Operations on structured numbers

Vincent Blondel

▶ **To cite this version:**

Vincent Blondel. Operations on structured numbers. [Research Report] RR-2464, INRIA. 1995. inria-00074211

HAL Id: inria-00074211
https://inria.hal.science/inria-00074211

Submitted on 24 May 2006

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

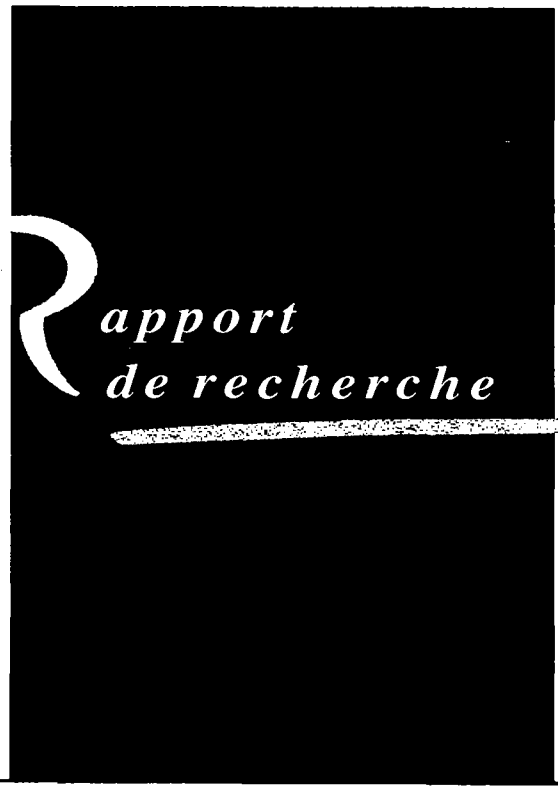
INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Operations on structured numbers

Vincent BLONDEL

N° 2464
Janvier 1995

PROGRAMME 5



Les rapports de recherche de l'INRIA
sont disponibles en format postscript sous
ftp.inria.fr (192.93.2.54)

si vous n'avez pas d'accès ftp
la forme papier peut être commandée par mail :
e-mail : dif.gesdif@inria.fr
(n'oubliez pas de mentionner votre adresse postale).

par courrier :
Centre de Diffusion
INRIA
BP 105 - 78153 Le Chesnay Cedex (FRANCE)

INRIA research reports
are available in postscript format
ftp.inria.fr (192.93.2.54)

if you haven't access by ftp
we recommend ordering them by e-mail :
e-mail : dif.gesdif@inria.fr
(don't forget to mention your postal address).

by mail :
Centre de Diffusion
INRIA
BP 105 - 78153 Le Chesnay Cedex (FRANCE)

Operations on structured numbers

Opérations sur les nombres structurés

Vincent Blondel

INRIA Rocquencourt, Domaine de Voluceau BP105, F-78153 Le Chesnay Cedex, France
vincent.blondel@inria.fr

Abstract. We propose the introduction of the concept of *structured number* as a possible extension of the positive integer number system. Structured numbers carry not only a quantitative information but also a structured information. The usual operations of addition, multiplication and exponentiation are defined for structured number and are presented as the three first members of a countable infinite number of operations on structured numbers. Structured numbers are isomorphic to binary trees; all results presented in the report equally apply to binary trees.

Résumé: Nous proposons l'introduction du concept de *nombre structuré* comme une extension possible des nombres naturels. Les nombres structurés véhiculent une information quantitative ainsi qu'une information de structure. Les opérations habituelles d'addition, de multiplication et d'exponentiation sont définies pour les nombres structurés et sont présentées comme les trois premiers membres d'une famille dénombrable d'opérations. Les nombres structurés sont isomorphes aux arbres binaires; tous les résutats présentés dans le rapport s'appliquent également à ces derniers.

1 Introduction

The product of two positive integers a and b is usually defined as the sum of b factors each equal to a

$$a.b = \underbrace{a + a + ... + a}_{b}$$

The b^{th} exponent of a, which we denote by $a \uparrow b$, is similarly defined as the product of b factors each equal to a

$$a \uparrow b = \underbrace{a.a.a....a}_{b}$$

The process of getting new operations by repeating old ones ends with exponentiation because this last operation is not associative. The definition

$$a \uparrow\uparrow b = \underbrace{a \uparrow a \uparrow a \uparrow ... \uparrow a}_{b} \tag{1}$$

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 (binary trees are rooted trees whose nodes have either no sons or have one left and one right son, see [11]). The number of external nodes (leaves) of a binary tree can be used to quantify the number of times the operation needs to be repeated, and the structure of the tree is then used to specify the order in which the operations need to be performed.
In this paper we think of binary trees as "structured numbers" (see [1]) and 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 of $\overset{1}{\cdot}$. This operation is not associative. The second operation $\overset{2}{\cdot}$ is then defined as follows: From the binary trees a and b we construct a new binary tree $c = a \overset{2}{\cdot} b$ 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. Thus, we eventually obtain countably many internal operations ($\overset{k}{\cdot}$ for $k \geq 1$) on binary trees; the operations being obtained by the recursion

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

in which b dictates the order in which the operations need to be performed.

The number of external nodes of the binary tree resulting from the operation $\overset{1}{\cdot}$, $\overset{2}{\cdot}$ and $\overset{3}{\cdot}$ can be proved equal to the sum, product and exponentiation of the number of external nodes of the operands. These three operations will therefore be thought of as binary trees counterparts

of the usual operations of addition, multiplication and exponentiation. The operations $\overset{k}{:}$ 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 the resulting tree.

The object of this paper is to study some of the properties of the operations $\overset{k}{:}$ described above and formalized in the second section of the paper. In Section 3 the operations are shown to satisfy a wide range of algebraic properties. In particular we show that they satisfy distributive properties that generalize known properties for integers and that they can be decomposed – in a unique way – as products of prime binary trees. In the third section we analyse various integer valued functions associated to trees such as the weight (size), height and Strahler number, and show how these functions behave with respect to $\overset{k}{:}$-products of binary trees. In a final section we argue that the notions introduced in the paper for finite binary trees can be generalized for infinite trees. This will be achieved by formalizing binary trees by means of factorial languages.

Different authors have proposed the introduction of operations of "superexponentiation" on integers. Knuth's recursive definition [12] is

$$a \uparrow\uparrow b = \underbrace{a \uparrow (a \uparrow (a \uparrow (a \ldots \uparrow a) \ldots))}_{b}$$

$$a \uparrow\uparrow\uparrow b = \underbrace{a \uparrow\uparrow (a \uparrow\uparrow (a \uparrow\uparrow (a \ldots \uparrow\uparrow a) \ldots))}_{b}$$

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

Such a definition has the disadvantage of making an arbitrary choice on how the non-associative operations are performed. As a result, these operations have only poor algebraic properties. They are defined in [12] to illustrate properties of large numbers rather than for algebraic analysis (we quote from [12]: "It seems to me that the magnitude of $10 \uparrow\uparrow\uparrow\uparrow 3$ is so large as to be beyond human comprehension").

Operations on graphs, trees and binary trees constitute a classical object of study in the theoretical computer science community [15, 11]. The number of operations defined on such objects is usually limited to two. We have found no reference that uses the particular structure of binary trees as a mean for defining countably many operations. The same remark applies to internal operations on languages [14].

The contribution that is probably closest to ours is the "arithmetic of shapes" developped by I. M. Etherington half a century ago [2, 3, 4]. Motivated by applications in genetics, Etherington has introduced in [3] the operations on binary trees that we have denoted by $\overset{1}{:}$, $\overset{2}{:}$ and $\overset{3}{:}$ and has founded a tradition in genetic algebra [4]. Other references that introduce the operations $\overset{1}{:}$, $\overset{2}{:}$ and $\overset{3}{:}$ include [16]. The operations $\overset{k}{:}$ for $k \geq 4$ were however never defined in these contexts.

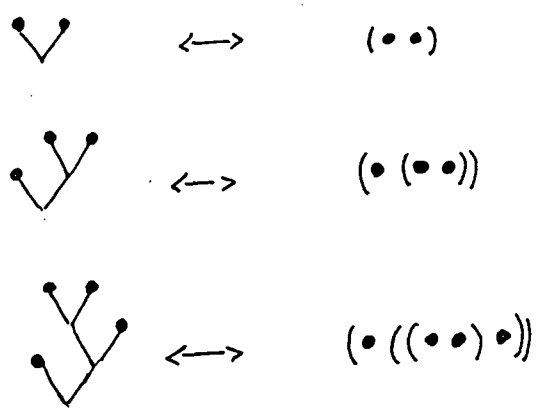


Figure 1: Binary trees and their corresponding parenthesed expansions.

2 The operations

By a binary tree we mean an ordered rooted tree whose nodes have either no sons or have one left and one right son [11]. The tree is ordered in that we make a distinction between left and right subtrees. The nodes that have no sons are called external and those with two sons are called internal. In a binary tree the number of internal nodes is always one unit less than the number of external nodes. The height of a node is equal to the length of its path to the root. The height of a tree is equal to the maximal height of its nodes.

For notational convenience we represent binary trees by means of horizontal parenthesed expressions (see Figure 1). $\bullet$ is used to represent the trivial tree consisting of a single node. If a and b are binary trees, then (ab) denotes the binary tree whose left and right subtrees are equal to a and b respectively. Every binary tree a is thus either equal to $\bullet$ or can be decomposed as $a = (a_L a_R)$ where a_L and a_R are binary trees and are uniquely determined. Our notation has the advantage to be one-dimensional.

Let BT denote the set of binary trees. The number of external nodes (leaves) of $a \in BT$ is called the weight of a (this definition is slightly different from the one given in [11] where internal – rather than external – nodes are counted). The function that maps binary trees to their weight is called the weight function.

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 now 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 let n be equal to 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 factors a in an order prescribed by the shape of b. For example

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

and

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

Simple examples of binary trees resulting from the operations $\overset{1}{\cdot}$, $\overset{2}{\cdot}$ and $\overset{3}{\cdot}$ are given in Figure 2.

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

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

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

In the sequel the three first operations $\overset{1}{\cdot}$, $\overset{2}{\cdot}$ and $\overset{3}{\cdot}$ on binary trees will be called addition, multiplication and exponentiation. We shall sometimes use $+, .$ and the superscript notation to denote these three operations on binary trees. There exist no natural number counterpart to $\overset{k}{\cdot}$ for $k \geq 4$ because in this case the weight of $a \overset{k}{\cdot} b$ depends not only on the weight of a and b but also on their respective shape.

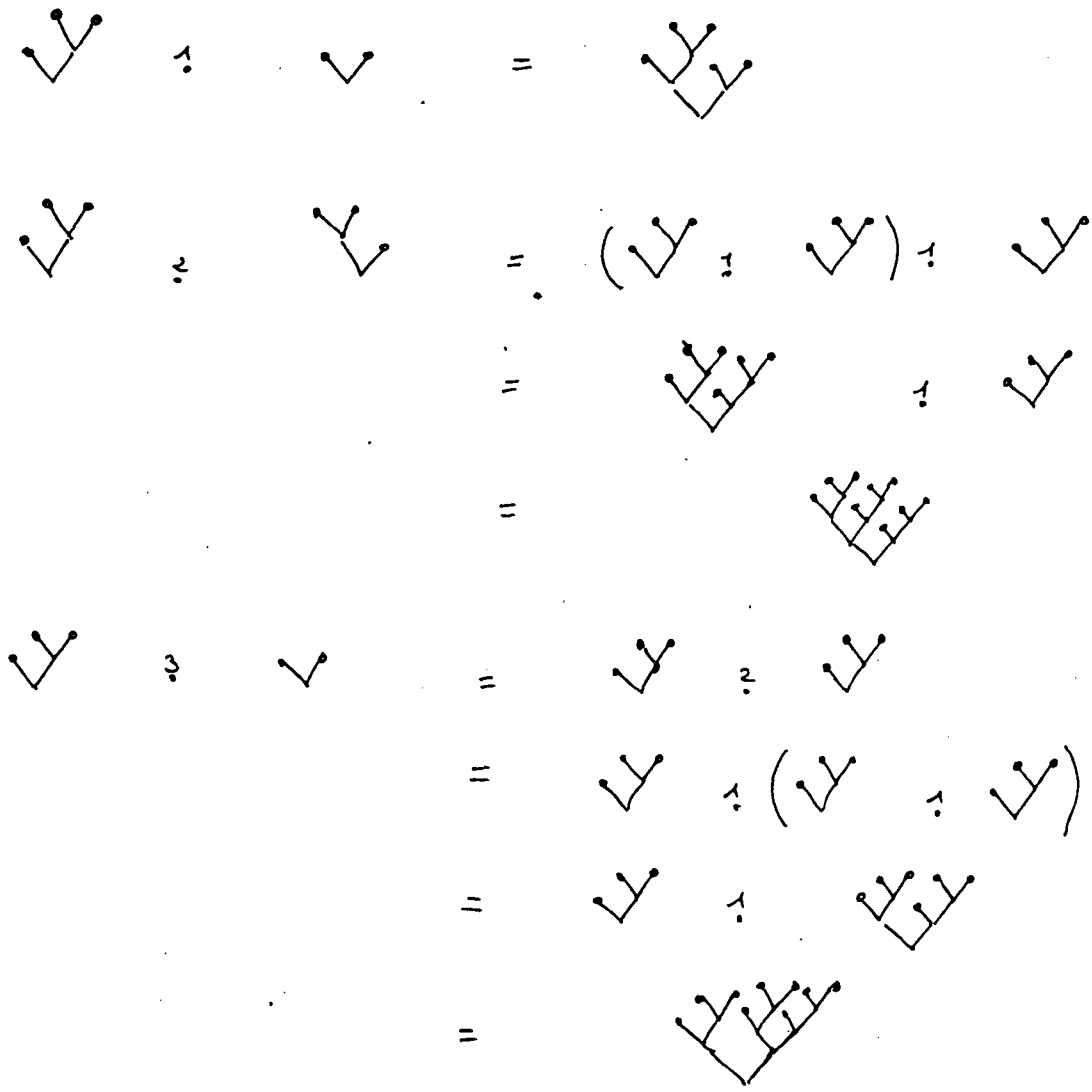


Figure 2: Binary trees resulting from the operations of addition, multiplication and exponentiation.

3 Algebraic properties

Some of the algebraic properties of the operations $\overset{k}{\cdot}$ are studied in this section. The main tool used is an adaptation of the induction principle.

Proposition 2 (Induction principle for binary trees) *Let Q be a predicate defined on binary trees. If the following two conditions are satisfied*

1. $Q(\bullet)$
2. *if* $Q(a_L)$ *and* $Q(a_R)$ *then* $Q(a_L a_R)$

Then $Q(a)$ is satisfied for all binary trees.

3.1 General properties

Neutral and absorbant. A first application of the induction principle allows us to give a short proof of binary trees identities corresponding to the usual identities $a.1 = 1.a$, $a^1 = a$ and $a^0 = 1$.

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

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

Proof. The equalities $a \overset{k}{\cdot} \bullet = a$ for $k \geq 2$ all 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 $\bullet \overset{k}{\cdot} a = a$ can similarly be proved by induction on a. □

Distributivity and associativity. The next two properties are valid for arbitrary $k \geq 2$. Both properties will be used extensively in the rest of the paper.

Theorem 2 (Distributive properties) *Let a, b, c be binary trees and $k \geq 2$. Then*

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

Proof. 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 distributive 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

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

for $c = (c_L c_R)$. Hence the result. □

When evaluated with $k = 2$ these distributive properties 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 **N** have thus counterparts for binary trees. Central in the sequel will be the fact that multiplication is associative.

Fixed point. The remarkable fact that $2+2 = 2.2 = 2^2$ carries over for operations on binary trees. Indeed, for any binary tree a and $k \geq 1$ we have $a \overset{k}{\cdot} a = a \overset{k+1}{\cdot} (\bullet\bullet)$. So that in particular $(\bullet\bullet) \overset{k+1}{\cdot} (\bullet\bullet) = (\bullet\bullet) \overset{k}{\cdot} (\bullet\bullet)$.

Cancellation rule. The operation of addition satisfies a strong form of the cancellation rule: If $a \overset{1}{\cdot} b = c \overset{1}{\cdot} d$, then $a = c$ and $b = d$. The operation of multiplication satisfies both the right and left cancellation rule.

Theorem 3 (Left and right cancellation rule for multiplication) *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. 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 $(a \overset{2}{\cdot} b_L) \overset{1}{\cdot} (a \overset{2}{\cdot} b_R) = (c \overset{2}{\cdot} b_L) \overset{1}{\cdot} (c \overset{2}{\cdot} b_R)$. By the cancellation rule for addition we obtain $a \overset{2}{\cdot} b_L = c \overset{2}{\cdot} b_L$ and the induction hypothesis then leads us to the conclusion. □

Counterexamples.
Commutativity. All 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, $(a \overset{k}{\cdot} \bullet) \overset{k}{\cdot} a \neq a \overset{k}{\cdot} (\bullet \overset{k}{\cdot} a)$. Thus, by Theorem 2, the only associative operation is $\overset{2}{\cdot}$.

	Commutativity	Associativity	neutral	cancellation rules
$\overset{1}{:}$	no	no	no	$a \overset{1}{:} b = c \overset{1}{:} d \Rightarrow a = c, b = d$
$\overset{2}{:}$	no	yes	$a \overset{2}{:} \bullet = a$ $\bullet \overset{2}{:} a = a$	$a \overset{2}{:} b = c \overset{2}{:} b \Rightarrow a = c$ $a \overset{2}{:} b = a \overset{2}{:} d \Rightarrow b = d$
$\overset{3}{:}$	no	no	$a \overset{3}{:} \bullet = a$ $\bullet \overset{3}{:} a = \bullet$	$a \overset{3}{:} b = c \overset{3}{:} b \Rightarrow a = c$ $a \overset{3}{:} b = a \overset{3}{:} d \Rightarrow weight(b) = weight(d)$
$\overset{4}{:}$	no	no	$a \overset{4}{:} \bullet = a$ $\bullet \overset{4}{:} a = \bullet$	$a \overset{4}{:} b = c \overset{4}{:} b \Rightarrow a = c$ $a \overset{4}{:} b = a \overset{4}{:} d \Rightarrow a \Diamond_4 b = a \Diamond_4 d$
$\overset{k}{:}$	no	no	$a \overset{k}{:} \bullet = a$ $\bullet \overset{k}{:} a = \bullet$	$a \overset{k}{:} b = c \overset{k}{:} b \Rightarrow^{?} a = c$ $a \overset{k}{:} b = a \overset{k}{:} d \Rightarrow a \Diamond_k b = a \Diamond_k d$

Table 1: Algebraic properties of the operations $\overset{k}{:}$.

Cancellation rule. The left cancellation rule does not hold for the operations $\overset{k}{:}$ when $k \geq 3$. Indeed, for any binary tree a and $k \geq 3$ we have $\bullet \overset{k}{:} a = \bullet$. Thus $\bullet \overset{k}{:} a = \bullet \overset{k}{:} b$ for all binary trees a and b which clearly shows that the left cancellation rule does not hold.
The right cancellation rule for $k \geq 3$ is much harder to analyse. It is known to hold for $k = 3$ and $k = 4$ but the general case is yet unsettled. We conjecture here that it holds for all $\overset{k}{:}$ $(k \geq 1)$.

Some of the algebraic properties of the operations $\overset{k}{:}$ are summarized in Table 1

3.2 Prime decomposition

Prime binary trees are defined in a natural way.

Definition 3 *The binary tree a is prime if it is different from $\bullet$ and $a = b \overset{?}{:} c$ implies that $b = \bullet$ or $c = \bullet$.*

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 assertion is not true. The binary tree $(\bullet(\bullet(\bullet)))$ for example has weight 4 but is a prime binary tree. It is easy to see that four of the five binary trees of weight tree are prime and that, in general, a number n is prime if and only if all binary trees of weight n are prime. A list of the number of prime binary trees of given weight is shown in Table 2. The number of binary trees of weigth n is given by the nth Catalan number $(2n-2)!/(n!(n-1)!)$ but we do not know a general formula for the number of composite (or prime) binary trees (for a proof that binary trees are counted by Catalan numbers see for example [9]; [8, 17] contain comprehensive lists of references on Catalan numbers and Catalan counted objects. See also [7]).

Binary trees different from $\bullet$ 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 $a_1 \overset{?}{:} a_2 = b_1 \overset{?}{:} b_2$. If a_1 and b_1 (or a_2 and b_2) are prime, then $a_1 = b_1$ and $a_2 = b_2$.*

weight	number of binary trees	number of composites
1	1	1
2	1	0
3	2	0
4	5	1
5	14	0
6	42	4
7	132	0
8	429	9
9	1430	4
10	4862	28
11	16796	0
12	58786	98

Table 2: Number of binary trees and prime binary trees of given weight.

Proof. If $a_1 = b_1$, then the left cancellation rule for multiplication gives the second identity. We proceed by induction on a_2 to prove that $a_1 = b_1$.
If $a_2 = \bullet$ then $a_1 = b_1 \stackrel{2}{:} b_2$. 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 $a_1 \stackrel{2}{:} a_2 = b_1 \stackrel{2}{:} b_2$ and $a_2 = (a_{2L} a_{2R})$. If $b_2 = \bullet$, then $a_1 \stackrel{2}{:} a_2 = b_1$ and the theorem is proved so assume $b_2 = (b_{2L} b_{2R})$. We have $(a_1 \stackrel{2}{:} a_{2L}) \stackrel{1}{:} (a_1 \stackrel{2}{:} a_{2R}) = (b_1 \stackrel{2}{:} b_{2L}) \stackrel{1}{:} (b_1 \stackrel{2}{:} b_{2R})$. 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 4 (Prime decomposition) *Let a be a binary tree different from $\bullet$. Then $a = a_1 \stackrel{2}{:} a_2 \stackrel{2}{:} \cdots \stackrel{2}{:} a_n$ for some $n \geq 1$ and a_i prime binary trees. If $a = b_1 \stackrel{2}{:} b_2 \stackrel{2}{:} \cdots \stackrel{2}{:} b_m$ 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 not prime, there exists a_1 and a_2 different from $\bullet$ and such that $a = a_1 \stackrel{2}{:} a_2$. 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 $a = a_1 \stackrel{2}{:} a_2 \stackrel{2}{:} \cdots \stackrel{2}{:} a_n$ for some $n \geq 1$ and a_i prime binary trees.
Assume now that $a = b_1 \stackrel{2}{:} b_2 \stackrel{2}{:} \cdots \stackrel{2}{:} b_m$ is another such decomposition. By the lemma we must have $a_1 = b_1$ and $a_2 \stackrel{2}{:} a_3 \stackrel{2}{:} \cdots a_n = b_2 \stackrel{2}{:} b_3 \stackrel{2}{:} \cdots b_m$. 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 i^{th} factor in the decomposition is uniquely determined and is called the i^{th} factor of a. We can exactly characterise those binary trees whose sum is prime.

Theorem 5 *Let a, b be binary trees different from $\bullet$. Then $a \stackrel{1}{:} b$ 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. □

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

Multiplication of binary trees is an associative operation. The result of $a \overset{3}{\cdot} b$ will therefore depend on a and on the weight of b but not on the shape of b. We have thus:

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

In view of this result we shall authorize ourselve to write $a \overset{3}{\cdot} n$ for $a \overset{3}{\cdot} b$ where b is any binary tree of weight n.
The result contained in Proposition 3 can be generalized to higher order operations.

Theorem 6 *Let a, b be binary trees and $k \geq 3$. Then $a \overset{k}{\cdot} b = a \overset{3}{\cdot} n$ for some $n = n(a, b) \geq 1$.*

Proof. We proceed by induction on k. For $k = 3$ the result is a consequence of Proposition 3. We assume that the result holds for $\overset{k}{\cdot}$ and show, by induction on b, that it then holds for $\overset{k+1}{\cdot}$. If $b = \bullet$, then $a \overset{k+1}{\cdot} \bullet = a = a \overset{3}{\cdot} 1$ so let $b = (b_L b_R)$ and assume that $a \overset{k+1}{\cdot} b_L = a \overset{3}{\cdot} n_L$ and $a \overset{k+1}{\cdot} b_R = a \overset{3}{\cdot} n_R$ for some $n_L, n_R \geq 1$. Then $a \overset{k+1}{\cdot} b = a \overset{k+1}{\cdot} (b_L b_R) = (a \overset{k+1}{\cdot} b_L) \overset{k}{\cdot} (a \overset{k+1}{\cdot} b_R) = (a \overset{3}{\cdot} n_L) \overset{k}{\cdot} (a \overset{3}{\cdot} n_R)$. By induction hypothesis on k we have then $a \overset{k+1}{\cdot} b = (a \overset{3}{\cdot} n_L) \overset{3}{\cdot} n'$ for some $n' \geq 1$. A final application of the second distributive property leads us to $a \overset{k+1}{\cdot} b = a \overset{k+1}{\cdot} (b_L b_R) = a \overset{3}{\cdot} n$ for $n = n_L . n'$ and concludes the proof. □

When $k = 3$ one has an explicit value for the value of n appearing in Theorem 6, namely $n = weight(b)$. When $k \geq 4$ the value taken by n will depend not only on the weight of b but also on the weight of a and on their respective shape. This dependency can be formalized by the definition of operations defined over BT and taking positive values.

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

1. $a \Diamond_3 b = weight(b)$
2. $$a \Diamond_k b = \begin{cases} 1 & \text{if } b = \bullet \\ (a \Diamond_k b_L).((a \overset{k}{\cdot} b_L) \Diamond_{k-1} (a \overset{k}{\cdot} b_R)) & \text{if } b = (b_L b_R) \end{cases}$$

With this purpose-built definition we have:

Theorem 7 *Let a, b be binary trees and $k \geq 3$. Then $a \overset{k}{\cdot} b = a \overset{3}{\cdot} (a \Diamond_k b)$.*

Proof. We proceed by induction on k. For $k = 3$ the result is contained in Proposition 6. We assume that the result holds for $\overset{k}{\cdot}$ and show, by induction on b, that it also holds for $\overset{k+1}{\cdot}$. If $b = \bullet$, then $a \overset{k}{\cdot} \bullet = a = a \overset{3}{\cdot} \bullet = a \overset{3}{\cdot} 1 = a \overset{3}{\cdot} (a \Diamond_k \bullet)$ so let $b = (b_L b_R)$ and assume that $a \overset{k}{\cdot} b_L = a \overset{3}{\cdot} (a \Diamond_k b_L)$ and $a \overset{k}{\cdot} b_R = a \overset{3}{\cdot} (a \Diamond_k b_R)$. We have then

$$\begin{aligned} a \overset{k}{\cdot} b &= a \overset{k}{\cdot} (b_L b_R) \\ &= (a \overset{k}{\cdot} b_L) \overset{k-1}{\cdot} (a \overset{k}{\cdot} b_R) \\ &= (a \overset{k}{\cdot} b_L) \overset{3}{\cdot} ((a \overset{k}{\cdot} b_L) \Diamond_{k-1} (a \overset{k}{\cdot} b_R)) \\ &= (a \overset{3}{\cdot} (a \Diamond_k b_L)) \overset{3}{\cdot} ((a \overset{k}{\cdot} b_L) \Diamond_{k-1} (a \overset{k}{\cdot} b_R)) \\ &= a \overset{3}{\cdot} ((a \Diamond_k b_L) \overset{2}{\cdot} ((a \overset{k}{\cdot} b_L) \Diamond_{k-1} (a \overset{k}{\cdot} b_R))) \\ &= a \overset{3}{\cdot} (a \Diamond_k b) \qquad \bullet \end{aligned}$$

and the theorem is proved. □

Cancellation rules for $\overset{1}{\cdot}$ and $\overset{2}{\cdot}$ were analysed in Section 3.1. For $k \geq 3$ we have argued that the left cancellation rule does not hold 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 7 our argument can now be made more precise.

Theorem 8 *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 $a \Diamond_k b = a \Diamond_k d$.*

Proof. (Necessity) By Theorem 7 we know that $a \overset{k}{\cdot} b = a \overset{3}{\cdot} (a \Diamond_k b)$ and $a \overset{k}{\cdot} d = a \overset{3}{\cdot} (a \Diamond_k d)$. Hence $a \overset{3}{\cdot} (a \Diamond_k b) = a \overset{3}{\cdot} (a \Diamond_k d)$. But then the prime decomposition theorem lead us to $a \Diamond_k b = a \Diamond_k d$ as requested.
(Sufficiency) This part is trivial. If $a \Diamond_k b = a \Diamond_k d$, then $a \overset{k}{\cdot} b = a \overset{3}{\cdot} (a \Diamond_k b) = a \overset{3}{\cdot} (a \Diamond_k d) = a \overset{k}{\cdot} d$.
□

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 $weight(b) = weight(d)$.*

4 Valuations

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

Definition 5 *Associated to $\Delta : \mathbb{Z} \times \mathbb{Z} \to \mathbb{Z}$ and $e \in \mathbb{Z}$ is a valuation $\mu : BT \to \mathbb{Z}$ 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 $e = 1$. Some other interesting inductive valuations are given hereafter. In the theorem that follows we show that the operations $\overset{k}{\cdot}$ enjoy remarkable properties with

$a\Delta b$	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

Table 3: Some inductive valuations.

respect to most of these valuations.

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 refered too 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 [5] it is called the "register function" and is used to calculate the minimal number of registers needed to evaluate an arithmetic expression (see also [10]). The same function is known in hydrology as the Horton-Strahler function and is used to describe characteristics of river flows (see [18, 20, 21] and references therein). The Strahler number of a tree is also equal to the height of the maximal complete tree that can be embedded in the tree [5]. 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. It is equal to the height of the largest complete binary tree that appear everywhere on the boundary of the tree. With our notations the 2-bud of a binary tree a can be shown equal the largest $n \geq 0$ for which the n^{th} first factor of a are all equal to $(\bullet\bullet)$.

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 $\overset{k}{:}$.

Theorem 9 *Assume $\Delta : \mathbb{Z}\times\mathbb{Z}\to\mathbb{Z}$, $e=0$, and let μ be the inductive valuation associated to Δ and e. If $a+(b\Delta c) = (a+b)\Delta(a+c)$ for all $a,b,c\in\mathbb{Z}$, then*

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

3. $\mu(a \overset{4}{:} b) = \mu(a)^{\mu(b)}$

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

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

The other identities can be proved similarly. □

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

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

Proof. The corresponding pairs (Δ, e) satisfy the conditions of Theorem 9. □

5 Infinite trees

The operations $\overset{k}{:}$ introduced for finite binary trees can be extended to infinite binary trees. For convenience we shall look at infinite trees as languages over 2-letters alphabets. We first briefly review elementary definitions and properties of languages and then analyse the properties of $\overset{k}{:}$ when applied to factorial languages over 2-letters alphabets.

5.1 Factorial languages

Let Σ be a finite alphabet, x a word of the language Σ^* generated by Σ and $L \subset \Sigma^*$ a language over Σ (for definitions see [14]). The product of x by L is the language $x.L = \{x.y : y \in L\}$ and the residual of L by x is the language $x^{-1}.L = \{y \in \Sigma^* : x.y \in L\}$. 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. The product of two languages L_1 and L_2 is the language $L_1.L_2 = \{x_1.x_2 : x_1 \in L_1, x_2 \in L_2\}$. For a language L and $n \geq 0$ we define $L^0 = \{\omega\}$ (ω denotes the empty word), $L^{n+1} = L^n.L$ and $L^* = \bigcup_{i=0}^{\infty} L^i$. Finally, a language L over Σ is factorial if $x, v \in \Sigma^*, x.v \in L$ implies that $x \in L$. Factorial languages always contain ω unless they are empty.

Theorem 10 *Let $L, L_1, L_2, \ldots$ be factorial languages over Σ, $x \in \Sigma^*$ and $n \geq 0$. Then the languages*

1. $x^{-1}.L$

2. $\lceil L \rceil_n$

3. $L = \bigcup_{i=0}^{\infty} L_i$

4. $L = \bigcap_{i=0}^{\infty} L_i$

5. $L_1.L_2$

6. L^*

are factorial.

Proof. Direct application of the definitions. □

Infinite trees can be represented by languages over alphabets with two letters. Indeed, any node in a binary tree 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. A node can thus be seen as a word over the alphabet $\Sigma = \{a, b\}$. A binary tree is entirely specified by its set of internal nodes. Thus a finite (infinite) tree will have a representation as a finite (infinite) language over Σ. To the trivial 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 an alphabet of two symbols can be seen as a representation of a particular binary tree. The corresponding binary tree is infinite if and only if the language is infinite. In the sequel we denote by FL the set of factorial languages over the two letter alphabet $\{a, b\}$.

5.2 Operations on factorial languages

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

$$L_1 \overset{k}{:} L_2 = \cup_{i=0}^{\infty} (\lceil L_1 \rceil_i \overset{k}{:} \lceil L_2 \rceil_i)$$

Theorem 11 *The operations* $\overset{k}{:}$ *are internal operations on factorial languages.*

Proof. The statement is clearly true for finite languages because any $\overset{k}{:}$ product of two finite languages can be expressed in terms of finitely many operations of the form given in Theorem 10. We complete the proof by observing that the result of a $\overset{k}{:}$ product of infinite languages is defined by a countable union of finite factorial languages. □

Most of the properties of the operations $\overset{k}{:}$ for finite binary trees were proved with the help of the induction principle. This principle does not anymore hold for infinite binary trees (or infinite factorial languages). It is nevertheless possible to show that the distributive properties

of Theorem 2 remain satisfied for infinite binary trees. On the other hand the statement of Theorem 3 is violated by infinite binary trees. Assume for example that $L_1 = \Sigma^*$, $L_2 = \{\omega\}$ and $L_3 = \{\omega, a, b\}$. Then $L_1 \overset{2}{:} L_2 = L_1 \overset{2}{:} L_3$ but $L_2 \neq L_3$.

Similarly to Theorem 6, the result of operations of index greater or equal to 3 can be expressed as $\overset{2}{:}$ products of identical factors. Infinitely many factors are multiplied when the second operand is infinite. The operation $\overset{2}{:}$ correspond to the usual multiplication of languages and the operation $\overset{3}{:}$ therefore degenerates into the Kleene "star" operation when the second operand is infinite.

Theorem 12 *Let L_1, L_2 be two factorial languages. Then*

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

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 $L_2 = L_2' \overset{1}{:} L_2''$ and assume that $L_1 \overset{3}{:} L_2' = L_1^{n'}$ and $L_1 \overset{3}{:} L_2'' = L_1^{n''}$ for some $n', n'' \geq 0$. We have then

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

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

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

Acknowledgements. We wish to express our sincere thanks to Professor D. Welsh, Oxford University, Professor P. Delsarte, University of Louvain, Dr S. Gaubert, INRIA, and Professor D. Knuth, Stanford University for commenting a first version of this paper.

References

[1] V. Blondel, Structured numbers, Technical Report TRITA/MAT-94-31, Department of mathematics, Royal Institute of Technology, S-10044 Stockholm, 1994.

[2] I. M. Etherington, Non associative products and a functional equation, *Math. Gazette* **21** (1937) 36-39.

[3] I. M. Etherington, On non-associative combinations, *Proc. Royal Soc. of Edinburgh* **58-59** (1937-1938) 153-162.

[4] I. M. Etherington, Genetic algebras, *Proc. Royal Soc. of Edinburgh* **58-59** (1937-1938) 242-258.

[5] P. Flajolet, J. C. Raoult, J. Vuillemin, The number of registers required for evaluating arithmetic expressions, *Theoret. Comp. Sc.* **9** (1979) 99-125.

[6] P. Flajolet, Z. Gao, A. Odlyzko, B. Richmond, The distribution of heights of binary trees and other simple trees, Technical Report INRIA 1749, 1992.

[7] M. Gardner, Mathematical games: Catalan numbers, *Sci. Amer.* (1976) 120-122.

[8] H. W. Gould, Research bibliography of two special number sequences, *Mathematica Monongaliae* **12** (1971).

[9] P. Hilton, J. Pedersen, Catalan numbers and their various uses, in: W. Lederman, ed., *Handbook of applicable mathematics* (John Wiley, Chichester, 1990).

[10] R. Kemp, The average number of register needed to evaluate a binary tree optimally, *Acta Informatica* **11** (1979) 363-372.

[11] D. E. Knuth, *The art of computer programming* (Addison-Wesley, 1968).

[12] D. E. Knuth, Mathematics and computer science: coping with finiteness, *Science* **194** (1976) 1235-1242.

[13] D. E. Knuth, Personal communication, October 10th, 1994.

[14] J. van Leeuwen, *Handbook of theoretical computer sciences, Vol. A and B* (North-Holland, 1990).

[15] C. Pair, A. Quere, Définition et étude des bilangages régulier, *Information and Control* **13** (1968) 565-593.

[16] G. Pólya, On picture-writing, *Amer. Math. Monthly* **63** (1956), 689-697.

[17] R. P. Stanley, *Enumerative combinatorics* (preliminary version, 1994).

[18] A. N. Strahler, Hypsomic analysis of erosional topography, *Bulletin Geological Society of America* **63** (1952) 1117-1142.

[19] J. Vannimenus, X. G. Viennot, Combinatorial tools for the analysis of ramified patterns *J. of Stat. Physics* **54** (1989) 1529-1538.

[20] X. G. Viennot, Trees, in: M. Lothaire, ed., *Mots. Mélanges offerts à M.-P. Schützenberger* (Hermes, Paris, 1990).

[21] X. G. Viennot, G. Eyrolles, N. Janey, D. Arquès, Combinatorial analysis of ramified patterns in computer imagery of trees, in: *Proc. SIGGRAPH* (1989).

Unité de recherche INRIA Rocquencourt
Domaine de Voluceau - Rocquencourt - B.P. 105 - 78153 Le Chesnay Cedex (France)

Unité de recherche INRIA Lorraine - Technopôle de Nancy-Brabois - Campus scientifique
615, rue du Jardin Botanique - B.P. 101 - 54602 Villers lès Nancy Cedex (France)
Unité de recherche INRIA Rennes - IRISA, Campus universitaire de Beaulieu 35042 Rennes Cedex (France)
Unité de recherche INRIA Rhône-Alpes 46, avenue Félix Viallet - 38031 Grenoble Cedex 1 (France)
Unité de recherche INRIA Sophia Antipolis - 2004, route des Lucioles - B.P. 93 - 06902 Sophia Antipolis Cedex (France)

Éditeur
INRIA - Domaine de Voluceau - Rocquencourt - B.P. 105 - 78153 Le Chesnay Cedex (France)

ISSN 0249 - 6399