Substitution in Non-wellfounded Syntax with Variable Binding [⋆]

Ralph Matthes [a] and Tarmo Uustalu [b,∗]

[a] *Institut für Informatik der Ludwig-Maximilians-Universität München Oettingenstraße 67, D-80538 München, Germany*

[b] *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. The main technical tool is a novel concept of heterogeneous substitution systems.

Key words: Substitution, Non-wellfounded Syntax, Variable Binding, Monad, Functor Category, Final Coalgebra, Primitive Corecursion

1 Introduction

Moss [30], Aczel, Adámek et al. [5,4], and Ghani et al. [18,17] have recently given a rather complete categorical analysis of *non-wellfounded* syntax, i.e., 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 [14] building on a suitable generalization of universal algebra—the theory of binding algebras [3,33,36]. In

[⋆] Expanded version of the conference paper [26].

[∗] Corresponding author.

Email addresses: matthes@informatik.uni-muenchen.de (Ralph Matthes), tarmo@cs.ioc.ee (Tarmo Uustalu).

both lines of work, the first thing 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 in the definitions of a heterogeneous signature and a 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. We emphasize that our focus is different from that of Fiore, Plotkin, Turi [15] in at least two aspects. First, Fiore, Plotkin and Turi were not interested in non-wellfounded syntax. Second, and more important, neither was it their specific goal to identify sufficient conditions under which a construction (other than a wellfounded term algebra), pretending to deserve to be called a language, is a monad: their binding algebras, i.e., their notion of models of languages with variable binding are, by definition, monads with extra structure, differently from the heterogeneous substitution systems of this work. On the other hand, we are not looking here into models of languages with variable binding.

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 adequate 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 one of the key concerns is reduction properties of intensional rewrite systems (as opposed to metatheory about extensional equational theories). In a type-theoretical system, 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 untyped lambda calculus syntax in the de Bruijn version in either a functional language or a typed lambda calculus is implicit in Bellegarde and Hook [9] and appears explicated in Altenkirch and Reus [7] and Bird and Paterson [12](remarkably, Altenkirch and Reus [7] discussed also the typed lambda calculus syntax). 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 [10,11,22,31]. Typed lambda calculi featuring heterogeneous and higher kind inductive and coinductive constructors are the topic of [25,1,2]. Fiore, Plotkin and Turi's original categorical analysis of de Bruijn-style abstract syntax [15], of which a tutorial-style presentation appears in [13], has been adapted to the setting of linear binders by Tanaka [37], and generalized further by Power [34]. Fiore [16] and Miculan and Scagnetto [27] have developed this approach further to cover multi-sorted and typed languages with variable binding. Hofmann [23] has given a category-theoretic explanation of the higher-order abstract syntax approach to variable binding [21,32].

To defend our engagement with non-wellfounded syntax (which arguably, in at least one sense, indeed is not syntax: by not admitting an initial algebra semantics), we also refer to some uses of this flavor of syntax. In proof theory, Mints' work [28] from the 1970s on the normalization of infinite derivations (continuous normalization) has recently been revived by him and others [29,6]. In rewriting, ongoing work on infinitary or coinductive lambda calculus [24,35] explains aspects of the model theory of the ordinary lambda calculus, and non-wellfounded syntax is relevant for lazy functional programming. Finally, also the syntax of Girard's Ludics [20], the language of designs, is non-wellfounded. We also stress that both wellfounded and non-wellfounded syntax are fundamentally about manipulating leaf-labelled trees whereas cosyntax is about node-labelled trees. Also: non-wellfounded syntax is not dual to wellfounded syntax; instead, it is dual to wellfounded cosyntax, see the discussions in [40,18].

The article is structured as follows. We begin in Section 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 Section 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 Section 4 (the central section), we define heterogeneous signatures and substitution systems for heterogeneous signatures and reprove the statements of Section 3 for these concepts. In Section 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., [39]. 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

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

We will not need primitive recursion and generalized coiteration and will therefore not introduce them either. These would be needed if we discussed wellfounded and non-wellfounded node-labelled trees (cosyntax). We reemphasize that wellfounded and non-wellfounded syntax are not dual to each other.

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 $h : \mu F \cdot Z \to X$ such that

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

The same kind of specialization of generalized iteration is carried out in [11], yielding "generalized folds".

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 (w.r.t. a system of equations), term graphs, rational terms etc.

Definition 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*

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

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*

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

i.e.,

$$\begin{tikzcd}
A \arrow[r, "\eta_A"] \arrow[dr, "f"'] & TA \arrow[d, "h"] & H(TA) \arrow[l, "\tau_A"'] \arrow[d, "Hh"] \\
 & TB & H(TB) \arrow[l, "\tau_B"]
\end{tikzcd}$$

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 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 *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. 4, 15.) (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. $\square$

The example of non-wellfounded H-terms, investigated in Moss [30] 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 4 (The substitution theorem of [30,5]) *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

$$\begin{tikzcd}
 & B + H(TA+TB) & A + H(TA+TB) \arrow[l, "{[\,(\mathsf{id}_B+H\mathsf{inr}_{TA,TB})\circ\alpha_B^{-1}\circ f,\mathsf{inr}\,]}"'] \arrow[d, "{\mathsf{id}_A+H[\,h,\mathsf{id}_{TB}\,]}"'] & A + H(TA) \arrow[l, "\mathsf{id}_A+H\mathsf{inl}_{TA,TB}"'] \arrow[dl, "\mathsf{id}_A+Hh"] \arrow[r, shift right, "\alpha_A"'] & TA \arrow[l, shift right, "\alpha_A^{-1}"'] \arrow[dd, "h"] \\
(*) & & A + H(TB) \arrow[dl, "{[\,\alpha_B^{-1}\circ f,\mathsf{inr}\,]}"'] \arrow[drr, "{[\,f,\tau_B\,]}"] & ** & \\
 & B + H(TB) \arrow[from=uu, "{\mathsf{id}_B+H[\,h,\mathsf{id}_{TB}\,]}"] \arrow[rrr, shift left, "\alpha_B"] & & & TB \arrow[lll, shift left, "\alpha_B^{-1}"]
\end{tikzcd}$$

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. $\square$

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 [7,12] 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 drawn 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 Section 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 Section 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 for the first-order signatures to be covered). 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 $\mathbf{Ptd}(\mathcal{C}) \to [\mathcal{C},\mathcal{C}]$. (Intuitively, a pointed functor is a "monad without multiplication", or a notion of syntax with variables but no substitution.)

Definition 5 *Given a category $\mathcal{C}$ with finite coproducts and 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*

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

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* $\{f\}$*, satisfying*

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

The operation $\{-\}$ is the substitution operation of the generalized substitution system.

Remark 6 *The conditions on* θ *are exactly those of a strength of* H*, except that we require* $\theta_{X,Z}$ *to be defined only when* Z *is pointed. This can be taken further: Notice that* $\mathsf{Id}+H-$ *delivers functors that are obviously pointed. We might therefore liberalize* $\mathsf{Id}+H-$ *to be an endofunctor on* $\mathbf{Ptd}(\mathcal{C})$ *and require* $\mathsf{id}_\sim + \theta_{-,\sim}$ *only to be a natural transformation between functors* $\mathbf{Ptd}(\mathcal{C}) \times \mathbf{Ptd}(\mathcal{C}) \to \mathbf{Ptd}(\mathcal{C})$*. We do not however have meaningful examples of* H*,* θ *where the definition of* θ *would use that its first argument is pointed. Having the second argument pointed will be crucial in the examples to follow.*

Definition 5 appears satisfactory in several ways. First of all, it covers our examples.

Example 7 (Homogeneous Signature) *For an endofunctor* K *on* $\mathcal{C}$*, we show that a substitution system* (T,α) *for* K *in the sense of Definition 1 with* K *seen as a homogeneous signature, gives rise to a heterogeneous substitution system.*

From Theorem 2 it follows that T *is even a functor (recall* $Tf = (\eta_B \circ f)^\star$ *for* $f : A \to B$*) and* η *a natural transformation. Also,* $\tau : K\cdot T \to T$ *is natural: For* $f : A \to B$*, one gets that* $Tf \circ \tau_A = \tau_B \circ K(Tf)$ *by appealing to the diagram characterizing* $(\eta_B \circ f)^\star$*. Define an endofunctor* H *on* $[\mathcal{C},\mathcal{C}]$ *by* $HX := K\cdot X$*,* $Hg := K\cdot g$ *for* $g : X \to Y$*. Define the natural transformation* θ *by*

$$\theta_{X,(Z,e)} := \mathsf{id}_{K\cdot X\cdot Z} : HX\cdot Z \to H(X\cdot Z)$$

Evidently, the two coherence conditions from Definition 5 are satisfied by θ*.* (T,α) *is a heterogeneous substitution system for* (H,θ)*: Assume* $f : (Z,e) \to$

(T,η). We want to find $h : T \cdot Z \to T$ satisfying the diagram above, i. e., having $h \circ (\eta \cdot Z) = f$ and $\tau \circ (K \cdot h) = h \circ (\tau \cdot Z)$. At object B of $\mathcal{C}$, this is precisely the condition for h_B being $(f_B)^\star$. (In the diagram in Definition 1, take $A := ZB$.) Thus, uniqueness of h is proved. The morphisms $(f_B)^\star$ for all B form a natural transformation from $T \cdot Z$ to T, to be proved by naturality of f and extensive use of the monad laws (again using the representation of Tf through Kleisli extension $-^\star$).

Example 8 (Lambda Calculus) *The lambda calculus signature is captured in H with $(HX)A = XA \times XA + X(1 + A)$, hence $H = H_1 + H_2$ with $H_1X = X \times X$ and $H_2X = X \cdot \Delta$ (again with $\Delta A = 1 + A$). For any $(\mathsf{Id} + H-)$-algebra (T,α), the $[\mathcal{C},\mathcal{C}]$-morphism α decomposes into the $[\mathcal{C},\mathcal{C}]$-morphisms $\eta : \mathsf{Id} \to T$ and $\tau : T \times T + T \cdot \Delta \to T$, where τ jointly represents application and lambda abstraction, and η turns variables into terms.*

The requirements for θ concerning naturality and the two coherence conditions are modular in the sense that such natural transformations can be provided separately for every summand of H, which we do here: Define

$$(\theta^1_{X,(Z,e)})_A := \mathsf{id}_{X(ZA)\times X(ZA)} : H_1X(ZA) \to H_1(X \cdot Z)A$$
$$(\theta^2_{X,(Z,e)})_A := X[\, e_{1+A} \circ \mathsf{inl}_{1,A}, Z\mathsf{inr}_{1,A}\,] : X(1 + ZA) \to X(Z(1 + A))$$

Then θ^i satisfies the required properties for H_i, $i \in \{1, 2\}$. (Certainly, naturality of θ^2 depends on the fact that pointed functor morphisms satisfy the additional requirement $f \circ e = e'$.)

Instantiating Definition 5, we get that (T,α) is a heterogeneous substitution system for $(H, \theta^1 + \theta^2)$ iff, for every $f : (Z, e) \to (T, \eta)$, there exists a unique $h : T \cdot Z \to T$, satisfying $h \circ (\eta \cdot Z) = f$ and, for all objects B of $\mathcal{C}$,

$$h_B \circ \tau_{ZB} = \tau_B \circ (h_B \times h_B + h_{1+B} \circ T[\, e_{1+B} \circ \mathsf{inl}_{1,B}, Z\mathsf{inr}_{1,B}\,]).$$

Below in Example 13, we will demonstrate that this implies the usual structural substitution.

Example 9 (Explicit Flattening) *We can also capture 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 $H := H_1 + H_3$ with H_1 taken from the previous example and $H_3X := X \cdot X$, hence $(HX)A = XA \times XA + X(XA)$. Given an $(\mathsf{Id}+H-)$-algebra (T,α), α decomposes into $\eta : \mathsf{Id} \to T$, as usual, and $\tau : T \times T + T \cdot T \to T$, which codes together application and formal flattening.*

We take θ^1 from the previous example and define

$$\theta^3_{X,(Z,e)} := X \cdot e \cdot X \cdot Z : X \cdot X \cdot Z \to X \cdot Z \cdot X \cdot Z,$$

hence $(\theta^3_{X,(Z,e)})_A = Xe_{X(ZA)}$. As in the previous example for θ^2, it is crucial for the definition of θ^3 that, in general, θ is parameterized in a pointed functor (Z, e), not just a functor Z. It is easy to show that H_3 and θ^3 satisfy the requirements, e. g., for the second equation, one has to show

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

which immediately follows from $e' \cdot e = (Z' \cdot e) \circ e'$.

(T, α) is a heterogeneous substitution system for $(H, \theta^1 + \theta^3)$ iff, for every $f : (Z, e) \to (T, \eta)$, there exists a unique $h : T \cdot Z \to T$, satisfying $h \circ (\eta \cdot Z) = f$ and, for all objects B of $\mathcal{C}$,

$$h_B \circ \tau_{ZB} = \tau_B \circ (h_B \times h_B + Th_B \circ h_{T(ZB)} \circ Te_{T(ZB)}).$$

In Example 14 below, a perspicuous application of this identity will be presented.

Lambda abstraction and "explicit flattening" can be combined by taking $H := (H_1 + H_2) + H_3$ and $\theta := (\theta^1 + \theta^2) + \theta^3$.

By replacing $(H_3X)A = X(XA)$ 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. The two formulations are obviously equivalent as $\int^B \coprod_{f \in \mathcal{C}(B,XA)} XB \cong (Lan_{\mathsf{Id}} X)(XA) \cong X(XA)$ (this is in categories; in a functional language, we would have to use an existential type instead of the coend, and then parametricity would be required to obtain the isomorphism). Why they really capture what is usually intended with explicit substitutions is discussed at length in [19].

There is also an instructive non-example adequately filtered out by Definition 5: the example of powerlists (also discussed in [2]). Powerlists, or perfectly balanced binary leaf trees, that is, all lists whose length is 2^n for some n, are represented by the initial $(\mathsf{Id}+H-)$-algebra of H given by $(HX)A = X(A\times A)$. There are two candidates for θ defined by

$$(\theta_{X,(Z,e)})_A = X(Z\langle \mathsf{id}_A, \mathsf{id}_A\rangle \circ x) : X(ZA \times ZA) \to X(Z(A \times A))$$

with x either $\mathsf{fst}_{ZA,ZA}$ or $\mathsf{snd}_{ZA,ZA}$, 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.

Definition 5 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. It also clarifies that parameterizing θ in a monad instead of a pointed functor would not work: in the proof, we need to instantiate $Z := T$, but for T we are just aiming at showing that it carries a monad structure. With a pointed functor parameter, we avoid this potential circularity.

Theorem 10 *If an $(\mathsf{Id} + H-)$-algebra (T,α) forms a heterogeneous substitution system for (H,θ) for some θ, then $(T,\eta,\{\mathsf{id}_{(T,\eta)}\})$ is a monad (in triple form).*

PROOF. The main structure of the proof is as follows: We define

$$\mu^{(0)} := \eta : \mathsf{Id} \to T$$

η is a $\mathbf{Ptd}(\mathcal{C})$-morphism from $(\mathsf{Id},\mathsf{id}_{\mathsf{Id}})$ to (T,η). Define

$$\mu^{(1)} := \{\eta\} : T \to T$$

By uniqueness of $\{\eta\}$, we get that (ii-a) $\mu^{(1)} = \mathsf{id}_T$, using the first coherence condition on θ. id_T is in turn a $\mathbf{Ptd}(\mathcal{C})$-morphism from (T,η) to (T,η). Define

$$\mu^{(2)} := \mu := \{\mathsf{id}_T\} : T \cdot T \to T.$$

We derive the monad law (i) $\mu \circ (\eta \cdot T) = \mathsf{id}_T$ from existence of $\{\mathsf{id}_T\}$ (the triangle part) and (ii-b) $\mu^{(1)} = \mu \circ (T \cdot \eta)$ from uniqueness of $\{\eta\}$, naturality of θ in the second argument and the existence of $\{\mathsf{id}_T\}$ (the rectangle part). Together with (ii-a), this yields the monad law (ii) $\mu \circ (T \cdot \eta) = \mathsf{id}_T$. Moreover, from (i) we get that $\mu : (T \cdot T, \eta \cdot \eta) \to (T,\eta)$ in $\mathbf{Ptd}(\mathcal{C})$. Define

$$\mu^{(3)} := \{\mu\} : T \cdot T \cdot T \to T.$$

In order to show the monad law (iii) $\mu \circ (\mu \cdot T) = \mu \circ (T \cdot \mu)$, we show (iii-a) $\mu^{(3)} = \mu \circ (\mu \cdot T)$ and (iii-b) $\mu^{(3)} = \mu \circ (T \cdot \mu)$. In each case, we need the uniqueness of $\{\mu\}$ (in conjunction with (i) and the existence of $\{\mathsf{id}_T\}$). For (iii-a), also naturality of θ in the first argument and the second coherence condition on θ is needed. For (iii-b), naturality of θ in the second argument is used.

Note, finally, that $\mu^{(i)} = \{\mu^{(i-1)}\}$ for $i \in \{1,2,3\}$.

We now fill the gaps.

- $\mu^{(1)} = \mathsf{id}_T$ is proved from the substitution system diagram not having more than one solution in h for $f := \eta$, $(Z,e) := (\mathsf{Id},\mathsf{id}_{\mathsf{Id}})$. Hence, we show $\mathsf{id}_T \circ$

$(\eta\cdot\mathsf{Id})=\eta$ and $\mathsf{id}_T\circ(\tau\cdot\mathsf{Id})=\tau\circ H\mathsf{id}_T\circ\theta_{T,(\mathsf{Id},\mathsf{id}_{\mathsf{Id}})}$. The first equation is trivial, the second holds by the first coherence condition on θ, applied for $X:=T$.

- (i) is trivial from $\mu=\{\mathsf{id}_T\}$ being a solution in h of the diagram above for $f:=\mathsf{id}_T$, $(Z,e):=(T,\eta)$.
- $\mu^{(1)}=\mu\circ(T\cdot\eta)$ is also proved from the diagram not having more than one solution in h for $f:=\eta$, $(Z,e):=(\mathsf{Id},\mathsf{id}_{\mathsf{Id}})$, i. e., we have to show $\mu\circ(T\cdot\eta)\circ(\eta\cdot\mathsf{Id})=\eta$ and

$$\tau\circ H(\mu\circ(T\cdot\eta))\circ\theta_{T,(\mathsf{Id},\mathsf{id}_{\mathsf{Id}})}=\mu\circ(T\cdot\eta)\circ(\tau\cdot\mathsf{Id})\tag{1}$$

By naturality of $\eta:\mathsf{Id}\to T$, $(T\cdot\eta)\circ(\eta\cdot\mathsf{Id})=(\eta\cdot T)\circ(\mathsf{Id}\cdot\eta)$, hence the first equation follows from (i). With first argument of θ fixed to T, it is a natural transformation from $HT\cdot U{\sim}$ to $H(T\cdot U{\sim})$. If this is applied to the $\mathbf{Ptd}(\mathcal{C})$-morphism $\eta:(\mathsf{Id},\mathsf{id}_{\mathsf{Id}})\to(T,\eta)$, one gets

$$H(T\cdot\eta)\circ\theta_{T,(\mathsf{Id},\mathsf{id}_{\mathsf{Id}})}=\theta_{T,(T,\eta)}\circ(HT\cdot\eta).$$

Thus, the left-hand side of (1) is equal to

$$\tau\circ H\mu\circ\theta_{T,(T,\eta)}\circ(HT\cdot\eta).$$

Since μ is the solution in h of the diagram for $f:=\mathsf{id}_T$, $(Z,e):=(T,\eta)$, one has

$$\tau\circ H\mu\circ\theta_{T,(T,\eta)}=\mu\circ(\tau\cdot T)\tag{2}$$

Therefore, it remains to show

$$\mu\circ(\tau\cdot T)\circ(HT\cdot\eta)=\mu\circ(T\cdot\eta)\circ(\tau\cdot\mathsf{Id}).$$

This is true because $(\tau\cdot T)\circ(HT\cdot\eta)=(T\cdot\eta)\circ(\tau\cdot\mathsf{Id})$ follows from naturality of $\tau:HT\to T$, applied to $\eta:\mathsf{Id}\to T$.
- μ is a $\mathbf{Ptd}(\mathcal{C})$-morphism: One has to check that $\mu\circ(\eta\cdot\eta)=\eta$. This is immediate from (i) since $\eta\cdot\eta=(\eta\cdot T)\circ\eta$.
- $\mu^{(3)}=\mu\circ(\mu\cdot T)$ comes from uniqueness of $\{\mu\}$ if we can prove that the right-hand side is a solution in h of the diagram for $f:=\mu$ and $(Z,e):=(T\cdot T,\eta\cdot\eta)$. We have to show $\mu\circ(\mu\cdot T)\circ(\eta\cdot T\cdot T)=\mu$ and

$$\tau\circ H(\mu\circ(\mu\cdot T))\circ\theta_{T,(T\cdot T,\eta\cdot\eta)}=\mu\circ(\mu\cdot T)\circ(\tau\cdot T\cdot T)\tag{3}$$

For the first equation, observe $(\mu\cdot T)\circ(\eta\cdot T\cdot T)=(\mu\circ(\eta\cdot T))\cdot T$ and use (i). For (3), use the second coherence condition on θ which yields

$$\theta_{T,(T\cdot T,\eta\cdot\eta)}=\theta_{T\cdot T,(T,\eta)}\circ(\theta_{T,(T,\eta)}\cdot T).$$

$H(\mu\cdot T)\circ\theta_{T\cdot T,(T,\eta)}$ can be calculated by using naturality of θ in the first argument: For the fixed object (T,η) of $\mathbf{Ptd}(\mathcal{C})$ as second argument, θ gives a natural transformation from $H-\cdot T$ to $H(-\cdot T)$. For the $[\mathcal{C},\mathcal{C}]$-morphism

$\mu : T \cdot T \to T$, this means

$$H(\mu \cdot T) \circ \theta_{T \cdot T,(T,\eta)} = \theta_{T,(T,\eta)} \circ (H\mu \cdot T).$$

Also using (2) again, the left-hand side of (3) becomes

$$\mu \circ (\tau \cdot T) \circ (H\mu \cdot T) \circ (\theta_{T,(T,\eta)} \cdot T) = \mu \circ ((\tau \circ H\mu \circ \theta_{T,(T,\eta)}) \cdot T).$$

We are done by another use of (2)!

- $\mu^{(3)} = \mu \circ (T \cdot \mu)$ is proved in the same manner. By uniqueness of $\{\mu\}$, it suffices to show $\mu \circ (T \cdot \mu) \circ (\eta \cdot T \cdot T) = \mu$ and

$$\tau \circ H(\mu \circ (T \cdot \mu)) \circ \theta_{T,(T \cdot T,\eta \cdot \eta)} = \mu \circ (T \cdot \mu) \circ (\tau \cdot T \cdot T) \tag{4}$$

Naturality of $\eta : \mathsf{Id} \to T$, applied to $\mu : T \cdot T \to T$, yields

$$(T \cdot \mu) \circ (\eta \cdot T \cdot T) = (\eta \cdot T) \circ (\mathsf{Id} \cdot \mu).$$

Therefore, the first equation follows from (i). Equation (4) is dealt with by naturality of θ in the second argument. As for the proof of (1), we fix the first argument to $X := T$. This time, we use naturality for the $\mathbf{Ptd}(\mathcal{C})$-morphism $\mu : (T \cdot T, \eta \cdot \eta) \to (T, \eta)$, which means

$$H(T \cdot \mu) \circ \theta_{T,(T \cdot T,\eta \cdot \eta)} = \theta_{T,(T,\eta)} \circ (HT \cdot \mu).$$

Using also (2), the left-hand side of (4) is equal to $\mu \circ (\tau \cdot T) \circ (HT \cdot \mu)$. Finally, naturality of $\tau : HT \to T$, applied to $\mu : T \cdot T \to T$ gives

$$(\tau \cdot T) \circ (HT \cdot \mu) = (T \cdot \mu) \circ (\tau \cdot T \cdot T). \qquad \square$$

Remark 11 *The proof above uses the fact that μ is a $\mathbf{Ptd}(\mathcal{C})$-morphism. This works for every $\{f\}$ instead of μ: If $f : (Z, e) \to (T, \eta)$ then $\{f\} : (T \cdot Z, \eta \cdot e) \to (T, \eta)$ in $\mathbf{Ptd}(\mathcal{C})$.*

PROOF. We have to show that $\{f\} \circ (\eta \cdot e) = \eta$. Since f is a pointed functor morphism, $f \circ e = \eta$. By existence of $\{f\}$ (the triangle part), $\{f\} \circ (\eta \cdot Z) = f$. We are done since $\eta \cdot e = (\eta \cdot Z) \circ e$. $\square$

Example 12 (Example 7, continued) *We found for the heterogeneous representation of homogeneous signature that, for $f : (Z, e) \to (T, \eta)$, $\{f\}_A = (f_A)^\star$. Consequently, $\{\mathsf{id}_{(T,\eta)}\}_A = (\mathsf{id}_{TA})^\star$ which is the definition of monad multiplication μ_A for the monad of Theorem 2. Hence, Theorem 10 yields the same monad and therefore extends Theorem 2 properly.*

Example 13 (Example 8, continued) *Assume a heterogeneous substitution system (T, α) for $(H, \theta^1 + \theta^2)$. By the previous theorem, (T, η, μ) is a monad* *where $\mu = \{\mathsf{id}_{(T,\eta)}\}$. For a substitution rule $f : A \to TB$, the associated substitution function $f^\star : TA \to TB$ is given as $f^\star = \mu_B \circ Tf$, as usual. Then*

$$f^\star \circ \tau_A = \tau_B \circ (f^\star \times f^\star + [\eta_{1+B} \circ \mathsf{inl}_{1,B}, T\mathsf{inr}_{1,B} \circ f]^\star).$$

It means that substitution $f^\star$ acts on an application homomorphically, and for lambda abstraction, it yields a lambda abstraction, with its kernel being the substitution function for $[\eta_{1+B} \circ \mathsf{inl}_{1,B}, T\mathsf{inr}_{1,B} \circ f] : 1 + A \to T(1+B)$, which is the appropriate lifting of f by the new variable represented by 1. ($T\mathsf{inr}_{1,B}$ should be seen as a weakening operation in the sense of logic.)

The proof is as follows: We have $f^\star \circ \tau_A = \mu_B \circ Tf \circ \tau_A$. By naturality of τ, the r.h.s. is $\mu_B \circ \tau_{TB} \circ HTf$. We saw previously that

$$\mu_B \circ \tau_{TB} = \tau_B \circ (\mu_B \times \mu_B + \mu_{1+B} \circ T[\eta_{1+B} \circ \mathsf{inl}_{1,B}, T\mathsf{inr}_{1,B}]).$$

Since $HTf = Tf \times Tf + T(\mathsf{id}_1 + f)$ and

$$T[\eta_{1+B} \circ \mathsf{inl}_{1,B}, T\mathsf{inr}_{1,B}] \circ T(\mathsf{id}_1 + f) = T[\eta_{1+B} \circ \mathsf{inl}_{1,B}, T\mathsf{inr}_{1,B} \circ f],$$

we are done.

Example 14 (Example 9, continued) *Now, we assume a heterogeneous substitution system (T, α) for $(H, \theta^1 + \theta^3)$ from Example 9. Again, we get monad multiplication $\mu = \{\mathsf{id}_{(T,\eta)}\}$ from the previous theorem. For a substitution rule $f : A \to TB$, the associated substitution function $f^\star : TA \to TB$ satisfies*

$$f^\star \circ \tau_A = \tau_B \circ (f^\star \times f^\star + Tf^\star).$$

For a proof, calculate $f^\star \circ \tau_A = \mu_B \circ Tf \circ \tau_A = \mu_B \circ \tau_{TB} \circ HTf$, as in the previous example. By Example 9,

$$\mu_B \circ \tau_{TB} = \tau_B \circ (\mu_B \times \mu_B + T\mu_B \circ \mu_{T(TB)} \circ T\eta_{T(TB)}).$$

By the second monad law, $\mu_{T(TB)} \circ T\eta_{T(TB)} = \mathsf{id}_{T(T(TB))}$. We are done since $HTf = Tf \times Tf + T(Tf)$.

Once generalized iteration in the sense of Section 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 Theorem 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 15 *If $[\mathcal{C}, \mathcal{C}]$ has an initial $(\mathsf{Id} + H-)$-algebra and a right adjoint for the functor $- \cdot Z : [\mathcal{C}, \mathcal{C}] \to [\mathcal{C}, \mathcal{C}]$ exists for every $\mathbf{Ptd}(\mathcal{C})$-object (Z, e), then (T, α) defined by*

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

is a heterogeneous substitution system for (H, θ) for every θ.

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

$$\{f\} = \mathsf{It}^{-\cdot Z, \mathsf{id}_Z + \theta_{-,(Z,e)}}_{\mathsf{Id}+H-, Z+H-}(\,[\,f, \tau\,]\,) \quad \text{for } f : (Z, e) \to (T, \eta)$$

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

$$\begin{tikzcd}
Z + (HT) \cdot Z \arrow[rr, "\alpha \cdot Z"] \arrow[d, "\mathsf{id}_Z + \theta_{T,(Z,e)}"'] & & T \cdot Z \arrow[dd, "h"] \\
Z + H(T \cdot Z) \arrow[d, "\mathsf{id}_Z + Hh"'] & & \\
Z + HT \arrow[rr, "{[\,f, \tau\,]}"] & & T
\end{tikzcd}$$

and this is the same square that $\{f\}$ is supposed to be the unique solution of in h. $\Box$

Example 16 (Resolving Explicit Substitutions) *Assume that $\mathcal{C}$, H and θ meet the requirements of the previous theorem, e. g., in case of the lambda calculus signature of Example 8, this would give the finite (well-founded) lambda terms. Set $\hat{H} := H + H_3$ (the signature extended by explicit flattening, as in Example 9). We assume that $\mathcal{C}$ meets the requirements of the previous theorem also for $\hat{H}$ and hence get the heterogeneous substitution systems (T, α) for (H, θ) and $(\hat{T}, \hat{\alpha})$ for $(\hat{H}, \theta + \theta^3)$. By Theorem 10, this gives rise to monads (T, η, μ) and $(\hat{T}, \hat{\eta}, \hat{\mu})$, respectively.*

Our aim is to show the existence of a natural transformation $\mathsf{eval} : \hat{T} \to T$ that resolves explicit substitution, expressed by explicit flattening. In order to state the result, it is necessary to consider the trivial embedding $\mathsf{emb} : T \to \hat{T}$ that "puts hats on every constructor", in view of the following definitions: α can be decomposed as $\alpha = [\,\eta, \tau\,]$. Likewise, $\hat{\alpha} = [\,\hat{\eta}, [\,\hat{\tau}, \widehat{\mathsf{flat}}\,]\,]$. (Note that μ is not a constructor for T, but a defined operation, hence we do not call it flat.) More precisely, set $\mathsf{emb} := \mathsf{It}_{\mathsf{Id}+H-}(\,[\,\hat{\eta}, \hat{\tau}\,]\,)$.

As before, given a substitution rule $f : A \to TB$, define the substitution function $f^\star := \mu_B \circ Tf : TA \to TB$. For $\hat{T}$, we do the same, but with explicit flattening instead of the operational flattening $\hat{\mu}$ which we will not need at all. Given $f : A \to \hat{T}B$, define the explicit substitution function $f^\sharp := \widehat{\mathsf{flat}}_B \circ \hat{T}f : \hat{T}A \to \hat{T}B$.

The property of eval we are interested in can now be formulated: Substitution for T-terms should be computable by embedding these terms into the syntax with explicit flattening, by the formal application of explicit substitution, and *finally by evaluation, i. e., for $f : A \to TB$, we want*

$$f^\star = \mathsf{eval}_B \circ (\mathsf{emb}_B \circ f)^\sharp \circ \mathsf{emb}_A.$$

This is, not surprisingly, achieved by defining

$$\mathsf{eval} := \mathsf{It}_{\mathsf{Id}+\hat{H}-}(\,[\eta, [\tau, \mu\,]\,]\,).$$

For a proof, one must first show $\mathsf{eval} \circ \mathsf{emb} = \mathsf{id}_T$, *which comes from* id_T *being the unique solution for* $\mathsf{It}_{\mathsf{Id}+H-}(\,\alpha\,)$. *By definition of iteration,* $\mathsf{eval} \circ \widehat{\mathsf{flat}} = \mu \circ (\mathsf{eval} \cdot \mathsf{eval})$. *Hence,*

$$\begin{aligned}
\mathsf{eval}_B \circ (\mathsf{emb}_B \circ f)^\sharp \circ \mathsf{emb}_A &= \mathsf{eval}_B \circ \widehat{\mathsf{flat}}_B \circ \hat{T}\mathsf{emb}_B \circ \hat{T} f \circ \mathsf{emb}_A \\
&= \mu_B \circ \mathsf{eval}_{TB} \circ \hat{T}\mathsf{eval}_B \circ \hat{T}\mathsf{emb}_B \circ \hat{T} f \circ \mathsf{emb}_A \\
&= \mu_B \circ \mathsf{eval}_{TB} \circ \hat{T} f \circ \mathsf{emb}_A \\
&= \mu_B \circ T f \circ \mathsf{eval}_A \circ \mathsf{emb}_A = \mu_B \circ T f = f^\star \qquad \square
\end{aligned}$$

The proof of being a heterogeneous substitution system in the case of non-wellfounded syntax follows that of the corresponding Theorem 4 in the previous section very closely.

Theorem 17 *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 for (H, θ) for every θ that satisfies naturality and the two coherence conditions of Definition 5.

PROOF. For $f : (Z, e) \to (T, \eta)$, define

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

The right-hand side is, 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., $\{f\}$, 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. □

5 Conclusions and future work

We have shown that a suitable generalization of substitution systems, inspired from [15], makes wellfounded and non-wellfounded term algebras given by signatures with variable binding analyzable as easily as are those given by first-order signatures. The key ingredient of a generalized signature is a kind of distributivity, or strength, parameterized in a pointed functor. With a more permissive functor parameter, neither lambda calculus nor languages with explicit substitution could be treated. We can prove that a heterogeneous substitution system always yields a monad and that both the wellfounded and non-wellfounded term algebras are substitution systems, thus obtaining a generalization of the substitution theorem of [30,5] which handles heterogeneous signatures. In a related piece of work [41], we show that the solution theorem of [30,5] generalizes as well. To this end, we use an alternative, equivalent definition of complete iterativeness of an ideal monad, which is more convenient for "point-free" arguments in functor categories than the original definition of [5].

One main direction for future work will be to obtain a deeper understanding of syntax with variable binding, in particular, of models of such syntax—a topic we have not touched here at all. Another goal will be to check how well the various versions of typed lambda calculus with monotone inductive and coinductive constructors of rank 2 introduced in [25,1,2] are suited for coding our constructions for non-wellfounded syntax with variable binding (from the point of view of achieving the intuitively desirable reduction behaviours). Some other possible directions for continuing this research are: to identify more examples of heterogeneous signatures and substitution systems, to see if a further generalization of the results of [38] along the lines of the present paper is possible.

Acknowledgements

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

R. Matthes participation in ETAPS'03 was partially financed by an EC HSC grant. T. Uustalu's research was 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. 5567. The visit of T. Uustalu to LMU München in Jan. 2002 was financed by the Graduiertenkolleg "Logik in der Informatik" of the Deutsche Forschungsgemeinschaft; his participation in ETAPS'03 was possible thanks to a travel grant from the Estonian Information Technology Foundation.

References

[1] A. Abel, R. Matthes, (Co-)iteration for higher-order nested datatypes, in H. Geuvers, F. Wiedijk (Eds.), Selected Papers from 2nd Int. Wksh. on Types for Proofs and Programs, TYPES 2002 (Berg en Dal, The Netherlands, 24–28 Apr. 2002), Vol. 2646 of Lect. Notes in Comp. Sci., Springer-Verlag, Berlin, 2003, pp. 1–20.

[2] A. Abel, R. Matthes, T. Uustalu, Generalized iteration and coiteration for higher-order nested datatypes, in: A. D. Gordon (Ed.), Proc. of 6th Int. Conf. on Foundations of Software Science and Computation Structures, FoSSaCS 2003 (Warsaw, Poland, 7–9 Apr. 2003), Vol. 2620 of Lect. Notes in Comp. Sci., Springer-Verlag, Berlin, 2003, pp. 54–69.

[3] P. Aczel, Frege structures and the notions of proposition, truth and set, in: J. Barwise, H. J. Keisler, K. Kunen (Eds.), The Kleene Symp.: Proc. of Symp. (Madison, WI, USA, 18–24 June 1978), Vol. 101 of Studies in Logic and the Foundations of Mathematics, North-Holland, Amsterdam, 1980, pp. 31–59.

[4] P. Aczel, Algebras and coalgebras, in: R. Backhouse, R. Crole, J. Gibbons (Eds.), Revised Lectures from Int. Summer School and Wksh. on Algebraic and Coalgebraic Methods in the Mathematics of Program Construction (Oxford, UK, 10–14 April 2000), Vol. 2297 of Lecture Notes in Computer Science, Springer-Verlag, Berlin, 2002, Ch. 3, pp. 79–88.

[5] P. Aczel, J. Adámek, S. Milius, J. Velebil, Infinite trees and completely iterative theories: a coalgebraic view, Theoretical Computer Science 300 (1–3) (2003) 1–45.

[6] K. Aehlig, F. Joachimski, On continuous normalization, in: J. C. Bradfield (Ed.), Proc. of 16th Int. Wksh. on Computer Science Logic, CSL 2002 (Edinburgh, UK, 22–25 Sept. 2002), Vol. 2471 of Lecture Notes in Computer Science, Springer-Verlag, Berlin, 2002, pp. 59–73.

[7] T. Altenkirch, B. Reus, Monadic presentations of lambda terms using generalized inductive types, in: J. Flum, M. Rodríguez-Artalejo (Eds.), Proc. of 13th Int. Workshop on Computer Science Logic, CSL'99 (Madrid, Spain, 20–25 Sept. 1999), Vol. 1683 of Lecture Notes in Computer Science, Springer-Verlag, Berlin, 1999, pp. 453–468.

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

[9] F. Bellegarde, J. Hook, Substitution: A formal methods case study using monads and transformations, Science of Computer Programming 23 (2–3) (1994) 287–311.

[10] R. Bird, L. Meertens, Nested datatypes, in: J. Jeuring (Ed.), Proc. of 4th Int. Conf. on Mathematics of Program Construction, MPC'98 (Marstrand, Sweden, 15–17 June 1998), Vol. 1422 of Lecture Notes in Computer Science, Springer-Verlag, Berlin, 1998, pp. 52–67.

[11] R. Bird, R. Paterson, Generalised folds for nested datatypes, Formal Aspects of Computing 11 (2) (1999) 200–222.

[12] R. S. Bird, R. Paterson, De Bruijn notation as a nested datatype, Journal of Functional Programming 9 (1) (1999) 77–91.

[13] R. L. Crole, Basic category theory for models of syntax, in: R. C. Backhouse, J. Gibbons (Eds.), Revised Lectures from Summer School and Wksh. on Generic Programming, SSGP 2002 (Oxford, UK, 26–30 Aug. 2002), Vol. 2793 of Lect. Notes in Comput. Sci., Springer-Verlag, Berlin, 2003, pp. 133–177.

[14] N. G. de Bruijn, Lambda calculus notation with nameless dummies, a tool for automatic formula manipulation, with applications to the Church-Rosser theorem, Indagationes Mathematicae (Koninglijke Nederlandse Akademie van Wetenschappen) 34 (1972) 381–392.

[15] M. Fiore, G. D. Plotkin, D. Turi, Abstract syntax and variable binding (extended abstract), in: Proc. of 14th Ann. IEEE Symp. on Logic in Computer Science, LICS'99, Trento, Italy, 2–5 July 1999, IEEE CS Press, Los Alamitos, CA, 1999, pp. 193–202.

[16] M. Fiore, Semantic analysis of normalisation by evaluation for typed lambda calculus, in: Proc. of 4th Int. ACM SIGPLAN Conf. on Principles and Practice of Declarative Programming, PPDP'02 (Pittsburgh, PA, USA, 6–8 Oct. 2002), ACM Press, New York, 2002, pp. 26–37.

[17] N. Ghani, C. Lüth, F. de Marchi, Coalgebraic monads, in: L. S. Moss (Ed.), Proc. of 5th Wksh. on Coalgebraic Methods in Computer Science, CMCS'02 (Grenoble, France, 6–7 Apr. 2002), Vol. 65(1) of Electronic Notes in Theoretical Computer Science, Elsevier, Amsterdam, 2002.

[18] N. Ghani, C. Lüth, F. de Marchi, J. Power, Dualising initial algebras, Mathematical Structures in Computer Science 13 (2) (2003) 349–370.

[19] N. Ghani, T. Uustalu, Explicit substitutions and higher-order syntax, in: F. Honsell, M. Miculan, A. Momigliano (Eds.), Proc. of 2nd ACM SIGPLAN Wksh. on Mechanized Reasoning about Languages with Variable Binding, MERLIN 2003 (Uppsala, Sweden, 26 Aug. 2003), ACM Press, New York, 2003.

[20] J.-Y. Girard, Locus solum: From the rules of logic to the logic of rules, Mathematical Structures in Computer Science 11 (3) (2001) 301–506.

[21] R. Harper, F. Honsell, G. Plotkin, A framework for defining logics, Journal of the ACM 40 (1) (1993) 143–184.

[22] R. Hinze, Polytypic functions over nested datatypes, Discrete Mathematics and Theoretical Computer Science 3 (4) (1999) 193–214.

[23] M. Hofmann, Semantical analysis of higher-order abstract syntax, in: Proc. of 14th Ann. IEEE Symp. on Logic in Computer Science, LICS'99 (Trento, Italy, 2–5 July 1999), IEEE CS Press, Los Alamitos, CA, 1999, pp. 204–213.

[24] J. R. Kennaway, J. W. Klop, M. R. Sleep, F.-J. de Vries, Infinitary lambda calculus, Theoretical Computer Science 175 (1) (1997) 93–125.

[25] R. Matthes, Monotone inductive and coinductive constructors of rank 2, in: L. Fribourg (Ed.), Proc. of 15th Int. Wksh. on Computer Science Logic, CSL'01 (Paris, France, 10–13 Sept. 2001), Vol. 2142 of Lecture Notes in Computer Science, Springer-Verlag, Berlin, 2001, pp. 600–614.

[26] R. Matthes, T. Uustalu, Substitution in Non-wellfounded Syntax with Variable Binding, in: H. P. Gumm (Ed.), Proc. of 6th Int. Wksh. on Coalgebraic Methods in Computer Science, CMCS'03 (Warsaw, Poland, 5–6 Apr. 2003), Vol. 82 (1) of Electronic Notes in Theoretical Computer Science, Elsevier, Amsterdam, 2003.

[27] M. Miculan, I. Scagnetto, A framework for typed HOAS and semantics, in: Proc. of 5th Int. ACM SIGPLAN Conf. on Principles and Practice of Declarative Programming, PPDP'03 (Uppsala, Sweden, 27-29 August 2003), ACM Press, New York, 2003, pp. 184–194.

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

[29] G. Mints, Reduction of finite and infinite derivations, Annals of Pure and Applied Logic 104 (1–3) (2000) 167–188.

[30] L. S. Moss, Parametric corecursion, Theoretical Computer Science 260 (1–2) (2001) 139–163.

[31] C. Okasaki, From fast exponentiation to square matrices: An adventure in types, in: Proc. of 5th ACM SIGPLAN Int. Conf. on Functional Programming, ICFP'99 (Paris, France, 27–29 Sept. 1999), Vol. 34 (9) of SIGPLAN Notices, ACM Press, New York, 1999, pp. 28–35.

[32] F. Pfenning, C. Elliot, Higher-order abstract syntax, in: Proc. of ACM SIGPLAN 1988 Conf. on Programming Language Design and Implementation, PLDI'88 (Atlanta, GA, 22-24 June 1988), Vol. 23 (7) of SIGPLAN Notices, ACM Press, New York, 1988, pp. 199–208.

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

[34] J. Power, A unified category-theoretic approach to variable binding, in: F. Honsell, M. Miculan, A. Momigliano (Eds.), Proc. of 2nd ACM SIGPLAN

Wksh. on Mechanized Reasoning about Languages with Variable Binding, MERLIN 2003 (Uppsala, Sweden, 26 Aug. 2003), ACM Press, New York, 2003.

[35] P. Severi, 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, Denmark, 22–24 July 2002), Vol. 2378 of Lecture Notes in Computer Science, Springer-Verlag, Berlin, 2002, pp. 159–173.

[36] Y. Sun, An algebraic generalization of Frege structures – binding signatures, Theoretical Computer Science 211 (1–2) (1999) 189–232.

[37] M. Tanaka, Abstract syntax and variable binding for linear binders, in: M. Nielsen, B. Rovan, eds., Proc. of 25th Int. Symp. on Math. Found. of Comp. Sci., MFCS 2000 (Bratislava, Slovakia, 28 Aug.–1 Sept. 2000), Vol. 1893 of Lecture Notes in Computer Science, Springer-Verlag, Berlin, 2000, pp. 670–679.

[38] T. Uustalu, Generalizing substitution, Theoretical Informatics and Applications 37 (4) (2003) 315–336.

[39] T. Uustalu, V. Vene, Primitive (co)recursion and course-of-value (co)iteration, categorically, Informatica 10 (1) (1999) 5–26.

[40] T. Uustalu, V. Vene, The dual of substitution is redecoration, in: K. Hammond, S. Curties (Eds.), Trends in Functional Programming 3, Intellect, Bristol / Portland, OR, 2001, pp. 99–110.

[41] T. Uustalu, V. Vene, An alternative definition of complete iterativeness, submitted.