A Brief History of Strahler Numbers

Javier Esparza, Michael Luttenberger, and Maximilian Schlund

Fakultät für Informatik, Technische Universität München, Germany

Abstract. The Strahler number or Horton-Strahler number of a tree, originally introduced in geophysics, has a surprisingly rich theory. We sketch some milestones in its history, and its connection to arithmetic expressions, graph traversing, decision problems for context-free languages, Parikh's theorem, and Newton's procedure for approximating zeros of differentiable functions.

1 The Strahler Number

In 1945, the geophysicist Robert Horton found it useful to associate a *stream order* to a system of rivers (geophysicists seem to prefer the term 'stream") [20].

> *Unbranched fingertip tributaries are always designated as of order 1, tributaries or streams of the 2d order receive branches or tributaries of the 1st order, but these only; a 3d order stream must receive one or more tributaries of the 2d order but may also receive 1st order tributaries. A 4th order stream receives branches of the 3d and usually also of lower orders, and so on.*

Several years later, Arthur N. Strahler replaced this ambiguous definition by a simpler one, very easy to compute [26]:

> *The smallest, or "finger-tip", channels constitute the first-order segments. [. . .]. A second-order segment is formed by the junction of any two first-order streams; a third-order segment is formed by the joining of any two second order streams, etc.*

Streams of lower order joining a higher order stream do not change the order of the higher stream. Thus, if a first-order stream joins a second-order stream, it remains a second-order stream. Figure 1 shows the Strahler number for a fragment of the course of the Elbe river with some of its tributaries. The stream system is of order 4.

From a computer science point of view, stream systems are just trees.

Definition 1. *Let t be a tree with root r. The* Strahler number *of t, denoted by $S(t)$, is inductively defined as follows.*

- *If r has no children (i.e., t has only one node), then $S(t) = 0$.*

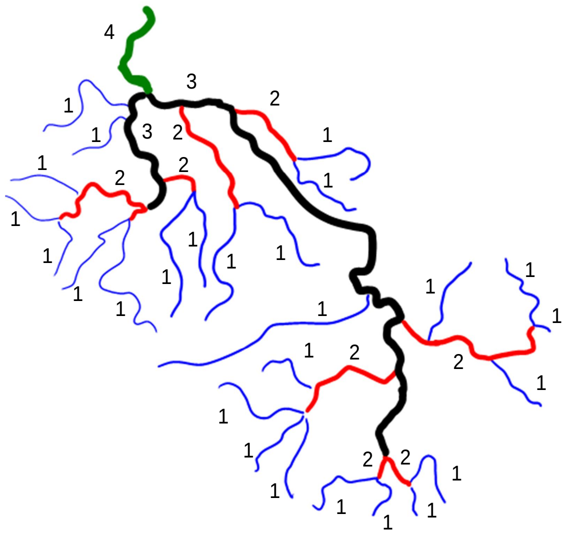


Fig. 1. Strahler numbers for a fragment of the Elbe river.

– *If r has children $r_1, \ldots, r_n$, then let $t_1, \ldots, t_n$ be the subtrees of t rooted at $r_1, \ldots, r_n$, and let $k = \max\{S(t_1), \ldots, S(t_n)\}$: if exactly one of $t_1, \ldots, t_n$ has Strahler number k, then $S(t) = k$; otherwise, $S(t) = k + 1$.*

Note that in this formal definition the Strahler number of a simple chain (a "finger-tip") is *zero*, and not *one*. This allows another characterization of the Strahler number of a tree t as the height of the largest minor of t that is a perfect binary tree (i.e., a rooted tree where every inner node has two children and all leaves have the same distance to the root): Roughly speaking, such a binary tree is obtained by, starting at the root, following paths along which the Strahler number never decreases by more than one unit at a time, and then contracting all nodes with only one child. If t itself is a binary tree, then this minor is unique. We leave the details as a small exercise.

Figure 2 shows trees with Strahler number 1, 2, and 3, respectively. Each node is labeled with the Strahler number of the subtree rooted at it.

Together with other parameters, like bifurcation ratio and mean stream length, Horton and Strahler used stream orders to derive quantitative empirical laws for stream systems. Today, geophysicists speak of the Strahler number (or Horton-Strahler number) of a stream system. According to the excellent Wikipedia article on the Strahler number (mainly due to David Eppstein), the Amazon and the Mississippi have Strahler numbers of 10 and 12, respectively.

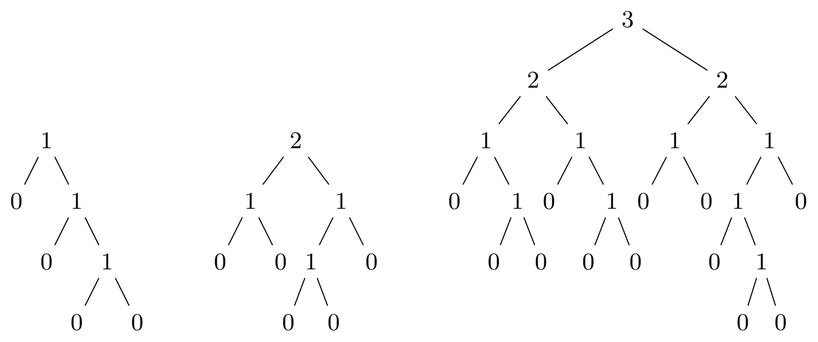
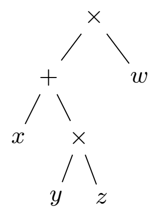


Fig. 2. Trees of Strahler number 1, 2, and 3.

2 Strahler Numbers and Tree Traversal

The first appearance of the Strahler number in Computer Science seems to be due to Ershov in 1958 [8], who observed that the number of registers needed to evaluate an arithmetic expression is given by the Strahler number of its syntax tree. For instance, the syntax tree of $(x + y \cdot z) \cdot t$, shown on the left of Figure 3, has Strahler number 2, and indeed can be computed with just two registers R_1, R_2 by means of the code shown on the right.



$R_1 \leftarrow y$
$R_2 \leftarrow z$
$R_2 \leftarrow R_1 \times R_2$
$R_2 \leftarrow x$
$R_1 \leftarrow R_1 + R_2$
$R_2 \leftarrow w$
$R_1 \leftarrow R_1 \times R_2$

Fig. 3. An arithmetic expression of Strahler number 2

The strategy for evaluating a expression $e = e_1 \; op \; e_2$ is easy: start with the subexpression whose tree has lowest Strahler number, say e_1; store the result in a register, say R_1; reuse all other registers to evaluate e_2; store the result in R_2; store the result of $R_1 \, op R_2$ in R_1.

Ershov's observation is recalled by Flajolet, Raoult and Vuillemin in [16], where they add another observation of their own: the Strahler number of a *binary* tree is the minimal stack size required to traverse it. Let us attach to each node of the tree the Strahler number of the subtree rooted at it. The traversing procedure follows again the "lowest-number-first" policy (notice that arithmetic expressions yield binary trees). If a node with number k has two children, then the traversing procedure moves to the child with lowest number, and pushes the (memory address of the) other child onto the stack. If the node is a leaf, then the procedure pops the top node of the stack and jumps to it. To prove that the stack size never exceeds the Strahler number, we observe that, if a node of number k has two children, then at least one of its children has number smaller than k. So the procedure only pushes a node onto the stack when it moves to a node of strictly smaller number, and we are done.

Notice, however, that the "lowest-number-first" policy requires to know the Strahler number of the nodes. If these are unknown, all we can say is that a nondeterministic traversing procedure always needs a stack of size at least equal to the Strahler number, and that it *may* succeed in traversing the tree with a stack of exactly that size.

2.1 Distribution of Strahler Numbers

The goal of Flajolet, Raoult and Vuillemin's paper is to study the distribution of Strahler numbers in the binary trees with a fixed number n of leaves. Let S_n be the random variable corresponding to the Strahler number of a binary tree (every node has either two or 0 children) with n internal nodes chosen uniformly at random. Since the Strahler number of t is the height of the largest perfect binary tree embeddable in t, we immediately have $S_n \leq \lfloor \log_2(n+1) \rfloor$. The paper shows that

$$Exp[S_n] \approx \log_4 n \qquad \text{and} \qquad Var[S_n] \in \mathcal{O}(1) \ .$$

In other words, when n grows the Strahler number of most trees becomes increasingly closer to $\log_4 n$. Independently of Flajolet, Raoult and Vuillemin, also Kemp derives in [21] the same asymptotic behaviour of the expected Strahler number of a random binary tree. Later, Flajolet and Prodinger extend the analysis to trees with both binary and unary inner nodes [17]. Finally, Devroye and Kruszewski show in [5] that the probability that the Strahler number of a random binary tree with n nodes deviates by at least k from the expected Strahler number of $\log_4 n$ is bounded from above by $\frac{2}{4^k}$, that is, the Strahler number is highly concentrated around its expected value.

2.2 Strahler Numbers in Language Theory: Derivation indices, Caterpillars, and Dimensions

Derivation indices and caterpillars. The Strahler number has been rediscovered (multiple times!) by the formal language community. In [19], Ginsburg and Spanier introduce the index of a derivation $S \Rightarrow \alpha_1 \Rightarrow \alpha_2 \Rightarrow \cdots \Rightarrow w$ of a given grammar as the maximal number of variables occurring in any of the sentential forms α_i (see also [27]). For instance, consider the grammar $X \rightarrow aXX \mid b$. The index of the derivations

$$X \Rightarrow aXX \Rightarrow aXaXX \Rightarrow abaXX \Rightarrow ababX \Rightarrow ababb$$

$$X \Rightarrow aXX \Rightarrow abaXX \Rightarrow abaXX \Rightarrow ababX \Rightarrow ababb$$

is 3 (because of $aXaXX$) and 2, respectively. For context-free grammars, where we have the notion of derivation tree of a word, we define the index of a derivation tree as the minimal index of its derivations. If the grammar is in Chomsky normal form, then a derivation tree has index k if and only if its Strahler number is $(k-1)$.

A first use of the Strahler number of derivation trees can be found in [4], where Chytil and Monien, apparently unaware of the Strahler number, introduce *k-caterpillars* as follows:

> *A caterpillar is an ordered tree in which all vertices of outdegree greater than one occur on a single path from the root to a leaf. A 1-caterpillar is simply a caterpillar and for $k > 1$ a k-caterpillar is a tree obtained from a caterpillar by replacing each hair by a tree which is at most $(k-1)$-caterpillar.*

Clearly, a tree is a k-caterpillar if and only if its Strahler number is equal to k.

Let $L_k(G)$ be the subset of words of $L(G)$ having a derivation tree of Strahler number at most k (or, equivalently, being a k-caterpillar). Chytil and Monien prove that there exists a nondeterministic Turing machine with language $L(G)$ that recognizes $L_k(G)$ in space $O(k \log |G|)$. Assume for simplicity that G is in Chomsky normal form. In order to nondeterministically recognize $w = a_1 a_2 \dots a_n \in L_k(G)$, we guess on-the-fly (i.e., while traversing it) a derivation tree of w with Strahler number at most k, using a stack of height at most k. The traversing procedure follows the "smaller-number-first" policy. More precisely, the nodes of the tree are triples (X, i, j) with intended meaning "X generates a tree with yield $a_i \dots a_j$". We start at node $(S, 1, n)$. At a generic node (X, i, j), we proceed as follows. If $i = j$, then we check that $X \to a_i$ is a production, pop a new node, and jump to it. If $i < j$, then we guess a production, say $X \to YZ$, and an index $i \leq l \leq j$, guess which of $(Y, 1, i)$ and (Z, l, j) generates the subtree of lowest number, say (Y, i, l), and jump to it, pushing (Z, l, j) onto the stack.

The traversing procedure can also be used to check emptiness of $L_k(G)$ in nondeterministic logarithmic space (remember: k is not part of the input) [13]. In this case we do not even need to guess indices: if the current node is labeled by X, then we proceed as follows. If X has no productions, then we stop. If G has a production $X \to a$ for some terminal a, we pop the next node from the stack and jump to it. If G has productions for X, but only of the form $X \to YZ$, then we guess one of them and proceed as above. Notice that checking emptiness of $L(G)$ is a P-complete problem, and so unlikely to be solvable in logarithmic space.

Tree dimension. The authors of this paper are also guilty of rediscovering the Strahler number. In [9] we defined the *dimension* of a tree, which is . . . nothing but its Strahler number.[1] Several papers [9, 11, 18, 13] have used tree dimension

[1] The name dimension was chosen to reflect that trees with Strahler number 1 are a chain (with hairs), trees of dimension 2 are chains of chains (with hairs), that can be

(that is, they have used the Strahler number) to show that $L_{n+1}(G)$, where n is the number of variables of a grammar G in Chomsky normal form, has interesting properties[2]:

(1) Every $w \in L(G)$ is a scattered subword of some $w' \in L_{n+1}(G)$ [13].
(2) For every $w_1 \in L(G)$ there exists $w_2 \in L_{n+1}(G)$ such that w_1 and w_2 have the same *Parikh image*, where the Parikh image of a word w is the function $\Sigma \to \mathbb{N}$ that assigns to every terminal the number of times it occurs in w. Equivalently, w and w' have the same Parikh image if w' can be obtained from w by reordering its letters [9].

The first property has already found at least one interesting application in the theory of formal verification (see [13]). The second property has been used in [12] to provide a simple "constructive" proof of Parikh's theorem. Parikh's theorem states that for every context-free language L there is a regular language L' such that L and L' have the same Parikh image (i.e., the set of Parikh images of the words of L and L' coincide). For instance, if $L = \{a^n b^n \mid n \geq 0\}$, then we can take $L' = (ab)^*$.

The proof describes a procedure to construct this automaton. By property (2), it suffices to construct an automaton A such that $L(A)$ and $L_{k+1}(G)$ have the same Parikh image. We construct A so that its runs "simulate" the derivations of G of index at most $k+1$. Consider for instance the context-free grammar with variables A_1, A_2 (and so $k = 2$), terminals a, b, c, axiom A_1, and productions

$$A_1 \;\to\; A_1 A_2 | a \qquad A_2 \;\to\; b A_2 a A_2 | c A_1$$

Figure 4 shows on the left a derivation of index 3, and on the right the run of A simulating it. The states store the current number of occurrences of A_1 and A_2, and the transitions keep track of the terminals generated at each derivation step. The run of A generates *bacaaca*, which has the same Parikh image as *abcaaca*.

The complete automaton is shown in Figure 5.

3 Strahler Numbers and Newton's method

Finally, we present a surprising connection between the Strahler number and Newton's method to numerically approximate a zero of a function. The connection works for multivariate functions, but in this note we just consider the univariate case.

Consider an equation of the form $X = f(X)$, where $f(X)$ is a polynomial with nonnegative real coefficients. Since the right-hand-side is a monotonic function, by Knaster-Tarski's or Kleene's theorem the equation has exactly one smallest

nicely drawn in the plane, trees of dimension 3 are chains of chains of chains (with hairs), with can be nicely displayed in 3-dimensional space, etc.

[2] For an arbitrary grammar G, the same properties hold for $L_{nm+1}(G)$, where m is the maximal number of variables on the right-hand-side of a production, minus 1. If G is in Chomsky normal form, then $m \leq 1$.

$$\begin{array}{lrl}
& A_1 & (0,1) \\
\Rightarrow A_1A_2 & \xrightarrow{\epsilon} & (1,1) \\
\Rightarrow A_1bA_2aA_2 & \xrightarrow{ba} & (1,2) \\
\Rightarrow A_1bcA_1aA_2 & \xrightarrow{c} & (2,1) \\
\Rightarrow abcA_1aA_2 & \xrightarrow{a} & (1,1) \\
\Rightarrow abcaaA_2 & \xrightarrow{a} & (0,1) \\
\Rightarrow abcaacA_1 & \xrightarrow{c} & (1,0) \\
\Rightarrow abcaaca & \xrightarrow{a} & (0,0)
\end{array}$$

Fig. 4. A derivation and its "simulation".

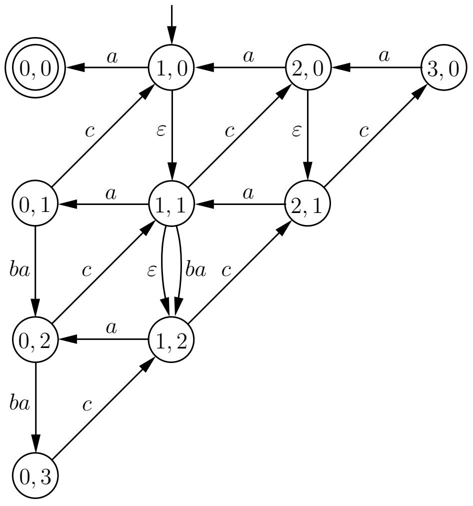


Fig. 5. The Parikh automaton of $A_1 \to A_1A_2|a,\ A_2 \to bA_2aA_2|cA_1$ with axiom A_1.

solution (possibly equal to ∞). We denote this solution by μf. It is perhaps less known that μf can be given a "language-theoretic" interpretation. We explain this by means of an example (see [14] for more details).

Consider the equation

$$X = \frac{1}{4}X^2 + \frac{1}{4}X + \frac{1}{2} \tag{1}$$

It is equivalent to $(X-1)(X-2) = 0$, and so its least solution is $X = 1$. We introduce identifiers a, b, c for the coefficients, yielding the formal equation

$$X = f(X) := aX^2 + bX + c\ . \tag{2}$$

We "rewrite" this equation as a context-free grammar in Greibach normal form in the way one would expect:

$$G:\ X \to aXX \mid bX \mid c\ , \tag{3}$$

Consider now the derivation trees of this grammar. It is convenient to rewrite the derivation trees as shown in Figure 6: We write a terminal not at a leaf, but at its parent node, and so we now write the derivation tree on the left of the figure in the way shown on the right. Notice that, since each production generates a different terminal, both representations contain exactly the same information. [3]

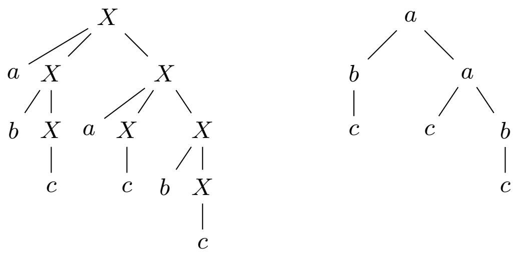


Fig. 6. New convention for writing derivation trees

We assign to each derivation tree t its *value* $V(t)$, defined as the product of the coefficients labeling the nodes. So, for instance, for the tree of Figure 6 we get the value $a^2 \cdot b^2 \cdot c^3 = (1/4)^4(1/2)^3 = 1/128$. Further, we define the value $V(T)$ of a set T of trees as $\sum_{t \in T} V(t)$ (which can be shown to be well defined, even if T is infinite). If we denote by T_G the set of all derivation trees of G, then

$$\mu f = V(T_G). \tag{4}$$

The earliest reference for the this theorem in all its generality we are aware of is Bozapalidis [2] (Theorem 16) to whom also [6] gives credit.

A well-known technique to approximate μf is *Kleene iteration*, which consists of computing the sequence $\{\kappa_i\}_{i \in \mathbb{N}}$ of *Kleene approximants* given by

$$\begin{aligned} \kappa_0 &= 0 \\ \kappa_{i+1} &= f(\kappa_i) \text{ for every } i \geq 0 \end{aligned}$$

It is easy to show that this corresponds to evaluating the derivation trees (with our new convention) *by height.* More precisely, if H_i is the set of derivation trees of T_G of height *less than* i, we get

$$\kappa_i = V(H_i) \tag{5}$$

In other words, the Kleene approximants correspond to evaluating the derivation trees of G by increasing height.

[3] This little change is necessary, because the tree of the derivation $X \Rightarrow c$ has Strahler number 1 if trees are drawn in the standard way, and 0 according to our new convention.

It is well known that convergence of Kleene iteration can be slow: in the worst case, the number of correct digits grows only logarithmically in the number of iterations. *Newton iteration* has much faster convergence (cf. [15, 10, 25]). Recall that Newton iteration approximates a zero of a differentiable function $g(X)$. For this, given an approximation ν_i of the zero, one geometrically computes the next approximation as follows:

– compute the tangent to $g(X)$ at the point $(\nu_i, g(\nu_i))$;
– take for ν_{i+1} the X-components of the intersection point of the tangent and the x-axis.

For functions of the form $g(X) = f(X) - X$, an elementary calculation yields the sequence $\{\nu_i\}_{i\in\mathbb{N}}$ of *Newton approximants*

$$\begin{aligned} \nu_0 &= 0 \\ \nu_{i+1} &= \nu_i - \frac{f(\nu_i) - \nu_i}{f'(\nu_i) - 1} \end{aligned}$$

We remark that in general choosing $\nu_0 = 0$ as the initial approximation may not lead to convergence – only in the special cases of the nonnegative reals or, more generally, ω-continuous semirings, convergence is guaranteed for $\nu_0 = 0$.

A result of [11] (also derived independently in [23]) shows that, if S_i is the set of derivation trees of T_G of Strahler number *less than* i (where trees are drawn according to our new convention), then

$$\nu_i = V(S_i) \tag{6}$$

In other words, the Newton approximants correspond to evaluating the derivation trees of G by increasing Strahler number!

The connection between Newton approximants and Strahler numbers has several interesting consequences. In particular, one can use results on the convergence speed of Newton iteration [3] to derive information on the distribution of the Strahler number in randomly generated trees. Consider for instance random trees generated according to the following rule.

> A node has three children with probability 0.1, two children with probability 0.2, one child with probability 0.1, and zero children with probability 0.6.

Let G the context-free grammar

$$X \to aXXX \mid bXX \mid cX \mid d$$

with valuation $V(a) = 0.1, V(b) = 0.2, V(c) = 0.1, V(d) = 0.6$. It is easy to see that the probability of generating a tree t is equal to its value $V(t)$. For instance, the tree t of Figure 7 satisfies $Pr[t] = V(t) = a \cdot b^2 \cdot c \cdot d^5$.

Therefore, the Newton approximants of the equation

$$X = 0.1X^3 + 0.2X^2 + 0.1X + 0.6$$

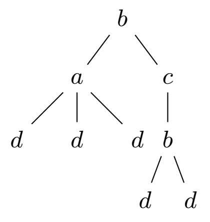


Fig. 7. A tree with probability $a \cdot b^2 \cdot c \cdot d^5$

give the distribution of the random variable S that assigns to each tree its Strahler number. Since $f(X) = 0.1X^3 + 0.2X^2 + 0.1X + 0.6$ and $f'(X) = 0.3X^2 + 0.4X^2 + 0.1$, we get

$$\begin{aligned} \nu_0 &= 0.6 \\ \nu_{i+1} &= \nu_i - \frac{\nu_i^3 + 2\nu_i^2 - 9\nu_i + 6}{3\nu_i^2 + 4\nu_i - 9} \end{aligned}$$

and so for the first approximants we easily obtain

$$\begin{aligned} \nu_0 &= Pr[S < 0] = 0 \\ \nu_1 &= Pr[S < 1] = 0.66\overline{7} \\ \nu_2 &= Pr[S < 2] \approx 0.904 \\ \nu_3 &= Pr[S < 3] \approx 0.985 \\ \nu_4 &= Pr[S < 4] \approx 0.999 \end{aligned}$$

As we can see, the probability converges very rapidly towards 1. This is not a coincidence. The function $f(X)$ satisfies $\mu f < 1$, and a theorem of [3] shows that for every f satisfying this property, there exist numbers $c > 0$ and $0 < d < 1$ such that

$$Pr[S \geq k] \leq c \cdot d^{2^k} \ .$$

4 Strahler numbers and . . .

We have exhausted neither the list of properties of the Strahler number, nor the works that have obtained them or used them. To prove the point, we mention some more papers.

In 1978, Ehrenfeucht et al. introduced the same concept for derivation trees w.r.t. ET0L systems in [7] where it was called *tree-rank*.

Meggido et al. introduced in 1981 the *search number* of an undirected tree [22]: the minimal number of police officers required to capture a fugitive when police officers may move along edges from one node to another, and the fugitive can move from an edge to an incident one as long as the common vertex is not blocked by a police officer; the fugitive is captured when he cannot move anymore. For trees, the search number coincides with the better known *pathwidth* (see e.g. [1]), defined for general graphs. In order to relate the pathwidth to the Strahler number, we need to extend the definition of the latter to undirected trees: let the Strahler number $S(t)$ of an *undirected* tree be the minimal Strahler number of all the directed trees obtained by choosing a node as root, and orienting all edges away from it. We can show that for any tree t:

$$\mathit{pathwidth}(t) - 1 \leq S(t) \leq 2 \cdot \mathit{pathwidth}(t)$$

Currently, we are studying the Strahler number in the context of natural language processing. Recall that the Strahler number measures the minimal height of a stack required to traverse a tree, or, more informally, the minimal amount of memory required to process it. We conjecture that most sentences of a natural language should have a small Strahler number – simply not to overburden the reader or listener. Table 1 contains the results of an examination of several publicly available tree banks (banks of sentences that have been manually parsed by human linguists), which seem to support this conjecture. For each language we have computed the average and maximum Strahler number of the parse trees in the corresponding tree bank. We are currently investigating whether this fact can be used to improve unlexicalized parsing of natural languages.

Language	Source	Average	Maximum
Basque	SPMRL‡	2.12	3
English	Penn♣	2.38	4
French	SPMRL	2.29	4
German	SPMRL	1.94	4
German	TueBa-D/Z♠	2.13	4
Hebrew	SPMRL	2.44	4
Hungarian	SPMRL	2.11	4
Korean	SPMRL	2.18	4
Polish	SPMRL	1.68	3
Swedish	SPMRL	1.83	4

Table 1. Average and maximum Strahler numbers for several treebanks of natural languages. ‡: SPMRL shared task dataset, ♣: 10% sample from the Penn treebank shipped with python nltk, ♠: TueBa-D/Z treebank.

5 Conclusions

We have sketched the history of the Strahler number, which has been rediscovered a surprising number of times, received a surprising number of different names (stream order, stream rank, index, tree rank, tree dimension, k-caterpillar . . .), and turns out to have a surprising number of applications and connections (Parikh's theorem, Newton's method, pathwidth . . .).

This paper is by no means exhaustive, and we apologize in advance to the many authors we have surely forgotten. We intend to extend this paper with further references. If you know of further work connected to the Strahler number, please contact us.

6 Acknowledgments

We thank Carlos Esparza for his help with some calculations.

References

1. D. Bienstock, N. Robertson, P. Seymour, and R. Thomas. Quickly excluding a forest. *Journal of Combinatorial Theory, Series B*, 52(2):274 – 283, 1991.
2. S. Bozapalidis. Equational elements in additive algebras. *Theory Comput. Syst.*, 32(1):1–33, 1999.
3. Tomás Brázdil, Javier Esparza, Stefan Kiefer, and Michael Luttenberger. Space-efficient scheduling of stochastically generated tasks. *Inf. Comput.*, 210:87–110, 2012.
4. Michal Chytil and Burkhard Monien. Caterpillars and context-free languages. In Christian Choffrut and Thomas Lengauer, editors, *STACS*, volume 415 of *Lecture Notes in Computer Science*, pages 70–81. Springer, 1990.
5. L. Devroye and P. Kruszewski. A note on the Horton-Strahler number for random trees. *Inf. Process. Lett.*, 56(2):95–99, 1995.
6. M. Droste, W. Kuich, and H. Vogler. *Handbook of Weighted Automata*. Springer, 2009.
7. Andrzej Ehrenfeucht, Grzegorz Rozenberg, and Dirk Vermeir. On et0l systems with finite tree-rank. *SIAM J. Comput.*, 10(1):40–58, 1981.
8. A. P. Ershov. On programming of arithmetic operations. *Comm. ACM*, 1(8):3–9, 1958.
9. J. Esparza, S. Kiefer, and M. Luttenberger. On fixed point equations over commutative semirings. In *STACS*, volume 4393 of *LNCS*, pages 296–307. Springer, 2007.
10. J. Esparza, S. Kiefer, and M. Luttenberger. Computing the least fixed point of positive polynomial systems. *SIAM J. Comput.*, 39(6):2282–2335, 2010.
11. J. Esparza, S. Kiefer, and M. Luttenberger. Newtonian program analysis. *J. ACM*, 57(6):33, 2010.
12. Javier Esparza, Pierre Ganty, Stefan Kiefer, and Michael Luttenberger. Parikhs theorem: A simple and direct automaton construction. *Inf. Process. Lett.*, 111(12):614–619, 2011.
13. Javier Esparza, Pierre Ganty, and Rupak Majumdar. Parameterized verification of asynchronous shared-memory systems. In Sharygina and Veith [24], pages 124–140.
14. Javier Esparza and Michael Luttenberger. Solving fixed-point equations by derivation tree analysis. In Andrea Corradini, Bartek Klin, and Corina Cîrstea, editors, *CALCO*, volume 6859 of *Lecture Notes in Computer Science*, pages 19–35. Springer, 2011.
15. K. Etessami and M. Yannakakis. Recursive markov chains, stochastic grammars, and monotone systems of nonlinear equations. *J. ACM*, 56(1), 2009.

16. P. Flajolet, J.-C. Raoult, and J. Vuillemin. The number of registers required for evaluating arithmetic expressions. *Theor. Comput. Sci.*, 9:99–125, 1979.
17. Philippe Flajolet and Helmut Prodinger. Register allocation for unary-binary trees. *SIAM J. Comput.*, 15(3):629–640, 1986.
18. Pierre Ganty, Rupak Majumdar, and Benjamin Monmege. Bounded underapproximations. *Formal Methods in System Design*, 40(2):206–231, 2012.
19. S. Ginsburg and E. Spanier. Derivation-bounded languages. *Journal of Computer and System Sciences*, 2:228–250, 1968.
20. R. E. Horton. Erosional development of streams and their drainage basins: hydrophysical approach to quantitative morphology. *Geol. Soc. Am. Bull.*, 56(3):275–370, 1945.
21. R. Kemp. The average number of registers needed to evaluate a binary tree optimally. *Acta Informatica*, 11:363–372, 1979.
22. N. Megiddo, S. Louis Hakimi, M. R. Garey, D. S. Johnson, and C. H. Papadimitriou. The complexity of searching a graph (preliminary version). In *FOCS*, pages 376–385. IEEE Computer Society, 1981.
23. C. Pivoteau, B. Salvy, and M. Soria. Algorithms for combinatorial structures: Well-founded systems and newton iterations. *J. Comb. Theory, Ser. A*, 119(8):1711–1773, 2012.
24. Natasha Sharygina and Helmut Veith, editors. *Computer Aided Verification - 25th International Conference, CAV 2013, Saint Petersburg, Russia, July 13-19, 2013. Proceedings*, volume 8044 of *Lecture Notes in Computer Science*. Springer, 2013.
25. A. Stewart, K. Etessami, and M. Yannakakis. Upper Bounds for Newton's Method on Monotone Polynomial Systems, and P-Time Model Checking of Probabilistic One-Counter Automata. In Sharygina and Veith [24], pages 495–510.
26. A. N. Strahler. Hypsometric (area-altitude) analysis of erosional topology. *Geol. Soc. Am. Bull.*, 63(11):1117–1142, 1952.
27. M.K. Yntema. Inclusion relations among families of context-free languages. *Information and Control*, 10:572–597, 1967.