Acta Informatica 35, 1–15 (1998)

© Springer-Verlag 1998

Structured numbers

Properties of a hierarchy of operations on binary trees

Vincent D. Blondel*

Institute of Mathematics, University of Liège, B-4000 Liège, Belgium (e-mail: vblondel@ulg.ac.be)

Received: 11 December 1995 / 30 December 1996

Abstract. We introduce a hierarchy of operations on (finite and infinite) binary trees. The operations are obtained by successive repetition of one initial operation. The first three operations are generalizations of the operations of addition, multiplication and exponentiation for positive integers.

1 Introduction

The product of two positive integers a and b is equal to the sum of b factors each equal to a. The bth exponent of a, denoted by $a \uparrow b$, can similarly be defined as the product of b factors each equal to a. The process of getting new operations by repeating old ones ends with exponentiation because this last operation is not associative. The definition

$$a \uparrow\uparrow b = \underbrace{a \uparrow a \uparrow a \uparrow \ldots \uparrow a}_{b \text{ factors}} \tag{1}$$

is ambiguous since for example $(4 \uparrow 4) \uparrow 4 \neq 4 \uparrow (4 \uparrow 4)$. For the the right hand side of (1) to be well defined both the number of factors and the order in which the operations $\uparrow$ are performed have to be specified.

A way of doing this is to ask the second operand to carry not only a quantitative information, the number of times a is repeated, but also a structured information, the order in which the operations $\uparrow$ are performed. Binary trees are naturally designated object to convey such a structured information. The number of external nodes (leaves) of a binary tree can be used to specify the number of factors, and the structure of the tree can then be used to specify the order in which the operations are performed.

* Parts of this work were completed while the author was at OCIAM Oxford, at KTH Stockholm and at INRIA Paris.

In this paper, we define countably many internal operations on binary trees. The first operation, which we denote by $\overset{1}{\cdot}$, is obtained by forming the binary tree whose left and right subtrees are equal to the operands. This operation is not associative. The second operation $\overset{2}{\cdot}$ is defined as follows: From the binary trees a and b we construct the binary tree $a \overset{2}{\cdot} b$ by repeating the operation $\overset{1}{\cdot}$ on the tree a with the structure dictated by b. In the same way, we define an operation $\overset{3}{\cdot}$ by repeating $\overset{2}{\cdot}$, an operation $\overset{4}{\cdot}$ by repeating $\overset{3}{\cdot}$, etc. We eventually obtain countably many internal operations ($\overset{k}{\cdot}$ for $k \geq 1$) with the definition

$$a \overset{k}{\cdot} b = \underbrace{a \overset{k-1}{\cdot} a \overset{k-1}{\cdot} a \overset{k-1}{\cdot} \ldots \overset{k-1}{\cdot} a}_{b \text{ factors}}$$

The number of external nodes of the binary tree resulting from the operation $\overset{1}{\cdot}$, $\overset{2}{\cdot}$ and $\overset{3}{\cdot}$ are equal to the sum, product and exponentiation of the number of external nodes of the operands. These three operations are thought of as binary trees counterparts of the usual operations of addition, multiplication and exponentiation. The operations $\overset{k}{\cdot}$ for $k \geq 4$ have no natural number counterparts since for these cases the structure of the trees have to be taken into account to compute the number of external nodes of a $\overset{k}{\cdot}$-product.

The object of this paper is to study some of the properties of the operations $\overset{k}{\cdot}$ described above and formalized in the second section of the paper. In Sect. 3 the operations are shown to satisfy algebraic properties that generalize elementary properties for integers. In Sect. 4 we show that binary trees can be decomposed in a unique way as products of prime binary trees. In Sect. 5 we analyse the operations $\overset{k}{\cdot}$ for $k \geq 4$. In Sect. 6 we describe various integer valued functions associated to trees and show how these functions behave with respect to $\overset{k}{\cdot}$-products of binary trees. In a final section we argue that the notions introduced for finite binary trees can be generalized for infinite trees. This is achieved by formalizing binary trees by means of factorial languages.

Different authors have proposed to continue the hierarchy $+, \times, \uparrow$ on natural numbers by introducing operations of "super-exponentiation". D. Knuth's recursive definition [18] is

$$\begin{aligned}
a \uparrow\uparrow b &= \underbrace{a \uparrow (a \uparrow (a \uparrow (a \ldots \uparrow a) \ldots))}_{b} \\
a \uparrow\uparrow\uparrow b &= \underbrace{a \uparrow\uparrow (a \uparrow\uparrow (a \uparrow\uparrow (a \ldots \uparrow\uparrow a) \ldots))}_{b} \\
a \underbrace{\uparrow \ldots \uparrow}_{k} b &= \underbrace{a \underbrace{\uparrow \ldots \uparrow}_{k-1} (a \underbrace{\uparrow \ldots \uparrow}_{k-1} a \ldots \underbrace{\uparrow \ldots \uparrow}_{k-1} a) \ldots))}_{b}
\end{aligned}$$

This definition coincide, modulo elementary notational modifications, with the definition originally given by Ackermann of a recursive function that is not primitive recursive (see [13]). The definition has the disadvantage of making an arbitrary choice on how the non-associative operations are performed and, as a result, these operations exhibit poor algebraic properties (see, however, [1]).

Operations on graphs, trees and binary trees constitute a classical object of study in theoretical computer science (see [20], [21], [19], [17, Vol. 1 Section 2.3]) but we have found no reference that uses the particular structure of binary trees as a mean for defining repeated operations. The contribution that is probably closest to ours is the "arithmetic of shapes" developed by I.M. Etherington half a century ago in the context of genetics. The transmission of a probability distribution of genes by mating is an operation that is commutative but not associative. In order to describe this operation, Etherington has introduced in [6] operations on trees that are similar to $\overset{1}{!}$, $\overset{2}{!}$ and $\overset{3}{!}$ and that have given rise to the widely studied genetic algebras (see [6], [15] and [5]). The operations $\overset{k}{!}$ for $k \geq 4$ are not defined in the context of genetic algebras because the trees considered there are not ordered and there is no natural definition of $\overset{k}{!}$ for $k \geq 4$ for unordered trees.

Motivated by the remarks made in this introduction, binary trees are introduced in [2], [3] and [4] as one possible representation of the concept of "structured number". A motivation for this terminology is justified by the following observation: Free groups with a simple generator are isomorphic to $(\mathbf{Z}, +)$, free monoids with a single generator are isomorphic to $(\mathbf{N}, +)$ and free groupoids with a single generator are isomorphic to $(BT, \overset{1}{!})$ where BT denotes the set of binary trees and $\overset{1}{!}$ is the first operation in our hierarchy. This analogy motivates the notation **SN** used in [2] for denoting the set of binary trees which can be seen as one possible representation of the notion of structured number.

2 The operations

Let BT be the set of binary trees [17, Vol. 1 Sect. 2.3] defined by the symbolic equation [9]:

$$BT = \bullet + \underset{BT\ BT}{\overset{\cdot}{/\ \backslash}} .$$

A tree $a \in BT$ different from the one node tree $\bullet$ is ordered in such a way that we can make a distinction between the left subtree $a_L \in BT$ and the right subtree $a_R \in BT$. For notational convenience we represent a binary tree $a \in BT$ by its horizontal paranthesed expression $\phi(a) \in \{\bullet, (,)\}^*$, where $\phi : BT \to \{\bullet, (,)\}^*$ is recursively defined by

$$\phi(\bullet) = \bullet$$

$$\phi\left(\underset{a_L\ a_R}{\overset{\cdot}{/\ \backslash}}\right) = (\phi(a_L)\ \phi(a_R))$$

In the sequel we identify $a \in BT$ with $\phi(a)$. A binary tree $a \in BT$ is represented by $\bullet$ if it is the one node tree, and by $(a_L a_R)$ if it has the left

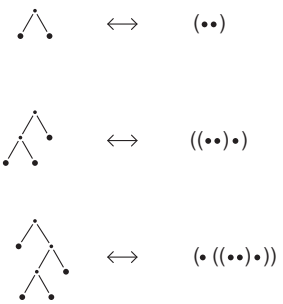

Fig. 1. Binary trees and their corresponding paranthesed expressions

and right subtree a_L and a_R, respectively. Some binary trees together with their paranthesed expressions are drawn in Fig. 1.

The number of external nodes (leaves) of a binary tree $a \in BT$ is called the weight of a.

Definition 1. *The weight function* $weight : BT \to \mathbf{N}$ *is defined inductively by*

$$weight(a) = \begin{cases} 1 & \text{if } a = \bullet \\ weight(a_L) + weight(a_R) & \text{if } a = (a_L a_R) \end{cases}$$

Binary trees that are distinct but that have identical weight are said to differ by their shape.

We define countably many operations on binary trees.

Definition 2. *The operation* $\overset{1}{\cdot}$ $: BT \times BT \to BT$ *is defined by* $a \overset{1}{\cdot} b = (ab)$. *For* $k \geq 2$, *the operations* $\overset{k}{\cdot}$ $: BT \times BT \to BT$ *are defined by*

$$a \overset{k}{\cdot} b = \begin{cases} a & \text{if } b = \bullet \\ (a \overset{k}{\cdot} b_L) \overset{k-1}{\cdot} (a \overset{k}{\cdot} b_R) & \text{if } b = (b_L b_R) \end{cases}$$

The integer k *is called the index of the operation* $\overset{k}{\cdot}$.

The operation $\overset{k+1}{\cdot}$ has a simple expression in terms of $\overset{k}{\cdot}$. Suppose a, b are binary trees and $n \geq 1$ is the weight of b. The binary tree $c = a \overset{k+1}{\cdot} b$ is then equal to the tree resulting from the $\overset{k}{\cdot}$-product of n trees a, in an order prescribed by the shape of b. For example

$$a \overset{k+1}{\cdot} (\bullet\bullet) = (a \overset{k}{\cdot} a)$$

and

$$a \overset{k+1}{\cdot} ((\bullet(\bullet\bullet))(\bullet\bullet)) = ((a \overset{k}{\cdot} (a \overset{k}{\cdot} a)) \overset{k}{\cdot} (a \overset{k}{\cdot} a))$$

Elementary examples of binary trees resulting from the operations $\overset{1}{\cdot}$, $\overset{2}{\cdot}$ and $\overset{3}{\cdot}$ are given in Fig. 2. The tree $a \overset{1}{\cdot} b$ is obtained by constructing the tree whose left and right subtrees are a and b respectively. The tree $a \overset{2}{\cdot} b$ is obtained by grafting

Fig. 2. Binary trees resulting from the operations of addition, multiplication and exponentiation

a copy of the tree a at the leaves of the tree b. No such simple geometrical description is available for $a \overset{3}{.} b$.

From the definition of the function $weight$ and of the operation $\overset{1}{.}$ we deduce that $weight(a \overset{1}{.} b) = weight(a) + weight(b)$. Since the operations of higher index are defined by successive repetition of $\overset{1}{.}$ the next result is easely obtained.

Proposition 1. *Let a, b be binary trees.*

1. $weight(a \overset{1}{.} b) = weight(a) + weight(b)$
2. $weight(a \overset{2}{.} b) = weight(a).weight(b)$
3. $weight(a \overset{3}{.} b) = weight(a)^{weight(b)}$

In the sequel the three first operations $\overset{1}{.}$, $\overset{2}{.}$ and $\overset{3}{.}$ on binary trees will be called addition, multiplication and exponentiation. There exist no natural number counterpart to $\overset{k}{.}$ for $k \geq 4$ because in this case the weight of $a \overset{k}{.} b$ depends on the weight of a and b but also on their respective shape.

3 Algebraic properties

The properties of the operations $\overset{1}{\cdot}$, $\overset{2}{\cdot}$, $\overset{3}{\cdot}$ for binary trees are similar to those of $+, ., \uparrow$ for natural numbers.

Theorem 1. *1. Let a be a binary tree and $k \geq 3$.*

1. $a \overset{2}{\cdot} \bullet = a = \bullet \overset{2}{\cdot} a$
2. $a \overset{k}{\cdot} \bullet = a$ *and* $\bullet \overset{k}{\cdot} a = \bullet$

2. Let a, b, c be binary trees and $k \geq 2$.

1. $a \overset{k}{\cdot} (b \overset{1}{\cdot} c) = (a \overset{k}{\cdot} b) \overset{k-1}{\cdot} (a \overset{k}{\cdot} c)$
2. $a \overset{k}{\cdot} (b \overset{2}{\cdot} c) = (a \overset{k}{\cdot} b) \overset{k}{\cdot} c$

3. Let a, b, c, d be binary trees.

1. *Left cancellation: If $a \overset{2}{\cdot} b = a \overset{2}{\cdot} c$ then $b = c$.*
2. *Right cancellation: If $a \overset{2}{\cdot} b = c \overset{2}{\cdot} b$ then $a = c$.*

Proof. 1. The equalities $a \overset{k}{\cdot} \bullet = a$ for $k \geq 2$ follow from the definition of $\overset{k}{\cdot}$. We prove $\bullet \overset{2}{\cdot} a = a$ by induction on a. The result is clearly true for $a = \bullet$ so let $a = (a_L a_R)$ and assume that $\bullet \overset{2}{\cdot} a_L = a_L$ and $\bullet \overset{2}{\cdot} a_R = a_R$. Then $a = a_L \overset{1}{\cdot} a_R = (\bullet \overset{2}{\cdot} a_L) \overset{1}{\cdot} (\bullet \overset{2}{\cdot} a_R) = \bullet \overset{2}{\cdot} (a_L \overset{1}{\cdot} a_R) = \bullet \overset{2}{\cdot} a$ and the theorem is proved. The equalities $\bullet \overset{k}{\cdot} a = a$ are proved by induction on a.

2. The first property is a rephrasement of the definition of $\overset{k}{\cdot}$. We prove the second property by induction on c. When $c = \bullet$ we have $a \overset{k}{\cdot} (b \overset{2}{\cdot} c) = a \overset{k}{\cdot} (b \overset{2}{\cdot} \bullet) = a \overset{k}{\cdot} b = (a \overset{k}{\cdot} b) \overset{k}{\cdot} \bullet = (a \overset{k}{\cdot} b) \overset{k}{\cdot} c$ so let $c = (c_L c_R)$ and assume that $a \overset{k}{\cdot} (b \overset{2}{\cdot} c_L) = (a \overset{k}{\cdot} b) \overset{k}{\cdot} c_L$ and $a \overset{k}{\cdot} (b \overset{2}{\cdot} c_R) = (a \overset{k}{\cdot} b) \overset{k}{\cdot} c_R$. Then by successive applications of the first property we obtain

$$\begin{aligned} (a \overset{k}{\cdot} (b \overset{2}{\cdot} c_L)) \overset{k-1}{\cdot} (a \overset{k}{\cdot} (b \overset{2}{\cdot} c_R)) &= ((a \overset{k}{\cdot} b) \overset{k}{\cdot} c_L) \overset{k-1}{\cdot} ((a \overset{k}{\cdot} b) \overset{k}{\cdot} c_R) \\ a \overset{k}{\cdot} ((b \overset{2}{\cdot} c_L) \overset{1}{\cdot} (b \overset{2}{\cdot} c_R)) &= (a \overset{k}{\cdot} b) \overset{k}{\cdot} (c_L \overset{1}{\cdot} c_R) \\ a \overset{k}{\cdot} (b \overset{2}{\cdot} (c_L \overset{1}{\cdot} c_R)) &= (a \overset{k}{\cdot} b) \overset{k}{\cdot} (c_L \overset{1}{\cdot} c_R) \end{aligned}$$

and thus

$$a \overset{k}{\cdot} (b \overset{2}{\cdot} c) = (a \overset{k}{\cdot} b) \overset{k}{\cdot} c.$$

3. We prove the right cancellation rule only, the proof of the left cancellation rule is similar. We proceed by induction on b. The result is clearly true for $b = \bullet$ so let $b = (b_L b_R)$ and assume that the result holds for b_L and b_R. If $a \overset{2}{\cdot} b = c \overset{2}{\cdot} b$, then $(a \overset{2}{\cdot} b_L) \overset{1}{\cdot} (a \overset{2}{\cdot} b_R) = (c \overset{2}{\cdot} b_L) \overset{1}{\cdot} (c \overset{2}{\cdot} b_R)$ and $a \overset{2}{\cdot} b_L = c \overset{2}{\cdot} b_L$. By the induction hypothesis we are lead to the conclusion. □

When evaluated with $k = 2$ the Properties 2.1 and 2.2 give

1. $a.(b + c) = a.b + a.c$
2. $a.(b.c) = (a.b).c$

whereas an evaluation with $k = 3$ gives

1. $a^{(b+c)} = a^b.a^c$

2. $a^{(b.c)} = (a^b)^c$

These four usual identities in $\mathbf{Z}$ have thus counterparts for binary trees. Central in the sequel is the fact that multiplication is associative.

Counterexamples

Commutativity. The operations $\overset{k}{\cdot}$ are non commutative. For $k = 1$ and $k = 2$ this can be seen from the examples $(\bullet\bullet)\overset{1}{\cdot}\bullet \neq \bullet\overset{1}{\cdot}(\bullet\bullet)$ and $(\bullet\bullet)\overset{2}{\cdot}(\bullet(\bullet\bullet)) \neq (\bullet(\bullet\bullet))\overset{2}{\cdot}(\bullet\bullet)$ whereas for the operations of higher index it suffices to notice that $a\overset{k}{\cdot}\bullet \neq \bullet\overset{k}{\cdot}a$ for any binary tree a different from $\bullet$.

Associativity. The operation $\overset{1}{\cdot}$ is not associative as is easily seen from the example $\bullet\overset{1}{\cdot}(\bullet\overset{1}{\cdot}\bullet) \neq (\bullet\overset{1}{\cdot}\bullet)\overset{1}{\cdot}\bullet$. The operations $\overset{k}{\cdot}$ for $k \geq 3$ are not associative either. For example, $(a\overset{k}{\cdot}\bullet)\overset{k}{\cdot}a \neq a\overset{k}{\cdot}(\bullet\overset{k}{\cdot}a)$ when $a \neq \bullet$. Thus, by Theorem 1, the only associative operation is $\overset{2}{\cdot}$.

Cancellation rule. The left cancellation rule does not hold for the operations $\overset{k}{\cdot}$ when $k \geq 3$. Indeed, for any binary tree a and $k \geq 3$ we have $\bullet\overset{k}{\cdot}a = \bullet$. Thus $\bullet\overset{k}{\cdot}a = \bullet\overset{k}{\cdot}b$ for all binary trees a and b which clearly shows that the left cancellation rule does not hold. In Sect. 5 we give necessary and sufficient conditions for the equality $a\overset{k}{\cdot}b = a\overset{k}{\cdot}d$.

The right cancellation rule is harder to analyse. It is known to hold for $k = 1$, $k = 2$, $k = 3$ and $k = 4$ but the general case is yet unsettled. We conjecture here that it holds for all $\overset{k}{\cdot}$ when $k \geq 1$.

4 Prime trees and prime decomposition

Definition 3. *A binary tree a is prime if it is different from the one node tree $\bullet$ and if $a = b\overset{2}{\cdot}c$ implies that $b = \bullet$ or $c = \bullet$. Trees that are not prime are composite.*

The weight of a product of binary trees is equal to the product of the weights. It is therefore clear that any binary tree whose weight is a prime number is automatically prime. The converse of this statement is not true. The binary tree $(\bullet(\bullet(\bullet\bullet)))$ has weight 4 and is a prime binary tree. It is easy to see that four of the five binary trees of weight four are prime and that, in general, a natural number n is prime if and only if all binary trees of weight n are prime.

In Table 1 we give, for the first values of $n \geq 1$, the number C_n of binary trees of weight n, the number I_n of composite trees of weight n, and the number P_n of prime trees of weight n. It is well-known that the number of binary trees of weigth n is equal to the nth Catalan number $C_n = (2n-2)!/(n!(n-1)!)$ (see [14], [12], [11] or [21]). No simple expression for I_n or P_n seems available. The sequence P_n does not appear in the recent encyclopedic list of integer sequences [22]. Ph. Flajolet has shown [9] that $T_n - I_n$ is equal to 0 if n is prime, is equal to $(C_p)^2$ if $n = p^2$ is the square of a prime, and is otherwise asymptotic to $2C_pC_{n/p}$ where p is the smallest prime factor of n.

Binary trees different from the one node tree can be decomposed into products of prime binary trees. The decomposition is unique up to, and including, the sequence in which the factors appear. We first need a lemma for proving this.

Lemma 1. *Let a_1, a_2, b_1, b_2 be binary trees such that $a_1 \overset{2}{\cdot} a_2 = b_1 \overset{2}{\cdot} b_2$. If a_1 and b_1 (or a_2 and b_2) are prime, then $a_1 = b_1$ and $a_2 = b_2$.*

Proof. If $a_1 = b_1$, then the left cancellation rule for multiplication shows that $a_2 = b_2$. We proceed by induction on a_2 to prove that $a_1 = b_1$. If $a_2 = \bullet$ then $a_1 = b_1 \overset{2}{\cdot} b_2$. Since a_1 is prime and b_1 is different from $\bullet$ we must have $b_2 = \bullet$ and thus $a_1 = b_1$. Assume now that $a_1 \overset{2}{\cdot} a_2 = b_1 \overset{2}{\cdot} b_2$ and $a_2 = (a_{2L} a_{2R})$. If $b_2 = \bullet$, then $a_1 \overset{2}{\cdot} a_2 = b_1$ and the theorem is proved, so assume $b_2 = (b_{2L} b_{2R})$. We have $(a_1 \overset{2}{\cdot} a_{2L}) \overset{1}{\cdot} (a_1 \overset{2}{\cdot} a_{2R}) = (b_1 \overset{2}{\cdot} b_{2L}) \overset{1}{\cdot} (b_1 \overset{2}{\cdot} b_{2R})$. By the cancellation rule for addition and the induction hypothesis we then conclude $a_1 = b_1$ as requested. □

It is now easy to show:

Theorem 2 (Existence and uniqueness of prime decomposition). *Let a be a binary tree different from $\bullet$. Then $a = a_1 \overset{2}{\cdot} a_2 \overset{2}{\cdot} \cdots \overset{2}{\cdot} a_n$ for some $n \geq 1$ and some prime binary trees a_i. If $a = b_1 \overset{2}{\cdot} b_2 \overset{2}{\cdot} \cdots \overset{2}{\cdot} b_m$ is another such decomposition. Then $n = m$ and $a_i = b_i$ for $i = 1, ..., n$.*

Proof. Let a be a binary tree different from $\bullet$. If a is prime, then $n = 1$ and $a_1 = a$ is the decomposition sought. If a is composite, there exists a_1 and a_2 different from $\bullet$ and such that $a = a_1 \overset{2}{\cdot} a_2$. The factors can then be further decomposed until prime factors are reached. Since the weight of the factors are positive and strictly decreasing, the procedure must end after a finite number of steps. Thus $a = a_1 \overset{2}{\cdot} a_2 \overset{2}{\cdot} \cdots \overset{2}{\cdot} a_n$ for some $n \geq 1$ and a_i prime binary trees.

Assume now that $a = b_1 \overset{2}{\cdot} b_2 \overset{2}{\cdot} \cdots \overset{2}{\cdot} b_m$ is another such decomposition. By the lemma we must have $a_1 = b_1$ and $a_2 \overset{2}{\cdot} a_3 \overset{2}{\cdot} \cdots a_n = b_2 \overset{2}{\cdot} b_3 \overset{2}{\cdot} \cdots b_m$. But then by successive repetition of the same argument we are lead to the conclusion. □

The decomposition given in the theorem is called a prime decomposition. The ith factor in the decomposition is uniquely determined and is the ith factor of a. We can characterise the binary trees whose sum is prime.

Theorem 3. *Let a, b be binary trees different from the one node tree $\bullet$. Then $a \overset{1}{\cdot} b$ is prime if and only if the first factors of a and b are distinct.*

Proof. (Necessity) Let the first factors of a and b be distinct and assume by contradiction that $a \overset{1}{\cdot} b$ is not prime. Then $a = c_1 \overset{2}{\cdot} c_2$ for some c_1 and c_2 different from $\bullet$. Since c_2 is different from $\bullet$ we may write $c_2 = c_{2L} \overset{1}{\cdot} c_{2R}$. Hence $a \overset{1}{\cdot} b = (c_1 \overset{2}{\cdot} c_{2L}) \overset{1}{\cdot} (c_1 \overset{2}{\cdot} c_{2R})$. By the cancellation rule this leads to $a = c_1 \overset{2}{\cdot} c_{2L}$ and $b = c_1 \overset{2}{\cdot} c_{2R}$. But since c_1 is different from $\bullet$ these last identies show that the first factors of a and b are identical and a contradiction is thus attained.
(Sufficiency) Let a, b be distinct from $\bullet$ and assume by contradiction that $a = c \overset{2}{\cdot} a'$ and $b = c \overset{2}{\cdot} b'$ for some c different from $\bullet$. Then $a \overset{1}{\cdot} b = (c \overset{2}{\cdot} a') \overset{1}{\cdot} (c \overset{2}{\cdot} b') = c \overset{2}{\cdot} (a' \overset{1}{\cdot} b') = c \overset{2}{\cdot} c'$ with c and c' different from $\bullet$. A contradiction is achieved and the theorem is proved. □

5 The operations $\overset{k}{\cdot}$ for $k \geq 3$

Multiplication of binary trees is associative. The result of $a \overset{3}{\cdot} b$ therefore depends on a and on the weight of b but not otherwise on the shape of b.

Table 1. Number of binary trees, composite trees and prime binary trees of given weight

weight n	C_n	I_n	P_n
1	1	1	0
2	1	0	1
3	2	0	2
4	5	1	4
5	14	0	14
6	42	4	38
7	132	0	132
8	429	9	420
9	1430	4	1426
10	4862	28	4834
11	16796	0	16796
12	58786	98	58688

Proposition 2. *Let a, b_1, b_2 be binary trees and assume that $weight(b_1) = weight(b_2)$. Then $a \overset{3}{\cdot} b_1 = a \overset{3}{\cdot} b_2$.*

We use this result for introducing a new notation. Let $n \geq 1$ and $a \in BT$. By $a \overset{3}{\cdot} n$ we mean the binary tree $a \overset{3}{\cdot} b$ where b is any binary tree of weight n. A similar construction is possible for operations of higher index.

Definition 4. *Let $k \geq 3$. The operations $\diamondsuit_k : BT \times BT \to \mathbf{N}$ are defined inductively by*

1. $a \diamondsuit_3 b = weight(b)$
2. $$a \diamondsuit_k b = \begin{cases} 1 & \text{if } b = \bullet \\ (a \diamondsuit_k b_L).((a \overset{k}{\cdot} b_L) \diamondsuit_{k-1} (a \overset{k}{\cdot} b_R)) & \text{if } b = (b_L b_R) \end{cases}$$

With this purpose-built definition we have:

Theorem 4. *Let a, b be binary trees and $k \geq 3$. Then $a \overset{k}{\cdot} b = a \overset{3}{\cdot} (a \diamondsuit_k b)$.*

Proof. We proceed by induction on k. For $k = 3$ the result is contained in Proposition 2. We assume that the result holds for $\overset{k-1}{\cdot}$ and show, by induction on b, that it also holds for $\overset{k}{\cdot}$. If $b = \bullet$, then $a \overset{k}{\cdot} \bullet = a = a \overset{3}{\cdot} \bullet = a \overset{3}{\cdot} 1 = a \overset{3}{\cdot} (a \diamondsuit_k \bullet)$ so let $b = (b_L b_R)$ and assume that $a \overset{k}{\cdot} b_L = a \overset{3}{\cdot} (a \diamondsuit_k b_L)$ and $a \overset{k}{\cdot} b_R = a \overset{3}{\cdot} (a \diamondsuit_k b_R)$. We have

$$\begin{aligned} a \overset{k}{\cdot} b &= a \overset{k}{\cdot} (b_L b_R) \\ &= (a \overset{k}{\cdot} b_L) \overset{k-1}{\cdot} (a \overset{k}{\cdot} b_R) \\ &= (a \overset{k}{\cdot} b_L) \overset{3}{\cdot} ((a \overset{k}{\cdot} b_L) \diamondsuit_{k-1} (a \overset{k}{\cdot} b_R)) \\ &= (a \overset{3}{\cdot} (a \diamondsuit_k b_L)) \overset{3}{\cdot} ((a \overset{k}{\cdot} b_L) \diamondsuit_{k-1} (a \overset{k}{\cdot} b_R)) \\ &= a \overset{3}{\cdot} ((a \diamondsuit_k b_L)((a \overset{k}{\cdot} b_L) \diamondsuit_{k-1} (a \overset{k}{\cdot} b_R))) \\ &= a \overset{3}{\cdot} (a \diamondsuit_k b) \end{aligned}$$

and the theorem is proved. □

Cancellation rules for $\overset{1}{\cdot}$ and $\overset{2}{\cdot}$ were analysed in Sect. 3. We have shown that the left cancellation rule does not hold for $k \geq 3$ since, for example, $\bullet \overset{k}{\cdot} a = \bullet \overset{k}{\cdot} b$ for any binary trees a and b. With the help of Theorem 4 this observation can now be made more precise.

Theorem 5. *Let a, b, d be binary trees different from $\bullet$ and $k \geq 3$. Then $a \overset{k}{\cdot} b = a \overset{k}{\cdot} d$ if and only if $a \diamondsuit_k b = a \diamondsuit_k d$.*

Proof. (Necessity) By Theorem 4 we know that $a \overset{k}{\cdot} b = a \overset{3}{\cdot} (a \diamondsuit_k b)$ and $a \overset{k}{\cdot} d = a \overset{3}{\cdot} (a \diamondsuit_k d)$. Hence $a \overset{3}{\cdot} (a \diamondsuit_k b) = a \overset{3}{\cdot} (a \diamondsuit_k d)$. But then the prime decomposition theorem leads to $a \diamondsuit_k b = a \diamondsuit_k d$ as requested.
(Sufficiency) This part is trivial. If $a \diamondsuit_k b = a \diamondsuit_k d$, then $a \overset{k}{\cdot} b = a \overset{3}{\cdot} (a \diamondsuit_k b) = a \overset{3}{\cdot} (a \diamondsuit_k d) = a \overset{k}{\cdot} d$. □

Corollary 1. *Let a, b, d be binary trees different from $\bullet$. Then $a \overset{3}{\cdot} b = a \overset{3}{\cdot} d$ if and only if $weight(b) = weight(d)$.*

Some of the algebraic properties of the operations $\overset{k}{\cdot}$ are summarized in Table 2

Table 2. Algebraic properties of the operations $\overset{k}{\cdot}$

	Commutativity	Associativity	neutral	cancellation rules
$\overset{1}{\cdot}$	no	no	no	$a \overset{1}{\cdot} b = c \overset{1}{\cdot} d \Leftrightarrow a = c$ and $b = d$
$\overset{2}{\cdot}$	no	yes	$a \overset{2}{\cdot} \bullet = a$ $\bullet \overset{2}{\cdot} a = a$	$a \overset{2}{\cdot} b = c \overset{2}{\cdot} b \Leftrightarrow a = c$ $a \overset{2}{\cdot} b = a \overset{2}{\cdot} d \Leftrightarrow b = d$
$\overset{3}{\cdot}$	no	no	$a \overset{3}{\cdot} \bullet = a$ $\bullet \overset{3}{\cdot} a = \bullet$	$a \overset{3}{\cdot} b = c \overset{3}{\cdot} b \Leftrightarrow a = c$ $a \overset{3}{\cdot} b = a \overset{3}{\cdot} d \Leftrightarrow weight(b) = weight(d)$
$\overset{4}{\cdot}$	no	no	$a \overset{4}{\cdot} \bullet = a$ $\bullet \overset{4}{\cdot} a = \bullet$	$a \overset{4}{\cdot} b = c \overset{4}{\cdot} b \Leftrightarrow a = c$ $a \overset{4}{\cdot} b = a \overset{4}{\cdot} d \Leftrightarrow a \diamondsuit_4 b = a \diamondsuit_4 d$
$\overset{k}{\cdot}$ ($k \geq 5$)	no	no	$a \overset{k}{\cdot} \bullet = a$ $\bullet \overset{k}{\cdot} a = \bullet$	Conjecture: $a \overset{k}{\cdot} b = c \overset{k}{\cdot} b \Leftrightarrow a = c$ $a \overset{k}{\cdot} b = a \overset{k}{\cdot} d \Leftrightarrow a \diamondsuit_k b = a \diamondsuit_k d$

6 Valuations

A valuation μ is a function defined on binary trees and taking values in $\mathbf{Z}$. Operations on integers can be used in a natural way to define valuations.

Definition 5. *Associated to $\Delta : \mathbf{Z} \times \mathbf{Z} \to \mathbf{Z}$ and $e \in \mathbf{Z}$ is a valuation $\mu : BT \to \mathbf{Z}$ defined inductively by*

$$\mu(a) = \begin{cases} e & \text{if } a = \bullet \\ \mu(a_L) \Delta \mu(a_R) & \text{if } a = (a_L a_R) \end{cases}$$

Valuations that can be obtained in this way are called inductive.

We immediatly recognise that the weight function is an inductive valuation obtained by setting $a \Delta b = a + b$ and $e = 1$. Other inductive valuations are given next.

Maximal height. If we set $a\Delta b = 1+\max(a,b)$ and $e=0$ the valuation obtained gives the maximal height of all external nodes. This quantity is referred to as the height of the tree and the corresponding valuation is denoted by *height*.

Minimal height. The valuation obtained with $a\Delta b = 1+\min(a,b)$ and $e=0$ gives the minimal height of all external nodes and is denoted *minheight*.

Strahler number. The valuation obtained with $a\Delta b = [a=b]+\max(a,b)$ and $e=0$ appear in various contexts (the expression $[a=b]$ outputs 1 when $a=b$ and outputs 0 otherwise). In [8] it is called the "register function" and is used to calculate the minimal number of registers needed to evaluate an arithmetic expression (see also [16]). The same function is known in hydrology as the Horton-Strahler function and is used to describe characteristics of river flows (see [23], [24] and references cited therein). The Strahler number of a tree is equal to the height of the maximal complete tree that can be embedded in the tree [8]. We denote this valuation by *Strahler*.

2-*bud.* The valuation obtained with $a\Delta b = [a=b]+\min(a,b)$ and $e=0$ (note the similarity with the definition of the Strahler number) has, to our knowledge, never been analysed. We denote this function by $2-bud$. The 2-bud of a binary tree a can be shown equal the largest $n\geq 0$ for which the nth first factors of a are equal to $(\bullet\bullet)$. The $2-bud$ of a binary tree a is thus equal to the largest n for which $a=((\bullet\bullet)^{3}_{\cdot}n)^{2}_{\cdot}b$ for some tree b. Because of the grafting interpretation of the operation $^{2}_{\cdot}$ this number corresponds also to the height n of the largest complete binary tree $(\bullet\bullet)^{3}_{\cdot}n$ that appear everywhere on the boundary of a.

Boolean valuation. The valuation obtained with $a\Delta b=[a=b]$ and $e=0$ outputs 1 if the weight of the tree is even and outputs 0 if it is odd.

In Table 3 we list some possible choices of operation Δ and their resulting valuations. Several of these inductive valuations have remarkable properties with respect to $^{k}_{\cdot}$.

Theorem 6. *Assume $\Delta:\mathbf{Z}\times\mathbf{Z}\to\mathbf{Z}$, $e=0$, and let μ be the inductive valuation associated to Δ and e. If $a+(b\Delta c)=(a+b)\Delta(a+c)$ for all $a,b,c\in\mathbf{Z}$, then*

1. $\mu(a^{2}_{\cdot}b)=\mu(a)+\mu(b)$
2. $\mu(a^{3}_{\cdot}b)=\mu(a).\mu(b)$
3. $\mu(a^{4}_{\cdot}b)=\mu(a)^{\mu(b)}$

Proof. We prove the first identity by induction on b. For $b=\bullet$ we have $\mu(a^{2}_{\cdot}\bullet)=\mu(a)=\mu(a)+\mu(\bullet)$ so let $b=(b_Lb_R)$ and assume that $\mu(a^{2}_{\cdot}b_L)=\mu(a)+\mu(b_L)$ and $\mu(a^{2}_{\cdot}b_R)=\mu(a)+\mu(b_R)$. We have then

$$\begin{aligned}
\mu(a^{2}_{\cdot}b) &= \mu(a^{2}_{\cdot}(b_L{}^{1}_{\cdot}b_R))\\
&= \mu((a^{2}_{\cdot}b_L)^{1}_{\cdot}(a^{2}_{\cdot}b_R))\\
&= \mu(a^{2}_{\cdot}b_L)\Delta\mu(a^{2}_{\cdot}b_R)\\
&= (\mu(a)+\mu(b_L))\Delta(\mu(a)+\mu(b_R))\\
&= \mu(a)+(\mu(b_L)\Delta\mu(b_R))\\
&= \mu(a)+\mu(b)
\end{aligned}$$

The other identities are similarly proved. □

Corollary 2. *Let μ be one of the valuations height, minheight, Strahler or $2-bud$. Then*

1. $\mu(a \overset{2}{\cdot} b) = \mu(a) + \mu(b)$
2. $\mu(a \overset{3}{\cdot} b) = \mu(a).\mu(b)$
3. $\mu(a \overset{4}{\cdot} b) = \mu(a)^{\mu(b)}$

Proof. The corresponding pairs (Δ, e) satisfy the conditions of Theorem 6. □

Table 3. Some inductive valuations

$a \Delta b$	e	resulting valuation
$a + b$	1	*weight*
$1 + \max(a, b)$	0	*height*
$1 + \min(a, b)$	0	*minheight*
$[a = b] + \max(a, b)$	0	*strahler*
$[a = b] + \min(a, b)$	0	$2 - bud$
$[a = b]$	0	1 if *weight* is even 0 if *weight* is odd

7 Infinite trees

The operations $\overset{k}{\cdot}$ introduced for finite binary trees can be extended to infinite binary trees. For convenience we shall look at infinite trees as languages over 2-letter alphabets.

Let Σ be a finite alphabet and $L_1, L_2 \subseteq \Sigma^*$ be two languages over Σ (for definitions see [19]). The product of L_1 and L_2 is the language $L_1 \cdot L_2 = \{x_1 \cdot x_2 : x_1 \in L_1, x_2 \in L_2\}$. Given $x \in \Sigma^*$, the language $x \cdot L$ and the residual $x^{-1} \cdot L$ of L by x are defined by $x \cdot L = \{x\} \cdot L$ and $x^{-1} \cdot L = \{y \in \Sigma^* : x \cdot y \in L\}$, respectively. If $n \geq 0$, the truncation of L at size n is the language $\lceil L \rceil_n = \{x \in L : |x| \leq n\}$ where $|x|$ is the length of x. For a language L and $n \geq 0$ we define $L^0 = \{\omega\}$ (ω denotes the empty word), $L^{n+1} = L^n \cdot L$ and $L^* = \bigcup_{i=0}^{\infty} L^i$. Finally, a language L over Σ is factorial if $x, v \in \Sigma^*$ and $x \cdot v \in L$ implies that $x \in L$. Factorial languages always contain ω unless they are empty.

Proposition 3. *Let $L, L_1, L_2, \ldots$ be factorial languages over Σ, $x \in \Sigma^*$ and $n \geq 0$. The languages $x^{-1} \cdot L, \lceil L \rceil_n, L = \bigcup_{i=0}^{\infty} L_i, L = \bigcap_{i=0}^{\infty} L_i, L_1 \cdot L_2$ and L^* are factorial.*

A binary tree is entirely specified by its set of internal nodes. Any internal node can be reached by starting from the root and specifying the finite sequence of left and right movements needed to reach it. Let us denote these movements by a and b respectively. A node can thus be seen as a word over the alphabet $\Sigma = \{a, b\}$. A finite (infinite) tree will therefore have a representation as a finite (infinite) language over Σ. To the one node binary tree $\bullet$ corresponds the empty language $L = \emptyset$, the tree $(\bullet\bullet)$ is represented by $L = \{\omega\}$ and the two binary trees

of weight 3 have $\{\omega, a\}$ and $\{\omega, b\}$ as corresponding languages. It is easy to see that languages generated by binary trees are factorial. The converse is also true: Any factorial language over a 2-letter alphabet can be seen as a representation of a particular binary tree, the corresponding binary tree is infinite when the language is. We denote by *FL* the set of factorial languages and by *FFL* the set of finite factorial languages over the two letter alphabet $\{a, b\}$. There is an obvious bijection between *FFL* and *BT*.

Definition 6. *The operation* $\overset{1}{:}$: $FFL \times FFL \to FFL$ *is defined by* $L_1 \overset{1}{:} L_2 = \{\omega\} \cup a \cdot L_1 \cup b \cdot L_2$. *For* $k \geq 2$, *the operations* $\overset{k}{:}$: $FFL \times FFL \to FFL$ *are defined inductively by*

$$L_1 \overset{k}{:} L_2 = \begin{cases} L_1 & \text{if } L_2 = \emptyset \\ (L_1 \overset{k}{:} a^{-1}L_2) \overset{k-1}{:} (L_1 \overset{k}{:} b^{-1}L_2) & \text{if } L_2 \neq \emptyset \end{cases}$$

We now remove the finiteness condition on the languages L_1 *and* L_2 *and define, for finite or infinite languages, the operations* $\overset{k}{:}$ $(k \geq 0)$ *by*

$$L_2 = L_2 \cdot L_1 \; 2. \qquad \overset{3}{:} \quad L_2 = \begin{cases} L_1^n & \text{is finite some } n \geq 1 \text{ if } L_2 \\ L_1 \quad L_1^* & \text{if } L_2 \text{ is infinite} \end{cases}$$

Proof. We prove the result for L_2 finite by induction on L_1. The result is clearly true for $L_1 = \emptyset$; so let $L_2 = L_2' \overset{1}{\cdot} L_2''$ and assume that $L_1 \overset{3}{\cdot} L_2' = L_1^{n'}$ and $L_1 \overset{3}{\cdot} L_2'' = L_1^{n''}$ for some $n', n'' \geq 0$. We have then

$$\begin{aligned} L_1 \overset{3}{\cdot} L_2 &= L_1 \overset{3}{\cdot} (L_2' \overset{1}{\cdot} L_2'') \\ &= (L_1 \overset{3}{\cdot} L_2') \overset{2}{\cdot} (L_1 \overset{3}{\cdot} L_2'') \\ &= L_1^{n'} \cdot L_1^{n''} \\ &= L_1^{n'+n''} \end{aligned}$$

as requested. Assume now that L_2 is infinite. We show that the languages L_1^* and $L_1 \overset{3}{\cdot} L_2$ coincide. Indeed, if $x \in \Sigma^*$ and $x \in L_1 \overset{3}{\cdot} L_2$, then $x \in (\lceil L_1 \rceil_i \overset{3}{\cdot} \lceil L_2 \rceil_i)$ for some i. But then by the finite case there exist some n for which $x \in (\lceil L_1 \rceil_i)^n \subseteq L_1^n \subseteq L_1^*$. Assume now that $x \in L_1^*$. Then $x \in L_1^n$ for some n. But then $x \in L_1 \overset{3}{\cdot} \lceil L_2 \rceil_i$ for some i and thus $x \in L_1 \overset{3}{\cdot} L_2$ as requested. □

Remark. The condition in Definition 6 that the languages be factorial is not essential. The operations $\overset{k}{\cdot}$ can equally be defined for arbitrary language defined over 2-letter alphabets.

Notes added in proof

1. The right cancellation rule conjectured at the end of Sect. 3 has recently been proved by Philippe Duchon ("Some new results on a family of operations for binary trees", submitted, 1997).
2. The results contained in this article have been presented at a seminar at INRIA-Paris in October 1995. An abstract of the seminar appears in "Algorithms Seminars 1994–1995", INRIA Technical Report 2669, Bruno Salvy (ed.), 1995.

Acknowledgements. We wish to express our sincere thanks to anonymous reviewers for their helpful remarks. We are also thankful to Professor D. Welsh, Oxford University, Professor Ph. Delsarte, University of Louvain, Professor Ph. Flajolet, INRIA, Dr S. Gaubert, INRIA, Professor M. Nivat, University of Paris, and Professor D. Knuth, Stanford University for commenting a first version of this paper.

References

1. G. R. Blakley, I. Borosh: Knuth's iterated powers, Adv. in Math. **34** (1979) 109-136.
2. V. Blondel: Structured numbers, Technical Report TRITA/MAT-94-31, Department of mathematics, Royal Institute of Technology, S-10044 Stockholm (1994).
3. V. Blondel: Operations on structured numbers, Research report 2464, INRIA BP 105, F-78153 Le Chesnay Cedex (1995).
4. V. Blondel: Une famille d'opérations sur les arbres binaires, C. R. Acad. Sci. Paris, Série 1 **321** (1995) 491-494.
5. R. Costa: Shape identities in genetic algebras, Lin. Algebra and its Appl. **214** (1995) 119-131.
6. I. M. Etherington: On non-associative combinations, Proc. Royal Soc. of Edinburgh **58-59** (1937-1938) 153-162.
7. I. M. Etherington: Genetic algebras, Proc. Royal Soc. of Edinburgh **58-59** (1937-1938) 242-258.
8. Ph. Flajolet, J. C. Raoult, J. Vuillemin: The number of registers required for evaluating arithmetic expressions, Theoret. Comp. Sc. **9** (1979) 99-125.
9. Ph. Flajolet: Analyse d'algorithmes de manipulation d'arbres et de fichiers, Cahiers BURO, **30-35** (1981) 1-209.
10. Ph. Flajolet: Personnal communication, 1996.

11. M. Gardner: Mathematical games: Catalan numbers, Sci. Amer. (1976) 120-122.
12. H. W. Gould: Research bibliography of two special number sequences, Mathematica Monongaliae **12** (1971).
13. J. W. Grossman, R. Z. Zeitman: An inherently iterative computation of Ackermann's function, Theoret. Comp. Science **57** (1988) 327-330.
14. P. Hilton, J. Pedersen: Catalan numbers and their various uses, in: W. Lederman, ed., Handbook of applicable mathematics. John Wiley, Chichester, 1990
15. J. H. Holgate: Population algebras, J. Royal Statist. Soc. Ser. B **43** (1981) 1-19.
16. R. Kemp: The average number of register needed to evaluate a binary tree optimally, Acta Informatica **11** (1979) 363-372.
17. D. E. Knuth: The art of computer programming, Vol. I and II. Addison-Wesley, Reading, MA, 1968.
18. D. E. Knuth: Mathematics and computer science: coping with finiteness, Science **194** (1976) 1235-1242.
19. J. van Leeuwen: Handbook of theoretical computer sciences, Vol. A and B. North-Holland, Amsterdam, 1990
20. C. Pair, A. Quere: Définition et étude des bilangages régulier, Information and Control **13** (1968) 565-593.
21. R. Sedgewick, Ph. Flajolet: An introduction to the analysis of algorithms. Addison-Wesley, Reading, MA, 1996
22. N. J. A. Sloane, S. Plouffe: The encyclopedia of integer sequences. Academic Press, New York, 1995
23. A. N. Strahler: Hypsomic analysis of erosional topography, Bulletin Geological Society of America **63** (1952) 1117-1142.
24. X. G. Viennot: Trees, in: M. Lothaire, ed., Mots. Mélanges offerts à M.-P. Schützenberger. Hermes, Paris, 1990.