Categorical Views on Computations on Trees (Extended Abstract)

Ichiro Hasuo[1], Bart Jacobs[1], and Tarmo Uustalu[2]

[1] Institute of Computing and Information Sciences, Radboud University Nijmegen, Postbus 9010, NL-6500 GL Nijmegen, The Netherlands
`http://www.cs.ru.nl/~{ichiro, bart}`
[2] Institute of Cybernetics at Tallinn University of Technology, Akadeemia tee 21, EE-12618 Tallinn, Estonia
`http://www.cs.ioc.ee/~tarmo`

Abstract. Computations on trees form a classical topic in computing. These computations can be described in terms of machines (typically called tree transducers), or in terms of functions. This paper focuses on three flavors of bottom-up computations, of increasing generality. It brings categorical clarity by identifying a category of tree transducers together with two different behavior functors. The first sends a tree transducer to a coKleisli or biKleisli map (describing the contribution of each local node in an input tree to the global transformation) and the second to a tree function (the global tree transformation). The first behavior functor has an adjoint realization functor, like in Goguen's early work on automata. Further categorical structure, in the form of Hughes's Arrows, appears in properly parameterized versions of these structures.

1 Introduction

Tree transformations are functions sending trees to trees. Such transformations are of broad interest in computing, notably in language processing, and are often studied in relation to certain types of realizing machines. They form a classical topic.

In this paper we aim at a systematic study of phenomena and constructions related to *bottom-up* tree transformations. We first sketch two motivating observations: these will later be given detailed accounts.

Behavior-realization adjunction It is a fundamental idea in computer science that we associate with a "computable" function a "machine" which realizes it. Those machines which realize tree transformations are often called *tree transducers* and have been extensively studied as a continuation of automata theory: see [10,11,2] and also more recently [1].

Here comes our first question. What do we mean by saying "a machine c realizes a transformation l"? Given a transformation l, is there a machine which realizes it? Is there a canonical choice among such realizers? We shall answer these questions, following the idea of Goguen's *behavior-realization adjunction* [3] for (a more elementary setting of) automata, see also [9].

Tree functions from local behaviors We start with relabeling bottom-up tree transformations that only change labels on each node of an input tree, like l on the left.

$$\begin{tikzcd}[column sep=tiny, row sep=tiny]
 & & a_4 \arrow[dl, dash] \arrow[dr, dash] & & & & & b_4 \arrow[dl, dash] \arrow[dr, dash] & & & & & a_4 \arrow[dl, dash] \arrow[dr, dash] & & & \\
 & a_2 \arrow[dl, dash] \arrow[dr, dash] & & a_3 & \overset{l}{\longmapsto} & & b_2 \arrow[dl, dash] \arrow[dr, dash] & & b_3 & \quad & & a_2 \arrow[dl, dash] \arrow[dr, dash] & & a_3 & \overset{k}{\longmapsto} & b_4 \\
a_0 & & a_1 & & & b_0 & & b_1 & & & a_0 & & a_1 & & &
\end{tikzcd} \qquad (1)$$

Now let us consider another function k which operates on the same input trees as l does but returns the root label of the output tree of l. That is, $k = \epsilon \circ l$ where ϵ extracts the root label. It may seem that k (which shall be called a *local behavior*) carries less information than l does—ϵ throws information away. But when l is relabeling bottom-up we can recover l from k.

Our main contribution is to give an account of some classes of tree transformations in terms of diagrams like this:

$$\begin{tikzcd}
\mathbf{TT} \arrow[dr, bend left, "TF"] \arrow[dd, bend left, "LBeh"] \arrow[dd, phantom, "\dashv"] & & \\
 & \mathbf{TF}_{\uparrow} \arrow[r, hook] & \mathbf{TF} \\
\mathbf{LBeh} \arrow[ur, bend right, "W"'] \arrow[uu, bend left, "Real"] & &
\end{tikzcd} \qquad (2)$$

Here, *TF* and *LBeh* are two *behavior functors* from the category of tree transducers ("machines") to tree functions and to local behaviors. For relabelings, the functor W is an isomorphism: this embodies the equivalence of the two behaviors *TF* and *LBeh* as hinted at above; for more general types of tree transformations, it will be epi. The category $\mathbf{TF}_{\uparrow}$ is included in $\mathbf{TF}$ of tree functions in general: we shall give a categorical characterization of being "bottom-up". The local behaviors are coKleisli maps of certain comonads, in one case biKleisli maps of a distributive law of a comonad over a monad, and agree with the idea of comonadic notions of computation as those that send "values-in-contexts" to "values" [13,12] (the latter reference deals with attribute grammars, another type of tree computations). The behavior-realization adjunction is presented as $Real \dashv LBeh$.

In each of the Sects. 2–4, we shall develop a situation like (2) for a specific class of tree transformations—and hence a corresponding class of tree transducers. Namely, *relabeling bottom-up tree transducers* in Sect. 2; *rebranching bottom-up tree transducers* in Sect. 3, and *bottom-up tree transducers* in full generality in Sect. 4. In Sect. 5 we generalize our categorical formulation in an orthogonal direction: we uncover further compositional structures using Hughes's Arrows [5], and thus a way to view tree transformations as "structured computations" in programming semantics.

2 Relabeling Bottom-Up Tree Transducers

In this section we will consider a class of tree transducers (TTs) that operate on well-founded trees of a fixed branching type F (a set functor), with labels at their nodes taken from a parameter set, or alphabet. These transducers take A-labeled trees to B-labeled trees for fixed alphabets A, B, but Section 5 will sketch a properly parameterized version. They work bottom-up by only changing the labels of an input tree and are thus shape-preserving.

For this class of TTs, we shall turn the informal diagram (2) from the introduction into the diagram below. It has: two behavior functors *LBeh* and *TF*; a functor W establishing equivalence of two kinds of behavior; and an adjunction $\mathit{Real} \dashv \mathit{LBeh}$.

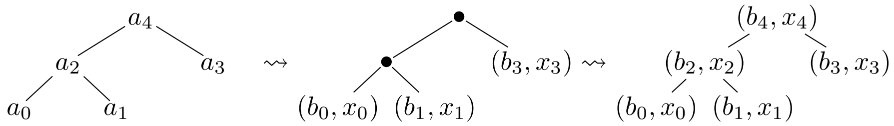
$$\begin{tikzcd} \mathbf{TT}(A,B) \arrow[dr, bend left, "TF"] \arrow[d, shift left, "LBeh"] \arrow[d, phantom, "\dashv"] & & \\ \mathbf{LBeh}(A,B) \arrow[u, shift left, "Real"] \arrow[r, bend right, "W"', "\cong"] & \mathbf{TF}_{\uparrow}(A,B) \arrow[r, hook] & \mathbf{TF}(A,B) \end{tikzcd} \tag{3}$$

That the branching type of our trees is expressed by a set functor F generalizes more traditional universal-algebraic signatures, given by a set Σ of operations f, each with an arity $|\mathsf{f}| \in \mathbb{N}$. Such a signature yields a functor $Z \mapsto \coprod_{\mathsf{f}\in\Sigma} Z^{|\mathsf{f}|}$. The A-labeled trees of the branching type F (for brevity, we also say A-trees) live in the initial algebra of the functor $A \times F\_$, whose carrier we denote by DA, leaving F implicit. The algebra structure $A \times FDA \xrightarrow{\cong} DA$ will be denoted by σ_A. Obviously $D1$ is the set of unlabelled trees or tree-shapes.

Definition 2.1 A (relabeling bottom-up) *tree transducer (TT)* is a function $A \times FX \xrightarrow{c} B \times X$ in **Sets**, where the set X is called the *state space*. A *morphism* of such TTs from $A \times FX \xrightarrow{c} B \times X$ to $A \times FY \xrightarrow{d} B \times Y$ is a function $f : X \to Y$ satisfying $(B \times f) \circ c = d \circ (A \times Ff)$.

TTs and morphisms between them form a category which we denote by $\mathbf{TT}(A, B)$, leaving again the dependence on F implicit. Obviously, $\mathbf{TT}(A, B)$ is nothing but the comma category $(A \times F\_ \downarrow B \times \_)$.

Example 2.2 The operation of a TT is best described on an example. As the branching type F we take $1 + (\_)^2$, describing well-founded binary trees. Consider a TT $A \times (1 + X^2) \xrightarrow{c} B \times X$ and the leftmost tree below as an input.



The bottom-up computation starts at the leaves: let $(a_0, \kappa_1(*)) \overset{c}{\mapsto} (b_0, x_0)$, where κ_1, κ_2 are coproduct injections. This assigns a label b_0 and a state x_0 to the corresponding leaf of the output tree. Similar mappings at the other leaves lead to the middle tree. At the inner position of a_2, the label on the output tree is determined by the input label a_2 as well as by the states x_0, x_1 of the successor nodes. They are already available precisely because we proceed in a bottom-up manner. Now we get (b_2, x_2) from the outcome $(a_2, \kappa_2(x_0, x_1)) \overset{c}{\mapsto} (b_2, x_2)$. We continue this way and get the tree on the right from $(a_4, \kappa_2(x_2, x_3)) \overset{c}{\mapsto} (b_4, x_4)$.

By forgetting about the states x_i, we finally obtain the output tree of the computation. It is obvious that the shape of the input tree is preserved. This will change in the next section.

For a TT c, we shall now define two behaviors $TF(c)$ and $LBeh(c)$. The former is a function that carries an A-tree to a B-tree; the latter carries an A-tree to an element in B, as hinted at in the introduction.

Definition 2.3 A TT $A \times FX \xrightarrow{c} B \times X$ induces its *tree function behavior* $TF(c) : DA \to DB$ and its *local behavior* $LBeh(c) : DA \to B$ via the following two diagrams, both using the initiality of σ_A.

$$
\begin{tikzcd}
A \times FDA \arrow[r, dashed] \arrow[d, "\sigma_A"', "\cong"] & A \times F(DB \times X) \arrow[d, "\check{c}"] \\
DA \arrow[r, dashed] \arrow[dr, bend right, "TF(c)"'] & DB \times X \arrow[d, "\pi_1"] \\
 & DB
\end{tikzcd}
\qquad
\begin{tikzcd}
A \times FDA \arrow[r, dashed] \arrow[dd, "\sigma_A"', "\cong"] & A \times F(B \times X) \arrow[d, "A \times F\pi_2"] \\
 & A \times FX \arrow[d, "c"] \\
DA \arrow[r, dashed] \arrow[dr, bend right, "LBeh(c)"'] & B \times X \arrow[d, "\pi_1"] \\
 & B
\end{tikzcd}
$$

where the algebra structure $\check{c}$ on the left is given by the composite

$$A \times \underline{F(DB \times X)} \xrightarrow{\langle F\pi_1, F\pi_2 \rangle} \underline{A} \times FDB \times \underline{FX} \xrightarrow{c} \underline{B \times FDB} \times X \xrightarrow{\sigma_B} DB \times X$$

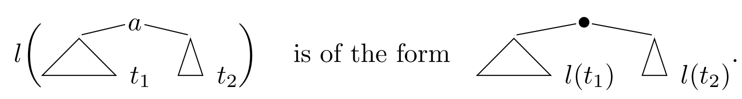
the underlining indicating what the maps act on.

The mapping $A \mapsto DA$ carries a comonad structure. It is the *cofree recursive comonad* on F [14]. A local behavior $LBeh(c) : DA \to B$ is a morphism $A \to B$ in the coKleisli category of the comonad D. This is a general phenomenon.

By a simple diagram chase it can be seen that a morphism of TTs is indeed a "behavior-preserving map" wrt. the above two behaviors.

Lemma 2.4 *Assume we have a morphism f from one TT c to another d. Then $LBeh(c) = LBeh(d)$ and $TF(c) = TF(d)$.* $\square$

In Example 2.2 we have illustrated how a TT acts in a bottom-up fashion on trees. Before we can show that the *TF* behavior from Def. 2.3 is indeed "bottom-up" we need a characterization of bottom-up tree functions. Intuitively, these are the functions $l : DA \to DB$ such that:



The following definition captures this intuition in categorical terms.

Definition 2.5 A tree function $l : DA \to DB$ is said to be (relabeling) *bottom-up* if it is a morphism of coalgebras, as in:

$$\begin{tikzcd} FDA \arrow[r, "Fl"] & FDB \\ A \times FDA \arrow[u, "\pi_2"] & B \times FDB \arrow[u, "\pi_2"] \\ DA \arrow[u, "\cong", "\sigma_A^{-1}"'] \arrow[r, "l"] & DB \arrow[u, "\cong", "\sigma_B^{-1}"'] \end{tikzcd} \tag{4}$$

Lemma 2.6 *For a TT $A \times FX \xrightarrow{c} B \times X$, the induced tree function $TF(c)$: $DA \to DB$ is bottom-up.* □

Now we can define the three semantic domains appearing in (3). We write:

- $\mathbf{LBeh}(A, B)$ for the set of maps $DA \to B$, i.e., $\mathbf{LBeh}(A, B) = \mathrm{Hom}_{\mathbb{C}}(DA, B)$;
- $\mathbf{TF}(A, B)$ for the set of maps $DA \to DB$, i.e., $\mathbf{TF}(A, B) = \mathrm{Hom}_{\mathbb{C}}(DA, DB)$;
- $\mathbf{TF}_{\uparrow}(A, B) \hookrightarrow \mathbf{TF}(A, B)$ for the subset of bottom-up maps $DA \to DB$.

These three sets are considered as discrete categories. This enables us to consider behavior mappings as functors from $\mathbf{TT}(A, B)$, in a degenerate fashion.

Lemma 2.7 *The mappings LBeh and TF in Def. 2.3 extend to functors*

$$LBeh : \mathbf{TT}(A, B) \to \mathbf{LBeh}(A, B) \quad \textit{and} \quad TF : \mathbf{TT}(A, B) \to \mathbf{TF}(A, B) \ .$$

The functor TF factors through the embedding $\mathbf{TF}_{\uparrow}(A, B) \hookrightarrow \mathbf{TF}(A, B)$. □

A realization functor *Real* : $\mathbf{LBeh}(A, B) \to \mathbf{TT}(A, B)$ is defined to send a local behavior $k : DA \to B$ to the TT $\langle k, DA \rangle \circ \sigma_A : A \times FDA \to B \times DA$. This TT has a canonical state space, namely the set DA of all A-trees; in all but degenerate cases, this state space is infinite. In fact *Real* yields the initial realization and we get a behavior-realization adjunction in the spirit of [3].

Theorem 2.8 *We have Real $\dashv$ LBeh, and since the category $\mathbf{LBeh}(A, B)$ is discrete, this adjunction is actually a coreflection.*

Proof. The statement is equivalent to the following. For a given local behavior $DA \xrightarrow{k} B$, the realization *Real*(k) is the initial one among those which yield k as their *LBeh* behavior. Let $A \times FX \xrightarrow{c} B \times X$ be one of such TTs. The following fact is shown by diagram chasing.

> $DA \xrightarrow{f} X$ is a morphism of TTs from *Real*(k) to c if and only if $\langle k, f \rangle$ is an algebra homomorphism from the initial algebra σ_A to $c \circ (A \times F\pi_2)$: $A \times F(B \times X) \to B \times X$.

Initiality of σ_A yields existence and uniqueness of such f, hence the initiality of *Real*(k). □

Next we shall establish an isomorphism between the two (local and tree function) behaviors, which we already discussed in the introduction. By Lemma 2.7, Theorems 2.8 and 2.9 we have established the situation (3).

Theorem 2.9 *The following composite W of functors is an isomorphism.*

$$W = \Big(\mathbf{LBeh}(A,B) \xrightarrow{Real} \mathbf{TT}(A,B) \xrightarrow{TF} \mathbf{TF}_{\uparrow}(A,B)\Big)$$

Proof. The functor W sends a map $k : DA \to B$ to its coKleisli extension $Dk \circ \delta_A \colon DA \to DDA \to DB$. Let $E : \mathbf{TF}_{\uparrow}(A,B) \to \mathbf{LBeh}(A,B)$ be the functor carrying a bottom-up tree function $l : DA \to DB$ to $\epsilon_B \circ l \colon DA \to DB \to B$. Thus E post-composes the tree function with extraction of the root label. Then $E \circ W = \mathrm{Id}$ because D is a comonad. For the opposite direction $W \circ E = \mathrm{Id}$, bottom-upness is crucial. □

3 Rebranching Bottom-Up Tree Transducers

In this section we pursue the same idea as in the previous section, but for a more general class of bottom-up TTs, namely *rebranching TTs*. They no longer preserve tree shapes, in fact they take trees of one branching type F to trees of a possibly different branching type G, by reorganizing the branching of any node of the input tree from type F to type G.

We shall establish the following situation, which is almost the same as (3). The main differences are: 1) the fixed parameters are now functors F, G for branching types (instead of sets A, B of labels) meaning that we consider transformations of F-branching trees (F-trees for short) into G-trees; 2) the isomorphism between $\mathbf{LBeh}$ and $\mathbf{TF}_{\uparrow}$ is not present.

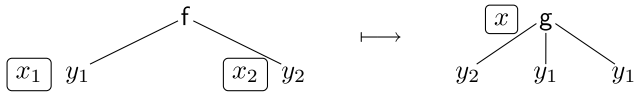
$$\begin{tikzcd}
\mathbf{TT}(F,G) \arrow[dr, "TF"] \arrow[dd, bend left, "LBeh"] \arrow[dd, phantom, "\dashv"] & & \\
& \mathbf{TF}_{\uparrow}(F,G) \arrow[r, hook] & \mathbf{TF}(F,G) \\
\mathbf{LBeh}(F,G) \arrow[uu, bend left, "Real"] \arrow[ur, two heads, "W"'] & &
\end{tikzcd} \tag{5}$$

A novelty in this section is what we call "placeholders-via-naturality". TTs are conventionally systems of transition rules in which placeholders appear explicitly. In our categorical approach, they have quite a different presentation as natural transformations (Def. 3.1). The correspondence between these seemingly different notions will be described via the Yoneda lemma.

Let us first present the conventional notion of rebranching TTs. Let Σ and Δ be universal-algebraic signatures: we consider transformations of Σ-trees into Δ-trees. Conventionally, a rebranching TT with a state space X is presented as an element of the set

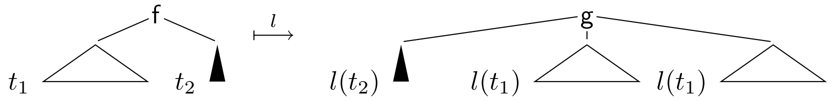
$$\textstyle\prod_{\mathsf{f}\in\Sigma}\Big(X^{|\mathsf{f}|} \;\longrightarrow\; (\coprod_{\mathsf{g}\in\Delta} |\mathsf{f}|^{|\mathsf{g}|}) \times X\Big) \;. \tag{6}$$

It is illustrative to think of the cardinality $|\mathsf{f}|$ as a set $\{y_1, \ldots, y_{|\mathsf{f}|}\}$ of placeholders, of the set $X^{|\mathsf{f}|}$ on the left as the set of graphs of functions from $|\mathsf{f}|$ to X and of the set $|\mathsf{f}|^{|\mathsf{g}|}$ on the right as the set of length-$|\mathsf{g}|$ lists over $|\mathsf{f}|$. For example, assume that some f is binary and a TT (6) carries $(\mathsf{f}, ((y_1, x_1), (y_2, x_2)))$ to $((\mathsf{g}, (y_2, y_1, y_1)), x)$ with a ternary g. This is understood graphically as follows.



(7)

This is "bottom-up" because the state x is determined by the states x_1, x_2 assigned to its successor nodes. Placeholders y_1, y_2 designate how the subtrees are reorganized in the bottom-up construction of a tree function behavior l.



The name *rebranching* comes from the fact that, on the right hand side of (7), exactly one function symbol occurs, so that a layer in a input tree is sent to exactly one layer of the output tree, and only the branching within the layer changes. In Sect. 4 we will abandon also this requirement.

We now present our categorical definition of TTs.

Definition 3.1 A (rebranching bottom-up) *TT* is a natural transformation $F(\_ \times X) \overset{\gamma}{\Longrightarrow} G\_ \times X$ between set functors. The set X is called its *state space*. A *morphism* of TTs from $F(\_ \times X) \overset{\gamma}{\Longrightarrow} G\_ \times X$ to $F(\_ \times Y) \overset{\delta}{\Longrightarrow} G\_ \times Y$ is a function $f : X \to Y$ satisfying $(G\_ \times f) \circ \gamma = \delta \circ F(\_ \times f)$. We denote by $\mathbf{TT}(F, G)$ the category of TTs and morphisms.

This categorical formulation may seem very different from the conventional one (6). But somewhat remarkably the two agree for functors arising from traditional signatures.

Let F, G be induced by universal-algebraic signatures Σ, Δ: namely, $F = \coprod_{\mathsf{f}\in\Sigma}(\_)^{|\mathsf{f}|}$ and $G = \coprod_{\mathsf{g}\in\Delta}(\_)^{|\mathsf{g}|}$. The following calculation shows the equivalence between (6) and Def. 3.1 via the Yoneda lemma.

$$
\begin{array}{ll}
\prod_{\mathsf{f}\in\Sigma}\bigl(X^{|\mathsf{f}|} \to (\coprod_{\mathsf{g}\in\Delta}|\mathsf{f}|^{|\mathsf{g}|}) \times X\bigr) & \\
= \prod_{\mathsf{f}\in\Sigma}\bigl(G|\mathsf{f}| \times X\bigr)^{X^{|\mathsf{f}|}} & \text{by definition of } G \\
= \prod_{\mathsf{f}\in\Sigma}\bigl((\_)^{|\mathsf{f}|} \Rightarrow (G\_ \times X)^{X^{|\mathsf{f}|}}\bigr) & \text{by Yoneda} \\
= \prod_{\mathsf{f}\in\Sigma}\bigl((\_ \times X)^{|\mathsf{f}|} \Rightarrow G\_ \times X\bigr) & \text{by } \_ \times X^{|\mathsf{f}|} \dashv (\_)^{X^{|\mathsf{f}|}} \\
= \bigl(\coprod_{\mathsf{f}\in\Sigma}(\_ \times X)^{|\mathsf{f}|}\bigr) \Rightarrow G\_ \times X & \\
= F(\_ \times X) \Rightarrow G\_ \times X & \text{by definition of } F \quad .
\end{array}
$$

On the third line the set of placeholders (the first occurrence of $|\mathsf{f}|$ on the second line) is absorbed into naturality, hence "placeholders-via-naturality".

We now proceed to the tree function behavior of our TTs. The tree functions here take F-trees to G-trees. Going slightly more general than necessary for this section (but preparing for the next), we write F^*Z for the carrier of the initial $(Z+F\_)$-algebra, i.e., the set of unlabelled F-trees with variables (graft-points) from a set Z. For the algebra structure $F(F^*Z) \xrightarrow{\cong} F^*Z$ we write α^F_Z. F-trees simpliciter (i.e., those without variables) arise as the special case F^*0. The set (or discrete category) of tree functions $F^*0 \to G^*0$ will be denoted by $\mathbf{TF}(F,G)$.

Definition 3.2 A TT $F(\_ \times X) \stackrel{\gamma}{\Rightarrow} G\_ \times X$ induces its *tree-function behavior* $TF(\gamma) \in \mathbf{TF}(F,G)$ by the following algebra initiality diagram.

$$\begin{tikzcd}
FF^*0 \arrow[r, dashed] \arrow[dd, "\alpha^F_0"', "\cong"] & F(G^*0 \times X) \arrow[d, "\gamma_{G^*0}"] \\
 & GG^*0 \times X \arrow[d, "\alpha^G_0 \times X", "\cong"'] \\
F^*0 \arrow[r, dashed, "\widetilde{\gamma}"] \arrow[dr, bend right, "TF(\gamma)"'] & G^*0 \times X \arrow[d, "\pi_1"] \\
 & G^*0
\end{tikzcd} \tag{8}$$

Here again, similarly to the situation for relabelings, not all the tree functions $F^*0 \to G^*0$ are induced by a TT but only "bottom-up" ones are.

Definition 3.3 A tree function $F^*0 \xrightarrow{l} G^*0$ is said to be *(rebranching) bottom-up*, if there exists a natural transformation called a *witness* $F(\_ \times F^*0) \stackrel{\omega}{\Longrightarrow} G\_$ which makes the following diagram commute.

$$\begin{tikzcd}
GF^*0 \arrow[rr, "Gl"] & & GG^*0 \\
F(F^*0 \times F^*0) \arrow[u, "\omega_{F^*0}"] & & \\
FF^*0 \arrow[u, "F\langle \mathrm{id}, \mathrm{id}\rangle"] & & \\
F^*0 \arrow[u, "(\alpha^F_0)^{-1}", "\cong"'] \arrow[rr, "l"'] & & G^*0 \arrow[uuu, "\cong", "(\alpha^G_0)^{-1}"']
\end{tikzcd} \tag{9}$$

By $\mathbf{TF}_{\uparrow}(F,G)$ we denote the set (discrete category) of tree functions $F^*0 \to G^*0$ which are rebranching bottom-up. We have $\mathbf{TF}_{\uparrow}(F,G) \hookrightarrow \mathbf{TF}(F,G)$.

Witnesses are not necessarily unique. A simple example is the tree function that sends an unlabelled binary tree to the unlabelled unary tree of its height.

Lemma 3.4 *For a TT $F(\_ \times X) \stackrel{\gamma}{\Rightarrow} G\_ \times X$, the induced tree function $TF(\gamma)$: $F^*0 \to G^*0$ is (rebranching) bottom-up.*

Proof. Take $\omega = \pi_1 \circ \gamma \circ F(\_ \times (\pi_2 \circ \widetilde{\gamma}))$, where $\widetilde{\gamma}$ is from (8). □

Definition 3.5 Given a TT $F(\_ \times X) \stackrel{\gamma}{\Rightarrow} G\_ \times X$, we define its *local behavior* $LBeh(\gamma)$ to be $F(\_ \times F^*0) \stackrel{\omega}{\Rightarrow} G\_$ from the proof of Lemma 3.4.

In Sect. 2 we observed that a local behavior $DA \to B$ is a coKleisli map. This is also the case in this section. In fact, the mapping $F \mapsto F(F^*0 \times \_)$ extends to a comonad on the functor category $[\mathbf{Sets}, \mathbf{Sets}]$, so that any natural transformation $F(F^*0 \times \_) \overset{\omega}{\Rightarrow} G\_$ is therefore a coKleisli map from F to G. We denote their set (discrete category) by $\mathbf{LBeh}(F, G)$.

Lemma 3.6 *The operations LBeh and TF in Definitions 3.5 and 3.2 extend to functors* $LBeh : \mathbf{TT}(F, G) \to \mathbf{LBeh}(F, G)$ *and* $TF : \mathbf{TT}(F, G) \to \mathbf{TF}_{\uparrow}(F, G)$. $\square$

Theorem 3.7 *We have an adjunction (actually a coreflection) Real $\dashv$ LBeh, where the realization functor for local behaviors* $Real : \mathbf{LBeh}(F, G) \to \mathbf{TT}(F, G)$ *sends a local behavior* $F(\_ \times F^*0) \overset{\omega}{\Rightarrow} G\_$ *to a TT with Z-components*

$$F(Z \times F^*0) \overset{\langle \omega_Z, F\pi_2 \rangle}{\to} GZ \times FF^*0 \overset{GZ \times \alpha_0^F}{\to} GZ \times F^*0 \ . \qquad \square$$

Proposition 3.8 *The functor* $W = (\, \mathbf{LBeh}(F, G) \overset{Real}{\to} \mathbf{TT}(F, G) \overset{TF}{\to} \mathbf{TF}_{\uparrow}(F, G)\,)$ *is an epimorphism.* $\square$

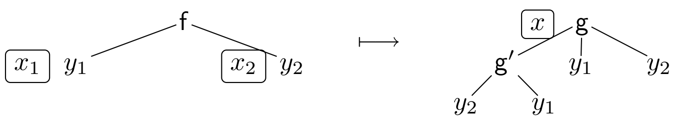
4 Relayering Bottom-Up Tree Transducers

In this section we will consider our most general class of bottom-up tree transformations, which can send a layer in an input tree to a truncated subtree in the output tree. For reasons of space, we must be fairly brief. We establish the same situation as in the previous section, except that we do not have to single out any condition of bottom-upness of tree functions. As we do not restrict state spaces to be finite, any tree function can arise as the behavior of a relayering bottom-up TT.

A categorical presentation of relayering TTs is obtained much like that of rebranching TTs in Sect. 3, using "placeholders-via-naturality". We recall the notation F^*Z for the carrier of the initial $(Z + F\_)$-algebra. It is now important for us that the functor F^* carries a monad structure, in particular a multiplication $\mu^F : F^*F^* \Rightarrow F^*$ that can be defined via initiality.

Definition 4.1 A (relayering bottom-up) TT is a natural transformation of the form $F(\_ \times X) \overset{\gamma}{\Longrightarrow} G^*\_ \times X$. Such TTs form a category $\mathbf{TT}(F, G)$ together with an obvious notion of morphism.

The difference from Def. 3.1 is that we have G^* instead of G in the codomain. This corresponds to allowing terms over placeholders rather than applications of single function symbols in the right-hand sides of transition rules (7): for example,



(10)

Definition 4.2 A TT $F(\_ \times X) \stackrel{\gamma}{\Rightarrow} G^*\_ \times X$ induces its *tree-function behavior* $TF(\gamma) : F^*0 \to G^*0$ by the following algebra initiality diagram.

$$\begin{tikzcd}
FF^*0 \arrow[r, dashed] \arrow[dd, "\alpha_0^F"', "\cong"] & F(G^*0 \times X) \arrow[d, "\gamma_{G^*0}"] \\
 & G^*G^*0 \times X \arrow[d, "\mu_0^G \times X"] \\
F^*0 \arrow[r, dashed, "\widetilde{\gamma}"] \arrow[dr, bend right, "TF(\gamma)"'] & G^*0 \times X \arrow[d, "\pi_1"] \\
 & G^*0
\end{tikzcd} \tag{11}$$

For relayering TTs any tree function is bottom-up: a tree function $l : F^*0 \to G^*0$ is realized by the TT whose Z-component is

$$F(Z \times F^*0) \xrightarrow{F\pi_2} FF^*0 \xrightarrow{\alpha_0^F} F^*0 \xrightarrow{\langle l, F^*0 \rangle} G^*0 \times F^*0 \xrightarrow{G^*! \times F^*0} G^*Z \times F^*0 \ ,$$

where ! denotes the empty map $0 \to Z$. This realization however does not give an adjunction.

The local behavior induced by a TT γ is a natural transformation $LBeh(\gamma) : F(\_ \times F^*0) \Rightarrow G^*\_$. Such natural transformations are biKleisli maps of a distributive law of the comonad $F \mapsto F(\_ \times F^*0)$ of the previous section over the free monad delivering monad $F \mapsto F^*$. We denote their set (discrete category) by $\mathbf{LBeh}(F, G)$.

For a realization functor for local behaviors $Real : \mathbf{LBeh}(F, G) \to \mathbf{TT}(F, G)$ we obtain an adjunction (actually a coreflection) $Real \dashv LBeh$, similarly to the rebranching case.

5 Allowing Parameters to Vary

In Sect. 2 we saw the fundamental diagram (3) relating tree transducers, local behaviors and tree functions. In that diagram we kept the alphabets A, B fixed. In this section we shall identify additional mathematical structure that emerges by allowing the alphabets to vary. For this purpose we utilize the notion of Arrows—as introduced by Hughes [5], but described more abstractly as monoids in categories of bifunctors in [4]—and also Freyd categories (or as fibered spans).

Arrows were devised for the purpose of reconciling impure "structured computations" with purely functional computation. Commonly an Arrow $\mathbf{A}(-, +)$ is a bifunctor $\mathbb{C}^{\mathrm{op}} \times \mathbb{C} \to \mathbf{Sets}$: in this case $\mathbf{A}(A, B)$ is the *set* of structured computations (of the kind designated by $\mathbf{A}$) from the type A to B. Since we want to consider $\mathbf{TT}(A, B)$ of relabeling transducers as a *category* of structured computation, we shall use **Cat**-*valued Arrows* instead: these are bifunctors $\mathbb{C}^{\mathrm{op}} \times \mathbb{C} \to \mathbf{Cat}$ with additional structure arr and $\ggg$.[3] The notion of **Cat**-valued Arrows are in fact the same thing as *Freyd categories* [8] (enriched by **Cat** in

[3] For the sake of brevity, we ignore here the compatibility with products which is usually given by an operation first.

a suitable way): this was shown in [7]. Moreover, a **Cat**-valued Arrow—as a bifunctor $\mathbb{C}^{\mathrm{op}}\times\mathbb{C}\to\mathbf{Cat}$—induces a *fibered span* via the generalized Grothendieck construction (see, e.g., [6, Ch. 9]).

In the remainder of the section we shall parameterize the diagram (3) and obtain the corresponding situation for Arrows. In this case we have $\mathbb{C}=\mathbf{Sets}$ as the base category. We do this only for relabelings due to limited space.

The bifunctor $\mathbf{TT}(-,+)$ is such that $\mathbf{TT}(A,B)$ is the category of relabelings from A-trees to B-trees. It sends a morphism $(\alpha,\beta):(A,B)\to(C,D)$ in $\mathbb{C}^{\mathrm{op}}\times\mathbb{C}$—hence $\alpha: C\to A$ and $\beta: B\to D$—to the functor $\mathbf{TT}(A,B)\to\mathbf{TT}(C,D)$ given as follows. On objects:

$$\big(A\times FX \xrightarrow{c} B\times X\big)\longmapsto\big(C\times FX \xrightarrow{\alpha\times FX} A\times FX \xrightarrow{c} B\times FX \xrightarrow{\beta\times FX} D\times FX\big)$$

and on morphisms it is the identity.

Interestingly, there is also a monoid structure $\mathbf{TT}\otimes\mathbf{TT}\overset{\ggg}{\to}\mathbf{TT}\overset{\mathsf{arr}}{\leftarrow} I$ on the bifunctor $\mathbf{TT}$—this makes $\mathbf{TT}$ an Arrow (see [4]). We shall describe it a bit more concretely. For TTs $A\times FX\xrightarrow{c} C\times X$ and $C\times FY\xrightarrow{d} B\times Y$ with matching output/input, their composition $c\ggg d$ has $X\times Y$ as its state space:

$$A\times\underline{F(X\times Y)}\xrightarrow{\langle F\pi_1,F\pi_2\rangle}\underline{A\times FX}\times FY\xrightarrow{c}\underline{C}\times X\times\underline{FY}\xrightarrow{d}B\times X\times Y\ .$$

The operation arr for $\mathbf{TT}$ carries a morphism $A\xrightarrow{f}B$ in $\mathbb{C}$ to a TT with a trivial state space 1: namely $A\times F1\xrightarrow{\pi_1}A\xrightarrow{f}B\xrightarrow{\cong}B\times 1$. It is easy to check that arr and $\ggg$ satisfy the appropriate naturality and monoid equations.

Just like $\mathbf{TT}(-,+)$ carries the structure of an Arrow we can identify similar structure on $\mathbf{LBeh}(-,+)$, $\mathbf{TF}(-,+)$ and $\mathbf{TF}_{\uparrow}(-,+)$. It then turns out that the diagram (3), but then without the fixed alphabets, also exists in parameterized form, even with preservation of this Arrow structure. For example, the behavior-realization adjunction is now described as an adjunction between Arrows.

Theorem 5.1 *We have the following situation in the 2-category* **Arrow***.*

$$\begin{tikzcd}
\mathbf{TT}(-,+) \arrow[dr, bend left, "TF"] \arrow[dd, bend left, "LBeh"] \arrow[dd, phantom, "\dashv"] & & \\
 & \mathbf{TF}_{\uparrow}(-,+) \arrow[r, hook] & \mathbf{TF}(-,+) \\
\mathbf{LBeh}(-,+) \arrow[ur, bend right, "W"'] \arrow[uu, bend left, "Real"] & &
\end{tikzcd} \tag{12}$$

□

6 Conclusions and Future Work

We have given a categorical account of three classes of bottom-up tree transformations. Notably, we have generalized traditional signatures to functors and replaced traditional descriptions of TTs based on placeholder notation with natural transformations, winning simplicity and clarity. In future work, we will elaborate on our basic picture in a form where, in addition to "extensional" tree functions, we also have "intensional" tree functions, capable of tracking which node in an input tree goes where in the output tree. And we will also include top-down computations, using the theory of containers, as well as bottom-up and top-down computations with look-ahead.

Acknowledgement T. Uustalu was partially supported by the Estonian Science Foundation grants No. 5567 and 6940.

References

1. Comon, H., Dauchet, M., Gilleron, R., Jacquemard, F., Lugiez, D., Tison, S., and Tommasi, M.: Tree automata techniques and applications. Book draft (2005)
2. Engelfriet, J.: Bottom-up and top-down tree transformations—a comparison. Math. Syst. Theory **9**(3) (1975) 198–231
3. Goguen, J.: Minimal realization of machines in closed categories. Bull. Amer. Math. Soc. **78**(5) (1972) 777–783
4. Heunen, C., Jacobs, B.: Arrows, like monads, are monoids. In Brookes, S., Mislove, M., eds.: Proc. of 22nd Conf. on Math. Found. of Program. Semantics, MFPS-XXII, Vol. 158 of Electr. Notes in Theor. Comput. Sci., Elsevier (2006) 219–236
5. Hughes, J.: Generalising monads to arrows. Sci. of Comput. Program. **37**(1–3) (2000) 67–111
6. Jacobs, B.: Categorical Logic and Type Theory. North Holland (1999)
7. Jacobs, B., Hasuo, I.: Freyd is Kleisli, for arrows. In McBride, C., Uustalu, T., eds.: Proc. of Wksh. on Mathematically Structured Functional Programming, MSFP '06, Electron. Wkshs. in Comput. Sci., BCS (2006)
8. Power, J., Robinson, E.: Premonoidal categories and notions of computation. Math. Struct. in Comput. Sci. **7**(5) (1997) 453–468
9. Rosebrugh, R. D., Sabadini, N., Walters, R. F. C.: Minimal realization in bicategories of automata. Math. Struct. in Comput. Sci. **8**(2) (1998) 93–116
10. Rounds, W. C.: Mappings and grammars on trees. Math. Syst. Theory **4**(3) (1970) 257–287
11. Thatcher, J. W.: Generalized sequential machine maps. J. Comput. Syst. Sci. **4**(4) (1970) 339–367
12. Uustalu, T., Vene, V.: Comonadic functional attribute evaluation. In van Eekelen, M., ed.: Trends in Functional Programming 6, Intellect (2007) 145–162
13. Uustalu, T., Vene, V.: The essence of dataflow programming. In Horváth, Z., ed.: Revised Selected Lectures from 1st Central European Functional Programming School, CEFP 2005, Vol. 4164 of Lect. Notes in Comput. Sci., Springer (2006) 135–167
14. Uustalu, T., Vene, V.: The dual of substitution is redecoration. In Hammond, K., Curtis, S., eds.: Trends in Functional Programming 3, Intellect (2002) 99–110