput. Science **65.1**, Elsevier, Amsterdam, 2002.

[17] Ghani, N., C. Lüth, F. de Marchi and J. Power, *Algebras, coalgebras, monads and comonads*, in: A. Corradini, M. Lenisa and U. Montanari, eds., *Proc. of 4th Wksh. on Coalgebraic Methods in Computer Science, CMCS'01 (Genova, Apr. 2001)*, Electr. Notes in Theoret. Comput. Sci. **44.1**, Elsevier, Amsterdam, 2001. Journal version to appear in Math. Struct. in Comput. Sci.

[18] Girard, J.-Y., *Locus solum: From the rules of logic to the logic of rules*, Math. Struct. in Comput. Sci. **11** (2001), pp. 301–506.

[19] Harper, H., F. Honsell and G. Plotkin. *A framework for defining logics.* J. of ACM **40** (1993), pp. 143–184.

[20] Hinze, R., *Polytypic functions over nested datatypes*, Discrete Math. and Theoret. Comput. Sci. **3** (1999), pp. 193–214.

[21] Hofmann, M., *Semantical analysis of higher-order abstract syntax*, in: *Proc. of 14th Ann. IEEE Symp. on Logic in Computer Science, LICS'99 (Trento, July 1999)*, IEEE CS Press, Los Alamitos, CA, 1999 pp. 204–213.

[22] Kennaway, J. R., J. W. Klop, M. R. Sleep and F. J. de Vries, *Infinitary lambda calculus*, Theoret. Comput. Sci. **175** (1997), pp. 93–125.

[23] Matthes, R., *Monotone inductive and coinductive constructors of rank 2*, in: L. Fribourg, ed., *Proc. of 15th Int. Wksh. on Computer Science Logic, CSL'01 (Paris, Sept. 2001)*, Lect. Notes in Comput. Sci. **2142**, Springer-Verlag, Berlin, 2001 pp. 600–614.

[24] Mints, G., *Finite investigations of infinite derivations*, Zap. Nauchn. Semin. Leningr. Otd. Matem. Inst. **49** (1975). Translation in: J. of Soviet Math. **10** (1978), pp. 548–596. Reprinted in: Mints, G., "Selected Papers in Proof Theory", Studies in Proof Theory: Monographs **3**, Bibliopolis, Napoli, 1992 pp. 17–72.

[25] Mints, G., *Reduction of finite and infinite derivations*, Ann. of Pure and Appl. Logic **104** (2000), pp. 167–188.

[26] Moss, L. S., *Parametric corecursion*, Theoret. Comput. Sci. **260** (2001), pp. 139–163.

[27] Okasaki, C., *From fast exponentiation to square matrices: An adventure in types*, in: *Proc. of 5th ACM SIGPLAN Int. Conf. on Functional Programming, ICFP'99 (Paris, Sept. 1999)*, ACM Press, New York, 1999 pp. 28–35.

[28] Pfenning, F. and C. Elliot, *Higher-order abstract syntax*, in: *Proc. of ACM SIGPLAN 1988 Conf. on Programming Language Design and Implementation, PLDI'88 (Atlanta, GA, June 1988)*, ACM Press, New York, 1988 pp. 199–208.

[29] Plotkin, G., *An illative theory of relations*, in: R. Cooper, K. Mukai and J. Perry, eds., *Situation Theory and Its Applications, Vol. 1*, CSLI Lect. Notes **22**, CSLI Publications, Stanford, CA, 1990 pp. 133–146.

[30] Severi, P. and F.-J. de Vries, *An extensional Böhm model*, in: S. Tison, ed., *Proc. of 13th Int. Conf. on Rewriting Theory and Applications, RTA 2002 (Copenhagen, July 2002)*, Lect. Notes in Comput. Sci. **2378**, Springer-Verlag, Berlin, 2002 pp. 159–173.

[31] Sun, Y., *An algebraic generalization of Frege structures – binding signatures*, Theoret. Comput. Sci. **211** (1999), pp. 189–232.

[32] Uustalu, T., *Generalizing substitution (extended abstract)*, in: Z. Ésik and A. Ingólfsdóttir, eds., *Prel. Proc. of 4th Int. Wksh. Fixed Points in Computer Science, FICS'02 (Copenhagen, July 2002)*, BRICS Notes Series NS-02-2, Dept. of Comput. Sci., Aarhus Univ., 2002 pp. 9–11. Full version submitted.

[33] Uustalu, T. and V. Vene, *Primitive (co)recursion and course-of-value (co)iteration, categorically*, Informatica **10** (1999), pp. 5–26.