SEMANTICS OF GRAMMARS AND ATTRIBUTES VIA INITIALITY

BART JACOBS AND TARMO UUSTALU

Institute for Computing and Information Sciences, Radboud University Nijmegen, P.O. Box 9010, 6500 GL Nijmegen, The Netherlands.
e-mail address: B.Jacobs@cs.ru.nl

Institute of Cybernetics, Tallinn University of Technology, Akadeemia tee 21, EE-12618 Tallinn, Estonia
e-mail address: tarmo@cs.ioc.ee

Dedicated to Henk Barendregt on the occasion of his 60th birthday

ABSTRACT. This paper uses elementary categorical techniques to systematically describe the semantics of context-free grammars and of attribute evaluation for such grammars. The novelty lies in capturing inherited attributes and their evaluation via exponents and naturality.

1. INTRODUCTION

Context free grammars form a fundamental topic in computer science, as the basis for compiler construction and language processing. The meaning of parse trees of such grammars is usually captured via attributes and semantic equations. They define the attribute values at any node of any given parse tree.

Knuth [7] is a classical paper on such semantics of context-free languages. A later paper [1], speaking of "Knuthian semantics" rephrases the material in terms of multi-sorted algebras and uses initiality for interpretation, following Goguen *et al.* [2, 3]. A complicating factor in this setting is that attributes come in two flavours, namely 'synthesised' (bottom-up) and 'inherited' (top-down), which may lead to circular dependencies. Much research has been devoted to avoiding such circularities via syntactic criteria. The problem is side-stepped in [1] by working in a domain-theoretic setting [3] in which the necessary fixed points always exist.

Here we 'modernise' the multi-sorted algebra approach of [1] by generalising it to the categorical theory of algebras of functors (see [6] for an introduction). This allows us to:

(1) see an attribute grammar built on a context-free grammar simply as an algebra of the functor associated with the context-free grammar;
(2) describe attribute evaluation systematically as a special form of tree relabeling.

Most of this 'modernisation' is straightforward. Nevertheless we spell it out in detail in order to make it accessible to readers who are less familiar with categorical techniques. The main (novel) contribution of the paper comes at the end, where a combination of exponents and naturality is used to capture inherited attributes. This is our way of side-stepping syntactic criteria.

REFLECTIONS ON TYPE THEORY, λ-CALCULUS, AND THE MIND
Essays Dedicated to Henk Barendregt on the Occasion of his 60th Birthday

2. From context free grammars to functors

A context free grammar (CFG) is a standard notion in language processing. It consists of a set of production rules, like $v \to a_1 v_1 v_2 a_2 v_3$, telling how non-terminals $v \in V$ can be replaced by a string of both non-terminals $v_i \in V$ and terminals $a_j \in \Sigma$. The right hand side is thus an element of the set $(V + \Sigma)^\star$ of words built from letters from either V or Σ. Here we write $+$ for the disjoint union of the two sets V, Σ. The grammar as a whole can thus be described by a single function of the form

$$V \xrightarrow{\quad f \quad} \mathcal{P}\Big((V + \Sigma)^\star\Big) \tag{2.1}$$

It maps a non-terminal $v \in V$ to a set of right hand sides $w \in f(v)$, written as $v \to w$. This description casts CFGs in "coalgebraic" form, following [4], with V as set of states. It allows for easy generalisations to stochastic CFGs by taking the distribution monad $\mathcal{D}$ instead of the powerset monad $\mathcal{P}$ in (2.1), or to weighted CFGs by taking the multiset monad $\mathcal{M}$ instead of $\mathcal{P}$. The coalgebraic representation leads to a trace semantics for such CFGs in the Kleisli category associated with the monad. It can be used to describe the associated skeleton parse trees and generated strings, see [4] for details.

Here we go in a different direction. We show how to associate with a CFG f as in (2.1) an endofunctor $F\colon \mathbf{Sets}^V \to \mathbf{Sets}^V$ on the category $\mathbf{Sets}^V$ of V-indexed families $(X_v)_{v\in V}$ of sets X_v. A morphism $(X_v)_{v\in V} \to (Y_v)_{v\in V}$ in this category consists of a collection of functions $\varphi_v\colon X_v \to Y_v$ for $v \in V$. Composition and identities are obtained "componentwise".

In order to define the functor F associated with f we need some notation. For a word $w \in (V + \Sigma)^\star$ of both non-terminals and terminals we write $\overline{w} \in V^\star$ for the word obtained from w by removing all terminals. Further, for a V-indexed collection $(X_v)_{v\in V}$ and a word $\langle v_1, \ldots, v_n\rangle \in V^\star$ we write $X_{\langle v_1,\ldots,v_n\rangle} = X_{v_1} \times \cdots \times X_{v_n}$. This Cartesian product is a singleton $1 = \{*\}$ in case the sequence is empty. Now we can define the functor $F\colon \mathbf{Sets}^V \to \mathbf{Sets}^V$ associated with f as:

$$F\Big((X_v)_{v\in V}\Big) \;=\; \Big(\coprod_{w\in f(v)} X_{\overline{w}}\Big)_{v\in V}. \tag{2.2}$$

The notation $\coprod$ is used for indexed disjoint union, as generalisation of the binary $+$. It is easy to see what the functor F does on morphisms $\varphi = \big(\varphi_v\colon X_v \to Y_v\big)_{v\in V}$, namely $F(\varphi)(\langle w, x_1, \ldots, x_n\rangle) = \langle w, \varphi_{v_1}(x_1), \ldots, \varphi_{v_n}(x_n)\rangle$, where $\overline{w} = \langle v_1, \ldots, v_n\rangle$.

The essence of this definition is already contained in [2, 1] where a multi-sorted signature is associated with a CFG. We shall illustrate this functor definition in our two leading examples below. The next section will show how such functors between categories of indexed sets will be used for providing semantics for languages.

2.1. **Binary tree grammar.** A simple grammar for binary trees is standardly described via productions as on the right, say with $E = \{e_1, e_2, \ldots\}$. The associated trees have labels from the set E at the leaves. We can describe this grammar in coalgebraic form (2.1) as follows. The state space is a singleton $\{S\}$, since this grammar is "single-sorted". The associated map $g\colon \{S\} \to \mathcal{P}((\{S\} + E)^\star)$ is given as $g(S) = \{\langle e\rangle \mid e \in E\} \cup \{\langle S, S\rangle\}$, in which the above two productions are recognisable. Since the category $\mathbf{Sets}^{\{S\}}$ is isomorphic to $\mathbf{Sets}$ we get an associated functor $G\colon \mathbf{Sets} \to \mathbf{Sets}$ given by:

$$\begin{array}{lcl} S & \to & e_1 \mid e_2 \mid \ldots \\ S & \to & SS \end{array}$$

$$G(X) \;=\; E + (X \times X).$$

Again, it reflects the productions in obvious manner.

2.2. **Binary number grammar.** Our next example from [7] is slightly less trivial. It describes, on the right, a grammar for numbers of the form β or $\beta.\gamma$, where $\beta, \gamma \in \{0,1\}^\star$ are bit strings. Now we have a "many-sorted" grammar with set of non-terminals $\{B, L, N\}$ and coalgebra map $h\colon \{B, L, N\} \to \mathcal{P}((\{B, L, N\} + \{0, 1, .\})^\star)$ given by three equations: $h(B) = \{\langle 0\rangle, \langle 1\rangle\}$, $h(L) = \{\langle B\rangle, \langle L, B\rangle\}$ and $h(N) = \{\langle L\rangle, \langle L, ., L\rangle\}$. Notice the role of the dot (.) as terminal. There is an associated endofunctor H on the category $\mathbf{Sets}^{\{B,L,N\}} \cong \mathbf{Sets}^3$. It is given, according to (2.2) by:

$$\begin{array}{rcl} B & \to & 0 \mid 1 \\ L & \to & B \mid LB \\ N & \to & L \mid L.L \end{array}$$

$$H(X_B, X_L, X_N) \quad = \quad (1+1, X_B + (X_L \times X_B), X_L + (X_L \times X_L)).$$

We shall illustrate how for instance the last component $X_L + (X_L \times X_L)$ on the right hand side of this definition arises. According to the general description (2.2) we have as third component, indicated by subscript $(-)_N$:

$$\begin{array}{rcl} H(X_B, X_L, X_N)_N & = & \coprod_{\sigma \in h(N)} (X_B, X_L, X_N)_{\overline{\sigma}} \\ & = & (X_B, X_L, X_N)_{\overline{\langle L\rangle}} + (X_B, X_L, X_N)_{\overline{\langle L,.,L\rangle}} \\ & = & (X_B, X_L, X_N)_{\langle L\rangle} + (X_B, X_L, X_N)_{\langle L, L\rangle} \\ & = & X_L + (X_L \times X_L). \end{array}$$

Notice how the overline mapping $\overline{(-)}$ removes the dot . since it is terminal.

It is not hard to see that categories of the form $\mathbf{Sets}^V$ have arbitrary products $\prod$ and coproducts $\coprod$ given by pointwise constructions. Also, exponents $(X_v)_{v\in V} \Rightarrow (Y_v)_{v\in V}$ are obtained pointwise, namely as $(X_v \Rightarrow Y_v)_{v\in V}$.

3. Parse trees as initial algebras

For an arbitrary endofunctor $F\colon \mathbb{C} \to \mathbb{C}$ on a category $\mathbb{C}$ an algebra is a map in $\mathbb{C}$ of the form $a\colon F(A) \to A$. A homomorphism (or map) of algebras, from $F(A) \xrightarrow{a} A$ to $F(B) \xrightarrow{b} B$ is a morphism $f\colon A \to B$ in $\mathbb{C}$ with $f \circ a = b \circ F(f)$. This yields a category $\mathbf{Alg}(F)$, with obvious forgetful functor $\mathbf{Alg}(F) \to \mathbb{C}$ which maps an algebra $F(A) \to A$ to its carrier $A \in \mathbb{C}$.

An initial algebra of an endofunctor F is an initial object in its category $\mathbf{Alg}(F)$ of algebras. It is an algebra $(FA \to A)$ with the special property that for each algebra $(FB \to B)$ there is a unique homomorphism of algebras $(FA \to A) \longrightarrow (FB \to B)$. We shall often write this homomorphism via "Scott" or "interpretation" brackets $[\![-]\!]\colon A \to B$, as in the diagram:

$$\begin{array}{ccc} FA & \xrightarrow{F([\![-]\!])} & FB \\ {\scriptstyle\cong}\downarrow & & \downarrow \\ A & \xrightarrow{[\![-]\!]} & B \end{array} \tag{3.1}$$

We have labeled the initial algebra map with the isomorphism symbol $\cong$ since it is by general reasoning an isomorphism. This fact is often called Lambek's lemma, see for instance [6].

Initial algebras are typically term algebras, formed by iteratively applying the rules for term formation. The map $[\![-]\!]$ obtained by initiality then provides the interpretation of terms in some other domain in a "compositional" manner. It corresponds to definition by induction, see [6] for details. Later on we shall extensively use this homomorphism property (3.1) of the mapping $[\![-]\!]$ for computing interpretations. Commutation of the diagram captures semantic equations.

The initial algebra of a functor F, if it exists, is sometimes written as $\mu X.\, F(X)$, or simply as μF. There are general criteria that guarantee existence of initial algebras, but they do not matter here. Fundamental for what follows is the following observation.

Fact 3.1. The initial algebra of the functor $F\colon \mathbf{Sets}^V \to \mathbf{Sets}^V$ associated as in (2.2) with a CFG $V \to \mathcal{P}((V + \Sigma)^\star)$ is given by the indexed collection $(P_v)_{v \in V}$ of sets P_v of v-rooted parse trees of the grammar.

We shall illustrate this result, and its application to attribute grammars, for our leading examples. These attribute grammars will be identified as algebras of the associated functor.

3.1. **Binary trees.** Recall the functor $G(X) = E + (X \times X)$ associated with the grammar for binary trees in Subsection 2.1. We shall write its initial algebra as set of binary (S-rooted) trees BT with algebra map as on the right. This says that an e-label goes to an S-rooted tree with just this label at its leaf, and that a pair of trees is combined to a single S-rooted tree. Clearly this is an isomorphism, because an arbitrary S-rooted tree $t \in BT$ has either such a direct leaf or a binary node with two subtrees.

$$\begin{array}{rcl} G(BT) = E + (BT \times BT) & \xrightarrow{\;\cong\;} & BT \\ e & \longmapsto & \left(\begin{array}{c} S \\ | \\ e \end{array}\right) \\ (t_1, t_2) & \longmapsto & \left(\begin{array}{ccc} & S & \\ t_1 & & t_2 \end{array}\right) \end{array}$$

An attribute grammar extends a CFG with attributes and so-called semantic equations for computing certain values for parse trees. We introduce this concept informally in examples. The attribute grammars of this section will only have what are called synthesised attributes. Inherited attributes will be considered later. Purely synthesized attribute grammars can be specified as algebras and initiality is then used to compute the attribute values of the root node.

A first example, from [8], is about the AVL property of binary trees. Recall that a tree is called AVL when it is balanced in the sense that the heights of each pair of (adjacent) subtrees differ at most by one. The AVL attribute grammar is based on the binary tree CFG. The only nonterminal S has two attributes *avl* and *ht* taking values from 2 and $\mathbb{N}$ where $2 = \{0, 1\}$ is the set of Booleans. The semantic equations associated to the production rules are

$$\begin{array}{rcll} \mathit{avl}(S) & = & 1 & \text{for } S \to e, \quad e \in E \\ \mathit{ht}(S) & = & 0 & \\ \mathit{avl}(S) & = & \mathit{avl}(S_\ell) \wedge \mathit{avl}(S_r) \wedge |\mathit{ht}(S_\ell) - \mathit{ht}(S_r)| \leq 1 & \text{for } S \to S_\ell S_r \\ \mathit{ht}(S) & = & \max(\mathit{ht}(S_\ell), \mathit{ht}(S_r)) + 1 & \end{array}$$

Here, subscripts are used for telling apart different occurrences of one nonterminal in one production (S occurs three times in the second production of the binary tree grammar).

For us, this grammar is an algebra of the functor G. The carrier set is $2 \times \mathbb{N}$, where the first component is for values of *avl* and the second for values of *ht*. The algebra structure is this:

$$\begin{array}{rcl} G(2 \times \mathbb{N}) = E + ((2 \times \mathbb{N}) \times (2 \times \mathbb{N})) & \longrightarrow & 2 \times \mathbb{N} \\ e & \longmapsto & (1, 0) \\ \langle (b_\ell, h_\ell), (b_r, h_r) \rangle & \longmapsto & (b_\ell \wedge b_r \wedge |h_\ell - h_r| \leq 1, \max(h_\ell, h_r) + 1) \end{array} \tag{3.2}$$

How to read this? The value $(1, 0)$ for a leaf e says that such a tree is AVL (value 1) and has height 0. The second assignment is more complicated: suppose for a left subtree we already have a Boolean value $b_\ell \in 2$ for AVL-ness and height $h_\ell \in \mathbb{N}$, and similarly $b_r \in 2$ and $h_r \in \mathbb{N}$ for a right subtree. For the tree combined from these subtrees we can then compute:

- the Boolean value for AVL-ness as: $b_\ell \wedge b_r \wedge |h_\ell - h_r| \leq 1$. Indeed the combined tree is AVL requires a conjunction of tree things: the left subtree is AVL (b_ℓ), the right subtree is AVL (b_r) and the difference of heights of the two subtrees is at most one: $|h_\ell - h_r| \leq 1$.
- the height as the maximum of the heights of the subtrees plus one: $\max(h_\ell, h_r) + 1$.

Initiality of BT gives an interpretation map $[\![-]\!] \colon BT \to 2 \times \mathbb{N}$ as in (3.1). It consists of a pair of maps $[\![-]\!]_1 \colon BT \to 2$ and $[\![-]\!]_2 \colon BT \to \mathbb{N}$, where $[\![-]\!]_1$ computes whether a tree is AVL and $[\![-]\!]_2$ computes the height. Commutation of the initiality diagram amounts to two "semantic" equations:

$$\begin{aligned} \left[\!\!\left[\begin{array}{c} S \\ | \\ e \end{array} \right]\!\!\right] &= (1, 0) \\ \left[\!\!\left[\begin{array}{ccc} & S & \\ t_\ell & & t_r \end{array} \right]\!\!\right] &= ([\![t_\ell]\!]_1 \wedge [\![t_r]\!]_1 \wedge |[\![t_\ell]\!]_2 - [\![t_r]\!]_2| \leq 1,\ \max([\![t_\ell]\!]_2, [\![t_r]\!]_2) + 1). \end{aligned}$$

Here is a simple illustration. For convenience we write indices i on tree nodes S_i (simply to distinguish them) and show during the computation only the relevant part of the tree at that point.

$$\begin{aligned} \left[\!\!\left[\begin{array}{ccccc} & & S_1 & & \\ & S_2 & & S_3 & \\ e_1 & & S_4 & & S_5 \\ & & e_2 & & e_3 \end{array} \right]\!\!\right]_1 &= \left[\!\!\left[\begin{array}{c} S_2 \\ | \\ e_1 \end{array} \right]\!\!\right]_1 \wedge \left[\!\!\left[\begin{array}{ccc} & S_3 & \\ \cdot & & \cdot \end{array} \right]\!\!\right]_1 \wedge \left| \left[\!\!\left[\begin{array}{c} S_2 \\ | \\ e_1 \end{array} \right]\!\!\right]_2 - \left[\!\!\left[\begin{array}{ccc} & S_3 & \\ \cdot & & \cdot \end{array} \right]\!\!\right] \right| \leq 1 \\ &= 1 \wedge \left[\!\!\left[\begin{array}{c} S_4 \\ | \\ e_2 \end{array} \right]\!\!\right]_1 \wedge \left[\!\!\left[\begin{array}{c} S_5 \\ | \\ e_3 \end{array} \right]\!\!\right]_1 \wedge \left| \left[\!\!\left[\begin{array}{c} S_4 \\ | \\ e_2 \end{array} \right]\!\!\right]_2 - \left[\!\!\left[\begin{array}{c} S_5 \\ | \\ e_3 \end{array} \right]\!\!\right]_2 \right| \leq 1 \wedge \\ &\qquad \left| 0 - (\max\left(\left[\!\!\left[\begin{array}{c} S_4 \\ | \\ e_2 \end{array} \right]\!\!\right]_2, \left[\!\!\left[\begin{array}{c} S_5 \\ | \\ e_3 \end{array} \right]\!\!\right]_2 \right) + 1) \right| \leq 1 \\ &= 1 \wedge 1 \wedge |0 - 0| \leq 1 \wedge |0 - (\max(0, 0) + 1)| \leq 1 \\ &= 1. \end{aligned}$$

Hence this tree is indeed AVL. Notice how the computation proceeds "bottom-up" in the sense that values computed at subtrees are needed for values higher up in the tree.

3.2. **Binary numbers.** We will now show that the initial algebra of the functor $H \colon \mathbf{Sets}^3 \to \mathbf{Sets}^3$ for the binary number grammar from Subsection 2.2 has the triple $BN = (BN_B, BN_L, BN_N)$ of B-, L-, and N-rooted trees as initial algebra. Such an initial algebra consists of an isomorphism $H(BN) \xrightarrow{\cong} BN$ in the category $\mathbf{Sets}^3$, and thus of a triple of isomorphisms $H(BN)_i \xrightarrow{\cong} BN_i$ for $i \in \{B, L, N\}$. They are given in the "obvious" manner by constructing trees, see Figure 1.

The following attribute grammar is from [7]. It gives a way to assign numerical meaning to parse trees of the binary numbers CFG. All nonterminals have an attribute *val* and the nonterminal L (for bitstrings) has a further attribute *len*; the *val* attributes of B and L and the *len* attribute are $\mathbb{N}$-valued, the *val* attribute for N takes values from $\mathbb{Q}$. The semantic equations are:

$$\begin{array}{lcll} \mathit{val}(B) & = & b & \text{for } B \to b, \quad b \in \{0, 1\} \\ \mathit{val}(L) & = & \mathit{val}(B) & \text{for } L \to B \\ \mathit{len}(L) & = & 1 & \\ \mathit{val}(L) & = & 2\mathit{val}(L') + \mathit{val}(B) & \text{for } L \to L'B \\ \mathit{len}(L) & = & \mathit{len}(L') + 1 & \\ \mathit{val}(N) & = & \mathit{val}(L) & \text{for } N \to L \\ \mathit{val}(N) & = & \mathit{val}(L_1) + \mathit{val}(L_2)/2^{\mathit{len}(L_2)} & \text{for } N \to L_1.L_2 \end{array}$$

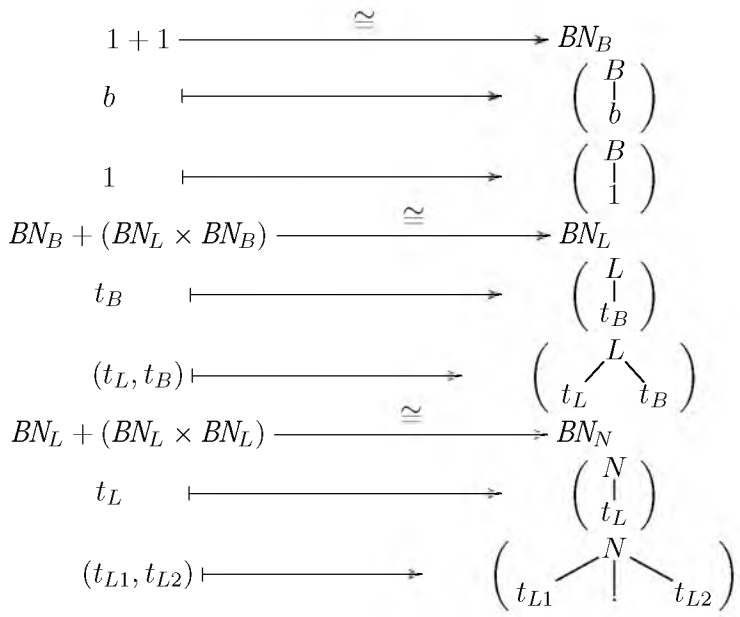


Figure 1: The initial algebra structure in $\mathbf{Sets}^3$ of binary numbers

The presentation as an algebra uses a carrier in $\mathbf{Sets}^3$ given by the triple of sets $(\mathbb{N}, \mathbb{N}^2, \mathbb{Q})$, with algebra/attribute structure $H(\mathbb{N}, \mathbb{N}^2, \mathbb{Q}) \to (\mathbb{N}, \mathbb{N}^2, \mathbb{Q})$ given by the three maps in:

$$\begin{array}{rcl}
1 + 1 & \longrightarrow & \mathbb{N} \\
0 & \longmapsto & 0 \\
1 & \longmapsto & 1 \\
\mathbb{N} + (\mathbb{N}^2 \times \mathbb{N}) & \longrightarrow & \mathbb{N}^2 \\
b & \longmapsto & (b, 1) \\
((n, m), b) & \longmapsto & (2n + b, m + 1) \\
\mathbb{N}^2 + (\mathbb{N}^2 \times \mathbb{N}^2) & \longrightarrow & \mathbb{Q} \\
(n, m) & \longmapsto & n \\
((n, m), (p, q)) & \longmapsto & n + \frac{p}{2^q}.
\end{array}$$

These mappings show that:

- the B-value in $\mathbb{N}$ gives the value of a bit;
- the L-value in $\mathbb{N}^2$ consists of a value of a bit string together with its length;
- the N-value in $\mathbb{Q}$ gives the ordinary bit string value, possibly with a quotient for the string after the dot.

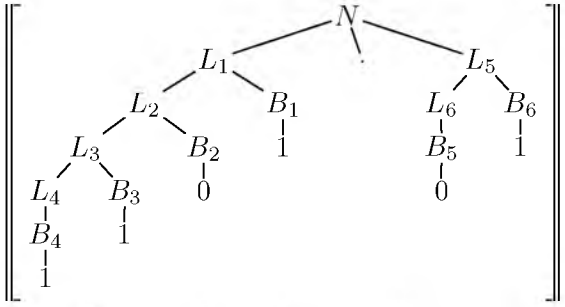
By initiality of $H(BN) \xrightarrow{\cong} BN$ we obtain an interpretation map $[\![-]\!] \colon (BN_B, BN_L, BN_N) \to (\mathbb{N}, \mathbb{N}^2, \mathbb{Q})$ in $\mathbf{Sets}^3$. We can write it as three separate maps $[\![-]\!]_B \colon BN_B \to \mathbb{N}$, $[\![-]\!]_L \colon BN_L \to \mathbb{N}^2$ and $[\![-]\!]_N \colon BN_N \to \mathbb{Q}$. The map $[\![-]\!]_L$ can then be split into two separate maps $[\![-]\!]_{L,i} \colon BN_L \to \mathbb{N}$, for $i = 1, 2$. Commutation of the initiality diagram amounts to the following equations.

$$\left[\!\!\left[\begin{array}{c} N \\ | \\ b \end{array} \right]\!\!\right]_B = b, \quad \text{for } b \in \{0, 1\}$$

$$\left[\!\left[\begin{array}{c} L \\ | \\ t \end{array} \right]\!\right]_L = (\llbracket t \rrbracket_B, 1) \qquad \left[\!\left[\begin{array}{ccc} & L & \\ \diagup & & \diagdown \\ t_L & & t_B \end{array} \right]\!\right]_L = (2\llbracket t_L \rrbracket_{L,1} + \llbracket t_B \rrbracket_B, \llbracket t_L \rrbracket_{L,2} + 1)$$

$$\left[\!\left[\begin{array}{c} N \\ | \\ t \end{array} \right]\!\right]_N = \llbracket t \rrbracket_{L,1} \qquad \left[\!\left[\begin{array}{ccc} & N & \\ \diagup & | & \diagdown \\ t_1 & \cdot & t_2 \end{array} \right]\!\right]_N = \llbracket t_1 \rrbracket_{L,1} + \frac{\llbracket t_2 \rrbracket_{L,1}}{2^{\llbracket t_2 \rrbracket_{L,2}}}.$$

These homomorphism equations are the basis of value calculations, for instance for the word 1101.01. For convenience the nodes in the parse tree below are labeled with subscripts in order to distinguish them; also, the "sort" subscripts B, N, L on the interpretation function $\llbracket - \rrbracket$ are omitted. In order to avoid complicated superscripts we write $\exp(2, n)$ for 2^n.



$$\begin{aligned}
&= \left[\!\left[\begin{array}{ccc} & L_1 & \\ \diagup & & \diagdown \\ \cdot & & \cdot \end{array} \right]\!\right]_1 + \left[\!\left[\begin{array}{ccc} & L_5 & \\ \diagup & & \diagdown \\ \cdot & & \cdot \end{array} \right]\!\right]_1 / \exp(2, \left[\!\left[\begin{array}{ccc} & L_5 & \\ \diagup & & \diagdown \\ \cdot & & \cdot \end{array} \right]\!\right]_2) \\
&= 2\left[\!\left[\begin{array}{ccc} & L_2 & \\ \diagup & & \diagdown \\ \cdot & & \cdot \end{array} \right]\!\right]_1 + \left[\!\left[\begin{array}{c} B_1 \\ | \\ 1 \end{array} \right]\!\right] + \left(2\left[\!\left[\begin{array}{c} L_6 \\ | \\ \cdot \end{array} \right]\!\right]_1 + \left[\!\left[\begin{array}{c} B_6 \\ | \\ 1 \end{array} \right]\!\right]\right) / \exp(2, \left[\!\left[\begin{array}{c} L_6 \\ | \\ \cdot \end{array} \right]\!\right]_2 + 1) \\
&= 4\left[\!\left[\begin{array}{ccc} & L_3 & \\ \diagup & & \diagdown \\ \cdot & & \cdot \end{array} \right]\!\right]_1 + 2\left[\!\left[\begin{array}{c} B_2 \\ | \\ 0 \end{array} \right]\!\right] + 1 + \left(2\left[\!\left[\begin{array}{c} B_5 \\ | \\ 0 \end{array} \right]\!\right]_1 + 1\right) / \exp(2, 2) \\
&= 8\left[\!\left[\begin{array}{c} L_4 \\ | \\ \cdot \end{array} \right]\!\right]_1 + 4\left[\!\left[\begin{array}{c} B_3 \\ | \\ 1 \end{array} \right]\!\right] + 1 + \tfrac{1}{4} = 8\left[\!\left[\begin{array}{c} B_4 \\ | \\ 1 \end{array} \right]\!\right]_1 + 5 + \tfrac{1}{4} = 13.25.
\end{aligned}$$

So far we have not said what attribute grammars are in general, but have described them as algebras for the functor associated with a grammar, as in (2.2). This semantical approach will be continued.

4. Attribute evaluation: the purely synthesised case

Attribute grammars are not only used for computing values for the root nodes of trees, as in the previous section, but also for computing values for the inner nodes—which were calculated implicitly in the earlier examples. In this section we show how to do such calculations explicitly. The attribute grammars that we considered so far are so-called purely synthesised ones, in which calculations on parse trees are performed "bottom-up", from children to parents. The general situation, also involving "top-down" calculations will be studied in Section 6.

The first step is to describe labeled trees abstractly.

Definition 4.1. For an arbitrary functor $F\colon \mathbb{C} \to \mathbb{C}$ and an object $A \in \mathbb{C}$ we write

$$F^{@}(A) = \mu X.A \times F(X).$$

Assuming that these initial algebras exist, we obtain a new functor $F^{@}\colon \mathbb{C} \to \mathbb{C}$.

$$\begin{array}{ccc} A \times F(F^{@}(A)) & \xrightarrow{\mathrm{id}_A \times F([\![-]\!])} & A \times F(Y) \\ {\scriptstyle \alpha_A}\downarrow{\scriptstyle \cong} & & \downarrow \\ F^{@}(A) & \xrightarrow{\qquad [\![-]\!] \qquad} & Y \end{array}$$

An arbitrary value $F^{@}(A)$ can be understood as A-labeled F-trees, as the examples below will illustrate. In general, a function $F^{@}(A) \to F^{@}(B)$ between such labeled trees may be called a relabeling function when it suitably preserves the tree structure (see below). We shall be especially interested in the case where $A = 1$. The value $F^{@}(1)$ at the final object 1 is the initial algebra $\mu X.F(X)$ of F-trees. Then it makes sense to talk of 'tree labeling' instead of 'relabeling'. Such labelings are also known as attribute evaluations. The initiality involved in $F^{@}(A)$ amounts to the following. For an arbitrary algebra $A \times F(Y) \to Y$ there is a unique map $[\![-]\!]$ as on the right. It can be shown that the functor $F^{@}$ is actually a comonad, but that is not very relevant here. We do however need the counit of this comonad structure. It is a special map $F^{@}(A) \to A$, namely:

$$\varepsilon_A \;=\; \Big(F^{@}(A) \xrightarrow[\cong]{\alpha_A^{-1}} A \times F(F^{@}(A)) \xrightarrow{\pi_1} A\Big)$$

This counit maps an A-labeled tree to the A-value at its root. The other (second) projection yields a map $\pi_2 \circ \alpha_A^{-1} \colon F^{@}(A) \to F(F^{@}(A))$ that forms a coalgebra of the functor F. It is used in [5] to define the property "bottom-up-ness" for tree transformers, namely as "coalgebra homomorphism". This property holds for the relabeling function $(\!|-|\!)$ that we are about to define.

Attribute evaluation as tree (re)labeling is based on the following easy result.

Proposition 4.2. *In the above situation with functors F and $F^{@}$, an algebra $F(B) \to B$ induces an attribute evaluation function $(\!|-|\!) \colon F^{@}(1) \to F^{@}(B)$ such that the following diagram commutes.*

$$\begin{array}{ccc} F^{@}(1) & \xrightarrow{(\!|-|\!)} & F^{@}(B) \\ & {\scriptstyle [\![-]\!]} \searrow & \downarrow{\scriptstyle \varepsilon_B} \\ & & B \end{array}$$

Proof The attribute evaluation function $(\!|-|\!)$ is obtained by initiality, in the square on the left below.

$$\begin{array}{ccccc} F(F^{@}(1)) & \xrightarrow{F((\!|-|\!))} & F(F^{@}(B)) & \xrightarrow{F(\varepsilon_B)} & F(B) \\ {\scriptstyle \alpha_1}\downarrow{\scriptstyle \cong} & & \downarrow{\scriptstyle \beta'} & & \downarrow{\scriptstyle \beta} \\ F^{@}(1) & \xrightarrow{(\!|-|\!)} & F^{@}(B) & \xrightarrow{\varepsilon_B} & B \\ & {\scriptstyle [\![-]\!]} \searrow & & \nearrow & \end{array}$$

The algebra $\beta \colon F(B) \to B$ is assumed, and $\beta' \colon F(F^{@}(B)) \to F^{@}(B)$ is obtained as:

$$\beta' = \Big(F(F^{@}(B)) \xrightarrow{\langle \beta \circ F(\varepsilon_B), \mathrm{id} \rangle} B \times F(F^{@}(B)) \xrightarrow[\cong]{\alpha_B} F^{@}(B)\Big).$$

The square on the right then commutes by construction. By (the uniqueness part of) initiality of α_1 we obtain that $\varepsilon_B \circ (\!|-|\!) = [\![-]\!]$. □

In terms of the comonad structure on $F^{@}$, the function $(\!|-|\!)$ is the coKleisli extension of $[\![-]\!]$.

The attribute grammar for AVL-trees from Subsection 3.1 was originally described as an algebra $G(2 \times \mathbb{N}) \to 2 \times \mathbb{N}$ in (3.2). The associated functor $G^{@}$ maps a set A to the initial algebra of the functor $X \mapsto A \times G(X) = A \times (E + (X \times X)) \cong (A \times E) + (A \times X \times X)$. It consists of binary trees with labels from $A \times E$ at the leaves and from A at the nodes. Proposition 4.2 associates with

the attribute grammar $G(2 \times \mathbb{N}) \to 2 \times \mathbb{N}$ a relabeling function $(\!|-|\!)\colon G^{@}(1) \to G^{@}(2 \times \mathbb{N})$. It does so via a map $G(G^{@}(2 \times \mathbb{N})) \to G^{@}(2 \times \mathbb{N})$, written as β' in the above proof. Explicitly:

$$G(G^{@}(2 \times \mathbb{N})) = E + (G^{@}(2 \times \mathbb{N}) \times G^{@}(2 \times \mathbb{N})) \longrightarrow G^{@}(2 \times \mathbb{N})$$

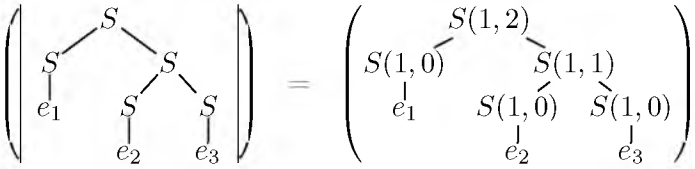
$$e \longmapsto \begin{pmatrix} S(1,0) \\ | \\ e \end{pmatrix}$$

$$(t_\ell, t_r) \longmapsto \begin{pmatrix} S(b,h) \\ t_\ell \quad t_r \end{pmatrix}$$

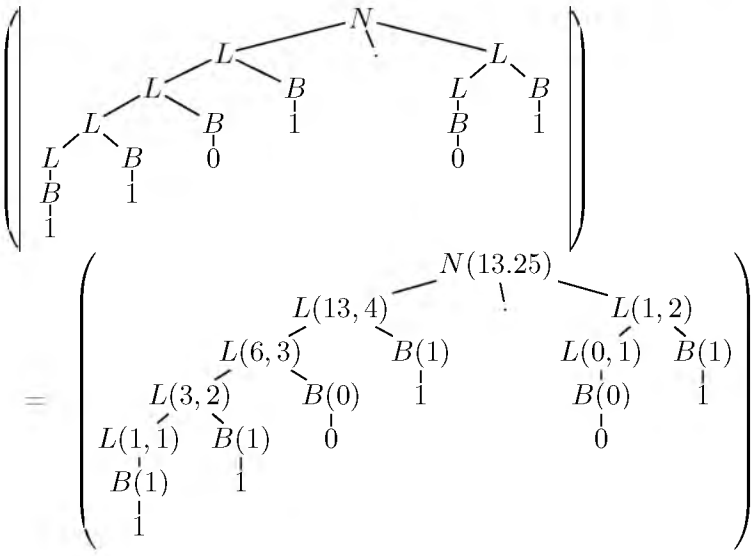
with, as in (3.2),

$$\begin{cases} b &= b_\ell \wedge b_r \wedge |h_\ell - h_r| \leq 1 \\ h &= \max(h_\ell, h_r) + 1 \end{cases} \quad \text{where} \quad \begin{cases} (b_\ell, h_\ell) &= \varepsilon(t_\ell) \\ (b_r, h_r) &= \varepsilon(t_r) \end{cases}$$

In the description of this mapping we use the convention to write the attribute values between brackets after the non-terminal at a node, as in $S(1,0)$. This yields, as example attribute evaluation:



In a similar way one may check that the attribute grammar $H(\mathbb{N}, \mathbb{N}^2, \mathbb{Q}) \to (\mathbb{N}, \mathbb{N}^2, \mathbb{Q})$ from Subsection 3.2 gives rise to a labeling (like in [7, (1.4)]):



The $[\![-]\!]$-calculation that we have done in Subsection 3.2 indeed yields the root value 13.25 of this labeled tree, obtained via ε, as formulated generally in the triangle in Proposition 4.2.

5. Inherited attributes

So far we have only given a limited view on attribute grammars, namely one in which only so-called synthesised attributes occur. Their values are obtained in bottom-up computations, out of values of the subtrees. There are also "inherited" attributes whose values depend on other values

higher up in the tree. Much research in the literature on attribute grammars is concerned with (syntactic) criteria for avoiding circularity between synthesised and inherited attributes. Here we only look at semantics, and simply claim (without proof) that non-circular attribute grammars with both synthesised and inherited attributes can be formulated as algebras (like in Section 3), but with exponents B^A as carriers, where the positive part B corresponds to the combined types of synthesised attributes, and the negative part A to the inherited ones.

In this section the claim will be illustrated for our leading examples of binary trees and binary numbers. How to do attribute evaluation in these cases will be described in the next section.

5.1. **Binary trees.** Our example comes again from [8] and involves pre-order numbering of the nodes of trees. It is described there via two $\mathbb{N}$-valued attributes *numin*, *numout* of the nonterminal S, which are given via the following semantic equations.

$$\begin{array}{lrcll} \text{(a)} & \mathit{numout}(S) & = & \mathit{numin}(S) & \text{for } S \to e, \quad e \in E \\ \text{(b)} & \mathit{numin}(S_\ell) & = & \mathit{numin}(S) + 1 & \text{for } S \to S_\ell S_r \\ \text{(c)} & \mathit{numin}(S_r) & = & \mathit{numout}(S_\ell) + 1 & \\ \text{(d)} & \mathit{numout}(S) & = & \mathit{numout}(S_r) & \end{array}$$

The attribute *numout* is auxiliary, and only used to compute the value *numin* that we are interested in. The inherited aspect appears in equation (b), making the *numin*-value of the left subtree dependent on the *numin* value of its parent. This will be made explicit in an algebraic description.

Indeed, we claim that we can capture this attribute grammar as an algebra of the functor $G(X) = E + (X \times X)$, introduced in Subsection 3.1 for binary trees. As carrier we take the exponent $\mathbb{N}^{\mathbb{N}}$ of inherited-to-synthesised attribute types. The algebra structure $G(\mathbb{N}^{\mathbb{N}}) \to \mathbb{N}^{\mathbb{N}}$ is:

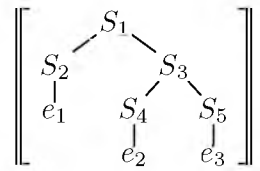
$$\begin{array}{rcl} G(\mathbb{N}^{\mathbb{N}}) = E + (\mathbb{N}^{\mathbb{N}} \times \mathbb{N}^{\mathbb{N}}) & \longrightarrow & \mathbb{N}^{\mathbb{N}} \\ e & \longmapsto & \lambda n \in \mathbb{N}.\, n \\ \langle f_\ell, f_r \rangle & \longmapsto & \lambda n \in \mathbb{N}.\, f_r(f_\ell(n+1)+1) \end{array} \tag{5.1}$$

A function in $\mathbb{N}^{\mathbb{N}}$ is seen as a mapping that takes the *numin* value of a node to its *numout* value. On leaves it must be the identity, by equation (a). If we already have two such functions f_ℓ, f_r for the left and right subtrees, then the resulting function f for their parent computes a *numout* value from a *numin* value n as:

- the output f_r of the right subtree—by equation (d), ...
- ... applied to the the *numin* value of the right subtree, which is the *numout* value f_ℓ of the left subtree plus one—by equation (c), ...
- ... applied to the *numin* value $n+1$ of the parent plus one—by equation (b).

Hence we have $f(n) = f_r(f_\ell(n+1)+1)$ as described in (5.1).

So what does this algebra $G(\mathbb{N}^{\mathbb{N}}) \to \mathbb{N}^{\mathbb{N}}$ give us? By initiality it leads to an interpretation map $[\![-]\!] : BT \to \mathbb{N}^{\mathbb{N}}$. The latter yields for a tree $t \in BT$ a function $[\![t]\!] \in \mathbb{N}^{\mathbb{N}}$ that computes the *numout* value of the root node from a given *numin* value of the root. For instance,



$$\begin{array}{rcl} [\![\cdots]\!](n) & = & [\![\cdot\!\!\overset{S_3}{\diagup\diagdown}\!\!\cdot]\!]\left([\![\overset{S_2}{|}\,e_1]\!](n+1)+1\right) \;=\; [\![\cdot\!\!\overset{S_3}{\diagup\diagdown}\!\!\cdot]\!](n+2) \\ & = & [\![\overset{S_5}{|}\,e_3]\!]\left([\![\overset{S_4}{|}\,e_2]\!](n+3)+1\right) \;=\; n+4. \end{array}$$

Section 6 describes how to do attribute evaluation also for the inner nodes.

5.2. **Binary numbers.** Knuth [7] describes a second semantics for binary numbers, which is "more close to the manner in which we usually think of the notation". It involves additional integer-valued *scale* attributes for bits B and bitstrings L, which are inherited. This *scale* attribute is used for a different interpretation for bits (B), namely not as $0, 1$ values, but as rationals depending on their position, given by the scale. The *val* and *len* attributes from Subsection 3.2 remain, but *val* is now rational-valued. The semantic equations are given on the right, in standard style.

$$
\begin{array}{rcll}
\mathit{val}(B) & = & 0 & \text{for } B \to 0 \\
\mathit{val}(B) & = & 2^{\mathit{scale}(B)} & \text{for } B \to 1 \\
\mathit{scale}(L) & = & \mathit{scale}(B) & \text{for } L \to B \\
\mathit{val}(L) & = & \mathit{val}(B) & \\
\mathit{len}(L) & = & 1 & \\
\mathit{scale}(L') & = & \mathit{scale}(L) + 1 & \text{for } L \to L'B \\
\mathit{scale}(B) & = & \mathit{scale}(L) & \\
\mathit{val}(L) & = & \mathit{val}(L') + \mathit{val}(B) & \\
\mathit{len}(L) & = & \mathit{len}(L') + 1 & \\
\mathit{scale}(L) & = & 0 & \text{for } N \to L \\
\mathit{val}(N) & = & \mathit{val}(L) & \\
\mathit{scale}(L_1) & = & 0 & \text{for } N \to L_1.L_2 \\
\mathit{scale}(L_2) & = & -\mathit{len}(L_2) & \\
\mathit{val}(N) & = & \mathit{val}(L_1) + \mathit{val}(L_2) &
\end{array}
$$

As objects of the category $\mathbf{Sets}^3$ in which we work for this example the types of the inherited attributes of the non-terminals (B, L, N) are given as $(\mathbb{Z}, \mathbb{Z}, 1)$. Those of the synthesized ones are given as $(\mathbb{Q}, \mathbb{Q} \times \mathbb{N}, \mathbb{Q})$. According to the claim mentioned in the beginning of this section we should be able to express Knuth's attribute grammar as an H-algebra with as carrier the exponent in the category $\mathbf{Sets}^3$:

$$(\mathbb{Q}, \mathbb{Q} \times \mathbb{N}, \mathbb{Q})^{(\mathbb{Z},\mathbb{Z},1)} \quad = \quad (\mathbb{Q}^{\mathbb{Z}}, (\mathbb{Q} \times \mathbb{N})^{\mathbb{Z}}, \mathbb{Q}^1) \quad \cong \quad (\mathbb{Q}^{\mathbb{Z}}, \mathbb{Q}^{\mathbb{Z}} \times \mathbb{N}^{\mathbb{Z}}, \mathbb{Q}).$$

The H-algebra structure on this exponent is given as follows.

$$
\begin{array}{rcl}
1 + 1 & \longrightarrow & \mathbb{Q}^{\mathbb{Z}} \\
0 & \longmapsto & \lambda s.\, 0 \\
1 & \longmapsto & \lambda s.\, 2^s \\
\mathbb{Q}^{\mathbb{Z}} + (\mathbb{Q}^{\mathbb{Z}} \times \mathbb{N}^{\mathbb{Z}}) \times \mathbb{Q}^{\mathbb{Z}} & \longrightarrow & \mathbb{Q}^{\mathbb{Z}} \times \mathbb{N}^{\mathbb{Z}} \\
b & \longmapsto & (b, \lambda s.\, 1) \\
(\langle f, g \rangle, b) & \longmapsto & (\lambda s.\, f(s+1) + b(s), \lambda s.\, g(s+1) + 1) \\
\mathbb{Q}^{\mathbb{Z}} \times \mathbb{N}^{\mathbb{Z}} + (\mathbb{Q}^{\mathbb{Z}} \times \mathbb{N}^{\mathbb{Z}}) \times (\mathbb{Q}^{\mathbb{Z}} \times \mathbb{N}^{\mathbb{Z}}) & \longrightarrow & \mathbb{Q} \\
(f, g) & \longmapsto & f(0) \\
\langle (f, g), (h, k) \rangle & \longmapsto & f(0) + h(-k(0)).
\end{array}
$$

By initiality we then get an interpretation map $[\![-]\!] \colon BN \to (\mathbb{Q}^{\mathbb{Z}}, \mathbb{Q}^{\mathbb{Z}} \times \mathbb{N}^{\mathbb{Z}}, \mathbb{Q})$ in $\mathbf{Sets}^3$. It allows us to compute the root value of a tree using the algebra homomorphism properties of $[\![-]\!]$, as in Figure 2.

In [1] a slightly simplified version of this algebra is described, namely with carrier $(\mathbb{Q}^{\mathbb{Z}}, \mathbb{Q}^{\mathbb{Z}} \times \mathbb{N}, \mathbb{Q})$. The difference lies in the second component: they use $\mathbb{Q}^{\mathbb{Z}} \times \mathbb{N}$ instead of our $\mathbb{Q}^{\mathbb{Z}} \times \mathbb{N}^{\mathbb{Z}}$ because they notice that the (synthesised) *len* attribute with type $\mathbb{N}$ does not depend on the (inherited) *scale* with type $\mathbb{Z}$. Here we stick with the general exponent form, which means that we have to pick an arbitrary value as input for the function k in the last line of our algebra description. The point here is that the attribute dependency analysis has to happen in the formulation itself of an attribute grammar as an algebra of a functor: the framework enforces it (which we see as advantage).

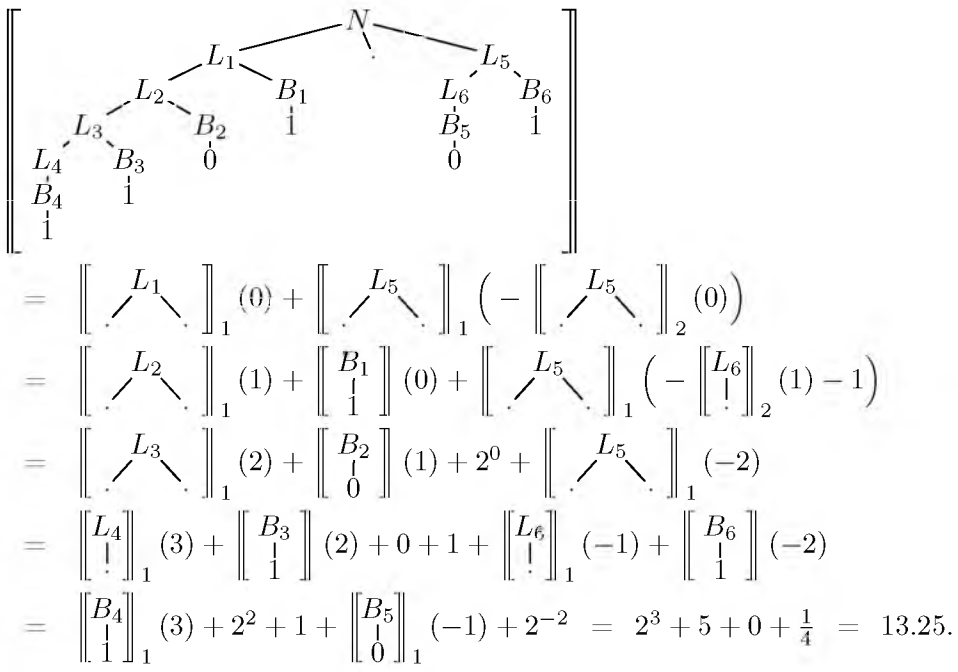


Figure 2: Sample calculation for the algebra from Subsection 5.2.

6. Attribute evaluation: the general case

We have described general attribute grammars with both synthesised and inherited attributes as algebras with an exponent as carrier. Here we describe attribute evaluation for such grammars, via an extension of Proposition 4.2 given below. It requires that the algebra $\gamma\colon F(B^A) \to B^A$ involved arises from a more general structure, namely a natural transformation with components of the form:

$$F\big((B \times X)^A\big) \xrightarrow{\Gamma_X} \big(B \times F(X)\big)^A \tag{6.1}$$

Such a natural transformation Γ is a collection of maps (Γ_X), indexed by objects X, which work "naturally" or "uniformly" in X. This means that for any function $f\colon X \to Y$ one has $(\mathrm{id}_B \times F(f))^A \circ \Gamma_X = \Gamma_Y \circ F((\mathrm{id}_B \times f)^A)$.

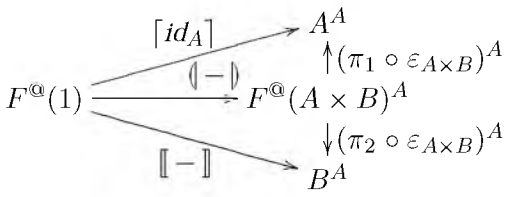
We then assume that the algebra $\gamma\colon F(B^A) \to B^A$ arises by taking $X = 1$. To be precise as:

$$\gamma = \Big(F(B^A) \xrightarrow{F(\langle \mathrm{id}, ! \rangle^A)} F\big((B \times 1)^A\big) \xrightarrow{\Gamma_1} \big(B \times F(1)\big)^A \xrightarrow{\pi_1^A} B^A\Big) \tag{6.2}$$

The extra generality given by Γ is needed to lead the values of the inherited attributes appropriately down the subtrees. If the functor F is "strong" and we already have an algebra $F(B^A) \to B^A$ then one can construct a trivial natural transformation Γ as in (6.1) by passing on the same argument downwards. But as the examples below will illustrate, Γ typically adapts the arguments.

We assume that the category in which we work is cartesian closed, so that we have exponent objects B^A, for arbitrary objects A, B, with evaluation maps $ev\colon B^A \times A \to B$ and abstractions $\Lambda(f)\colon C \to B^A$ for $f\colon C \times A \to B$, satisfying the standard equations $ev \circ (\Lambda(f) \times \mathrm{id}) = f$, $\Lambda(f) \circ g = \Lambda(f \circ (g \times \mathrm{id}))$ and $\Lambda(ev) = \mathrm{id}$, see any basic text on category theory. Maps of the form f^A, as already used above, are defined as $f^A = \Lambda(f \circ ev)\colon X^A \to Y^A$, where $f\colon X \to Y$.

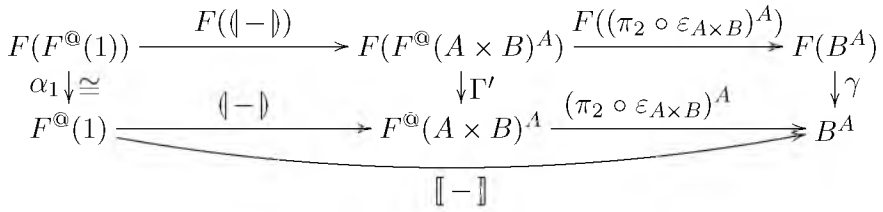
Proposition 6.1. *Assume an endofunctor $F\colon \mathbb{C} \to \mathbb{C}$ on a cartesian closed category $\mathbb{C}$ with associated labeled tree functor $F^{@}$ as in the beginning of Section 4. A natural transformation $\Gamma\colon F((B \times -)^A) \Rightarrow (B \times F(-))^A$ as in (6.1) induces an attribute evaluation function $(\!|-|\!)\colon F^{@}(1) \to F^{@}(A \times B)^A$ such that the following diagram commutes.*



where $\lceil id_A \rceil$ is the constant map yielding the identity function on A.

The attribute evaluation map $(\!|-|\!)\colon F^{@}(1) \to F^{@}(A \times B)^A$ takes a parse tree $t \in F^{@}(1)$ and an initial "inherited" value $a \in A$ to a labeled tree $(\!|t|\!)(a) \in F^{@}(A \times B)$ with combined inherited and synthesised labels from $A \times B$. Of course we can concentrate on the A-labels or the B-labels alone, by composing with a suitable projection $F^{@}(\pi_i)$.

Proof Assuming the natural transformation Γ with associated algebra γ as in (6.2) we construct Γ' in the middle of the diagram below.



The map Γ' is obtained as composite:

$$
\begin{array}{l}
F\big(F^{@}(A \times B)^A\big) \\
\qquad \downarrow F(\langle \pi_2 \circ \varepsilon, \mathrm{id} \rangle^A) \\
F\big((B \times F^{@}(A \times B))^A\big) \xrightarrow[\Gamma]{} \big(B \times F(F^{@}(A \times B))\big)^A \\
\qquad\qquad\qquad\qquad \downarrow \Lambda(\langle \langle \pi_2, \pi_1 \circ ev \rangle, \pi_2 \circ ev \rangle) \\
\qquad\qquad\qquad\qquad \big((A \times B) \times F(F^{@}(A \times B))\big)^A \xrightarrow[\alpha^A]{} F^{@}(A \times B)^A
\end{array}
$$

We show that the right-hand-square above commutes, via a tedious but elementary calculation.

$$
\begin{aligned}
(\pi_2 \circ \varepsilon)^A \circ \Gamma' &= \Lambda(\pi_2 \circ \pi_1 \circ \alpha^{-1} \circ ev) \circ \Lambda(\alpha \circ \langle \langle \pi_2, \pi_1 \circ ev \rangle, \pi_2 \circ ev \rangle) \circ \\
&\qquad \Gamma \circ F(\langle \pi_2 \circ \varepsilon, \mathrm{id} \rangle^A) \\
&= \Lambda(\pi_2 \circ \pi_1 \circ \alpha^{-1} \circ \alpha \circ \langle \langle \pi_2, \pi_1 \circ ev \rangle, \pi_2 \circ ev \rangle) \circ \\
&\qquad \Gamma \circ F(\langle \pi_2 \circ \varepsilon, \mathrm{id} \rangle^A) \\
&= \Lambda(\pi_1 \circ ev) \circ \Gamma \circ F(\langle \pi_2 \circ \varepsilon, \mathrm{id} \rangle^A) \\
&= \pi_1^A \circ (\mathrm{id} \times F(!))^A \circ \Gamma \circ F(\langle \pi_2 \circ \varepsilon, \mathrm{id} \rangle^A) \\
&= \pi_1^A \circ \Gamma \circ F((\mathrm{id} \times !)^A) \circ F(\langle \pi_2 \circ \varepsilon, \mathrm{id} \rangle^A) \quad \text{by naturality of } \Gamma \\
&= \pi_1^A \circ \Gamma \circ F((\langle \mathrm{id}, ! \rangle \circ \pi_2 \circ \varepsilon)^A) \\
&= \gamma \circ F((\pi_2 \circ \varepsilon)^A).
\end{aligned}
$$

In a similar, but easier, way one checks that $(\pi_1 \circ \varepsilon)^A \circ \Gamma' = \Lambda(\pi_2) = \lceil \mathrm{id} \rceil$. □

We shall make the general recipe for attribute evaluation from Proposition 6.1 more concrete by demonstrating it for our two running examples.

6.1. **Binary trees.** We first show how the attribute grammar $G(\mathbb{N}^{\mathbb{N}}) \to \mathbb{N}^{\mathbb{N}}$ from Subsection 5.1 arises in fact from a natural transformation with components $G((\mathbb{N} \times X)^{\mathbb{N}}) \to (\mathbb{N} \times G(X))^{\mathbb{N}}$ given as follows.

$$\begin{array}{rcl} E + (\mathbb{N} \times X)^{\mathbb{N}} \times (\mathbb{N} \times X)^{\mathbb{N}} & \longrightarrow & (\mathbb{N} \times (E + X \times X))^{\mathbb{N}} \\ e & \longmapsto & \lambda n \in \mathbb{N}.\, \langle n, e \rangle \\ \langle f_\ell, f_r \rangle & \longmapsto & \lambda n \in \mathbb{N}.\, \langle \pi_1 f_r(m+1), \langle \pi_2 f_\ell(n+1), \pi_2 f_r(m+1) \rangle \rangle \\ & & \text{where } m = \pi_1 f_\ell(n+1) \end{array}$$

This definition clearly contains the one for the algebra $G(\mathbb{N}^{\mathbb{N}}) \to \mathbb{N}^{\mathbb{N}}$ from (5.1). But it does a bit more: the argument n is also passed on to the subtrees via the X-component, in a similar manner. The arguments are thus not only needed for computing a value at a particular point in the tree, but also for attribute evaluation in the subtrees. According to Proposition 6.1 this natural transformation yields an attribute evaluation map $(\!|-|\!)\colon G^{\mathbb{Q}}(1) \to G^{\mathbb{Q}}(\mathbb{N} \times \mathbb{N})^{\mathbb{N}}$ via the algebra $G(G^{\mathbb{Q}}(\mathbb{N} \times \mathbb{N})^{\mathbb{N}}) \to G^{\mathbb{Q}}(\mathbb{N} \times \mathbb{N})^{\mathbb{N}}$, called Γ' in the above proof. In this case it is:

$$\begin{array}{rcl} E + G^{\mathbb{Q}}(\mathbb{N}^2)^{\mathbb{N}} \times G^{\mathbb{Q}}(\mathbb{N}^2)^{\mathbb{N}} & \longrightarrow & G^{\mathbb{Q}}(\mathbb{N}^2)^{\mathbb{N}} \\ e & \longmapsto & \lambda n.\, \begin{pmatrix} S(n,n) \\ | \\ e \end{pmatrix} \\ \langle f_\ell, f_r \rangle & \longmapsto & \lambda n.\, \begin{pmatrix} S(n,p_r) \\ \diagup \quad \diagdown \\ f_\ell(n+1) \quad f_r(p_\ell+1) \end{pmatrix} \text{ with } \begin{cases} p_\ell = \pi_2 \varepsilon(f_\ell(n+1)) \\ p_r = \pi_2 \varepsilon(f_r(p_\ell+1)) \end{cases} \end{array}$$

These p_ℓ and p_r are the second components of the root values of the subtrees.

Then we can also compute the (*numin*, *numout*)-values also for the inner nodes in the example calculation at the end of Subsection 5.1: the pre-order numbering starting at $n \in \mathbb{N}$ is:

$$\left(\begin{array}{ccccc} & & S & & \\ & \diagup & & \diagdown & \\ S & & & & S \\ | & & & \diagup & \diagdown \\ e_1 & & S & & S \\ & & | & & | \\ & & e_2 & & e_3 \end{array} \right)(n) \quad = \quad \left(\begin{array}{ccccc} & & S(n, n+4) & & \\ & \diagup & & \diagdown & \\ S(n+1, n+1) & & & & S(n+2, n+4) \\ | & & & \diagup & \diagdown \\ e_1 & & S(n+3, n+3) & & S(n+4, n+4) \\ & & | & & | \\ & & e_2 & & e_3 \end{array} \right)$$

6.2. **Binary numbers.** As before we first notice that the H-algebra structure on the exponent $(\mathbb{Q}, \mathbb{Q} \times \mathbb{N}, \mathbb{Q})^{(\mathbb{Z},\mathbb{Z},1)} \cong (\mathbb{Q}^{\mathbb{Z}}, \mathbb{Q}^{\mathbb{Z}} \times \mathbb{N}^{\mathbb{Z}}, \mathbb{Q})$ from Subsection 5.2 arises in fact from a natural transformation

$$H\Big(((\mathbb{Q}, \mathbb{Q} \times \mathbb{N}, \mathbb{Q}) \times X)^{(\mathbb{Z},\mathbb{Z},1)}\Big) \longrightarrow \Big((\mathbb{Q}, \mathbb{Q} \times \mathbb{N}, \mathbb{Q}) \times H(X)\Big)^{(\mathbb{Z},\mathbb{Z},1)}$$

By unravelling the definition of H for $X = (X_B, X_L, X_N)$ and using the componentwise description of exponents in $\mathbf{Sets}^3$ we see that this natural transformation consists of three maps of the

form:

$$1+1 \longrightarrow \big(\mathbb{Q}\times(1+1)\big)^{\mathbb{Z}}$$

$$\big(\mathbb{Q}\times X_B\big)^{\mathbb{Z}} + \big(\mathbb{Q}\times\mathbb{N}\times X_L\big)^{\mathbb{Z}} \times \big(\mathbb{Q}\times X_B\big)^{\mathbb{Z}} \longrightarrow \big((\mathbb{Q}\times\mathbb{N}\times(X_B + X_L\times X_B)\big)^{\mathbb{Z}}$$

$$\big(\mathbb{Q}\times\mathbb{N}\times X_L\big)^{\mathbb{Z}} + \big(\mathbb{Q}\times\mathbb{N}\times X_L\big)^{\mathbb{Z}} \times \big(\mathbb{Q}\times\mathbb{N}\times X_L\big)^{\mathbb{Z}} \longrightarrow \mathbb{Q}\times(X_L + X_L \times X_L).$$

It is given along the lines of the algebra definition in Subsection 5.2:

$$\begin{cases} 0 \longmapsto \lambda s.\,\langle 0,0\rangle \\ 1 \longmapsto \lambda s.\,\langle 2^s,1\rangle \end{cases}$$

$$\begin{cases} b \longmapsto \lambda s.\,\langle b_1(s),1,b_2(s)\rangle \\ \langle f,b\rangle \longmapsto \lambda s.\,\langle f_1(s+1)+b_1(s),\, f_2(s+1)+1,\, \langle f_3(s+1),b_2(s)\rangle\rangle \end{cases}$$

$$\begin{cases} f \longmapsto \langle f_1(0),f_3(0)\rangle \\ \langle f,g\rangle \longmapsto \langle f_1(0)+g_1(-g_2(0)),\langle f_3(0),g_3(-g_2(0))\rangle\rangle. \end{cases}$$

In the proof of Proposition 6.1 we then see how we get an H-algebra on

$$H^{@}((\mathbb{Z},\mathbb{Z},1)\times(\mathbb{Q},\mathbb{Q}\times\mathbb{N},\mathbb{Q}))^{(\mathbb{Z},\mathbb{Z},1)} \;\cong\; H^{@}(\mathbb{Z}\times\mathbb{Q},\mathbb{Z}\times\mathbb{Q}\times\mathbb{N},\mathbb{Q})^{(\mathbb{Z},\mathbb{Z},1)},$$

which we can describe explicitly as:

$$\begin{cases} 0 \longmapsto \lambda s.\,\begin{pmatrix} B(s,0) \\ | \\ 0 \end{pmatrix} \\ 1 \longmapsto \lambda s.\,\begin{pmatrix} B(s,2^s) \\ | \\ 1 \end{pmatrix} \end{cases}$$

$$\begin{cases} b \longmapsto \lambda s.\,\begin{pmatrix} L(s,v,1) \\ | \\ b(s) \end{pmatrix} \quad \text{where } \begin{cases} v = \pi_2\varepsilon(b(s)), \text{the second component of} \\ \quad \text{the root value of the subtree } b(s) \end{cases} \\ \langle f,b\rangle \longmapsto \lambda s.\,\begin{pmatrix} L(s,v,l) \\ \swarrow \quad \searrow \\ f(s+1) \quad b(s) \end{pmatrix} \quad \text{where } \begin{cases} v = \pi_2\varepsilon(f(s+1)) + \pi_2\varepsilon(b(s)) \\ l = \pi_3\varepsilon(f(s+1)) + 1 \end{cases} \end{cases}$$

$$\begin{cases} f \longmapsto \begin{pmatrix} N(v) \\ | \\ f(0) \end{pmatrix} \quad \text{where } v = \pi_2\varepsilon(f(0)) \\ \langle f,g\rangle \longmapsto \begin{pmatrix} \quad N(v) \quad \\ \swarrow \quad | \quad \searrow \\ f(0) \quad ! \quad g(-s) \end{pmatrix} \quad \text{where } \begin{cases} s = \pi_3\varepsilon(g(0)) \\ v = \pi_2\varepsilon(f(0)) + \pi_2\varepsilon(g(-s)) \end{cases} \end{cases}$$

Figure 3 illustrates the attribute evaluation function $(\!|-|\!)$ resulting from this algebra for our running example. It reconstructs equation (1.6) from [7] in a systematic manner via initiality.

7. Future work

After these first steps in a categorical reformulation of classical work in computer science many issues remain, among which we are particularly interested in the following three.

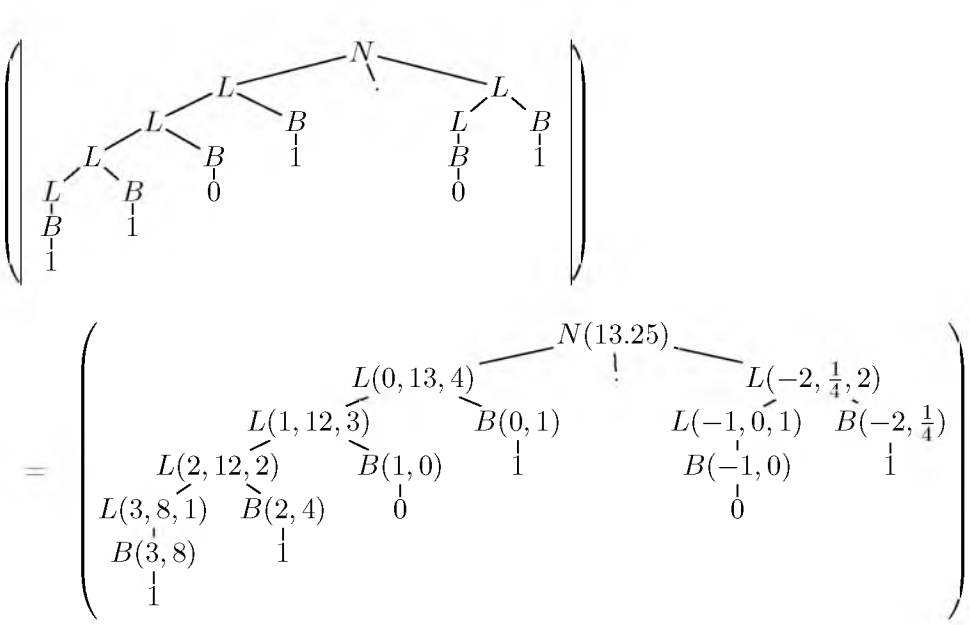


Figure 3: Attribute evaluation for the binary number example from Subsection 5.2

- Investigation of the generality obtained by a natural transformation $\Gamma\colon F((B \times -)^A) \Rightarrow (B \times F(-))^A$ as in the beginning of Section 6. In this paper, we required no conditions on such Γ. Is this appropriate, also in examples?
- Formulation of a natural transformation from a given set of semantical equations: is there a canonical way to do so?
- Implementation, for instance in the programming language Haskell, of attribute evaluation as described above.

References

[1] L.M. Chirica and D.F. Martin. An order-algebraic definition of knuthian semantics. *Math. Syst. Theory*, 13:1–27, 1979.

[2] J.A. Goguen and J. Thatcher. Initial algebra semantics. In *Symposium on Switching and Automata Theory*, pages 63–77. IEEE, 1974.

[3] J.A. Goguen, J. Thatcher, E. Wagner, and J. Wright. Initial algebra semantics and continuous algebras. *Journ. ACM*, 24(1):68–95, 1977.

[4] I. Hasuo and B. Jacobs. Context-free languages via coalgebraic trace semantics. In J.L. Fiadeiro, N. Harman, M. Roggenbach, and J. Rutten, eds., *Algebra and Coalgebra in Computer Science, CALCO 2005*, v. 3629 in *Lect. Notes in Comput. Sci.*, pages 213–231. Springer, 2005.

[5] I. Hasuo, B. Jacobs, and T. Uustalu. Categorical views on computations on trees. In L. Arge, C. Cachin, T. Jurdzinski, and A. Tarlecki, eds., *Int. Coll. on Automata, Languages and Programming, ICALP 2007*, v. 4596 in *Lect. Notes in Comput. Sci.*, pp. 619–630. Springer, 2007.

[6] B. Jacobs and J. Rutten. A tutorial on (co)algebras and (co)induction. *EATCS Bulletin*, 62:222–259, 1997.

[7] D.E. Knuth. Semantics of context-free languages. *Math. Syst. Theory*, 2:127–145, 1968.

[8] T. Uustalu and V. Vene. Comonadic functional attribute evaluation. In M. van Eekelen, ed., *Trends in Functional Programming 6*, pp. 145-162. Intellect, 2007.