Available online at www.sciencedirect.com

SciVerse ScienceDirect

Electronic Notes in Theoretical Computer Science 276 (2011) 237–246

Electronic Notes in Theoretical Computer Science

www.elsevier.com/locate/entcs

Realization of Coinductive Types

Dexter Kozen[1,2]

Department of Computer Science
Cornell University
Ithaca, New York 14853–7501, USA

Abstract

We give an explicit combinatorial construction of final coalgebras for a modest generalization of polynomial functors on Set. Type signatures are modeled as directed multigraphs instead of endofunctors. The final coalgebra for a type signature F involves the notion of Brzozowski derivative on sets of paths in F.

Keywords: coalgebra, coinduction, recursive types, Brzozowski derivative

1 Introduction

Final F-coalgebras for endofunctors F on Set are useful in defining semantics of coinductive datatypes. The existence of final coalgebras under very general conditions has been studied in several papers [1,2,3,4,5,7,11,12,14,15,16]. These studies are mostly undertaken from an abstract categorical viewpoint, typically involving inverse limits, Cauchy completions, or bisimulation quotients of large coproducts. Aside from a few specific examples [7,11], general *concrete* constructions seem to be lacking. It is stated in [2] that "it is well-known that a final coalgebra. . . can be described as the coalgebra of all properly labelled ordered trees," but this informal statement is not completely accurate without further qualification; at any rate, its informality contrasts sharply with the formality of the ensuing abstract development. An accessible concrete construction would be of use to anyone interested in formal semantics and logics for reasoning about coinductive datatypes.

Of lesser concern, but still an issue, is that the traditional representation of type signatures as polynomial functors on Set is not always adequate the case of

[1] Thanks to Jean-Baptiste Jeannin, Bobby Kleinberg, Alexandra Silva, Navin Sivakumar, and the anonymous reviewers for insightful comments.

[2] Email: `kozen@cs.cornell.edu`

doi:10.1016/j.entcs.2011.09.024

mutually recursively defined types; and even when it is, it can introduce undesirable asymmetries.

Ordinary deterministic finite automata over an alphabet Σ form a family of coalgebras of a particularly simple form [11,13]. Final coalgebras of this type can be constructed explicitly in terms of the *Brzozowski derivative*

$$D_a(A) = \{x \mid ax \in A\} \tag{1}$$

for $A \subseteq \Sigma^*$ and $a \in \Sigma$.

In this paper we give an explicit Brzozowski-like construction of final coalgebras for type signatures corresponding to polynomial functors on Set^V, where V is a set of sorts. However, instead of functors, we represent type signatures as directed multigraphs with nodes designated as either *existential* or *universal.*

This representation has a number of advantages. Normally, polynomial functors on Set are built from product, coproduct, total and partial functions from a fixed set, constant functors, and compositions thereof; but all these can be modeled with existential and universal nodes. Many of the abstract constructions impose finiteness conditions to ensure continuity, but we require no such restrictions. Most importantly, the multigraph provides a platform for a definition of a Brzozowski derivative on sets of paths.

2 Brzozowski Derivatives

Before proceeding with the construction in §3, it is instructive to review the role of the Brzozowski derivative [6] in the construction of final coalgebras for ordinary deterministic finite automata. Classically, a deterministic finite automaton (DFA) over an alphabet Σ consists of a finite set of states S, a transition function $\delta : S \to \Sigma \to S$, a start state, and a set of accepting states $F \subseteq S$.

As observed in [11,13], ignoring the start state, a DFA is just a coalgebra for the polynomial endofunctor $(\Sigma \to -) \times \mathbb{2}$. In general, a coalgebra of this signature consists of a set of states S (not necessarily finite) and a structure map $\alpha : S \to (\Sigma \to S) \times \mathbb{2}$. The value $\alpha(s)$ is a pair in $(\Sigma \to S) \times \mathbb{2}$, of which the first component determines the transition function δ and the second determines whether $s \in F$.

Now associate with every state s the set of strings $L(s)$ that would be accepted by the automaton *were s the start state.* The map L satisfies the two properties:

(i) If $t = \delta(s)(a)$, then $L(t) = D_a(L(s))$, where $D_a(A)$ is the Brzozowski derivative of A with respect to $a \in \Sigma$ as defined in (1). That is, the string ax is accepted starting from the state s iff the string x is accepted starting from the state $\delta(s)(a)$.

(ii) The null string $\varepsilon \in L(s)$ iff s is an accept state.

Essentially, the subsets of Σ^*, along with the Brzozowski derivatives D_a and a function E determining whether $\varepsilon \in A$, form the final coalgebra for this signature, and L is the unique coalgebra homomorphism from the DFA to this final coalgebra. Formally, the transition function and accept states of the final coalgebra are given

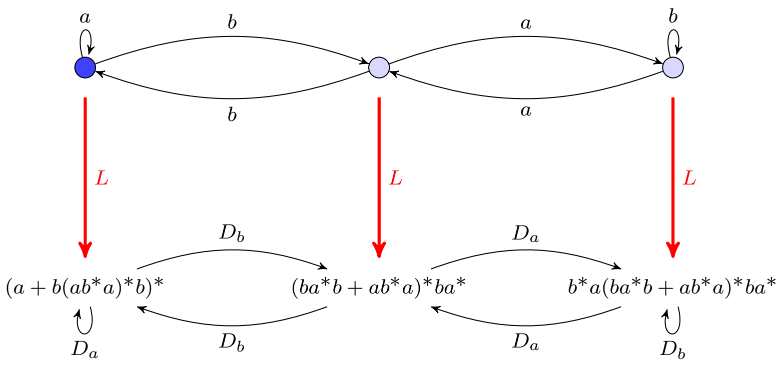

Fig. 1. An automaton and its image in the final coalgebra. Final states (shown a darker color) map to sets containing ε. Transitions in the automaton correspond to derivatives in the final coalgebra.

by

$$D(A)(a) = D_a(A) \qquad\qquad E(A) = \begin{cases} 1 & \text{if } \varepsilon \in A \\ 0 & \text{if } \varepsilon \notin A. \end{cases}$$

The relevant property that makes L a coalgebra homomorphism is that it commutes with the structure maps of the two coalgebras. This is the content of properties (i) and (ii) above. An example is illustrated in Fig. 1.

The Brzozowski derivatives D_a and the homomorphism E can be defined syntactically on regular expressions:

$$\begin{aligned} D_a(e_1 + e_2) &= D_a(e_1) + D_a(e_2) & E(e_1 + e_2) &= E(e_1) + E(e_2) \\ D_a(e_1e_2) &= D_a(e_1)e_2 + E(e_1)D_a(e_2) & E(e_1e_2) &= E(e_1)E(e_2) \\ D_a(e^*) &= D_a(e)e^* & E(e^*) &= 1 \\ D_a(b) &= \begin{cases} 1 & \text{if } a = b,\ a, b \in \Sigma \\ 0 & \text{if } a \neq b,\ a, b \in \Sigma \end{cases} & E(b) &= 0,\ b \in \Sigma \\ D_a(1) &= 0 & E(1) &= 1 \\ D_a(0) &= 0 & E(0) &= 0. \end{aligned}$$

This is a key ingredient of Kleene's theorem establishing the equivalence of finite automata and regular expressions [6]; see [13] for a thorough exposition.

3 Main Results

3.1 Directed Multigraphs

A *directed multigraph* is a structure $G = (V, E, \mathsf{src}, \mathsf{tgt})$ with nodes V, edges E, and two maps $\mathsf{src}, \mathsf{tgt} : E \to V$ giving the source and target of each edge, respectively. We write $e : s \to t$ if $s = \mathsf{src}\, e$ and $t = \mathsf{tgt}\, e$. When specifying multigraphs, we will sometimes use the notation $s \xrightarrow{n} t$ for the metastatement, "There are exactly n edges from s to t."

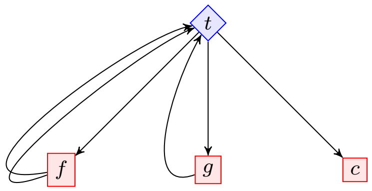

Fig. 2. A multigraph representing a single-sorted algebraic signature. Blue diamonds represent existential nodes and red squares universal nodes.

A *path* is a finite alternating sequence of nodes and edges

$$s_0\, e_0\, s_1\, e_1\, s_2\, \cdots\, s_{n-1}\, e_{n-1}\, s_n,$$

$n \geq 0$, such that $e_i : s_i \to s_{i+1}$, $0 \leq i \leq n-1$. These are the arrows of the free category generated by G. The *length* of a path is the number of edges. A path of length 0 is just a single node. The first and last nodes of a path p are denoted $\mathsf{src}\, p$ and $\mathsf{tgt}\, p$, respectively. As with edges, we write $p : s \to t$ if $s = \mathsf{src}\, p$ and $t = \mathsf{tgt}\, p$.

A *multigraph homomorphism* $\ell : G_1 \to G_2$ is a map $\ell : V_1 \to V_2$, $\ell : E_1 \to E_2$ such that if $e : s \to t$ then $\ell(e) : \ell(s) \to \ell(t)$. This lifts to a functor on the free categories generated by G_1 and G_2.

3.2 Type Signatures

A *type signature* is a directed multigraph F along with a designation of each node of F as either *existential* or *universal*. The existential and universal nodes correspond respectively to coproduct and product constructors. The directed edges of the graph represent the corresponding destructors.

For example, consider an algebraic signature consisting of a binary function symbol f, a unary function symbol g, and a constant c. This would ordinarily be represented by the polynomial endofunctor $F = -^2 + - + \mathbb{1}$, or in OCaml by

```
type t = F of t * t | G of t | C
```

We would represent this signature by a directed multigraph consisting of four nodes $\{t, f, g, c\}$, of which t is existential and f, g, c are universal, along with edges

$$t \xrightarrow{1} f \qquad t \xrightarrow{1} g \qquad t \xrightarrow{1} c \qquad f \xrightarrow{2} t \qquad g \xrightarrow{1} t.$$

The multigraph is illustrated in Fig. 2.

Here is a more involved example from [8]. In that paper, the state of a computation of a higher-order language with closures is defined in terms of a recursive type definition

$$\begin{aligned} \mathsf{Val} &= \mathsf{Const} + \mathsf{Cl} && \text{values} \\ \mathsf{Cl} &= \lambda\text{-}\mathsf{Abs} \times \mathsf{Env} && \text{closures} \\ \mathsf{Env} &= \mathsf{Var} \rightharpoonup \mathsf{Val} && \text{closure environments} \end{aligned}$$

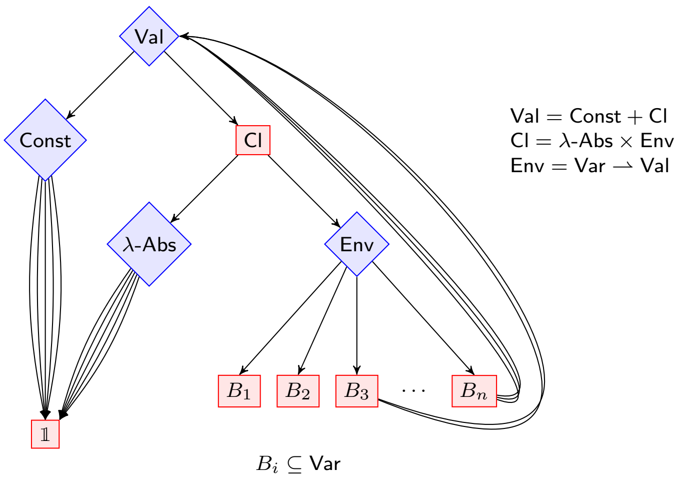

Fig. 3. A multigraph representing a multisorted signature.

where Const is a fixed set of constants, λ-Abs is a fixed set of λ-abstractions, and Var is a fixed set of variables (the exact nature of these sets is not important). The set of values is a solution to the recursive equation

$$\mathsf{Val} = \mathsf{Const} + (\lambda\text{-}\mathsf{Abs} \times (\mathsf{Var} \rightharpoonup \mathsf{Val})),$$

which would ordinarily be modeled by an endofunctor

$$F = \mathsf{Const} + (\lambda\text{-}\mathsf{Abs} \times (\mathsf{Var} \rightharpoonup -))$$

on Set. In OCaml, we might write

```
type value = Const of int | Closure of closure
and closure = labs * env
and env = var -> value
```

We model this type signature by a multigraph with existential nodes Val, Const, λ-Abs, and Env and universal nodes Cl, $\mathbb{1}$, and a node for each $B \subseteq \mathsf{Var}$. The edges are

$$\begin{array}{cc}
\mathsf{Val} \xrightarrow{1} \mathsf{Const} & \mathsf{Val} \xrightarrow{1} \mathsf{Cl} \\
\mathsf{Cl} \xrightarrow{1} \lambda\text{-}\mathsf{Abs} & \mathsf{Cl} \xrightarrow{1} \mathsf{Env} \\
\mathsf{Const} \xrightarrow{c} \mathbb{1},\ c = |\mathsf{Const}| & \lambda\text{-}\mathsf{Abs} \xrightarrow{d} \mathbb{1},\ d = |\lambda\text{-}\mathsf{Abs}| \\
\mathsf{Env} \xrightarrow{1} B,\ B \subseteq \mathsf{Var} & B \xrightarrow{b} \mathsf{Val},\ b = |B|
\end{array}$$

Note that we regard a partial function $\mathsf{Var} \rightharpoonup \mathsf{Val}$ on the fixed set Var as a dependent coproduct $\sum_{B \subseteq \mathsf{Var}} \mathsf{Val}^B$. This is modeled by an existential node to select the domain $B \subseteq \mathsf{Var}$, followed by a universal node to select the value of the function Val^B on that domain. The multigraph is illustrated in Fig. 3.

3.3 Coalgebras and Realizations

Let F be a type signature with nodes V_F. An *F-coalgebra* is a V_F-indexed collection of pairs $(A_s,\ \alpha_s)$, where the A_s are sets and the α_s are set functions

$$\alpha_s : A_s \to \begin{cases} \sum_{\mathsf{src}\, e=s} A_{\mathsf{tgt}\, e}, & \text{if } s \text{ is existential,} \\ \prod_{\mathsf{src}\, e=s} A_{\mathsf{tgt}\, e}, & \text{if } s \text{ is universal.} \end{cases}$$

A morphism of F-coalgebras is a V_F-indexed collection of set maps h_s that commute with the α_s in the usual way. This corresponds to the traditional definition of an F-coalgebra for an endofunctor F on Set^V.

Coalgebras are equivalent to *realizations.* An *F-realization* is a directed multigraph G along with a multigraph homomorphism $\ell : G \to F$, called a *typing*, with the following properties.

- If $\ell(u)$ is existential, then there is exactly one edge of G with source u.
- If $\ell(u)$ is universal, then ℓ is a bijection between the edges of G with source u and the edges of F with source $\ell(u)$.

A homomorphism of F-realizations is a multigraph homomorphism that commutes with the typings.

Theorem 3.1 *The categories of F-coalgebras and F-realizations are equivalent* (*in the sense of* [10, §IV.4]).

Proof. We must exhibit a pair of functors between F-coalgebras and F-realizations, one in each direction, and show that they are inverses up to natural isomorphisms.

We first construct a coalgebra from a given realization G with nodes V_G and typing $\ell : G \to F$. For each node s of F, let $A_s = \{u \in V_G \mid \ell(u) = s\}$ and define α_s as follows:

- If s is existential and $u \in A_s$, let $d : u \to v$ be the unique edge in G with $\mathsf{src}\, d = u$, let $e = \ell(d)$, and let $t = \ell(v) = \mathsf{tgt}\, e$. Define $\alpha_s(u) = \mathsf{in}_e(v)$, where $\mathsf{in}_e : A_t \to \sum_{\mathsf{src}\, e=s} A_{\mathsf{tgt}\, e}$ is the natural injection into the coproduct.
- If s is universal and $u \in A_s$, let $\alpha_s(u) \in \prod_{\mathsf{src}\, e=s} A_{\mathsf{tgt}\, e}$ be the unique element of the product such that for any edge $d : u \to v$ with $\mathsf{src}\, d = u$, if $e = \ell(d)$ and $t = \ell(v) = \mathsf{tgt}\, e$, then $\pi_e(\alpha_s(u)) = v$, where $\pi_e : \prod_{\mathsf{src}\, e=s} A_{\mathsf{tgt}\, e} \to A_t$ is the natural projection from the product.

Conversely, given an F-coalgebra with data $(A_s,\ \alpha_s)$, we can construct a realization. The nodes of the realization are elements of the coproduct

$$\sum_{s \in V_F} A_s = \bigcup_{s \in V_F} \{\mathsf{in}_s(u) \mid u \in A_s\} \tag{2}$$

with $\ell(\mathsf{in}_s(u)) = s$ for $u \in A_s$. If $u \in A_s$ and s is existential, then $\alpha_s(u) = \mathsf{in}_e(v) \in \sum_{\mathsf{src}\, e=s} A_{\mathsf{tgt}\, e}$ for some $e : s \to t$ and $v \in A_t$. Add an edge $\langle u, e\rangle$ to the realization with $\langle u, e\rangle : \mathsf{in}_s(u) \to \mathsf{in}_t(v)$ and $\ell(\langle u, e\rangle) = e$. If $u \in A_s$ and s is universal, then

$\alpha_s(u) \in \prod_{\mathsf{src}\,e=s} A_{\mathsf{tgt}\,e}$. For each $e : s \to t$, let $v_e = \pi_e(\alpha_s(u)) \in A_t$. Add an edge $\langle u, e\rangle$ to the realization with $\langle u, e\rangle : \mathsf{in}_s(u) \to \mathsf{in}_t(v_e)$ and $\ell(\langle u, e\rangle) = e$.

In this construction, we may take the edge $\langle u, e\rangle$ to be the ordered pair (u, e). Because u and e determine v, s, and t uniquely, each such ordered pair appears at most once as an edge $\langle u, e\rangle$, so there is no danger of duplication.

To finish the proof, we must verify that these two constructions are inverses up to natural isomorphisms. Given a realization G with typing $\ell : G \to F$, applying the first construction followed by the second yields a realization naturally isomorphic to G via an isomorphism that maps the node u to $\mathsf{in}_{\ell(u)}(u)$ and the edge d to $\langle \mathsf{src}\,d, \ell(d)\rangle$. It is easily checked that the maps are bijections on nodes and edges, preserve adjacency, and commute with the typing maps.

Similarly, given a coalgebra with data (A_s, α_s), performing the second construction followed by the first yields a coalgebra with data

$$A'_s = \{\mathsf{in}_s(u) \mid u \in A_s\}$$

$$\alpha'_s : A'_s \to \begin{cases} \sum_{\mathsf{src}\,e=s} A'_{\mathsf{tgt}\,e} & \text{if } s \text{ is existential,} \\ \prod_{\mathsf{src}\,e=s} A'_{\mathsf{tgt}\,e} & \text{if } s \text{ is universal.} \end{cases}$$

If s is existential, then $\alpha'_s(\mathsf{in}_s(u)) = \mathsf{in}_e(\mathsf{in}_t(v))$, where $\alpha_s(u) = \mathsf{in}_e(v)$ and $t = \mathsf{tgt}\,e$. If s is universal, then $\pi_e(\alpha'_s(\mathsf{in}_s(u))) = \mathsf{in}_t(v)$, where $\pi_e(\alpha_s(u)) = v$ and $t = \mathsf{tgt}\,e$. It is routine to verify that the two coalgebras are naturally isomorphic via the isomorphism with components $\mathsf{in}_s : A_s \to A'_s$.

Note that the A'_s are always pairwise disjoint, whereas the A_s may not be. However, this does not preclude isomorphism, because a morphism in the category of F-coalgebras consists of a collection of set maps indexed by nodes of F, which may take different values on the same element. □

3.4 Final Coalgebras

Realizations allow us to give a concrete construction of final coalgebras that is reminiscent of the Brzozowski derivative on sets of strings (1). Here, instead of strings, the derivative acts on certain sets of paths of the type signature.

Let F be a type signature. Construct a realization R_F, ℓ_F as follows. A node of R_F is a set A of finite paths in F such that

(i) A is nonempty and prefix-closed;

(ii) all paths in A have the same first node, which we define to be $\ell_F(A)$;

(iii) if p is a path in A of length n and $\mathsf{tgt}\,p$ is existential, then there is exactly one path of length $n + 1$ in A extending p;

(iv) if p is a path in A of length n and $\mathsf{tgt}\,p$ is universal, then all paths of length $n + 1$ extending p are in A.

The edges of R_F are defined as follows. Let A be a set of paths in F and e an edge

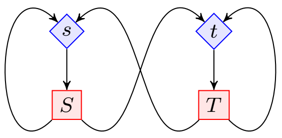

Fig. 4. A multisorted signature.

of F. Define the *Brzozowski derivative* of A with respect to e to be

$$D_e(A) = \{p \mid (\mathsf{src}\, e)\, e\, p \in A\},$$

the set of paths obtained by removing the initial edge e from paths in A that start with that edge. If A is a node of R_F and $D_e(A)$ is nonempty, we include exactly one edge

$$\langle A, e\rangle : A \to D_e(A)$$

in R_F and take $\ell_F(\langle A, e\rangle) = e$. It is readily verified that $\mathsf{tgt}\,\langle A, e\rangle = D_e(A)$ satisfies properties (i)–(iv) and that $\ell_F(D_e(A)) = \mathsf{tgt}\, e$, so ℓ_F is a typing.

Theorem 3.2 *The realization R_F, ℓ_F is final in the category of F-realizations. The corresponding F-coalgebra as constructed in Theorem 3.1 is final in the category of F-coalgebras.*

Proof. Let G, ℓ be an arbitrary realization. The unique homomorphism $h : G, \ell \to R_F, \ell_F$ is given by: $h(s)$ is the set of paths in F that are images under ℓ of paths in G starting with node s.

The second statement of the theorem follows from the equivalence of the two categories (Theorem 3.1). □

4 Discussion

4.1 Multisorted Signatures and Asymmetry

In the introduction, we mentioned that polynomial endofunctors on Set do not appear to be adequate for modeling some coinductive types that deserve to be regarded as polynomial types. We also mentioned that they can introduce undesirable asymmetries. For example, it is not clear how to model the mutually dependent coinductive types

```
type s = C of s * t
and t = D of s * t
```

using an endofunctor on Set; it appears that Set^2 is required. The multigraph corresponding to this signature is illustrated in Fig. 4.

Endofunctors on Set are adequate in the single-sorted case. They are also adequate in the multisorted example

$$\mathsf{Val} = \mathsf{Const} + \mathsf{Cl} \qquad \mathsf{Cl} = \lambda\text{-}\mathsf{Abs} \times \mathsf{Env} \qquad \mathsf{Env} = \mathsf{Var} \rightharpoonup \mathsf{Val}$$

of §3.2 because there is a node that meets all cycles; in fact, there are three such nodes. However, we must still choose where to break the cycle, and this is the undesirable asymmetry. In this case, we could choose any of the three options

$$\begin{aligned} F_{\mathsf{Val}} &= \mathsf{Const} + (\lambda\text{-}\mathsf{Abs} \times (\mathsf{Var} \rightharpoonup -)) \\ F_{\mathsf{Cl}} &= \lambda\text{-}\mathsf{Abs} \times (\mathsf{Var} \rightharpoonup (\mathsf{Const} + -)) \\ F_{\mathsf{Env}} &= \mathsf{Var} \rightharpoonup (\mathsf{Const} + (\lambda\text{-}\mathsf{Abs} \times -)), \end{aligned}$$

but then we would be left with the task of proving that the choice does not matter.

We conjecture that endofunctors on Set are adequate exactly when there exists a set of nodes A of the type signature such that every cycle contains exactly one node of A.

4.2 Final Coalgebras as Labeled Trees

In this section, we wish to expand on the statement of Adámek [2] that "a final coalgebra. . . can be described as the coalgebra of all properly labelled ordered trees" and draw a relationship to the Brzozowski construction of §3.4. In that paper and [3], one finds an explicit tree-like construction for a single-sorted polynomial signature such as the one illustrated in Fig. 2. Worrell [16] gives a construction for unordered trees.

A subtlety arises when one tries to define *labeled trees* formally. The issue is how to define the nodes and edges so that one obtains unique representatives in the final coalgebra. For traditional algebraic signatures involving n-ary functions $f : A^n \to A$, one can define the nodes of the tree as a prefix-closed, nonempty subset of ω^* such that if α is a node, then αi is a node for all $0 \le i < n$, where n is the arity of the node's label. This construction appears for example in [3,9]. However, it is not immediately clear what to do for unordered trees or more general type signatures. In [16], it is stated that "We consider trees that are isomorphic as directed graphs. . . to be identical," thus trees are isomorphism classes. But of what?

Thinking about this issue leads naturally to idea of type signatures as directed multigraphs F. This allows us to construct labeled trees whose nodes are paths in F. Instead of natural numbers, the children of a node are indexed by the edges of F.

To characterize the elements of the final coalgebra as labeled trees, we can start from the final realization R_F, ℓ_F constructed in §3.4. Each node A of R_F corresponds to a labeled tree $\tau(A)$ as follows. The root of $\tau(A)$ is $\ell_F(A)$. The nodes of $\tau(A)$ are the elements of A, which are paths in F. There is an edge in $\tau(A)$ from p to q if p is a prefix of q and their lengths differ by one. The labeling function labels a path p with its final node $\mathsf{tgt}\, p$. In this construction, $\tau(D_e(A))$ is the eth maximal proper subtree of $\tau(A)$.

References

[1] Aczel, P. and P. Mendler, *A final coalgebra theorem*, in: *Category Theory and Computer Science*, Lecture Notes in Computer Science **389**, Springer, 1989 pp. 357–365.

[2] Adámek, J., *On final coalgebras of continuous functors*, Theor. Comput. Sci. **294** (2003), pp. 3–29.

[3] Adámek, J. and V. Koubek, *On the greatest fixed point of a set functor*, Theor. Comput. Sci. **150** (1995), pp. 57–75.

[4] America, P. and J. Rutten, *Solving reflexive domain equations in a category of complete metric spaces*, J. Comput. Syst. Sci. **39** (1989), pp. 343–375.

[5] Barr, M., *Terminal coalgebras in well-founded set theory*, Theor. Comput. Sci. **114** (1993), pp. 299–315.

[6] Brzozowski, J. A., *Derivatives of regular expressions*, J. Assoc. Comput. Mach. **11** (1964), pp. 481–494.

[7] Hausmann, D., "Data Types and Computability via Final Coalgebras," Ph.D. thesis, Dresden University (2004).

[8] Jeannin, J.-B. and D. Kozen, *Computing with capsules*, Technical Report `http://hdl.handle.net/1813/22082`, Computing and Information Science, Cornell University (2011).

[9] Kozen, D., J. Palsberg and M. I. Schwartzbach, *Efficient recursive subtyping*, Mathematical Structures in Computer Science **5** (1995), pp. 113–125.

[10] MacLane, S., "Categories for the Working Mathematician," Springer-Verlag, New York, 1971.

[11] Rutten, J., *Universal coalgebra: a theory of systems*, Theor. Comput. Sci. **249** (2000), pp. 3–80.

[12] Santocanale, L., *Logical construction of final coalgebras*, in: H. P. Gumm, editor, *Proc. Workshop on Coalgebraic Methods in Computer Science, Warsaw, Poland*, Electronic Notes in Theoretical Computer Science **82** (2003), pp. 1–20.

[13] Silva, A., "Kleene Coalgebra," Ph.D. thesis, University of Nijmegen (2010).

[14] Smyth, M. B. and G. D. Plotkin, *The category-theoretic solution of recursive domain equations*, SIAM J. Comput. **11** (1982), pp. 761–783.

[15] Worrell, J., *Coinduction for recursive data types: partial orders, metric spaces and Ω-categories.*, Electr. Notes in Theor. Comput. Sci. **33** (2000), pp. 337–356.

[16] Worrell, J., *On the final sequence of a finitary set functor*, Theor. Comput. Sci. **338** (2005), pp. 184–199.