EQUATIONAL PROPERTIES OF RECURSIVE PROGRAM SCHEME SOLUTIONS

It is a pleasure to dedicate this paper to Jiří Adámek. Ever since the field of coalgebra caught his attention, people in it learned a tremendous amount from him. Most of the background notions used in this paper were studied by Jiří and his colleagues, and so we feel his influence strongly as we write this paper. We wish Jiří many years to come, and many more strong contributions to mathematics and computer science.

by Stefan MILIUS and Lawrence S. MOSS

Abstract

Dans leurs précédents travaux [17, 18, 19], les auteurs ont proposé une théorie générale des schémas de programmes récursifs et de leurs solutions. Ces travaux généralisaient des approches plus anciennes, qui utilisaient les ensembles ordonnés ou les espaces métriques en offrant une théorie utilisant le concept de coalgèbre finale, d'algèbre d'Elgot, et une grande partie de ce que l'on sait à leur sujet. La théorie donnait l'existence et l'unicité des solutions de schémas de programmes récursifs non interprétés trés généraux. En outre, nous donnions aussi une théorie des solutions interprétées. Cet article poursuit le développement de la théorie. Il fournit des principes généraux qui sont utilisés pour montrer que deux schémas de programmes récursifs dans notre sens ont les solutions non interprétées identiques ou liées, ou qu'ils ont des solutions correctement liés à l'interprétation, identiques ou liées.

1 Introduction

The theory of *recursive program schemes* (rps's) is concerned with function definitions such as

$$f(n) \quad \approx \quad \mathsf{ifzero}(n, \mathsf{one}, f(\mathsf{pred}(n)) * n)\,. \tag{1}$$

The intention is that (1) be a *definition* of the factorial function on the natural numbers in terms of other functions (ifzero, one, pred, and $*$) which are known to exist before one writes (1). There are several aspects of the theory, including the study of several kinds of semantics for schemes, connections with operational semantics and rewriting, and questions of the equivalence of functions defined by various kinds of recursive program schemes. For example, using the fact that multiplication is commutative, it should be the case that

$$g(n) \quad \approx \quad \mathsf{ifzero}(n, \mathsf{one}, n * g(\mathsf{pred}(n))) \tag{2}$$

again defines the factorial function. This kind of fact goes back very far. One early source is Burstall and Darlington [7]. They write, “We may transform an equation by using on its right-hand expression any laws we have about the primitives k, l, ... (commutativity, associativity, etc.), obtaining a new equation.”

Though it produced sources such as Courcelle [9], Guessarian [10], and Nivat [23], the area is no longer as active as it once was. The authors of this paper proposed in [17, 18, 19] a category-theoretic generalization of part of the theory, including treatments of *uninterpreted* and *interpreted* recursion. The goal is to work with as few assumptions as possible and to obtain a theory that covers as much as possible in terms of standard examples and treatments of recursion. To see what has been done, it is useful to make a digression to a form of recursion that differs from what we have seen with the factorial function in (1) above.

Let $C(I)$ be the set of non-empty compact subsets of the unit interval $I = [0, 1]$. We shall be interested in several operations on $C(I)$:

$$\begin{array}{lcl} s^* & = & \{1 - x : x \in s\} \\ E(s,t) & = & \frac{1}{3}s \,\cup\, (\frac{2}{3} + \frac{1}{3}t) \\ F(s,t) & = & E(s,t^*) \\ G(s) & = & \frac{1}{3} + \frac{1}{3}s \end{array} \tag{3}$$

Here $\frac{1}{3}s = \{\frac{1}{3}x : x \in s\}$, and adding a number like $\frac{1}{3}$ or $\frac{2}{3}$ to a given set means adding the number to each element of it. So $G(s)$ is like s but linearly squashed to the interval $[\frac{1}{3}, \frac{2}{3}]$. We shall be interested in functions on $C(I)$ defined in terms of E, F, and G by *fixed-point equations* or systems, such as

$$\varphi(s) \quad \approx \quad E(s, \varphi(s)) \tag{4}$$

This is *not* a recursion as it is usually studied: there are no base cases. However, it turns out that there is a unique $\varphi^\dagger : C(I) \to C(I)$ with these properties; moreover, $\varphi^\dagger$ is continuous when $C(I)$ is taken to be a metric space under the Hausdorff metric. (We review the definition in Example 2.19(iii).) This falls out as a special case of our theory from [17, 18, 19]. In fact, we treat (4) as an *interpreted solution* of a very general form of *recursive program scheme*. Again, the main goal of our work has been to put forward a theory of these general recursive program schemes, their uninterpreted solutions, and also their interpreted solutions.

The theory is not about metric spaces or topology, or even about functions on the natural numbers. It is much more abstract, calling on ideas originating in recent work in the field of coalgebra. In discussions of examples such as (4), all of the topological work would go into showing that various spaces, maps, and functors have certain very general properties; after that, the general theory takes over and one need not look at the particular metric, or prove anything by recursion, etc. To see some of the subtlety, note that $\chi(s) \approx G(\chi(s))$ has a unique solution: $\chi^\dagger(s) = \{\frac{1}{2}\}$ for all s. However, $\chi(s) \approx \chi(s)^*$ has many solutions.

What is new in this paper is the consideration of *equational properties* of solutions. To see what this is about, we consider (4) and an analogous function $\psi(s)$ given by

$$\psi(s) \quad \approx \quad E(\psi(s), s) \tag{5}$$

A few moments of thought shows that the following equation should hold:

$$\varphi^\dagger(s)^* \quad = \quad \psi^\dagger(s^*) \tag{6}$$

One hint that (6) is true comes from the observation that $E(s^*, t^*) = E(t, s)^*$. Assuming that $\varphi^\dagger(s)^* = \psi^\dagger(s^*)$, we conclude that

$$\varphi^\dagger(s)^* \;=\; E(\varphi^\dagger(s), s)^* \;=\; E(s^*, \varphi^\dagger(s)^*) \;=\; E(s^*, \psi^\dagger(s^*)) \;=\; \psi^\dagger(s^*)\;.$$

But this is as circular as can be! (Correct reasoning may be found in Example 2.23.)

Here is another example: it is easy to check that

$$F(F(s,t), F(t,s)) \quad = \quad E(F(s,t), F(s,t))$$

And from this and the principle we quoted above in connection with Burstall and Darlington [7] it should follow that the two functions defined below are equal:

$$\begin{array}{rcl}\varphi(s) & \approx & F(F(s, \varphi(s)), F(\varphi(s), s)) \\ \psi(s) & \approx & E(F(s, \psi(s)), F(s, \psi(s)))\end{array}$$

One very specific goal of our work would be to show that the two functions just above are indeed equal.

Contents. The next section is a summary of our previous work on recursive program schemes. Although it may look long as a summary, we have cut all the corners and only provided those definitions, results, and examples that are needed in the rest of this paper. In addition, we re-arranged the order in which topics are introduced at several places, because this seemed to make for a shorter presentation. Accordingly, Section 2 would not be a good way to learn the theory. Section 3 presents the laws of *first-order recursion* as a kind of preparation for the work of the rest of the paper. Section 4 is a treatment of several of the main laws of recursive program scheme solutions in the uninterpreted setting, where one works with infinite trees only. On our account as well as on the classical one, this study is needed before we can tackle the more interesting case of equations on interpreted settings (Section 5); these are the kinds of things we presented in our opening discussion.

J. Mersch's work. Prior work in our direction comes from the dissertation and paper of J. Mersch [14, 15]. His project begins with the logical language FLR_0 proposed by Moschovakis [22] (a fragment of a larger language called FLR for *formal language of recursion*) and studied by Hurkens et al [11]. Mersch extends FLR_0 to a language $FLRS$ (*formal language of recursive schemes*) which interprets fixed point terms and their uninterpreted semantics. He presents axioms and rules of inference for equations between these fixed point terms. The axioms are quite related to our work in Section 4. Our treatment is somewhat more general, precisely because we are not restricted to working with terms in a particular syntax. (In this sense, what we are doing is closer like the *iteration theories* of Bloom and Ésik [5], and we foreshadow the connection in Section 3.) At the same time, because $FLRS$ (like FLR before it) permits terms like x **where** $x = x$, the semantics requires work that we did not pursue. The biggest difference between our work and Mersch's is that our main focus here is on *interpreted* schemes (Section 5). Following this paper, one next step should be to take what we have done in this paper and combine it with [14, 15] to study a formal language of interpreted recursive program schemes.

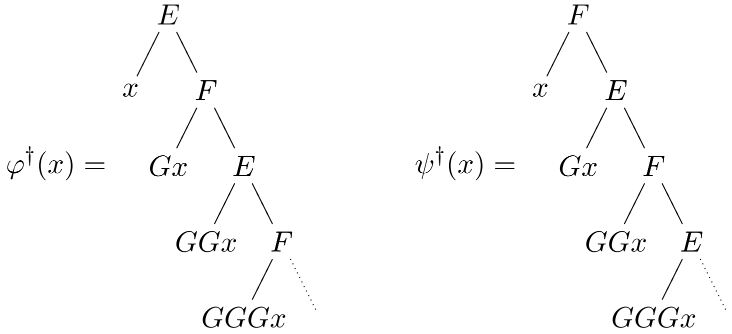
2 Background

We illustrate our account by going back and forth between a very specific example and a very general theory. The example is the rps

$$\begin{array}{rcl} \varphi(x) & \approx & E(x, \psi(Gx)) \\ \psi(x) & \approx & F(x, \varphi(Gx)) \end{array} \tag{7}$$

There are two possible interpretations. First, it might be that E, F, and G are functions which are "known" and that (7) should uniquely define functions in terms of them. This would be an *interpreted* rps. Our Introduction was about problems in interpreted recursion. The *uninterpreted* interpretation of (7) regards E, F, and

G as merely syntactic objects. Then (7) defines infinite trees $\varphi^\dagger$ and $\psi^\dagger$, as shown below:



(8)

Our account begins with two *signatures*; these are just collections of symbols with specified arities. In our example, these are: Σ, the signature with a unary symbol G and two binary ones E and F; and Φ, the signature with unary symbols φ and ψ. We then associate to Σ and Φ two *signature functors* on Set. We illustrate this with Σ. Regard it as a functor $\Sigma : \mathbb{N} \to \mathsf{Set}$, where $\mathbb{N}$ is the discrete category on the natural numbers. More to the point, each $\Sigma(n)$ is the set of function symbols of arity n. Let $J : \mathbb{N} \to \mathsf{Set}$ be the inclusion functor defined by $Jn = \{\, 0, \ldots, n-1 \,\}$. For any set X, and any $n \in \mathbb{N}$, $\mathsf{Set}(Jn, X)$ is isomorphic to X^n. We usually identify these sets. We associate to Σ the endofunctor $H_\Sigma : \mathsf{Set} \to \mathsf{Set}$ given by

$$H_\Sigma(X) = \coprod_{n \in \mathbb{N}} \Sigma(n) \times \mathsf{Set}(Jn, X). \tag{9}$$

with the action of H_Σ on morphisms defined in the obvious way. Moreover, for every endofunctor G of Set there is a bijective correspondence

$$\frac{\Sigma \to G \cdot J}{H_\Sigma \to G}. \tag{10}$$

We return to our example signatures Σ and Φ from above. We have associated signature functors H_Σ and H_Φ on Set. Since these functors are so important, and since they occur so often in our notation, we find it convenient to simplify our notation and use single letters for them. To simplify notation, we shall write H for H_Σ and V for H_Φ.

2.1 Iteratable functors

At this point, we step back and develop the overall theory. The basis for this theory includes many standard definitions from category theory which we shall not review, such as algebras and coalgebras for functors, initial and final (terminal) objects, monads and their Eilenberg-Moore and Kleisli categories; see Mac Lane [13], for example.

Definition 2.1. Let $\mathcal{A}$ be a category with finite coproducts denoted by $+$. If $H : \mathcal{A} \to \mathcal{A}$ is a functor and c is an object of $\mathcal{A}$, then the functor $H(-) + c$ is the sum of H and the constant functor with value c. H is *iteratable* if for all c, $H(-) + c$ has a final coalgebra. In this case, we adopt special notation for a final coalgebra for $H(-) + c$:

$$\alpha_c : Tc \to HTc + c$$

Example 2.2. Of the many examples, we only mention those which are needed in this paper. Further examples may be found in [17, 18]. Take $\mathcal{A}$ to be Set. Then every signature functor H_Σ is iteratable. For a set c, the final coalgebra Tc for $H_\Sigma(-) + c$ consists of all (finite and infinite) Σ-trees over c, i. e., rooted and ordered trees where all inner nodes with $n > 1$ children are labeled by operation symbols from $\Sigma(n)$ and all leaves are labeled in $\Sigma(0)$ or the set c.

Let CMS be the category of complete metric spaces, where distances are measured in the interval $[0, 1]$, with non-expanding maps as morphisms. We take for the coproduct the disjoint union, with points in different copies taken to have distance 1. Each homset $\mathsf{CMS}(X, Y)$ is again an object, using the sup metric $d_{X,Y}$ on functions. Also, for each space X and each $\epsilon < 1$, we get a space ϵX by keeping the points and scaling the metric by ϵ. Let Σ be a signature. We turn Σ into a functor $H_\Sigma : \mathsf{CMS} \to \mathsf{CMS}$ via

$$H_\Sigma(X) = \coprod_{n \in \mathbb{N}} \Sigma(n) \times \frac{1}{3}\mathsf{CMS}(Jn, X).$$

The product is the usual one, and $J : \mathbb{N} \to \mathsf{CMS}$ takes n to the discrete space on $\{0, \ldots, n-1\}$. (The $\frac{1}{3}$ comes from our examples in the introduction.)

Let CPO be the category of complete partial orders, i. e., posets (not necessarily with a least element) where all ω-chains have joins; morphisms are the maps preserving these joins, which are called *continuous maps*. Notice that products and coproducts are in CPO are formed as in the category of sets; and each homset $\mathsf{CPO}(X, Y)$ is itself a cpo with the pointwise order. So for every signature Σ we again have the associated signature functor given on objects by

$$H_\Sigma(X) = \coprod_{n \in \mathbb{N}} \Sigma(n) \times \mathsf{CPO}(Jn, X),$$

where $J : \mathbb{N} \to \mathsf{CPO}$ maps every number n to the discrete cpo on $\{0, \ldots, n-1\}$.

Returning to iteratable functors: we are so concerned with them in this paper that from now on, the letters H, V, etc., denote *iteratable* endofunctors on some underlying category with finite coproducts.

There is a rich general theory of iteratable functors. Everything we do depends on some central results from this theory, and so we review them. Let H be iteratable on $\mathcal{A}$. For each object c, α_c is invertible by Lambek's Lemma [12], and we write its inverse as $[\tau_c, \eta_c]$. So $\tau_c \ : \ H(Tc) \ \to \ Tc$, and $\eta_c \ : \ c \ \to \ Tc$. Moreover, $\alpha_c \cdot \tau_c \ = \ \mathsf{inl}_{HTc+c}$ and $\alpha_c \cdot \eta_c \ = \ \mathsf{inr}_{HTc+c}$. (We are using inl and inr as the coproduct injections in all categories.) Notice that τ_c gives Tc the structure of an H-algebra. It turns out that T is a functor, α, η, and τ are natural transformations: $\alpha : T \to HT + Id$, $\eta : Id \to T$, and $\tau : HT \to T$. We define $\kappa : H \to T$ to be $\tau \cdot H\eta$.

Further, there is a natural transformation $\mu : TT \to T$ with many important properties. First, (T, η, μ) is a monad, see e. g. [1, 20]. This monad is of course defined from H, and to emphasize this we write T^H for the monad; we also use this notation for the functor part T of this monad. Similarly, we write η^H, μ^H, τ^H and κ^H.

Proposition 2.3. *The diagrams below commute.*

$$\begin{tikzcd}
HTT \arrow[r, "\tau T"] \arrow[d, "H\mu"'] & TT \arrow[d, "\mu"] \\
HT \arrow[r, "\tau"'] & T
\end{tikzcd}
\qquad
\begin{tikzcd}
HT \arrow[r, "\tau"] \arrow[d, "\kappa T"'] & T \\
TT \arrow[ur, "\mu"'] &
\end{tikzcd}$$

(We shall state but not re-prove any results from our earlier papers [17, 18, 19].)

2.2 First-order recursion and the Substitution Theorem

Recall that our purpose in this paper is to study equational properties of solutions to recursive program schemes such as (7), repeated below:

$$\begin{array}{rcl}
\varphi(x) & \approx & E(x, \psi(Gx)) \\
\psi(x) & \approx & F(x, \varphi(Gx))
\end{array}$$

We are discussing some general category-theoretic background used in our account of rps solutions. The next step is to consider solutions of systems of equations, but ones which are simpler than those just above. Consider first a *first-order recursion* such as

$$x \ \approx \ F(x)$$

The "x" here is any object which we take to be a variable. We aim to solve this equation for x, obtaining a tree over the signature consisting of a unary function symbol F. Naturally enough, the solution is the infinite term

$$x^\dagger \quad = \quad F(F(F(F(\cdots))))$$

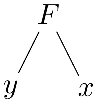
A bit more generally, we would like to consider systems whose solutions are not so easily drawn, and also whose right-hand sides might contain other "variables", that is, other objects besides the variables and the function symbols from the signature used on the right. For example,

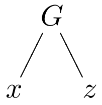
$$\begin{array}{rcl} x & \approx & F(y,x) \\ y & \approx & G(x,z) \end{array} \tag{11}$$

Now our set of variables is $\{x,y\}$, the signature Σ on the right consists of two binary symbols F and G, and z is a fresh object. Write a for $\{x,y\}$ and b for $\{z\}$; also let $H = H_\Sigma$ and let (T,η,μ) be the monad associated to H as discussed in Section 2.1 just above. Here is a result which guarantees the existence and uniqueness of solutions to systems such as (11):

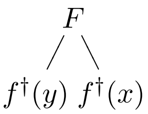
Theorem 2.4 ([1, 20]). *Let $f : a \to T(a+b)$ factor through τ_{a+b}. Then there is a unique $f^\dagger : a \to Tb$ such that $f^\dagger = \mu_b \cdot T[f^\dagger, \eta_b] \cdot f$. Moreover, $f^\dagger$ factors through τ_b.*

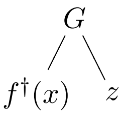
Example 2.5. We continue our discussion of (11). The system itself is modeled by the function $f : a \to T(a+b)$. This function is described in pictures as

$f(x) =$ 

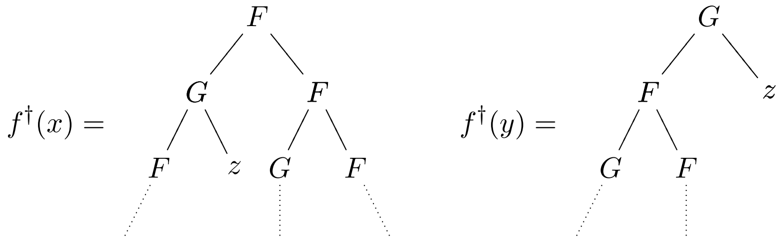
$f(y) =$ 

Then $f^\dagger$ is the unique function from $a = \{x,y\}$ to Σ-trees over $\{z\}$ such that

$f^\dagger(x) =$ 

$f^\dagger(y) =$ 

$f^\dagger$ may be pictured more explicitly:



We next want to mention a result that shows that the account which we have provided covers the phenomenon of *substitution* of variables by infinite trees.

Theorem 2.6 (Substitution Theorem [1])**.** *Let $f : a \to Tb$, and let $f^* : Ta \to Tb$ be $\mu_b \cdot Tf$. Then f^* is the unique morphism with the following two properties:*

(i) f^* is a morphism of H-algebras:

$$\begin{tikzcd}
H(Ta) \arrow[r, "\tau_a"] \arrow[d, "Hf^*"'] & Ta \arrow[d, "f^*"] \\
H(Tb) \arrow[r, "\tau_b"'] & Tb
\end{tikzcd}$$

(ii) $f = f^* \cdot \eta_a$.

$(-)^*$ **has the properties of a Kleisli operation.** One indication that our theory is on the right track is that we are able to prove the important properties of substitution. These are familiar from the theory of finite terms on a signature. The important point is that they hold in the infinite case. For that matter, the theory only requires the apparatus of iteratable functors, and so it is much more general.

Consider the operation $(-)^*$ taking a morphism of the form $f : a \to Tb$ to the associated $f^* : Ta \to Tb$. It is not hard to show that the triple $(T, \eta, (-)^*)$ satisfies the properties of a Kleisli triple: $(\eta_a)^* = id_{Ta}$; and if $f : a \to Tb$ and $g : b \to Tc$, then $f^* \cdot \eta_a = f$ and $(g^* \cdot f)^* = g^* \cdot f^*$.

A final note: our review here is developing things in a different order than in papers in the literature. It is more convenient to define the Kleisli operation before the monad multiplication μ, and indeed to use the operation to define multiplication by $\mu_a = (id_{Ta})^*$.

2.3 Second-order Substitution

A moment's look at (7) and (11) shows that Theorem 2.6 is not strong enough to provide an account of solutions to recursive program schemes such as (7). To see the point clearly, consider the difference between the following two equations:

$$x \approx F(x, G(x)) \tag{12}$$

$$\varphi(x) \approx F(x, \varphi(G(x))) \tag{13}$$

Equation (12) is a first-order recursion as we have studied in our last section. Its solution would essentially be a single tree $x^\dagger$ satisfying $x^\dagger = F(x^\dagger, G(x^\dagger))$. On the other hand, the solution to (13) is more complicated, since it asks not for a single tree but rather a *function* $\varphi^\dagger$ from trees (on F and G) to trees. It would satisfy $\varphi^\dagger(x) = F(x, \varphi^\dagger(G(x)))$. That is, the required property does not concern a single tree but rather a function. This equation (13) is a recursive program scheme, and an account of these schemes we must be different from an account of first-order recursion. We provided a treatment in our papers [17, 18, 19], and we review it in the next section.

We first need an account of *second-order substitution*, i. e., substitution of operation symbols from one signature in a tree by trees over another signature. In fact, such an account follows from the characterization of T^H as the free *completely iterative monad* on H, see [1, 16]. We shall not recall that notion and the main result here. Instead we merely state Theorem 2.9. It is an easy consequence of the freeness result in loc. cit.

Definition 2.7. Let H and K be iteratable endofunctors of $\mathcal{A}$. A natural transformation $\sigma : H \to T^K$ is called *ideal* if it factors as $\tau^K \cdot \sigma'$ for some natural transformation $\sigma' : H \to KT^K$.

Example 2.8. The canonical natural transformation $\kappa = \tau \cdot H\eta : H \to T$ is ideal.

Theorem 2.9. *Let H and K be iteratable endofunctors. Then for every ideal natural transformation $\sigma : H \to T^K$ there exists a unique monad morphism $\overline{\sigma} : T^H \to T^K$ such that $\overline{\sigma} \cdot \kappa^H = \sigma$.*

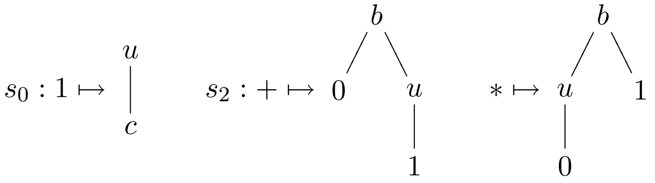
Let us explain how the monad morphism from Theorem 2.9 provides a modeling of second-order substitution. Let Σ and Γ be signatures (considered as functors $\mathbb{N} \to \mathsf{Set}$). Each symbol $f \in \Sigma(n)$ is considered as a flat tree in n variables. A second-order substitution gives an "implementation" to each such f as a Γ-tree in the same n variables. We model this by a natural transformation $s : \Sigma \to T_\Gamma \cdot J$, i. e., a family of maps $s_n : \Sigma(n) \to T_\Gamma\{\,0, \ldots n-1\,\}$, $n \in \mathbb{N}$. By the bijective correspondence (10), this gives rise to a natural transformation $\sigma : H_\Sigma \to T_\Gamma$. We are only interested in *non-erasing* substitutions, those where s assigns to each Σ-symbol a Γ-tree which is not just single node tree labelled by a variable. Translated along (10) that means precisely that σ is an ideal natural transformation. Thus, from Theorem 2.9 we get a monad morphism $\overline{\sigma} : T_\Sigma \to T_\Gamma$. For any set X of variables, the action of $\overline{\sigma}$ is that of second-order substitution: $\overline{\sigma}_X$ replaces every Σ-symbol in a tree t from $T_\Sigma X$ by its implementation according to σ. More precisely, let t be a tree from $T_\Sigma X$. If $t = x$ is a variable from X, then $\overline{\sigma}_X(t) = x$. Otherwise we have $t = f(t_1, \dots, t_n)$ with $f \in \Sigma(n)$ and $t_i \in T_\Sigma X$, $i = 1, \dots, n$. Let

$$s_n(f) = t'(0, \dots, n-1) \in T_\Gamma\{\,0, \dots, n-1\,\}.$$

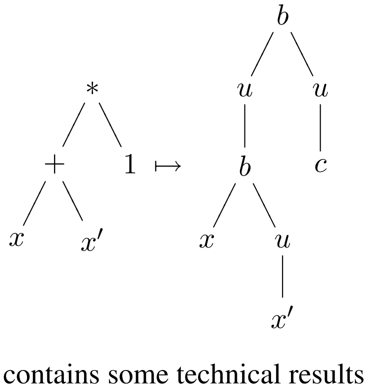
Then $\overline{\sigma}_X$ would satisfy the following equation

$$\overline{\sigma}_X(t) = t'(\overline{\sigma}_X(t_1), \dots, \overline{\sigma}_X(t_n)).$$

Example 2.10. Suppose that Σ consists of two binary symbols $+$ and $*$ and a constant 1, and Γ consists of a binary symbol b, a unary one u and a constant c. Furthermore, let σ be given by $s : \Sigma \to T_\Gamma \cdot J$ as follows:



and else s_n is the unique map from the empty set. For the set $X = \{\,x, x'\,\}$, the second-order substitution morphism $\overline{\sigma}_X$ acts for example as follows:



The rest of this section contains some technical results which we shall use later in the paper. It may be omitted on first reading without loss of continuity.

Lemma 2.11 ([17, 18])**.** *If H and K are iteratable endofunctors, $\sigma : H \to T^K$, $\sigma' : H \to KT^K$, and $\sigma = \tau^K \cdot \sigma'$, then for the unique induced monad morphism $\overline{\sigma}$ the diagram below commutes:*

$$\begin{tikzcd}
HT^H \arrow[r, "\sigma' * \overline{\sigma}"] \arrow[d, "\tau^H"'] & KT^K T^K \arrow[r, "K\mu^K"] & KT^K \arrow[d, "\tau^K"] \\
T^H \arrow[rr, "\overline{\sigma}"'] & & T^K
\end{tikzcd}$$

(The symbol $*$ denotes parallel composition of natural transformations.)

We would like to turn Theorem 2.9 into a statement about the assignment $H \mapsto T^H$. However, it is not clear how to formulate a result of this type. First of all, iteratability is a special property not enjoyed by all endofunctors on a given category $\mathcal{A}$. Further, it is not even known in general whether the iteratability of a functor H implies that of T^H. The best we can say seems to be the following result, one which we shall use.

Proposition 2.12. *Let H, J and K be iteratable endofunctors of $\mathcal{A}$. Then for all ideal natural transformations $s : H \to T^J$ and $t : J \to T^K$ we have the following Kleisli laws for $(T^{\text{-}}, \kappa^{(\text{-})}, \overline{(\_)})$:*

(i) $\overline{s} \cdot \kappa^H = s$,

(ii) $\overline{\kappa^H} = id_{T^H}$, *and*

(iii) $\overline{t} \cdot \overline{s} = \overline{\overline{t} \cdot s}$.

Proof. The three properties follow easily from the universal property of T^H as stated in Theorem 2.9. □

In the following result we denote by $\mathbf{It}[\mathcal{A}, \mathcal{A}]$ the category of iteratable endofunctors on $\mathcal{A}$ and natural transformations, and we write $\mathbf{CIM}(\mathcal{A})$ for the category of completely iterative monads on $\mathcal{A}$ and monad (homo)morphisms.

Corollary 2.13. *The assignment $H \mapsto T^H$ extends to a functor*

$$T^{\text{-}} : \mathbf{It}[\mathcal{A}, \mathcal{A}] \to \mathbf{CIM}(\mathcal{A})$$

which assigns to any natural transformation $n : H \to K$ between iteratable endofunctors the monad morphism $T^n = \overline{\kappa^K \cdot n}$.

We need the following properties of $T^{\text{-}}$.

Corollary 2.14. *Let H, J and K be iteratable endofunctors on $\mathcal{A}$, and let $n : H \to J$.*

(i) The diagram below commutes:

$$\begin{tikzcd}
HT^H \arrow[r, "n*T^n"] \arrow[d, "\tau^H"'] & JT^J \arrow[d, "\tau^J"] \\
T^H \arrow[r, "T^n"'] & T^J
\end{tikzcd}$$

(ii) For every ideal natural transformation $t : J \to T^K$ the equation $\overline{t}\cdot T^n = \overline{t \cdot n}$ holds.

Proof. Part (i) follows from Lemma 2.11, and part (ii) is immediate from Proposition 2.12(iii) and Corollary 2.13. □

2.4 Recursive program schemes

As we near the end of our preliminary sections, we discuss our formulation of recursive program schemes in terms of notation we have already introduced.

Definition 2.15 ([17])**.** Let V and H be endofunctors on $\mathcal{A}$. Assume that H, V, and $H+V$ are all iteratable. A *recursive program scheme* (or *rps*, for short) is a natural transformation

$$e : V \to T^{H+V}.$$

We sometimes call V the *variables*, and H the *givens*. The rps e is called *guarded* if there exists a natural transformation $f : V \to HT^{H+V}$ such that the diagram on the left below commutes:

$$\begin{tikzcd}
V \arrow[r, "e"] \arrow[ddr, bend right, "f"'] & T^{H+V} \\
& (H+V)T^{H+V} \arrow[u, "\tau^{H+V}"'] \\
& HT^{H+V} \arrow[u, "\mathsf{inl}*T^{H+V}"']
\end{tikzcd}
\qquad
\begin{tikzcd}
V \arrow[r, "e^\dagger"] \arrow[d, "e"'] & T^H \\
T^{H+V} \arrow[ur, "\overline{[\kappa^H, e^\dagger]}"'] &
\end{tikzcd}
\tag{14}$$

A *solution* of e is an ideal natural transformation $e^\dagger : V \to T^H$ such that the right-hand triangle above commutes.

Theorem 2.16 ([17])**.** *Every guarded rps has a unique solution.*

Example 2.17. We return to (7), repeated below:

$$\begin{array}{rcl}\varphi(x) & \approx & E(x, \psi(Gx)) \\ \psi(x) & \approx & F(x, \varphi(Gx))\end{array}$$

Let Σ be the signature that contains a unary operation symbol G and a binary one F—so we have $\Sigma_1 = \{ G \}$, $\Sigma_2 = \{E, F \}$ and $\Sigma_n = \emptyset$ else. The signature Φ of recursively defined operations consists of two unary symbols φ and ψ. Consider the recursive program scheme above as a natural transformation $r : \Phi \to T_{\Sigma+\Phi} \cdot J$ with the components given by

$$r_1 : \varphi \mapsto E(0, \psi(G0)) \qquad \psi \mapsto F(0, \varphi(G0))$$

(we write trees as terms above) and where r_n, $n \neq 1$, is the empty map. The bijective correspondence (10) yields a natural transformation $e : H_\Phi \to T^{H_\Sigma + H_\Phi}$. This is our formulation of (7) as a recursive program scheme in the sense of this paper.

Continuing, we may turn the trees in (8) into a natural transformation $e^\dagger : H_\Phi \to T^{H_\Sigma}$. It is not hard to finally check that $\varphi^\dagger$ is the solution to φ in our sense; that is, $e^\dagger = \overline{[\kappa^{H_\Sigma}, e^\dagger]} \cdot e$.

2.5 Completely iterative algebras

The notion of solution in Theorem 2.16 is that of an *uninterpreted* solution to an rps. We are especially interested in interpreted solutions, and to study those we must work on algebras with enough "solutions to equations" in which to interpret recursive program schemes. Our work has isolated two main classes of algebras for this, *completely iterative algebras* and *Elgot algebras*.

Definition 2.18. Let $H : \mathcal{A} \to \mathcal{A}$ be an endofunctor. A *flat equation morphism* in an object A (*of parameters*) is a morphism of the form $e : X \to HX + A$. Let $a : HA \to A$ be an H-algebra. We say that $e^\dagger : X \to A$ is a *solution* of e *in* (A, a) if the square below commutes:

$$\begin{tikzcd} X \arrow[r, "e^\dagger"] \arrow[d, "e"'] & A \\ HX + A \arrow[r, "He^\dagger + A"'] & HA + A \arrow[u, "{[a,A]}"'] \end{tikzcd} \qquad (15)$$

A is a *completely iterative algebra* (*cia*) if every flat equation morphism in A has a unique solution in A.

Example 2.19. We only give a few very specific examples; for more, see Milius [16] and Adámek et al [2].

(i) Let H be iteratable on $\mathcal{A}$. For each object a, $(Ta, \tau_a : HTa \to Ta)$ is a cia for H; see Milius [16].

(ii) Let $H : \mathsf{CMS} \to \mathsf{CMS}$ be a signature endofunctor as in Example 2.2. Then any non-empty H-algebra (A, a) is a cia. More generally, recall that a *contracting* endofunctor H of CMS is one for which there exists a constant $\varepsilon < 1$ such that every derived map $\mathsf{CMS}(X, Y) \to \mathsf{CMS}(HX, HY)$ is an ε-contraction, i. e., we have $d_{HX,HY}(Hf, Hg) \leq \varepsilon d_{X,Y}(f, g)$ for every $f, g : X \to Y$. Then every non-empty algebra for H is a cia; see [2]. In fact, given any flat equation morphism $e : X \to HX + A$ in CMS, we know that $\mathsf{CMS}(X, A)$ is itself an object in the category, and it is not difficult to prove that the assignment $s \mapsto a \cdot (Hs + A) \cdot e$ is a contracting function from it to itself, see [2]. Then, by the Banach Fixed Point Theorem, there exists a unique fixed point of that contracting function, viz. a unique solution $e^\dagger$ of e.

(iii) Here is a specific case of the last point which is related to one of our examples in the introduction. Let I be the unit interval, and let $C(I)$ be the complete metric space of non-empty compact subspaces of I with the Hausdorff metric h; for two compact subsets s and t of I, $h(s, t) = \max\{\, d(s \to t), d(t \to s)\,\}$, where $d(s \to t) = \max_{x \in s} \min_{y \in t} d(x, y)$. We take the signature Σ with binary E and F, and unary G. For the algebra of interest, we take $(C(I), a : HC(I) \to C(I))$ given by the operations back in (3) in our introduction. In a little more detail, we mean that for sets s and t in $C(I)$, $a(E, s, t)$ is $\frac{1}{3}s \cup (\frac{2}{3} + \frac{1}{3}t)$, etc. This gives a cia. Note that although we have an operation $(-)^*$ in (3), this is not the interpretation of any symbol in our signature. This is not an accident: we may solve $x = G(x)$ uniquely in $C(I)$ (the solution is $\{\frac{1}{2}\}$) and we may even solve $x = E(x, x)$ uniquely (the solution is the Cantor set). But we cannot solve $x = x^*$ uniquely. So this function $x \mapsto x^*$ must be treated on a different level.

(iv) More generally, it is well-known that for a complete metric space X the non-empty compact subspaces of X with the Hausdorff metric form a complete metric space $(C(X), h)$; see, e.g. [4]. Furthermore, if $f_i : X \to X$, $i = 1, \ldots, n$, are contractions of the space X with contraction factors c_i, $i = 1, \ldots, n$, then it is easy to show that the map

$$a_X : C(X)^n \to C(X) \qquad (A_i)_{i=1,\ldots,n} \mapsto \bigcup_{i=1}^{n} f_i[A_i] \tag{16}$$

is a contraction with contraction factor $c = \max_i c_i$ (the product $C(X)^n$ is, of course, equipped with the maximum metric). In other words, given the f_i, we obtain on $C(X)$ the structure a_X of an H-algebra for the contracting endofunctor $H(X, d) = (X^n, c \cdot d_{\max})$. Thus, if X is not empty and thus has a compact subset, then $(C(X), a_X)$ is a cia for H.

As an illustration we come back to (iii) above and show that the Cantor "middle third" set c may be obtained via the cia structure on a certain space. Recall that c is the unique non-empty closed subset of the unit interval which satisfies $c = \frac{1}{3}c \cup (\frac{2}{3} + \frac{1}{3}c)$. So here we take $X = I$ and we note that the algebraic structure a on $C(I)$ is given as in (16) above, e. g. $a(E, s, t) = f(s) \cup g(t)$ for the $\frac{1}{3}$-contracting functions $f(x) = \frac{1}{3}x$ and $g(x) = \frac{1}{3}x + \frac{2}{3}$ on I.

Now consider the formal equation

$$x \approx E(x, x)\,.$$

It gives rise to a flat equation morphism $e : 1 \to H1 + C(I)$ which maps the element x of the trivial one point space 1 to the element (E, x, x) of $H1$. The unique solution $e^\dagger : 1 \to C(I)$ picks a non-empty closed set c satisfying $c = E(c, c) = f[c] \cup g[c]$. Hence $e^\dagger$ picks the Cantor set.

(v) Continuing with our last point, for each non-empty closed $s \in C(I)$, there is a unique $c(s)$ with $c(s) = E(s, c(s))$. In addition c is a continuous function. The argument is similar as above. But here we study the recursive program scheme (4) and solve this in $(C(I), a)$ in the appropriate sense.

2.6 Complete Elgot algebras

In many settings, one studies a fixed point operation on a structure like a complete partial order. And in such settings, one typically does not have *unique* fixed points. So completely iterative algebras are *not* the unifying concept capturing precisely what is needed to solve recursive program schemes. Instead, we shall need a weakening of the notion of a cia. This notion is that of a *(complete) Elgot algebra*, which was introduced in [2]. An Elgot algebra is a triple $(A, a, (\_)^\dagger)$, where $a : HA \to A$ is an H-algebra and $(\_)^\dagger$ is an operation taking a flat equation morphism $e : X \to HX + A$ to a solution $e^\dagger : X \to A$. Two properties are required of $(\_)^\dagger$. But since our work does not explicitly call on these properties, we shall not mention them.

Example 2.20. We mention some important examples of Elgot algebras.

(i) Every cia for H is an Elgot algebra, see [1, 16].

(ii) Let H be a locally continuous endofunctor on the category CPO; i. e., every derived map $\mathsf{CPO}(X, Y) \to \mathsf{CPO}(HX, HY)$ is continuous. It is shown in [2] that any H-algebra (A, a) with a least element $\perp$ is an Elgot algebra when to a flat equation morphism $e : X \to HX + A$ the least solution $e^\dagger$ is assigned. More precisely, define $e^\dagger$ as the join of the following increasing ω-chain in $\mathsf{CPO}(X, A)$: $e_0^\dagger$ is the constant function $\perp$; and given $e_n^\dagger$ let $e_{n+1}^\dagger = [a, A] \cdot (He_n^\dagger + A) \cdot e$.

(iii) Finally here is a specific case that we shall use later in our examples. Let Σ be the signature consisting of a constant one, two unary symbols succ and pred, a binary symbol $*$ and a ternary symbol ifzero. This gives rise to a signature functor H_Σ on CPO as explained in Example 2.2. Now consider the set $\mathbb{N}_\perp = \{\perp, 0, 1, 2, \ldots\}$ with the so-called flat cpo structure, i. e., $x \leq y$ iff $x = y$ or $x = \perp$. We interpret the operation symbols from Σ by the usual operations on $\mathbb{N}$—the constant 1 and the successor, predecessor, multiplication and conditional functions—extended to $\perp$ in the obvious way. Then $\mathbb{N}_\perp$ is an H_Σ-algebra, whence an Elgot algebra for that functor.

Theorem 2.21 ([2])**.** *The category* $\mathsf{Alg}^\dagger\, H$ *of Elgot H-algebras is isomorphic to the Eilenberg-Moore category $\mathcal{A}^T$ of monadic algebras for the free completely iterative monad T on H.*

It follows that for every Elgot algebra $(A, a, (\_)^\dagger)$ there is an associated structure $\widetilde{a} : TA \to A$ of an Eilenberg-Moore algebra, which we call *evaluation morphism* to remind us that in the special case of a signature functor H_Σ of Set this morphism evaluates every Σ-tree over A in the algebra A.

2.7 Standard interpreted solutions

We now summarize the theory of *standard interpreted solutions* to our formalization of recursive program schemes. For more on it, see [17, 18, 19].

Let $A = (A, a, (\_)^\dagger)$ be an Elgot algebra for an iteratable endofunctor H. We consider A as an Eilenberg-Moore algebra $\widetilde{a} : T^H A \to A$. Let $e : V \to T^{H+V}$ be a guarded recursive program scheme, and let $e^\dagger : V \to T^H$ be its unique (uninterpreted) solution. The *standard interpreted solution* of e is $e_A^\ddagger : VA \to A$, given by

$$e_A^\ddagger = \widetilde{a} \cdot (e^\dagger)_A \tag{17}$$

This solution extends the given algebraic structure $a : HA \to A$ in the sense that there is an operation $(-)^+$ taking solutions to flat equations under which $(A, [a, e^{\ddagger}_A], (-)^+)$ is an Elgot algebra for $H + V$. For the associated evaluation morphism $\widetilde{[a, e^{\ddagger}_A]} : T^{H+V}A \to A$ we have two important properties:

$$\widetilde{[a, e^{\ddagger}_A]} = \widetilde{a} \cdot \overline{[\kappa^H, e^{\dagger}]}_A \qquad (18)$$

$$e^{\ddagger}_A = \widetilde{[a, e^{\ddagger}_A]} \cdot e_A \qquad (19)$$

In addition, there is an important characterization result for interpreted solutions on CMS and CPO.

Theorem 2.22. *Let H be a contracting endofunctor on* CMS*, let A be a non-empty H-algebra, and let $e : V \to T^{H+V}$ be a guarded rps. The standard interpreted solution $e^{\ddagger}_A : VA \to A$ of e in A is the unique fixed point of the continuous function R on* $\mathsf{CMS}(VA, A)$ *defined by $R(f) = \widetilde{[a, f]} \cdot e_A$:*

$$R(f) = VA \xrightarrow{e_A} T^{H+V}A \xrightarrow{\widetilde{[a,f]}} A \qquad (20)$$

On CPO*, if H is locally continuous and A is an H-algebra with a least element, an rps $e : V \to T^{H+V}$ gives rise to a continuous operation R defined again by (20), and the standard interpreted solution $e^{\ddagger}_A$ of e in A is the least fixed point of R.*

Example 2.23. At this point, we return to a discussion in the Introduction. Although we did not say it at the time, we were concerned with interpreted solutions of rps's in a particular object in CMS. Let Σ be a signature with binary symbols E and F, let $H = H_\Sigma$ be the associated endofunctor on CMS, and let $A = C(I)$ be the H-algebra as in Example 2.19(iii). (This means that the interpretation of the symbols E and F are the functions with the same name given in (3).) Let V be the endofunctor obtained in the same way from a unary operation φ (or ψ, respectively). We are concerned with the two rps's $\varphi(s) \approx E(s, \varphi(s))$ and $\psi(s) \approx E(\psi(s), s)$. These equations correspond to two rps's $V \to T^{H+V}$ and, by Theorem 2.22, the standard interpreted solutions are given by two operations $\varphi^{\dagger}, \psi^{\dagger} : A \to A$. Our goal here is to show (6), repeated below:

$$\varphi^{\dagger}(s)^* = \psi^{\dagger}(s^*) \qquad (21)$$

For this, consider $\chi(s) \approx F(\chi(s), s)$ as an rps $e : V \to T^{H+V}$. By Theorem 2.22, $e^{\ddagger}_A : VA \to A$ is given by the unique non-expanding $f : A \to A$ such that for all

$s \in C(I)$,

$$f(s) = F(f(s), s) = E(f(s), s^*).$$

Now we claim that g and h also satisfy this, where $g(s) = \varphi^\dagger(s)^*$ and $h(s) = \psi^\dagger(s^*)$. This will verify (21). These functions g and h are non-expanding since the $(\_)^*$ operation preserves distances. What we do know, by Theorem 2.22 again, is that $g(s) = E(s, \varphi^\dagger(s))^*$ and $h(s) = E(\psi^\dagger(s^*), s^*)$. We reason as follows:

$$\begin{array}{lclclcl} g(s) & = & E(s, \varphi^\dagger(s))^* & = & E(\varphi^\dagger(s)^*, s^*) & = & E(g(s), s^*) \\ h(s) & = & E(\psi^\dagger(s^*), s^*) & = & E(h(s), s^*) & & \end{array}$$

This completes our verification.

3 Equational properties of first-order recursion

The main point of this paper is to consider properties of recursive program scheme solutions. It therefore makes sense to write the general properties of the dagger operation which we just defined. These are the equational properties studied in *iteration theory* [5]. However, please note that these are the properties of *ideal* morphisms, where a morphism $f : a \to Tb$ is *ideal* if it factors through τ_b; more precisely, $f = \tau_b \cdot f'$ for some $f' : a \to HTb$. As we stated in Theorem 2.4, for ideal $f : a \to T(a + b)$, there is a unique $f^\dagger : a \to Tb$ such that $f^\dagger = \mu_b \cdot T[f^\dagger, \eta_b] \cdot f$. We are interested in the operation $f \mapsto f^\dagger$, and the point of this section is to isolate the relevant algebraic laws concerning it and the rest of the category-theoretic machinery which we have been studying.

This section offers a preparation for Section 4 below. But the work here is not really needed for in Section 4. Much of the content of this section was known in slightly different formulations in other papers. For example, for the base category $\mathcal{A} = \mathsf{Set}$, Moss [21] studies laws of an iteration operator which makes sense even for non-ideal morphisms (one has a fixed element $\perp$ for "ungrounded" recursions solving equations of the form $x = x$). The laws there are somewhat more complicated to state, but they are more fully equational since they do not depend on the notion of an ideal morphism. Also, a completeness result is found in [21] for interpretations of the resulting logical system. In recent work by Adámek, Milius and Velebil [3] equational laws of an iteration operation which applies to all equation morphisms in iterative monads are studied for more general base categories than Set.

3.1 Preliminaries

This section presents four laws of the dagger operation in Sections 3.2–3.5. The verification of some of these use some general facts which we now quote.

Proposition 3.1. *Let* $f : a \to Tb$ *and* $h : d \to a$. *Then* $(f \cdot h)^* = f^* \cdot Th$.

Proof. $f^* \cdot Th \cdot \eta_d = f^* \cdot \eta_a \cdot h = f \cdot h$. Appealing to Theorem 2.4, we only need to check that $f^* \cdot Th$ is a morphism of H-algebras. But $f^* \cdot Th \cdot \tau_d = f^* \cdot \tau_a \cdot HTh = HTf^* \cdot HTh \cdot \tau_b = HT(f^* \cdot Th) \cdot \tau_b$. □

Proposition 3.2. *Let* $\mathsf{inr} : a \to a + b$ *be a coproduct injection. Then* $T\mathsf{inr} = (\eta_{a+b} \cdot \mathsf{inr})^*$. *Similarly,* $T\mathsf{inl} = (\eta_{a+b} \cdot \mathsf{inl})^*$.

Proof. By naturality of η, $T\mathsf{inr} \cdot \eta_a = \eta_{a+b} \cdot \mathsf{inr}$. And by naturality of τ, $T\mathsf{inr}$ is an H-algebra morphism. □

3.2 The fixed point identity

For all ideal $f : a \to T(a + b)$, $[f^\dagger, \eta_b]^* \cdot f = f^\dagger$.

Proof. The definition of $f^\dagger$ (in this paper) is that it is the unique morphism such that

$$f^\dagger = \mu_b \cdot T[f^\dagger, \eta_b] \cdot f$$

Since $[f^\dagger, \eta_b]^* = \mu_b \cdot T[f^\dagger, \eta_b]$, we see that $f^\dagger = [f^\dagger, \eta_b]^* \cdot f$, as desired. □

3.3 The pairing identity

Let $f : a \to T(a + b + c)$ and $g : b \to T(a + b + c)$ be ideal. Then $[f, g]$ is also ideal, and

$$[f, g]^\dagger \quad = \quad [[h, \eta_c]^* \cdot f^\dagger, h^\dagger],$$

where $h : b \to T(b + c)$ is $[f^\dagger, \eta_{b+c}]^* \cdot g$.

Proof. We omit the verification that $[f, g]$ is ideal. Instead, we shall verify that

$$[[h^\dagger, \eta_c]^* \cdot f^\dagger, h^\dagger] = [[[h^\dagger, \eta_c]^* \cdot f^\dagger, h^\dagger], \eta_c]^* \cdot [f, g]. \tag{22}$$

The pairing identity then follows from the fact that $[f, g]^\dagger$ is uniquely determined (cf. Theorem 2.4). To simplify notation, let

$$k = [[h^\dagger, \eta_c]^* \cdot f^\dagger, h^\dagger].$$

Then (22) reduces to the following two identities:

$$\begin{aligned} [h^\dagger, \eta_c]^* \cdot f^\dagger &= [k, \eta_c]^* \cdot f \\ h^\dagger &= [k, \eta_c]^* \cdot g \end{aligned} \tag{23}$$

In order to show these, we consider the diagram below:

$$\begin{tikzcd}
a \arrow[r, "f^\dagger"] \arrow[d, "f"'] & T(b+c) \arrow[dr, "{[h^\dagger,\eta_c]^*}"'] & b \arrow[l, "h"'] \arrow[d, "h^\dagger"] \\
T(a+b+c) \arrow[ur, "{[f^\dagger,\eta_{b+c}]^*}"'] \arrow[rr, "{[k,\eta_c]^*}"'] & & Tc
\end{tikzcd}$$

The triangles on the left and right commute by the fixed point identity. For the central triangle, we calculate using the Kleisli laws of $(-)^*$:

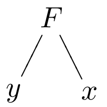
$$\begin{aligned} [h^\dagger, \eta_c]^* \cdot [f^\dagger, \eta_{b+c}]^* &= ([h^\dagger, \eta_c]^* \cdot [f^\dagger, \eta_{b+c}])^* \\ &= [[h^\dagger, \eta_c]^* \cdot f^\dagger, [h^\dagger, \eta_c]^* \cdot \eta_{b+c}]^* \\ &= [k, \eta_c]^* \end{aligned}$$

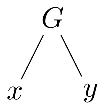
The first equation of (23) follows from the commutativity of the left and center. For the second equation of (23), we use the triangles on the right and in the center:

$$[k, \eta_c]^* \cdot g = [h^\dagger, \eta_c]^* \cdot [f^\dagger, \eta_{b+c}]^* \cdot g = [h^\dagger, \eta_c]^* \cdot h = h^\dagger.$$

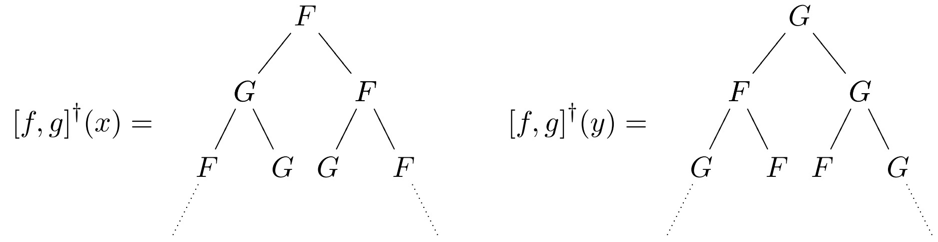
This completes the proof. □

Example 3.3. Let $\mathcal{A} = \mathsf{Set}$, let Σ be a signature with binary operations symbol F and G, and let $H : \mathsf{Set} \to \mathsf{Set}$ be the endofunctor corresponding to Σ. Let $a = \{x\}$, let $b = \{y\}$, and let c be arbitrary. Let $f : a \to T(a+b+c)$ and $g : b \to T(a+b+c)$ be described by

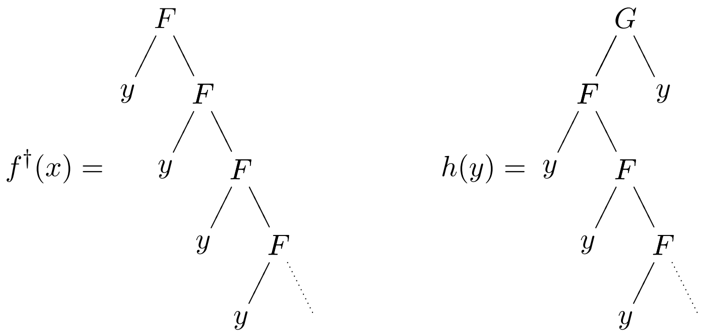
$f(x) =$ 

$g(y) =$ 

(In this example, we are going to omit all of the injections.) Then $[f, g]^\dagger$ is essentially given by the infinite trees



The pairing identity gives us a two-step procedure for finding these trees. First, we find $f^\dagger : a \to T(b+c)$ and $h = [f^\dagger, \eta_{b+c}]^* \cdot g : b \to T(b+c)$. These are given by



(24)

We obtained $f^\dagger$ by "solving" f, and in doing this we think of y as a constant. Then we obtained h by plugging $f^\dagger(x)$ for x in $g(y)$. And now we solve h in the same way, obtaining $[f,g]^\dagger(y)$. Finally, substitute this tree $[f,g]^\dagger(y)$ for y in $f^\dagger(x)$, and we obtain $[f,g]^\dagger(x)$. All of these points follow from the pairing identity.

3.4 The parameter identity

Let $f : a \to T(a+b)$ be ideal, and also let $g : b \to Tc$. Then $[T\mathsf{inl} \cdot \eta_a, T\mathsf{inr} \cdot g]^* \cdot f$ is ideal, and

$$([T\mathsf{inl} \cdot \eta_a, T\mathsf{inr} \cdot g]^* \cdot f)^\dagger = g^* \cdot f^\dagger.$$

Proof. Again, we omit the verification that the desired morphism is ideal, and we only check that its solution is $g^* \cdot f^\dagger$. To shorten the notation, let

$$i = [T\mathsf{inl} \cdot \eta_a, T\mathsf{inr} \cdot g].$$

We need only verify that the outside of the diagram below commutes.

$$\begin{tikzcd}
a \arrow[r, "f"] \arrow[d, "f^\dagger"'] & T(a+b) \arrow[r, "i^*"] \arrow[dl, "{[f^\dagger,\eta_b]^*}"'] \arrow[d, "{[g^*\cdot f^\dagger,g]^*}"] & T(a+c) \arrow[dl, bend left, "{[g^*\cdot f^\dagger,\eta_c]^*}"] \\
Tb \arrow[r, "g^*"'] & Tc &
\end{tikzcd}$$

We check that the inside parts commute. The leftmost triangle commutes by the fixed point identity. The central triangle commutes since $g^* \cdot [f^\dagger, \eta_b]^* = (g^* \cdot [f^\dagger, \eta_b])^* = [g^* \cdot f^\dagger, g^* \cdot \eta_b]^* = [g^* \cdot f^\dagger, g]^*$. The most interesting verification is

$$\begin{aligned}
& [g^* \cdot f^\dagger, \eta_c]^* \cdot i^* \\
= \; & [g^* \cdot f^\dagger, \eta_c]^* \cdot [T\mathsf{inl} \cdot \eta_a, T\mathsf{inr} \cdot g]^* \\
= \; & [g^* \cdot f^\dagger, \eta_c]^* \cdot [\eta_{a+c} \cdot \mathsf{inl}, (\eta_{a+c} \cdot \mathsf{inr})^* \cdot g]^* \\
= \; & ([g^* \cdot f^\dagger, \eta_c]^* \cdot [\eta_{a+c} \cdot \mathsf{inl}, (\eta_{a+c} \cdot \mathsf{inr})^* \cdot g])^* \\
= \; & [[g^* \cdot f^\dagger, \eta_c]^* \cdot \eta_{a+c} \cdot \mathsf{inl}, [g^* \cdot f^\dagger, \eta_c]^* \cdot (\eta_{a+c} \cdot \mathsf{inr})^* \cdot g]^* \\
= \; & [g^* \cdot f^\dagger, ([g^* \cdot f^\dagger, \eta_c]^* \cdot \eta_{a+c} \cdot \mathsf{inr})^* \cdot g]^* \\
= \; & [g^* \cdot f^\dagger, \eta_c^* \cdot g]^* \\
= \; & [g^* \cdot f^\dagger, g]^*
\end{aligned}$$

We used Proposition 3.2 in asserting $T\mathsf{inr} = (\eta_{a+c} \cdot \mathsf{inr})^*$. □

Example 3.4. Let $f : a \to T(a+b)$ be given similarly as in Example 3.3, and let $g : b \to Tc$ be given by $g(y) = F(z,z)$. Then $[T\mathsf{inl} \cdot \eta_a, T\mathsf{inr} \cdot g]^* \cdot f$ is given by $x \mapsto F(F(z,z),x)$. We may of course solve this directly. The parameter identity tells us that we would get the same tree as if we took the tree for $f^\dagger(x)$ from (24) and substituted $F(z,z)$ for y.

3.5 The functorial dagger implication for base morphisms

Let $f : a \to T(a+c)$ and $g : b \to T(b+c)$ be ideal, and let $h : a \to b$. Suppose that $T(h + id_c) \cdot f = g \cdot h$. Then, $f^\dagger = g^\dagger \cdot h$.

The proof goes by examining the diagram below:

$$\begin{tikzcd}
a \arrow[r, "f"] \arrow[d, "h"'] & T(a+c) \arrow[d, "T(h+id_c)"] \\
b \arrow[r, "g"] \arrow[d, "g^\dagger"'] & T(b+c) \arrow[dl, "{[g^\dagger, \eta_c]^*}"] \\
Tc &
\end{tikzcd}$$

Both parts commute, and by Proposition 3.1, $[g^\dagger, \eta_c]^* \cdot T(h + id_c) = [g^\dagger \cdot h, \eta_c]^*$.

Example 3.5. For a very quick example, let $a = \{x, y\}$, $b = \{z\}$, and $h : a \to b$ the constant function, and f and g given by

$$\begin{array}{rcl} x & \approx & F(x, y, w) \\ y & \approx & F(y, y, w) \end{array} \qquad \text{and} \qquad z \approx F(z, z, w),$$

respectively. The functorial dagger implication applies because $T(h+id_c)\cdot f = g\cdot h$. It tells us that $f^\dagger(x) = g^\dagger(z) = f^\dagger(y)$.

Summary. The point of this section was to provide examples of equational properties of interest concerning a simple kind of recursive definition, namely first order recursion. The identities in this section are not so surprising, since they correspond to the laws of iteration theory [21]. What is more interesting is that the same *kind* of laws hold when we move to the more involved setting of rps solutions. We wish to emphasize, however, that the properties we establish below for rps solutions do not follow from the earlier work: the definitions of guardedness differ, as do points in the formulations of the identities themselves. We are not aware of any unified treatment of the work in this section with what we do in Section 4 below.

4 Equational properties of uninterpreted rps solutions

Let us reiterate what we have done so far in this paper. We began with a few puzzles concerning particular interpreted recursions. Then we launched into a much broader discussion of both interpreted and uninterpreted recursion in Section 2. In this section we will study equational laws of the operation of taking the unique solution of a guarded rps. The laws we will establish are inspired by the laws studied in iteration theory [5], and so we use the same names for our laws here. Notice however, that these laws are new for rps solutions in the setting that we study in the present paper.

4.1 The fixed point identity

Let $e : V \to T^{H+V}$ be a guarded rps. Then $\overline{[\kappa^H, e^\dagger]} \cdot e = e^\dagger$ (recall from Theorem 2.9 that $\overline{[\kappa^H, e^\dagger]}$ denotes the unique extension of the ideal natural transformation $[\kappa^H, e^\dagger]$ to a monad morphism).

This identity is just a restatement of the definition of the solution natural transformation $e^\dagger$.

4.2 The functoriality law

Suppose that we have an rps defining operations from a signature Φ recursively from given operations from a signature Σ. The functoriality law states that we can rename the symbols of Φ or permute argument variables of symbols from Φ without changing the solution of our rps as long as we rename or permute on both sides of the formal equations of the rps consistently. The categorical formulation of this fact is this:

Proposition 4.1. *Suppose that $e : V \to T^{H+V}$ and $f : W \to T^{H+W}$ are guarded rps's, and let $n : V \to W$ be a natural transformation such that the square*

$$\begin{tikzcd}
V \arrow[r, "e"] \arrow[d, "n"'] & T^{H+V} \arrow[d, "T^{H+n}"] \\
W \arrow[r, "f"'] & T^{H+W}
\end{tikzcd} \tag{25}$$

commutes. Then $f^\dagger \cdot n = e^\dagger$.

Proof. We need only show that $f^\dagger \cdot n$ solves e. Our candidate solution $f^\dagger \cdot n$ is obviously an ideal natural transformation since $f^\dagger$ is one. Now consider the diagram

$$\begin{tikzcd}
V \arrow[r, "n"] \arrow[dd, "e"'] & W \arrow[r, "f^\dagger"] \arrow[d, "f"'] & T^H \\
 & T^{H+W} \arrow[ur, "\overline{[\kappa^H, f^\dagger]}"'] & \\
T^{H+V} \arrow[ur, "T^{H+n}"'] \arrow[uurr, bend right, "\overline{[\kappa^H, f^\dagger \cdot n]}"'] & &
\end{tikzcd}$$

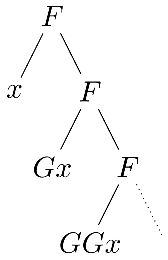
All its inner parts commute; the upper left-hand part commutes by hypothesis, the upper right-hand one since $f^\dagger$ is a solution of f, and the lower part commutes by Corollary 2.14(ii). □

Example 4.2. Let $\mathcal{A} = \mathsf{Set}$ and let Σ be a signature with one binary operation symbol F and one unary symbol G. Let Φ be a signature with a unary symbol φ, and let Ψ have one binary symbol ψ. Let H, V, and W be the endofunctors on Set associated to Σ, Φ and Ψ, respectively. Consider the guarded rps's $e : V \to T^{H+V}$ and $f : W \to T^{H+W}$ given by the formal equations

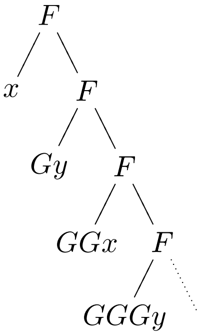
$$\varphi(x) \approx F(x, \varphi(Gx)) \qquad \text{and} \qquad \psi(x,y) \approx F(x, \psi(Gy, Gx))\,,$$

respectively. Let $n : V \to W$ be the natural transformation with components n_X given by $n_X(\varphi, x) = (\psi, x, x)$. Clearly, n makes the square (25) in Proposition 4.1 commute. The solutions $e^\dagger : V \to T^H$ and $f^\dagger : W \to T^H$ are given by

$\varphi^\dagger(x) =$



$\psi^\dagger(x, y) =$



(26)

Obviously, we have $\varphi^\dagger(x) = \psi^\dagger(x, x)$. This follows from the functoriality law.

4.3 The parameter identity

Suppose again that we have an rps defining operations from a signature Φ recursively from given operations from a signature Σ. Let us assume that we want to substitute the symbols from Σ by terms or even infinite trees over another signature Γ. There are two ways to obtain a solution for operations of Φ as Γ-trees: either one first substitutes the symbols from Σ in the rps and then solves the resulting rps, or one first solves the rps with givens Σ and then substitutes all Σ-symbols. Proposition 4.3 below states that these two ways to solve our given rps are the same.

Proposition 4.3. *Let H, K and V be iteratable endofunctors on $\mathcal{A}$, and let $e : V \to T^{H+V}$ be a guarded rps. For any ideal natural transformation $n : H \to T^K$ consider the ideal natural transformation*

$$s \equiv H + V \xrightarrow{n+\kappa^V} T^K + T^V \xrightarrow{[T^{\mathsf{inl}}, T^{\mathsf{inr}}]} T^{K+V}$$

and form the rps

$$n \bullet e \equiv \; V \xrightarrow{e} T^{H+V} \xrightarrow{\overline{s}} T^{K+V} .$$

Then $n \bullet e$ is a guarded rps, and its solution is related to that of e by

$$(n \bullet e)^\dagger = \overline{n} \cdot e^\dagger .$$

Proof. Observe first that s is indeed an ideal natural transformation since n and κ^V are ideal natural transformations and by Lemma 2.11 applied to T^{inl} and T^{inr}:

$$\begin{tikzcd}
H \arrow[rr, "\mathsf{inl}"] \arrow[d, "n'"'] \arrow[dr, "n"] \arrow[dd, bend right=90, "s'\cdot\mathsf{inl}"'] & & H+V \arrow[d, "n+\kappa^V"] & & V \arrow[ll, "\mathsf{inr}"'] \arrow[dl, "\kappa^V"'] \arrow[d, "V\eta^V"] \arrow[dd, bend left=90, "s'\cdot\mathsf{inr}"] \\
KT^K \arrow[r, "\tau^K"] \arrow[d, "\mathsf{inl}*T^{\mathsf{inl}}"'] & T^K \arrow[r, "\mathsf{inl}"] \arrow[dr, "T^{\mathsf{inl}}"'] & T^K + T^V \arrow[d, "\mathsf{can}"] & T^V \arrow[l, "\mathsf{inr}"'] \arrow[dl, "T^{\mathsf{inr}}"] & VT^V \arrow[l, "\tau^V"'] \arrow[d, "\mathsf{inr}*T^{\mathsf{inr}}"] \\
LT^L \arrow[rr, "\tau^L"'] & & T^L & & LT^L \arrow[ll, "\tau^L"]
\end{tikzcd} \tag{27}$$

In this diagram, L is $K+V$, and the middle vertical arrow is s. Also, $s' : H+V \to (K+V)T^{K+V}$ is defined so that the two outside parts commute. All the inner parts commute, easily. And so we see that $s = \tau^{K+V} \cdot s'$. This shows that s is an ideal natural transformation.

Now to see that $n \bullet e$ is guarded, we use the diagram below, where we use L for $K+V$.

$$\begin{tikzcd}
V \arrow[r, "e"] \arrow[ddr, bend right] & T^{H+V} \arrow[rr, "\overline{s}"] & & T^L \\
& (H+V)T^{H+V} \arrow[u, "\tau^{H+V}"'] \arrow[r, "s'*\overline{s}"] & LT^LT^L \arrow[r, "L\mu^L"] & LT^L \arrow[u, "\tau^L"'] \\
& HT^{H+V} \arrow[u, "\mathsf{inl}*T^{H+V}"'] \arrow[r, "(KT^{\mathsf{inl}}\cdot n')*\overline{s}"'] & KT^LT^L \arrow[u, "\mathsf{inl}*T^L*T^L"'] \arrow[r, "K\mu^L"'] & KT^L \arrow[u, "\mathsf{inl}*T^L"']
\end{tikzcd}$$

Its left-hand part commutes since e is guarded, the upper square commutes by Lemma 2.11, and the lower right-hand square is obvious. For the remaining lower left-hand square it suffices to consider the two parallel components on the left and right-hand side of $*$ separately. The right-hand component clearly commutes, and the left-hand one also does by the left-most part of Diagram (27) above. The outside of the figure shows the desired guardedness.

Next, we prove that $\overline{n} \cdot e^\dagger$ solves $n \bullet e$. Then, the desired result follows by the uniqueness of solutions. In fact, it is clear that $\overline{n} \cdot e^\dagger$ is an ideal natural transformation since $e^\dagger$ is one and by applying Lemma 2.11 to $\overline{n}$. Consider the diagram

$$\begin{tikzcd}
V \arrow[r, "e^\dagger"] \arrow[d, "e"'] \arrow[dd, bend right=60, "n \bullet e"'] & T^H \arrow[r, "\overline{n}"] & T^K \\
T^{H+V} \arrow[ur, "\overline{[\kappa^H, e^\dagger]}"'] \arrow[d, "\overline{s}"'] & & \\
T^{K+V} \arrow[uurr, "\overline{[\kappa^K, \overline{n} \cdot e^\dagger]}"'] & &
\end{tikzcd}$$

The left-hand part is the definition of $n \bullet e$. The upper left triangle commutes since $e^\dagger$ is a solution of e. Thus, we are done if we show that the lower right-hand part commutes. By Proposition 2.12(iii), it suffices to show that the following square

$$\begin{tikzcd}
H + V \arrow[r, "{[\kappa^H, e^\dagger]}"] \arrow[d, "s"'] & T^H \arrow[d, "\overline{n}"] \\
T^{K+V} \arrow[r, "\overline{[\kappa^K, \overline{n} \cdot e^\dagger]}"'] & T^K
\end{tikzcd} \tag{28}$$

commutes. We consider the two coproduct components of Diagram (28) separately. For the left-hand component, the upper right passage gives $\overline{n} \cdot \kappa^H = n$, and we need only show that the lower left passage also yields n. To see this, consider

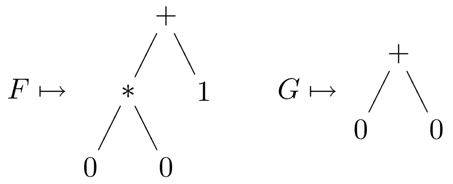
$$\begin{tikzcd}
H + V \arrow[r, "s"] & T^{K+V} \arrow[r, "\overline{[\kappa^K, \overline{n} \cdot e^\dagger]}"] & T^K \\
H \arrow[u, "\mathsf{inl}"] \arrow[r, "n"'] & T^K \arrow[u, "T^{\mathsf{inl}}"'] \arrow[ur, "\overline{\kappa^H} = id"'] &
\end{tikzcd}$$

The square commutes by definition of s, and the triangle by Corollary 2.14(ii). We conclude with the right-hand component of Diagram (28). The upper right passage yields $\overline{n} \cdot e^\dagger$. To see that the lower left passage yields the same thing, consider

$$\begin{tikzcd}
H + V \arrow[r, "s"] & T^{K+V} \arrow[r, "\overline{[\kappa^K, \overline{n} \cdot e^\dagger]}"] & T^K \\
V \arrow[u, "\mathsf{inr}"] \arrow[r, "\kappa^V"'] \arrow[urr, bend right=50, "\overline{n} \cdot e^\dagger"'] & T^V \arrow[u, "T^{\mathsf{inr}}"'] \arrow[ur, "\overline{n} \cdot e^\dagger"'] &
\end{tikzcd}$$

Its left-hand square commutes by the definition of s, the right-hand triangle commutes by Corollary 2.14(ii), and the lower part commutes due to Proposition 2.12(i). □

Example 4.4. Let $\mathcal{A} = \mathsf{Set}$, let Σ be a signature that consists of a unary symbol G and a binary one F, and let Φ have only the unary symbol φ. Furthermore, let Γ consist of two binary operation symbols $+$ and $*$. Suppose that H, V and K are the endofunctors of Set associated to Σ, Φ and Γ, respectively. We specify $n : H \to T^K$ by instead giving an associated $\Sigma \to T_\Gamma \cdot J$ and then using the correspondence in (10). In pictures, we take



(29)

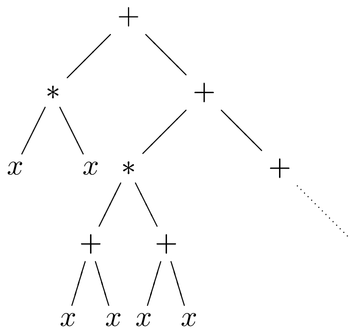
Finally, let the guarded rps $e : V \to T^{H+V}$ be given by the formal equation

$$\varphi(x) \approx F(x, \varphi(Gx)), \tag{30}$$

whose solution $e^\dagger : V \to T^H$ is shown in (26) above as $\varphi^\dagger(x)$. The rps $n \bullet e$ is given by replacing in the right-hand side of (30) all Σ-symbols according to (29), i. e., $n \bullet e$ is described by the formal equation

$$\varphi(x) \approx (x * x) + \varphi(x + x),$$

where we write $+$ and $*$ in infix notation as usual. The solution $(n \bullet e)^\dagger$ is essentially given by the infinite tree



The parameter identity confirms that we get the same tree when we form $\overline{n} \cdot e^\dagger$, i. e., we take $\varphi^\dagger(x)$ and perform second-order substitution (i. e., substitution of operation symbols from Σ by Γ-trees) according to (29).

4.4 The pairing identity

Let H, V and W be iteratable endofunctors of $\mathcal{A}$. Suppose we have a guarded rps factoring as

$$\begin{tikzcd}
V+W \arrow[r, "{[e,f]}"] \arrow[dr, "{[e^*,f^*]}"'] & T^{H+V+W} \\
& HT^{H+V+W} \arrow[u, "\tau^{H+V+W}\cdot(\mathsf{inl}*id)"']
\end{tikzcd} \tag{31}$$

We wish to compute the solution of $[e,f]$ by first solving $f : W \to T^{(H+V)+W}$, which is also a guarded rps, and then plugging into e the solution $f^\dagger : W \to T^{H+V}$ for W. More formally, we consider the rps

$$g \equiv V \xrightarrow{e} T^{H+V+W} \xrightarrow{\overline{[\kappa^{H+V}, f^\dagger]}} T^{H+V}\,. \tag{32}$$

The solution $g^\dagger : V \to T^H$ is an ideal natural transformation, and so we can form the monad morphism $\overline{[\kappa^H, g^\dagger]} : T^{H+V} \to T^H$. The pairing identity now states that the equation

$$[e,f]^\dagger = [g^\dagger, \overline{[\kappa^H, g^\dagger]} \cdot f^\dagger] : V+W \to T^H \tag{33}$$

holds.

Proposition 4.5. *For a guarded rps $[e,f]$ as above, the rps g from (32) is guarded, and the pairing identity (33) holds.*

Proof. Throughout this proof we write G as an abbreviation for $H+V$, and we write σ for $[\kappa^G, f^\dagger]$. In order to prove that g is guarded, we establish below that the outside of the following diagram commutes:

$$\begin{tikzcd}
V \arrow[r, "e"] \arrow[rrr, bend left, "g"] \arrow[ddr, "e^*"'] & T^{G+W} \arrow[rr, "\overline{\sigma}"] & & T^G \\
& (G+W)T^{G+W} \arrow[u, "\tau^{G+W}"'] \arrow[r, "{[G\eta^G, f^{\dagger\prime}]*\overline{\sigma}}"] & GT^GT^G \arrow[r, "G\mu^G"] & GT^G \arrow[u, "\tau^G"'] \\
& HT^{G+W} \arrow[u, "\mathsf{inl}*id"'] \arrow[r, "H\eta^G*\overline{\sigma}"'] & HT^GT^G \arrow[u, "\mathsf{inl}*id*id"'] \arrow[r, "H\mu^G"'] & HT^G \arrow[u, "\mathsf{inl}*id"']
\end{tikzcd}$$

We must explain $f^{\dagger\prime}$. The unique solution $f^\dagger : W \to T^G$ is an ideal natural transformation, and so we have $f^{\dagger\prime} : W \to GT^G$ with $\tau^G \cdot f^{\dagger\prime} = f^\dagger$. Now all inner parts of the above diagram commute: the left-hand part due to the guardedness of $[e, f]$, the upper right-hand square commutes by Lemma 2.11, and the two lower right-hand squares are obvious.

Next, we show (33). For this, we show that $[g^\dagger, \overline{[\kappa^H, g^\dagger]} \cdot f^\dagger]$ is a solution of $[e, f]$, and then appeal to the uniqueness of solutions. We shall first establish that $f^\dagger$ factors as $\tau^G \cdot (\mathsf{inl} * id) \cdot f^{\dagger\prime\prime}$, where

$$f^{\dagger\prime\prime} \equiv V \xrightarrow{f^*} HT^{G+W} \xrightarrow{H\overline{\sigma}} HT^G \, ,$$

Let K be a shorthand for $G + W$, and consider the diagram

$$\begin{tikzcd}
W \arrow[rrrr, "f^\dagger"] \arrow[d, "f^*"'] \arrow[dddd, bend right=60, "f"'] & & & & T^G \\
HT^K \arrow[r, "H\overline{\sigma}"] \arrow[ddd, "\mathsf{inl}*id"'] \arrow[drr, bend right, "H\eta^G*\overline{\sigma}"'] & HT^G \arrow[r, equal] \arrow[dr, "H\eta^G T^G"] & HT^G \arrow[r, "\mathsf{inl}*id"] & GT^G \arrow[ur, "\tau^G"] & \\
& & HT^G T^G \arrow[u, "H\mu^G"'] \arrow[dr, "\mathsf{inl}*id*id"] & & \\
KT^K \arrow[rrr, "{[G\eta^G, f^{\dagger\prime}]*\overline{\sigma}}"'] \arrow[d, "\tau^K"'] & & & GT^G T^G \arrow[uu, "G\mu^G"'] & \\
T^K \arrow[rrrruuuu, bend right=40, "\overline{\sigma}"'] & & & &
\end{tikzcd}$$

We are to show that the upper part commutes. The left-hand part commutes by (31), the lower part commutes due to Lemma 2.11, and all other remaining parts are clear. The outward shape commutes since $f^\dagger$ solves f. It then follows that the desired upper part also does.

In order to complete the proof we have to show that the triangle

$$\begin{tikzcd}
V + W \arrow[r, "{[g^\dagger, \overline{[\kappa^H, g^\dagger]} \cdot f^\dagger]}"] \arrow[d, "{[e,f]}"'] & T^H \\
T^{H+V+W} \arrow[ur, "\overline{[\kappa^H, g^\dagger, \overline{[\kappa^H, g^\dagger]} \cdot f^\dagger]}"'] &
\end{tikzcd} \tag{34}$$

commutes. We consider its coproduct components separately. For the left-hand component, consider the diagram

$$\begin{tikzcd}
V \arrow[rr, "g^\dagger"] \arrow[dr, "g"] \arrow[dd, "e"'] & & T^H \\
& T^{H+V} \arrow[ur, "\overline{[\kappa^H, g^\dagger]}"'] & \\
T^{H+V+W} \arrow[ur, "\overline{[\kappa^{H+V}, f^\dagger]}"'] \arrow[rruu, bend right, "\overline{[\kappa^H, g^\dagger, \overline{[\kappa^H, g^\dagger]}\cdot f^\dagger]}"'] & &
\end{tikzcd} \tag{35}$$

Its left-hand triangle is the definition (32) of g, and its upper part commutes since $g^\dagger$ is a solution of g. To see that the lower right-hand part commutes, it suffices by Proposition 2.12(iii) to show that the following diagram commutes:

$$\begin{tikzcd}
H+V+W \arrow[r, "{[\kappa^{H+V}, f^\dagger]}"] \arrow[dr, "{[\kappa^H, g^\dagger, \overline{[\kappa^H, g^\dagger]}\cdot f^\dagger]}"'] & T^{H+V} \arrow[d, "\overline{[\kappa^H, g^\dagger]}"] \\
& T^H
\end{tikzcd}$$

But this is trivial; for the left-hand component $H + V$ use Proposition 2.12(i), and for the right-hand one nothing needs to be shown.

Finally, we show that the right-hand component of Diagram (34) commutes. To this end consider the commutative diagram

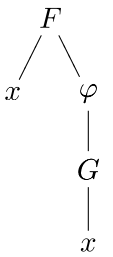
$$\begin{tikzcd}
W \arrow[r, "f^\dagger"] \arrow[d, "f"'] & T^{H+V} \arrow[r, "\overline{[\kappa^H, g^\dagger]}"] & T^H \\
T^{H+V+W} \arrow[ur, "\overline{[\kappa^{H+V}, f^\dagger]}"'] \arrow[rru, bend right, "\overline{[\kappa^H, g^\dagger, \overline{[\kappa^H, g^\dagger]}\cdot f^\dagger]}"'] & &
\end{tikzcd}$$

The left-hand triangle commutes since $f^\dagger$ solves f, and the right-hand part commutes as we have already seen in Diagram (35). This completes our proof. $\square$

Example 4.6. Let $\mathcal{A} = \mathsf{Set}$ and let Σ be a signature with a unary symbol G and two binary ones E and F. Let Φ and Ψ be signatures expressing unary symbols φ and ψ, respectively. Finally, let U, V and W be the signature functors of Set associated to Σ, Φ and Ψ, respectively. Consider the rps $[e, f] : V + W \to T^{U+V+W}$ given by the recursive definition in (7) again, repeated below:

$$\begin{array}{rcl}
\varphi(x) & \approx & E(x, \psi(Gx)) \\
\psi(x) & \approx & F(x, \varphi(Gx))
\end{array}$$

The pairing identity tells us that we can solve this recursive program scheme in a step-by-step fashion. We take first the solution $f^\dagger : W \to T^{U+V}$ of the second equation in (7), where φ is considered as a given operation symbol, i. e., $f^\dagger$ essentially is described by the $(\Sigma + \Phi)$-tree



(36)

This is the same as the right-hand side of $\psi(x)$ in (7) because in this right-hand side no "recursive call" to ψ is made. The guarded rps $g : V \to T^{U+V}$ expresses the following recursion

$$\varphi(x) \approx E(x, F(Gx, \varphi(GGx)))\,. \tag{37}$$

That is, we have plugged in (36) for $\psi(x)$ in (7). The solution $g^\dagger : V \to T^U$ yields the uninterpreted solution for $\varphi(x)$ and is given by the infinite tree $\varphi^\dagger(x)$ shown on the left in (8) above. Plugging $g^\dagger$ into $f^\dagger$ corresponds to plugging $\varphi^\dagger(x)$ into (36). In this way, we obtain the uninterpreted solution for $\psi(x)$, shown on the right in (8).

4.5 A derived property: the Bekić-Scott law

Using the properties we have established in the previous sections we will now derive a simplified but often useful version of the pairing identity which we will call *the Bekić-Scott law*.

Definition 4.7. Let H, V and K be endofunctors with $H + K$ iteratable. We call a natural transformation $e : V \to T^{H+K}$ *guarded by* H if there exists a factorization e^* of e of the form

$$\begin{tikzcd}
V \arrow[r, "e"] \arrow[ddr, bend right, "e^*"'] & T^{H+K} \\
& (H+K)T^{H+K} \arrow[u, "\tau^{H+K}"'] \\
& HT^{H+K} \arrow[u, "\mathsf{inl}*T^{H+K}"']
\end{tikzcd}$$

Let H, V and W be iteratable endofunctors of $\mathcal{A}$. Suppose we have a guarded rps $e : V \to T^{H+V}$ and an rps $f : W \to T^{H+V+W}$ which is guarded by H. We wish to compute the uninterpreted solution of the guarded rps

$$[e', f] : V + W \to T^{H+V+W} ,$$

where

$$e' \equiv V \xrightarrow{e} T^{H+V} \xrightarrow{T^{\mathsf{inl}}} T^{H+V+W} .$$

In fact, one first solves e and f to obtain $e^\dagger : V \to T^H$ and $f^\dagger : W \to T^{H+V}$, and then plugs the solution $e^\dagger$ into $f^\dagger$. More formally, the Bekić-Scott law states that

$$[e', f]^\dagger = [e^\dagger, \overline{[\kappa^H, e^\dagger]} \cdot f^\dagger] : V + W \to T^H . \tag{38}$$

Proposition 4.8. *Let e and f be rps's as above. Then the Bekić-Scott law (38) holds.*

Proof. To see that $[e', f]$ is a guarded rps we consider the coproduct components separately. Indeed, nothing has to be shown for the right-hand component f as it is guarded by H. And for the left-hand component we have the commutative diagram

$$\begin{tikzcd}
V \arrow[r, "e"] \arrow[rr, bend left, "e'"] \arrow[ddr, bend right, "e^*"'] & T^{H+V} \arrow[r, "T^{\mathsf{inl}}"] & T^{H+V+W} \\
& (H+V)T^{H+V} \arrow[u, "\tau^{H+V}"'] \arrow[r, "\mathsf{inl} * T^{\mathsf{inl}}"'] & (H+V+W)T^{H+V+W} \arrow[u, "\tau^{H+V+W}"'] \\
& HT^{H+V} \arrow[u, "\mathsf{inl} * id"'] \arrow[r, "HT^{\mathsf{inl}}"'] & HT^{H+V+W} \arrow[u, "\mathsf{inl} * id"']
\end{tikzcd}$$

Here e^* is defined so that the left-hand region commutes; we obtain e^* from the guardedness of e. The upper right-hand square commutes by Corollary 2.14(ii), and the lower right-hand square is obvious. So the outside of the figure commutes, and we have the desired guardedness of $[e', f]$.

The equation (38) follows immediately from the pairing identity (33) if we show that for the rps g formed from e' similarly as in (32) we have $g^\dagger = e^\dagger$. But in fact, here we have $g = e$:

$$\begin{tikzcd}
V \arrow[r, "e"] \arrow[ddr, bend right, "g"'] & T^{H+V} \arrow[d, "T^{\mathsf{inl}}"] \arrow[dd, bend left=70, "\overline{\kappa^{H+V}}"] \\
& T^{H+V+W} \arrow[d, "\overline{[\kappa^{H+V}, f^{\dagger}]}"] \\
& T^{H+V}
\end{tikzcd}$$

In fact, the left-hand part is the definition of g, and the right hand part commutes by Corollary 2.14(ii). Finally, use that $\overline{\kappa^{H+V}} = id$ holds by Proposition 2.12(ii). □

Remark 4.9. Using the parameter identity we easily derive from (38) the following equation

$$[e', f]^{\dagger} = [e^{\dagger}, (\overline{h} \cdot f)^{\dagger}] \,,$$

where

$$h \equiv H + V + W \xrightarrow{[\kappa^H, e^{\dagger}] + \kappa^W} T^H + T^W \xrightarrow{[T^{\mathsf{inl}}, T^{\mathsf{inr}}]} T^{H+W} \,.$$

In fact, for $n = [\kappa^H, e^{\dagger}]$ we have $\overline{h} \cdot f = n \bullet f$, whence $(\overline{h} \cdot f)^{\dagger} = \overline{n} \cdot f^{\dagger}$ by the parameter identity, whence the above equation follows from the Bekić-Scott law (38).

5 Properties of Standard Interpreted Solutions

In general, rps's have many solutions in a given Elgot algebra. So to prove properties relating solutions of different rps's, we must fix on some canonical solution operation. Fortunately, we have isolated the concept of a standard interpreted solution of an rps in an Elgot algebra, see Section 2.7.

In this section, we establish some properties of standard interpreted rps solutions. The work here builds on what we did in the previous section, but we also need a new definition and a result pertaining to it.

Definition 5.1. Let A be an object of $\mathcal{A}$, and consider two iteratable functors H and K on $\mathcal{A}$, and an ideal natural transformation $n : H \to T^K$. Let $A(H) = (A, a, (\_)^*)$ be an Elgot algebra for H, and let $A(K) = (A, b, (-)^+)$ be an Elgot algebra for K. Let $\widetilde{a}$ and $\widetilde{b}$ be the associated Eilenberg-Moore structures for $A(H)$ and $A(K)$, respectively. We say that $A(H)$ and $A(K)$ are *n-related* if $\widetilde{a} = \widetilde{b} \cdot \overline{n}_A$.

Proposition 5.2. *Let H, K, and $n : H \to T^K$ be as above, and let $A(K) = (A, b, (\_)^*)$ be a K-Elgot algebra. Then there is an Elgot H-algebra $A(H)$ which is n-related to $A(K)$. The H-algebra structure underlying $A(H)$ is (A, a), where $a = \widetilde{b} \cdot n_A$.*

Proof. Let $\widetilde{a} : T^H A \to A$ be $\widetilde{b} \cdot \overline{n}_A$. Then $(A, \widetilde{a})$ is an Eilenberg-Moore algebra for T^H. This follows from the fact that $\widetilde{b}$ is an Eilenberg-Moore algebra for T^K and $\overline{n} : T^H \to T^K$ is a monad morphism, see e. g. Proposition 4.5.9 in [6].

For the second assertion, our general theory tells us that the H-algebra structure a of the Elgot algebra $A(H)$ associated to $(A, \widetilde{a})$ is $\widetilde{a} \cdot \kappa^H_A$. Thus, the desired result follows by using Proposition 2.12(i):

$$a = \widetilde{a} \cdot \kappa^H_A = \widetilde{b} \cdot \overline{n}_A \cdot \kappa^H_A = \widetilde{b} \cdot n_A \,.$$

□

5.1 Using laws about the givens in CMS and CPO

Proposition 5.3. *Let H and V be contracting endofunctors of* CMS *(or locally continuous on* CPO*). Let $e, f : V \to T^{H+V}$ be guarded recursive program schemes over* CMS *(or* CPO*), let (A, a) be an H-algebra which is non-empty (or which has a least element), and hence is a cia (or Elgot algebra) for H. Assume that e and f have the following equivalence property: for all morphisms $b : VA \to A$, $\widetilde{[a, b]} \cdot e_A = \widetilde{[a, b]} \cdot f_A$. Under these assumptions, $e^\ddagger_A = f^\ddagger_A$.*

Proof. First notice that the coproduct $H + V$ is contracting (or locally continuous). So for every $b : VA \to A$ the morphism $[a, b]$ is part of the structure of a cia (or Elgot algebra) for $H + V$, and we can form $\widetilde{[a, b]}$. Next, we apply the condition in the hypothesis, taking $e^\ddagger_A$ for b, and also equation (19) to see that

$$\widetilde{[a, e^\ddagger_A]} \cdot f_A \quad = \quad \widetilde{[a, e^\ddagger_A]} \cdot e_A \quad = \quad e^\ddagger_A \,.$$

At this point, we need slightly different arguments for CMS and CPO. In both cases, we use Theorem 2.22. For CMS, we have $e^\ddagger_A = f^\ddagger_A$ because $e^\ddagger_A$ is a fixed point of an operation whose only fixed point is $f^\ddagger_A$. For CPO, we only have $f^\ddagger_A \leq e^\ddagger_A$. But then interchanging e and f shows the converse inequality $e^\ddagger_A \leq f^\ddagger_A$. □

Example 5.4. We return to an example from the introduction. Let $\mathcal{A}$ and H be as in Example 2.23, and let V be a functor on CMS corresponding to a single unary symbol. Let e and f correspond to φ and ψ, respectively:

$$\begin{array}{lcl} \varphi(s) & \approx & F(F(s,\varphi(s)),F(\varphi(s),s)) \\ \psi(s) & \approx & E(F(s,\psi(s)),F(s,\psi(s))) \end{array}$$

In our algebra $A = C(I)$, we have $F(F(s,t),F(t,s)) = E(F(s,t),F(s,t))$ for all s and t. This translates to an assertion which implies the hypothesis $\widetilde{[a,b]} \cdot e_A = \widetilde{[a,b]} \cdot f_A$ for all $b : VA \to A$. We thus conclude that $e^\ddagger_A = f^\ddagger_A$.

5.2 The interpreted fixed point identity

For every guarded program scheme $e : V \to T^{H+V}$ the standard interpreted solution satisfies Equation (19) repeated below:

$$e^\ddagger_A = \widetilde{[a, e^\ddagger_A]} \cdot e_A.$$

This interpreted fixed point identity was established in [17, 18] (in fact, see Equation (7.4) in [18]). It also follows easily from Equation (18):

$$\begin{array}{rcll} e^\ddagger_A & = & \widetilde{a} \cdot {e_A}^\dagger & \text{by (17)} \\ & = & \widetilde{a} \cdot \overline{[\kappa^H, e^\dagger]}_A \cdot e_A & \text{see Section 4.1} \\ & = & \widetilde{[a, e^\ddagger_A]} \cdot e_A & \text{by (18)} \end{array}$$

5.3 The interpreted functoriality law

Proposition 5.5. *Suppose that* $e : V \to T^{H+V}$ *and* $f : W \to T^{H+W}$ *are guarded rps's, and let* $n : V \to W$ *be a natural transformation such that the square*

$$\begin{tikzcd} V \arrow[r, "e"] \arrow[d, "n"'] & T^{H+V} \arrow[d, "T^{H+n}"] \\ W \arrow[r, "f"'] & T^{H+W} \end{tikzcd}$$

commutes. Let $(A, a, (\_)^*)$ *be an Elgot* H*-algebra. Then* $e^\ddagger_A = f^\ddagger_A \cdot n_A$.

Proof. As always let $\widetilde{a}$ be the associated Eilenberg-Moore algebra structure for $(A, a, (\_)^*)$. Then the desired result follows at once from the (uninterpreted) functoriality law by virtue of the following computation:

$$\begin{array}{rcll} e^\ddagger_A & = & \widetilde{a} \cdot e^\dagger_A & \text{by (17)} \\ & = & \widetilde{a} \cdot f^\dagger_A \cdot n_A & \text{by Proposition 4.1} \\ & = & f^\ddagger_A \cdot n_A & \text{by (17)} \end{array}$$

□

As the next example shows, the functoriality law may be used to show that interpreted solutions behave in the expected way with respect to renaming recursively defined function symbols in recursion schemes, and with respect to permutations of argument variables.

Example 5.6. Let $\mathcal{A}$ and H be as in Example 2.23, and let V and W be the endofunctors on CMS obtained from binary function symbols φ and ψ, respectively. Let $e : V \to T^{H+V}$ and $f : W \to T^{H+W}$ be guarded rps's expressing one of the following recursions each:

$$\begin{array}{lcl} \varphi(x,y) & \approx & F(\varphi(x,y), \varphi(y, \varphi(x,y))) \\ \psi(x,y) & \approx & F(\psi(y,x), \psi(\psi(y,x), y)) \end{array}$$

Let $n : V \to W$ be given by $\varphi \mapsto (\psi, 1, 0)$. That is, for all sets X and all $x, y \in X$, $n_X(\varphi, x, y) = (\psi, y, x)$. Then $f \cdot n = T^{H+n} \cdot e$. It follows from Proposition 5.5 that $e^{\ddagger}_{A} = f^{\ddagger}_{A} \cdot n_A$. This has a clearer interpretation if we write $\varphi^{\dagger} : A^2 \to A$ for $x, y \mapsto e^{\ddagger}_{A}(\varphi, x, y)$, and similarly for $\psi^{\dagger}$. Then the relation of $\varphi^{\dagger}$ and $\psi^{\dagger}$ is that for all $s, t \in A$, $\varphi^{\dagger}(s,t) = \psi^{\dagger}(t,s)$.

5.4 The interpreted parameter identity

Proposition 5.7. *Let H, K and V be iteratable endofunctors of $\mathcal{A}$, and let $e : V \to T^{H+V}$ be a guarded rps. Let $n : H \to T^K$ be an ideal natural transformation. Let $A(K) = (A, b, (\_)^*)$ be a K-Elgot algebra, and let $A(H)$ be the n-related H-Elgot algebra structure according to Proposition 5.2. Let $n \bullet e$ be the guarded rps*

$$n \bullet e \equiv \; V \xrightarrow{e} T^{H+V} \xrightarrow{\bar{s}} T^{K+V} .$$

where

$$s \equiv H + V \xrightarrow{n + \kappa^V} T^K + T^V \xrightarrow{[T^{\mathsf{inl}}, T^{\mathsf{inr}}]} T^{K+V} ,$$

Then the standard solutions are related as follows:

$$(n \bullet e)^{\ddagger}_{A(K)} = e^{\ddagger}_{A(H)}$$

(Note that on the left we interpret $n \bullet e$ in the K-Elgot algebra $A(K)$, whereas on the right we interpret e in the H-Elgot algebra $A(H)$.)

Proof. Let $\widetilde{a}$ and $\widetilde{b}$ be the Eilenberg-Moore algebra structures corresponding to the Elgot algebras $A(H)$ and $A(K)$, respectively. We then argue as follows:

$$\begin{array}{rcll} (n \bullet e)^{\ddagger}_A & = & \widetilde{b} \cdot (n \bullet e)^{\dagger}_A & \text{by (17)} \\ & = & \widetilde{b} \cdot \overline{n}_A \cdot e^{\dagger}_A & \text{by Proposition 4.3} \\ & = & \widetilde{a} \cdot e^{\dagger}_A & \text{since } A(H) \text{ is } n\text{-related to } A(K) \\ & = & e^{\ddagger}_A & \text{by (17)} \end{array}$$

□

Example 5.8. Consider the interpretation of

$$r(x) \approx \mathsf{ifzero}(x, \mathsf{zero}, \mathsf{succ}(\mathsf{square}(r(\mathsf{pred}(x))))) \tag{39}$$

in $\mathbb{N}_\perp$. That is, let Σ be the signature with all symbols above, except r, and let H be the corresponding Set-endofunctor. Let V correspond to a new unary symbol r, and let K correspond to Σ, except that square is replaced by a binary symbol $\times$. Let $e : V \to T^{H+V}$ be the ideal natural transformation expressing (39). Consider an Elgot K-algebra, say $\mathbb{N}_\perp$ extended with the (strictly extended) multiplication function $a, b \mapsto a \cdot b$ as its interpretation of $\times$. Let $n : H \to T^K$ be the natural transformation corresponding to the substitution $\mathsf{square}(x) \mapsto x \times x$. The rps $n \bullet e$ corresponds to the equation

$$r(x) \approx \mathsf{ifzero}(x, \mathsf{zero}, \mathsf{succ}(r(\mathsf{pred}(x)) \times r(\mathsf{pred}(x)))) .$$

Then n induces an n-related Elgot H-algebra structure on $\mathbb{N}_\perp$; intuitively, one takes a subtree $\mathsf{square}(t)$ and replaces it with $t \times t$ “corecursively.” The point of the interpreted parameter identity is that e and $n \bullet e$ have the same interpreted solution in the two (different, but related) Elgot algebras.

5.5 The interpreted pairing identity

Let H, V and W be iteratable endofunctors of $\mathcal{A}$. Suppose that we have a guarded rps

$$[e, f] : V + W \to T^{H+V+W} ,$$

and an Elgot algebra $A(H) = (A, a, (\_)^*)$. To compute the standard interpreted solution of $[e, f]$ we first solve the guarded rps f and plug its solution into e to obtain the guarded rps

$$g \equiv V \xrightarrow{e} T^{H+V+W} \xrightarrow{\overline{[\kappa^{H+V}, f^\dagger]}} T^{H+V} , \tag{40}$$

see (32). The standard interpreted solution $g^{\ddagger}_{A(H)} : VA \to A$ gives an Elgot algebra $[a, g^{\ddagger}_{A(H)}] : (H+V)A \to A$, which we denote by $A(H+V)$. The interpreted pairing identity states that the equation

$$[e, f]^{\ddagger}_{A(H)} = [g^{\ddagger}_{A(H)}, f^{\ddagger}_{A(H+V)}] : (V+W)A \to A \tag{41}$$

holds.

Proposition 5.9. *For a guarded rps $[e, f]$ as above and an Elgot algebra A for H, the interpreted pairing identity (41) holds.*

Proof. Recall that g is indeed guarded, see Proposition 4.5. To establish (41) we first apply equation (18) to g and $A(H)$ to obtain the equation

$$\widetilde{[a, g^{\ddagger}_{A(H)}]} = \widetilde{a} \cdot \overline{[\kappa^H, g^{\dagger}]}_A \,. \tag{42}$$

Then the following equations hold (we write $A = A(H)$ and $A' = A(H+V)$):

$$\begin{array}{rcll} [e, f]^{\ddagger}_A & = & \widetilde{a} \cdot [e, f]^{\dagger}_A & \text{by (17)} \\ & = & \widetilde{a} \cdot [g^{\dagger}, \overline{[\kappa^H, g^{\dagger}]} \cdot f^{\dagger}]_A & \text{by Proposition 4.5} \\ & = & [\widetilde{a} \cdot g^{\dagger}_A, \widetilde{[a, g^{\ddagger}_A]} \cdot f^{\dagger}_A] & \text{by (42)} \\ & = & [g^{\ddagger}_A, f^{\ddagger}_{A'}] & \text{by (17)} \end{array}$$

This completes the proof. □

Remark 5.10. The interpreted pairing identity shows that we can solve interpreted recursive program schemes in a step-by-step fashion. The fact that in the formulation (41) above the guarded rps g is formed using the uninterpreted solution $f^{\dagger}$ may seem odd at first. However, we shall now illustrate with an example that this cannot be avoided in general when one wants to solve $[e, f]$ in two successive steps.

Example 5.11. We work with $\mathcal{A} = \mathsf{Set}$, and we consider the signature Σ with the constant zero, the unary symbols pred and succ, the binary symbol $*$ and the ternary symbol ifzero. Let H be the corresponding signature functor of Set, let $WX = X \times X$ be a functor expressing one binary operation symbol q, and let $V = \mathit{Id}$ express the unary symbol r. Finally, let $e : V \to T^{H+V+W}$ and $f : W \to T^{H+V+W}$ be the rps's expressing the following recursive equations, respectively:

$$\begin{array}{rcl} r(x) & \approx & \mathsf{ifzero}(x, \mathsf{succ}(\mathsf{zero}), q(r(\mathsf{pred}\, x) * x, x)) \\ q(x, y) & \approx & \mathsf{ifzero}(y, x, r(\mathsf{pred}\, y)) \,. \end{array} \tag{43}$$

As interpretation of the givens in Σ we consider as always the Elgot algebra $\mathbb{N}_\perp$. The interpreted pairing identity tells us how to solve the above system (43) in $\mathbb{N}_\perp$. Notice that it is impossible to first obtain an interpreted solution for q, and then use this to solve r because we do not know how to interpret r in $\mathbb{N}_\perp$. Similarly, we cannot first obtain the solution for r to use it to solve q. The interpreted pairing identity tells us to take first the uninterpreted solution for q obtained from $f^\dagger$. Since in the second equation in (43) there is no "recursive call" to q, this uninterpreted solution is simply

$$q^\dagger(x,y) = \mathsf{ifzero}(y, x, r(\mathsf{pred}\, y)) \, .$$

Now we form the guarded rps g of (40) by plugging in $q^\dagger(x,y)$ for q in e; and so g expresses the recursive equation

$$r(x) \approx \mathsf{ifzero}(x, \mathsf{succ}(\mathsf{zero}), \mathsf{ifzero}(x, r(\mathsf{pred}\, x) * x, r(\mathsf{pred}\, x))) \, .$$

It is not difficult to see by induction that the interpreted solution $g^\ddagger_{\mathbb{N}_\perp}$ gives for r the constant function on 1 (extended strictly, of course). Thus, using this interpretation of r we obtain the standard interpreted solution of f w.r.t. the Elgot algebra $[a, g^\ddagger_{\mathbb{N}_\perp}] : (H+V)\mathbb{N}_\perp \to \mathbb{N}_\perp$. It gives the following operation on $\mathbb{N}_\perp$:

$$q_{\mathbb{N}_\perp}(x,y) = \begin{cases} x & \text{if } y = 0 \\ \perp & \text{if } y = \perp \\ 1 & \text{else} \end{cases}$$

5.6 The interpreted Bekić-Scott law

Let $e : V \to T^{H+V}$ be a guarded rps and let $f : W \to T^{H+V+W}$ be an rps which is guarded by H, see Definition 4.7. Suppose that $A(H) = (A, a, (\_)^*)$ is an Elgot algebra for H. We wish to compute the standard interpreted solution of the guarded rps

$$[e', f] : V + W \to T^{H+V+W} \, ,$$

where

$$e' \equiv V \xrightarrow{\ e\ } T^{H+V} \xrightarrow{\ T^{\mathsf{inl}}\ } T^{H+V+W} \, .$$

In fact, one first solves e in A to obtain the standard interpreted solution $e^\ddagger_A : VA \to A$ which gives an Elgot algebra $[a, e^\ddagger_A] : (H+V)A \to A$. We denote this Elgot algebra by $A(H+V)$, and we use it as an interpretation to solve f. Thus, the interpreted Bekić-Scott law states that for the standard interpreted solutions the equation below holds:

$$[e', f]^\ddagger_{A(H)} = [e^\ddagger_{A(H)}, f^\ddagger_{A(H+V)}] : (V+W)A \to A \qquad (44)$$

Proposition 5.12. *Let e and f be rps's as in Section 4.5, and let $A(H) = (A, a, (\_)^*)$ be an Elgot algebra for H. Then the standard interpreted solution of the guarded rps $[e', f]$ obeys the interpreted Bekić-Scott law (44).*

Proof. As in the proof of Proposition 5.9 the equation (42) follows from equation (18). Therefore we can calculate as follows (writing $A = A(H)$ and $A' = A(H + V)$):

$$\begin{array}{rcll}
[e', f]^\ddagger_A & = & \widetilde{a} \cdot [e', f]^\dagger_A & \text{by (18)} \\
& = & \widetilde{a} \cdot [e^\dagger, \overline{[\utilde{\kappa^H, e^\dagger}]} \cdot f^\dagger]_A & \text{by Proposition 4.8} \\
& = & [\widetilde{a} \cdot e^\dagger_A, [a, e^\ddagger_A] \cdot f^\dagger_A] & \text{by (42)} \\
& = & [e^\ddagger_A, f^\ddagger_{A'}] & \text{by (18)}
\end{array}$$

This completes the proof. □

Example 5.13. Consider the interpretations in $\mathbb{N}_\perp$ of

$$\begin{array}{rcl}
q(x, y) & \approx & \mathsf{ifzero}(y, x, \mathsf{succ}(\mathsf{succ}(q(x, \mathsf{pred}(y))))) \,, \quad \text{and} \\
r(x) & \approx & \mathsf{ifzero}(x, q(x, x), q(r(\mathsf{pred}(x)) * x, x))
\end{array}$$

Let H be the Set-endofunctor corresponding to the symbols zero, succ, and pred, let V (and W) correspond to q (and r). Let $e : V \longrightarrow T^{H+V}$ and $f : W \longrightarrow T^{H+V+W}$ be the natural transformations expressing the two recursions above. To obtain $[e', f]^\ddagger_{\mathbb{N}_\perp}$, we can first determine $e^\ddagger_{\mathbb{N}_\perp}$. It is easy to see this interpreted solution corresponds to the function $q(x, y) = 2x + y$. Then we may go back to the r equation and recast it via the interpretation to

$$r(x) \approx \mathsf{ifzero}(x, 2x + x, (2 * r(\mathsf{pred}(x)) * x) + x) \,.$$

In general, the interpreted Bekić-Scott law as we have formulated it tells us that we can solve interpreted recursive program schemes in an Elgot algebra in a step-by-step fashion, interpreting at some step all function symbols whose right hand sides only contain the same symbols or symbols interpreted at some previous step. Notice that in order to be able to do the latter the different parts of a system solved in each step must not be mutually recursive. This is the difference from the interpreted pairing identity, where the different parts of the given system solved in each step may be mutually recursive, see Example 5.11.

6 Conclusion

The aims of this work were: (1) to show that the same general tools needed to prove the existence and uniqueness of uninterpreted solutions of recursive program schemes also are sufficient to prove the basic laws of these solutions; (2) to similarly show that the tools for studying interpreted solutions, especially for schemes over CMS or over CPO, also are sufficient to study interpreted solutions. We were especially interested in studying interpreted solutions. It turned out that main theorems in [17, 18, 19] provide most of what is needed in this paper. In addition (and as one would expect), we did need to formulate a few new results (Proposition 2.12, Corollaries 2.13 and 2.14, and Propositions 5.2 and 5.3). But the most important finding in this paper is that the classical results on recursive program scheme solutions generalize from the classical settings to the level of Elgot algebras for iteratable functors.

References

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

[2] J. Adámek, S. Milius and J. Velebil, Elgot Algebras, *Log. Methods Comput. Sci.*, Vol. 2 (5:4), 31 pp.

[3] J. Adámek, S. Milius and J. Velebil, Equational Properties of Iterative Monads, submitted.

[4] M. F. Barnsley, *Fractals Everywhere*, Academic Press 1988.

[5] S. L. Bloom and Z. Ésik, *Iteration Theories: The Equational Logic of Iterative Processes,* EATCS Monographs on Theoretical Computer Science, Springer Verlag, 1993.

[6] F. Borceux, *Handbook of Categorical Algebra 2: Categories and Structures*, Vol. 51 of Encyclopedia of Mathematics and its Applications, Cambridge University Press, 1994.

[7] R. M. Burstall and J. Darlington, A Transformation System for Developing Recursive Programs, *J. ACM*, 24:1 (1977), 44–67.

[8] B. Courcelle, Fundamental Properties of Infinite Trees, *Theoret. Comput. Sci.* 25 (1983), no. 2, 95–169.

[9] B. Courcelle, Recursive Applicative Program Schemes. In J. van Leeuwen, ed., *Handbook of Theoretical Computer Science*, Vol. B, 459–492, Elsevier, Amsterdam, 1990.

[10] I. Guessarian, *Algebraic Semantics*. Lecture Notes in Comput. Sci., Vol. 99, Springer Verlag, 1981.

[11] A. J. C. Hurkens, M. McArthur, Y. N. Moschovakis, L. S. Moss and G. Whitney, The Logic of Recursive Equations, *J. Symbolic Logic* 63 (1998), no. 2, 451–478.

[12] J. Lambek, A Fixpoint Theorem for Complete Categories, *Math. Z.* 103 (1968), 151–161.

[13] S. Mac Lane, *Categories for the Working Mathematician*, 2nd edition, Springer Verlag, 1998.

[14] J. G. Mersch, Equational Logic of Recursive Program Schemes, PhD thesis, Indiana University, Bloomington, 2004.

[15] J. G. Mersch, Equational Logic of Recursive Program Schemes, in J. Fiadeiro et al (eds.): *Algebra and Coalgebra in Computer Science: First International Conference (CALCO 2005)*, Proceedings, Lecture Notes in Comput. Sci., Vol. 3629, 278–292, Springer Verlag, 2005.

[16] S. Milius, Completely Iterative Algebras and Completely Iterative Monads, *Inform. and Comput.* 196 (2005), 1–41.

[17] S. Milius and L. S. Moss, The Category Theoretic Solution of Recursive Program Schemes, *Theoret. Comput. Sci.* 366 (2006), 3–59.

[18] S. Milius and L. S. Moss, The Category Theoretic Solution of Recursive Program Schemes, full version, available at the URL `http://www.stefan-milius.eu`.

[19] S. Milius and L. S. Moss, Corrigendum to [17], *Theoret. Comput. Sci.* 403 (2008), 409–415.

[20] L. S. Moss, Parametric Corecursion, *Theoret. Comput. Sci.* 260 (2001), no. 1–2, 139–163.

[21] L. S. Moss, Recursion and Corecursion Have the Same Equational Logic, *Theoret. Comput. Sci.* 294 (2003), no. 1–2, 233–267.

[22] Y. Moschovakis, The Logic of Functional Recursion. In M. L. Dalla Chiara et al (eds.) *Logic and scientific methods*, Synthese Lib., 259, Kluwer Acad. Publ., Dordrecht, 1997, 179–207.

[23] M. Nivat, On the Interpretation of Recursive Polyadic Program Schemes, *Symposia Mathematica XV* (1975), 255–281.

Stefan Milius
Institute of Theoretical Computer Science, Technical University, Braunschweig, Germany
`mail@stefan-milius.eu`

Lawrence S. Moss
Department of Mathematics, Indiana University, Bloomington, IN, USA 47405
`lsm@cs.indiana.edu`