Parallel Enumeration of Parse Trees

Margarita Mikhelson
Department of Mathematics and Computer Science, Saint Petersburg State University, Russia

Alexander Okhotin
Department of Mathematics and Computer Science, Saint Petersburg State University, Russia

Abstract

A parallel algorithm for enumerating parse trees of a given string according to a fixed context-free grammar is defined. The algorithm computes the number of parse trees of an input string; more generally, it applies to computing the weight of a string in a weighted grammar. The algorithm is first implemented on an arithmetic circuit of depth $O((\log n)^2)$ with $O(n^6)$ elements. Then, it is improved using fast matrix multiplication to use only $O(n^{5.38})$ elements, while preserving depth $O((\log n)^2)$.

2012 ACM Subject Classification Theory of computation → Grammars and context-free languages; Theory of computation → Parallel algorithms

Keywords and phrases Context-free grammars, weighted grammars, parsing, parallel algorithms, matrix multiplication

Digital Object Identifier 10.4230/LIPIcs.MFCS.2023.67

Funding This work was supported by the Russian Science Foundation, project 23-11-00133.

1 Introduction

The classical cubic-time Cocke–Kasami–Younger parsing algorithm can be adapted to solving such problems as counting the number of parse trees of an input string, or computing the probability of an input string in a probabilistic grammar – and, more generally, computing the weight of a string in a weighted grammar. This is possible, because the algorithm considers the substrings in a certain predictable order, and investigates each partition of each substring exactly once, and hence never enumerates the same parse tree twice.

Another classical parsing algorithm discovered by Valiant [9] computes the same values in the time of matrix multiplication $O(n^\omega)$, where $\omega < 3$. In spite of quite a nontrivial order of processing the substrings and their partitions, it maintains the same property: every partition of every substring is considered only once, there is no double counting. Hence, the algorithm can be applied to computing weights in any ring [2], see Okhotin [6] for an elementary presentation.

These are the sequential parsing algorithms. Yet another classical parsing method is a parallel algorithm discovered independently by Brent and Goldschlager [3] and by Rytter [7]. Their algorithm, if implemented on a parallel architecture with concurrent reads and exclusive writes, works in time $O((\log n)^2)$ and uses $O(n^6)$ processors. If implemented on a uniform family of Boolean circuits (the model of parallel computation assumed in this paper), the algorithm by Brent, Goldschlager and Rytter uses circuits of depth $O((\log n)^2)$ with $O(n^6 \log n)$ elements, that is, NC^2-circuits.

What is important to know about the Brent–Goldschlager–Rytter parallel parsing algorithm is that it *considers the same trees multiple times*, and this happens in different branches of parallel computation: each tree is obtained in different ways by combining different subtrees. An example of this is given in Figure 1: here the input string is $a_1a_2a_3a_4$, and the algorithm obtains its single parse tree in two ways. One way is to produce a subtree of a_1a_2 with root A, another tree with root S with a "hole" instead of A and with the

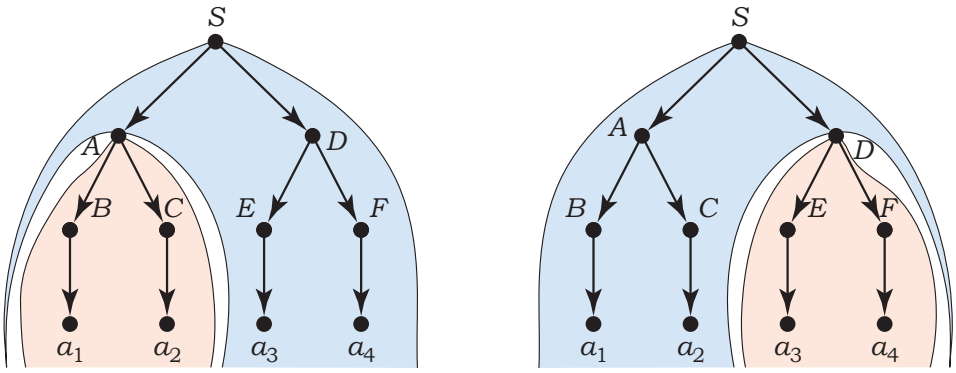

Figure 1 Multiple partitions of a tree into a "tree with a hole" and a subtree.

leaves a_3a_4 to the right of the hole, and then to fill this hole with the former subtree, see Figure 1(left). The other way is similar, using a subtree of a_3a_4 with root D to fill a hole in another "tree with a hole". The algorithm will construct these subtrees in different branches of parallel computation, each time deducing the existence of a parse tree of $a_1a_2a_3a_4$. If the algorithm were modified to count the number of trees, then the tree in the figure will be counted twice, as the algorithm has no means to check whether these trees were the same or not. Hence, the algorithm can only determine the existence of a tree, but cannot compute a reliable count of the number of trees, as well as cannot compute the weight of the input string in most semirings.

In a recent paper by Bakinova et al. [1], a simple variant of the Brent–Goldschlager–Rytter algorithm for computing *the number of trees modulo two* was presented: the algorithm uses a Boolean circuit of the same depth $O((\log n)^2)$, whereas the number of elements is $O(n^7)$. This algorithm considers each tree only once, and therefore it can be adapted to compute the precise number of trees, if the Boolean circuit is replaced with an arithmetic circuit. In the same way, one can compute the weight in any semiring. This raises a natural question: could the number of elements in $(\log n)^2$-depth circuits solving this problem be reduced to $O(n^6)$ or below?

This paper presents a new parallel algorithm for enumerating all parse trees of an input string, which maintains the same circuit depth $O(\log^2 n)$, and is more efficient in terms of the number of elements. First, in Section 3, the problem is solved using a circuit with $O(n^6)$ elements. The algorithm is formulated as computing the number of trees, and then it is adapted to compute the weight of an input string in any semiring.

Next, in Section 4, this algorithm is improved by using fast matrix multiplication. The algorithm elaborates the algorithm in Section 3 in the same way as Valiant's [9] algorithm elaborates the Cocke–Kasami–Younger algorithm: it computes the same values as the original algorithm, but rearranges the order of computation so that the arithmetical operations are combined into matrix multiplications. As compared to Valiant's [9] algorithm, the proposed algorithm has to handle trees with holes, which require a more difficult rearrangement of the computation.

By abuse of notation, let $\omega \geqslant 2$ be a number, such that for every n and for every ring there is a circuit of depth $O(\log n)$ with $O(n^\omega)$ elements computing operations in the ring, which multiplies any two $n \times n$ matrices. Strassen's algorithm [8] can be implemented on such a circuit. Also it is known that the Coppersmith–Winograd [4] algorithm can be implemented on a logarithmic-depth circuit as well – see, e.g., Gazit and Miller [5]. Then

the new algorithm for enumerating parse trees maintains circuit depth $O(\log^2 n)$, whereas the number of elements in it is reduced to $O(n^{\omega+3})$, that is, does not exceed $O(n^{5.38})$. The algorithm is applicable to computing the weight of a string in any ring.

2 Definitions

▶ **Definition 1.** *A grammar in the Chomsky normal form is a quadruple $G = (\Sigma, N, R, S)$, where Σ is a finite alphabet; N is a finite set of nonterminal symbols; R is a finite set of grammar rules, each of the form $A \to BC$, with $A, B, C \in N$, or $A \to a$, with $A \in N$ and $a \in \Sigma$; $S \in N$ is the initial symbol.*

▶ **Definition 2.** *Let $G = (\Sigma, N, R, S)$ be a grammar in the Chomsky normal form. A parse tree in G is a tree, in which the root is labelled with S, every internal vertex is labelled with any nonterminal symbol $A \in N$, and every leaf is labelled with a symbol $a \in \Sigma$. Every vertex $A \in N$ may either have two sons labelled B and C, for some rule $A \to BC$, or one son labelled a, if there is a rule $A \to a$. Vertices with two sons have their sons ordered, which induces an order on the leaves. If $w \in \Sigma^+$ is the string formed by the leaves of a tree, this is a* parse tree of w.

The language defined by the grammar, denoted by $L(G)$ is the set of all strings $w \in \Sigma^+$ that have at least one parse tree.

Under this definition, it is easy to generalize grammars to weighted grammars over a semiring.

▶ **Definition 3.** *Let $G = (\Sigma, N, R, S)$ be a grammar in the Chomsky normal form, let $\mathcal{S}$ be a semiring, and let $\Phi\colon R \to \mathcal{S}$ be a function assigning weights in $\mathcal{S}$ to rules. The triple $(G, \mathcal{S}, \Phi)$ is called a weighted grammar.*

Then, the weight of a parse tree is the product of weights of all rules used in the tree. The weight of a string $w \in \Sigma^*$ is the sum of weights of all parse trees of w. The grammar defines a mapping from Σ^* to $\mathcal{S}$.

If $\mathcal{S}$ is the Boolean semiring and all rules have weight 1, then a weighted grammar become an ordinary grammar, which defines the language $\{\, w \mid w \text{ has weight } 1 \,\}$; and if $\mathcal{S}$ is a two-element field, the weighted grammar becomes a GF(2)-grammar. If $\mathcal{S}$ is the set of non-negative integers with addition and multiplication, and all rules have weight 1, then the weighted grammar defines the number of parse trees in G. If $\mathcal{S}$ is the set of probabilities in $[0, 1]$ with addition and multiplication, this is a probabilistic grammar.

This paper uses arithmetic circuits as a model of parallel computation.

▶ **Definition 4.** *An* arithmetic circuit *over natural numbers $\mathbb{N} = \{0, 1, 2, \ldots\}$ is a directed acyclic graph, in which all nodes with no incoming arcs are called the* inputs *and are labelled with distinct input variables, and each of the rest of the nodes, called* elements*, has two incoming arcs and is labelled either with addition, or with multiplication. Input variables are natural numbers, each element computes a function of the input variables. The circuit is said to compute the function computed in a designated* output element*.*

An arithmetic circuit over integers $\mathbb{Z}$ additionally may use the subtraction operation. Arithmetic circuits over a semiring $\mathcal{S}$ and over a ring $\mathcal{R}$ generalize these circuits by computing values in the corresponding algebraic structures.

The depth *of a circuit is the length of the longest path.*

A parsing algorithm is implemented on a circuit, which is constructed for a fixed grammar $G = (\Sigma, N, R, S)$ and a fixed length of input strings n. The circuit has $|\Sigma| \cdot n$ inputs, each indicating whether there is a particular symbol in a particular position. The output element gives the number of parse trees of the given string.

Since a different circuit is needed for each input length, the following standard uniformity condition is assumed.

▶ **Definition 5.** *A family of arithmetic circuits is* uniform*, if there exists a deterministic logarithmic-space Turing machine that, for every input length n given in unary notation, produces the circuit for input strings of length n.*

3 A circuit with $O(n^6)$ elements

▶ **Theorem 6.** *For every grammar $G = (\Sigma, N, R, S)$ in the Chomsky normal form there is a uniform family of arithmetic circuits, for each length of input strings $n \geqslant 1$, which computes the number of parse trees of an input string of length n, has at most $|N|^2 \cdot |R| \cdot n^6$ elements and is of depth at most $21 \log_2^2 n + 7 \log_2 n \log_2 |R|$.*

Proof. Denote $w = a_1 \dots a_n$. The circuit consists of the following elements.

- An element $A(i,j)$, for all $A \in N$ and i, j, with $0 \leqslant i < j \leqslant n$, computes the number of parse trees from A for the string $a_{i+1} \dots a_j$; these trees are schematically presented as in Figure 2(left).
  An example of such a tree for $A(0,2)$ can be seen in Figure 1(left).
- An element $\frac{A}{D}(i,k,\ell,j)$, for all $A, D \in N$ and $0 \leqslant i \leqslant k < \ell \leqslant j \leqslant n$, computes the number of parse trees for the string $a_{i+1} \dots a_j$ from A, with a hole instead of a subtree $a_{k+1} \dots a_\ell$ from D, depicted as in Figure 2(middle).
  Such a "tree with a hole" for $\frac{S}{A}(0,0,2,4)$ can be seen in Figure 1(left).
- An element $\frac{A}{D}\big|_q(i,k,\ell,j)$, for all $A, D \in N$ and $0 \leqslant i \leqslant k < \ell \leqslant j \leqslant n$, with $(k-i) + (j-\ell) > 0$ and $q \geqslant 0$, computes the number of parse trees for the string $a_{i+1} \dots a_j$ from A, with a hole instead of a subtree $a_{k+1} \dots a_\ell$ from D, such that at the first branching the subtree with the hole contains at most q leaves, as shown in Figure 2(right).
  The tree in the earlier Figure 1(left) is too small to illustrate this, because after the first branching the subtree with the hole is empty (that is, contains 0 leaves).
- Some extra elements are used to compute the values of the named elements listed above: e.g., each element $A(i,j)$ is computed by a circuit that consists of $O(n^3)$ extra elements, etc.

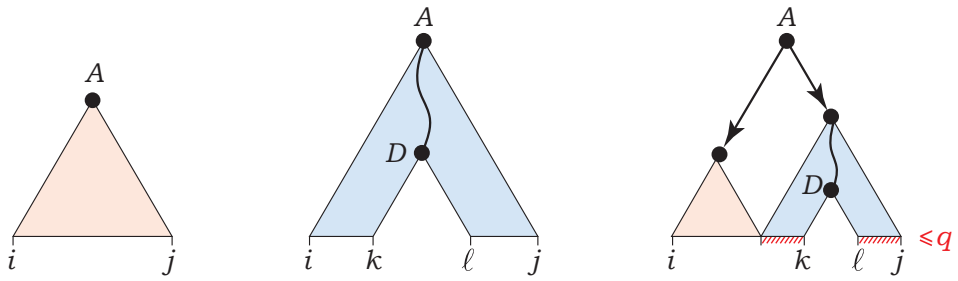

Figure 2 (left) $A(i,j)$; (middle) $\frac{A}{D}(i,k,\ell,j)$; (right) $\frac{A}{D}\big|_q(i,k,\ell,j)$.

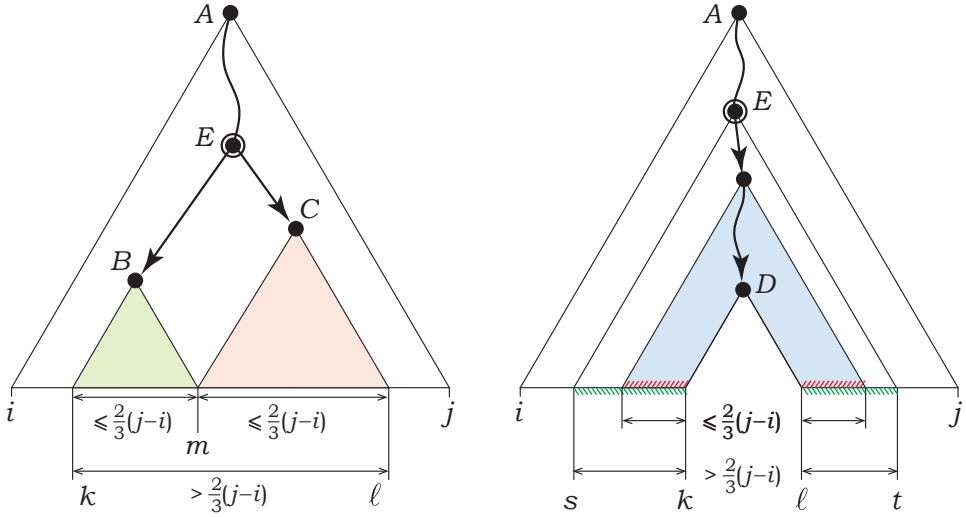

Figure 3 (left) Intermediate vertex E in a tree $A(i,j)$; (right) Intermediate vertex E on the path to the hole in a tree $\frac{A}{D}(i,k,\ell,j)$.

Denote the inputs of the circuit by $In(a,i)$, for all $a \in \Sigma$ and $1 \leqslant i \leqslant n$, where $In(a,i) = 1$ if the i-th symbol of an input string is equal to a and $In(a,i) = 0$ otherwise. Elements of the circuit corresponding to one-symbol strings are initialized directly from the inputs as $A(i-1,i) = \sum_{A \to a \in R} In(a,i)$, for all $A \in N$ and $1 \leqslant i \leqslant n$. Also, trees with a hole containing no leaves are defined for every pair of positions: $\frac{A}{D}(i,i,j,j) = 1$, for $A, D \in N$ and $0 \leqslant i < j \leqslant n$.

The desired number of parse trees of w is computed by the element $S(0,n)$.

Consider an arbitrary parse tree τ, corresponding to $A(i,j)$ which has $j-i$ leaves. Denote by E the deepest vertex having more than $\frac{2}{3}(j-i)$ leaves in its subtree. Such a vertex exists, moreover it is unique. Denote by B and C the children of E. Suppose that the subtree of B corresponds to leaves $a_{k+1} \ldots a_m$, and the subtree of C corresponds to leaves $a_{m+1} \ldots a_\ell$, as illustrated in Figure 3(left). Taking into account that E is the deepest vertex with more than $\frac{2}{3}(j-i)$ leaves, both B and C may only have at most $\frac{2}{3}(j-i)$ leaves each, in other words, $\ell - k > \frac{2}{3}(j-i)$ and $\max(m-k, \ell-m) \leqslant \frac{2}{3}(j-i)$.

Hence, the number of parse trees for $A(i,j)$ can be calculated as

$$A(i,j) = \sum_{\substack{E \to BC \in R \\ k,\ell,m: i \leqslant k < m < \ell \leqslant j \\ \ell - k > \frac{2}{3}(j-i) \\ \max(m-k, \ell-m) \leqslant \frac{2}{3}(j-i)}} \frac{A}{E}(i,k,\ell,j) \cdot B(k,m) \cdot C(m,\ell)$$

Every parse tree τ has been taken into account, because there is such a vertex E in every tree. Furthermore, the vertex E is unique for the given parse tree τ, therefore every τ is taken into account only once.

The above formula for $A(i,j)$ is implemented on a circuit as follows. For each triple (k,ℓ,m), one extra element is used to multiply $\frac{A}{E}(i,k,\ell,j)$ by $B(k,m)$, and another extra element multiplies the result by $C(m,\ell)$. It remains to sum up the resulting products: as there are $O(n^3)$ of them, up to $O(n^3)$ extra elements are needed, and they are organized into a tree of logarithmic depth.

Similar arguments allow us to deduce a formula for $\frac{A}{D}(i,k,\ell,j)$ and to implement it on a circuit. Denote by E the deepest vertex on the path from A to D with more than $\frac{2}{3}(k-i+j-\ell)$ leaves in its subtree, and denote these leaves by $a_{s+1}\dots a_k$ and $s_{\ell+1}\dots s_t$, see Figure 3(right). Then

$$\frac{A}{D}(i,k,\ell,j) = \sum_{\substack{E\in N\\ s:i\leqslant s\leqslant k\\ t:\ell\leqslant t\leqslant j\\ k-s+t-\ell>\frac{2}{3}(k-i+j-\ell)}} \frac{A}{E}(i,s,t,j)\cdot \frac{E}{D}\Big|_{\frac{2}{3}(k-i+j-\ell)}(s,k,\ell,t)$$

Indeed, the intermediate positions s and t are chosen so that the subtree E has more than $\frac{2}{3}(k-i+j-\ell)$ leaves. On the other hand, the definition of $\frac{E}{D}|_{\frac{2}{3}(k-i+j-\ell)}$ guarantees that no other vertex on the path from E to D has more than $\frac{2}{3}(k-i+j-\ell)$ leaves. Hence, E is indeed the deepest vertex on the path from A to D with more than $\frac{2}{3}(k-i+j-\ell)$ leaves. Since such a vertex is unique for each tree, every parse tree for $\frac{A}{D}(i,k,\ell,j)$ is counted only once.

A circuit computing $\frac{A}{D}(i,k,\ell,j)$ by the above formula uses $O(n^2)$ extra elements.

In order to compute $\frac{E}{D}\big|_q(s,k,\ell,t)$, let us consider the rule used at the root E of a tree with a hole D. Let $E\to BC$ be this rule, and let m be the position in the input such that leaves up to a_m belong to subtree B, and leaves beginning with a_{m+1} belong to subtree C. Then $\frac{E}{D}\big|_q(s,k,\ell,t)$ can be computed as follows:

$$\frac{E}{D}\Big|_q(s,k,\ell,t) = \sum_{\substack{E\to BC\in R\\ m:s<m\leqslant k\\ k-m+t-\ell\leqslant q}} B(s,m)\cdot\frac{C}{D}(m,k,\ell,t) + \sum_{\substack{E\to BC\in R\\ m:\ell\leqslant m<t\\ k-s+m-\ell\leqslant q}} \frac{B}{D}(s,k,\ell,m)\cdot C(m,t)$$

A circuit computing an element $\frac{E}{D}\Big|_q(s,k,\ell,t)$ by this formula uses $O(n)$ extra elements.

The entire circuit consists of all the above elements and computes the desired values.

Number of elements in the circuit. There are $O(n^2)$ elements $A(i,j)$ and computing each of them takes $O(n^3)$ extra elements. Similarly, each of the $O(n^4)$ elements $\frac{A}{D}(i,k,\ell,j)$ takes $O(n^2)$ extra elements to compute, whereas the computation of each of the $O(n^5)$ elements $\frac{E}{D}(s,k,\ell,t)|_q$ takes $O(n)$ extra elements. This accounts for $O(n^6)$ elements in the entire circuit; the calculation of the dependence on the grammar is omitted due to space constraints.

Depth of the circuit. The elements $A(i,j)$ directly depend on elements, which have at most $\frac{2}{3}(j-i)$ leaves in their subtrees.

For each element $\frac{A}{D}(i,k,\ell,j)$, let $\alpha = k-i+j-\ell$ be the number of leaves in its subtree. Then it directly depends on elements $\frac{E}{D}(\cdot,\cdot,\cdot,\cdot)$ with at most $\frac{1}{3}\alpha$ leaves, and elements $\frac{E}{D}\big|_q(\cdot,\cdot,\cdot,\cdot)$ with at most α leaves, and with $q\leqslant\frac{2}{3}\alpha$. The latter element depends on $B(\cdot,\cdot)$ with at most α leaves, and on $\frac{C}{D}(\cdot,\cdot,\cdot,\cdot)$ with at most $\frac{2}{3}\alpha$. Overall, $\frac{A}{D}$ depends on elements with at most $\frac{2}{3}\alpha$ leaves in three steps.

Each element $\frac{E}{D}\big|_q(s,k,\ell,t)$ depends on elements $\frac{A}{D}$ and B with fewer than α leaves.

Thus, for an element of every type, whether it is $A(\cdot,\cdot)$, $\frac{E}{D}\big|_q(\cdot,\cdot,\cdot,\cdot)$ or $\frac{A}{D}(\cdot,\cdot,\cdot,\cdot)$, after a quadruple application of the rules, the number of leaves is reduced at least by a factor of $\frac{3}{2}$.

At each level of these dependencies, a sum of multiple values is computed using a binary tree of logarithmic depth in the number of arguments, whence overall depth $O(\log^2 n)$. ◀

The above circuit actually performs operations in the ring of natural numbers, adding constant 1 for each parse tree accounted for. This can be generalized to compute the value of a string in a weighted grammar over every semiring.

▶ **Theorem 7.** *For every weighted grammar $G = (\Sigma, N, R, S)$ in the Chomsky normal form, with weights in a semiring $\mathcal{S}$, there is a uniform family of arithmetic circuits computing elements over $\mathcal{S}$, for each length of input strings $n \geqslant 1$, which computes the value of an input string of length n in the semiring, has $O(n^6)$ elements and is of depth $O(\log^2 n)$.*

4 A smaller circuit using fast matrix multiplication

In the circuit constructed in Theorem 6, all elements $A(\cdot,\cdot)$ are computed using $\Theta(n^5)$ operations in total, and computation of elements of both types $\frac{A}{D}$ and $\frac{A}{D}\big|_q$ takes $\Theta(n^6)$ operations. Hence, in order to reduce the total number of elements in the circuit, computation of elements $\frac{A}{D}$ and $\frac{A}{D}\big|_q$ should be optimized. It turns out that such an optimization could be done via fast matrix multiplication.

4.1 Matrix multiplication for $\frac{A}{D}$

Let us show how fast matrix multiplication can be used to compute the same values $\frac{A}{D}(i,k,\ell,j)$ using fewer elements.

In the following, matrices with rows and columns indexed by *substrings*, that is, pairs of two positions in the string, will be considered. Denote $I = (i,j)$ and $|I| = j - i$; similarly, let $K = (k,\ell)$, $|K| = \ell - k$, $S = (s,t)$, $|S| = t - s$. For every pair of non-terminal symbols A, D let us define a matrix of size $\frac{n(n+1)}{2} \times \frac{n(n+1)}{2}$, consisting of elements $\frac{A}{D}[I][K] = \frac{A}{D}(i,k,\ell,j)$ with $i \leqslant k < \ell \leqslant j$ and $\frac{A}{D}[I][K] = 0$ otherwise. Then the formula for computing the value $\frac{A}{D}(i,k,\ell,j)$ can be rewritten in the new notation as follows.

$$\frac{A}{D}[I][K] = \sum_{\substack{E \in N \\ S : K \subset S \subset I \\ |S|-|K| > \frac{2}{3}(|I|-|K|)}} \frac{A}{E}[I][S] \cdot \frac{E}{D}\Big|_{\frac{2}{3}(|I|-|K|)}[S][K]$$

In this notation, the formula resembles a matrix product, and suggests that some elements of a matrix $\frac{A}{D}[I][K]$, with rows indexed by I and columns indexed by K, could be computed as a product of some submatrices of $\frac{A}{E}$ and of $\frac{E}{D}|_{\frac{2}{3}(|I|-|K|)}$ (as long as $|I|-|K|$ is fixed). However, an algorithm cannot compute the product of these matrices in one step, since the elements corresponding to the longer substrings indirectly depend on the elements corresponding to the shorter substrings. For this reason, those matrices shall be split into several submatrices, which shall be multiplied separately at different stages of the computation.

The matrix $\frac{A}{D}[I][K]$ is split into submatrices with fixed $|I|$ and $|K|$. The submatrix with $|I| = \alpha$ and $|K| = \beta$, is denoted by $\left\{\frac{A}{D}[I][K]\right\}_{|I|=\alpha,|K|=\beta}$, and is of size $(n-\alpha+1)\times(n-\beta+1)$. This submatrix is computed as the following sum of matrix products.

$$\left\{\frac{A}{D}[I][K]\right\}_{\substack{|I|=\alpha \\ |K|=\beta}} = \sum_{\substack{E \in N \\ \gamma : \frac{2}{3}\alpha + \frac{1}{3}\beta < \gamma < \alpha}} \left\{\frac{A}{E}[I][S]\right\}_{\substack{|I|=\alpha \\ |S|=\gamma}} \cdot \left\{\frac{E}{D}\Big|_{\frac{2}{3}(\alpha-\beta)}[S][K]\right\}_{\substack{|S|=\gamma \\ |K|=\beta}}$$

(the condition $\frac{2}{3}\alpha + \frac{1}{3}\beta < \gamma < \alpha$ is obtained from the inequalities $|S| - |K| > \frac{2}{3}(|I| - |K|)$ and $|S| < |I|$)

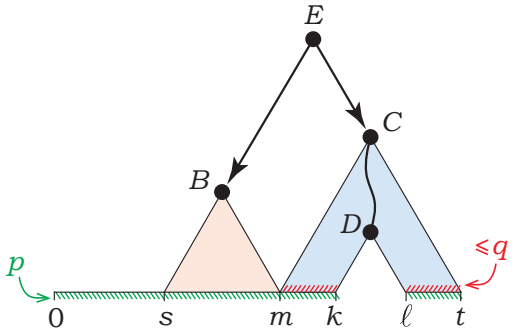

Figure 4 The computation of $\frac{E}{D}\big|_q(s,k,\ell,t)$ in the case of branching to the left.

The submatrices being multiplied include some elements $\frac{A}{E}[I][S]$ with $S \subsetneq I$, and $\frac{E}{D}\big|_{\frac{2}{3}(\alpha-\beta)}[S][K]$ with $K \subsetneq S$, which are not in the formula for $\frac{A}{D}[I][K]$; however, since these elements are zero by the definition of the matrices, these unintended terms in the sums have no effect on the result.

Also note that the computation of the element $\frac{A}{D}[I][K]$ with $\alpha-\beta$ leaves, requires only the elements $\frac{A}{E}[\cdot][\cdot]$ with at most $\frac{1}{3}(\alpha-\beta)$ leaves, and the elements $\frac{E}{D}\big|_q[\cdot][\cdot]$ with at most $\alpha-\beta$ leaves. The overall order of computation will be set accordingly, so that by the time the product is computed, its arguments will have already been computed.

4.2 Matrix multiplication for elements $\frac{E}{D}\big|_q$

Let us now consider the elements $\frac{E}{D}\big|_q(s,k,\ell,t)$. As it was mentioned above, each of these elements can be calculated as

$$\frac{E}{D}\Big|_q(s,k,\ell,t) = \sum_{\substack{E\to BC\in R\\ m:s<m\leqslant k\\ k-m+t-\ell\leqslant q}} B(s,m)\cdot\frac{C}{D}(m,k,\ell,t) + \sum_{\substack{E\to BC\in R\\ m:\ell\leqslant m<t\\ k-s+m-\ell\leqslant q}} \frac{B}{D}(s,k,\ell,m)\cdot C(m,t)$$

Let us show how fast matrix multiplication can be used in order to compute the first sum. The second sum is computed similarly.

For each element $\frac{E}{D}\big|_q(s,k,\ell,t)$, define a value p, which equals $p=k+t-\ell$, as it is shown in Figure 4. The proposed order of computations is designed in a such way that elements $\frac{E}{D}\big|_q$ with the same p are calculated together.

Let us fix $p>0$ and consider computations of $\frac{E}{D}\big|_q(s,k,\ell,t)$ with $t-\ell+k=p$. Denote a matrix $U^{p,B}$ of size $p\times p$, such that for all $s,m=0,1,\dots p-1$, $U^{p,B}[s][m]=B(s,m)$ if $s<m$, and $U[s][m]=0$ otherwise. In other words, U is a top-left corner of the matrix B. Also denote a matrix $W^{p,q,C,D}$ of size $p\times n^2$, where the first coordinate is $m=0,1\dots p-1$, and the second is a triple of k,ℓ,t, such that $k+t-\ell=p$, and $W^{p,q,C,D}[m][k,\ell,t]=\frac{C}{D}(m,k,\ell,t)$ if $m\leqslant k\leqslant\ell\leqslant t$ and $m\geqslant p-q$, and $W^{p,q,C,D}[m][k,\ell,t]=0$ otherwise.

It is claimed that the product of $U^{p,B}$ and $W^{p,q,C,D}$ contains the exact values needed for the computation of $\frac{E}{D}\big|_q$.

$$(U^{p,B}\cdot W^{p,q,C,D})[s][k,\ell,t] = \sum_{m=0}^{p-1} U^{p,B}[s][m]\cdot W^{p,q,C,D}[m][k,\ell,t]$$

This sum has $U^{p,B}[s][m] = 0$ for $m \leqslant s$, and $W^{p,q,C,D}[m][k,\ell,t] = 0$ for $m \geqslant k$ and for $m \leqslant p-q$. Since $p = k+t-\ell \geqslant k$, the upper limit $m < p$ becomes redundant. Therefore, the range of the sum can be reduced as follows.

$$(U^{p,B} \cdot W^{p,q,C,D})[s][k,\ell,t] = \sum_{m=0}^{p-1} U^{p,B}[s][m] \cdot W^{p,q,C,D}[m][k,\ell,t] = \\ = \sum_{\substack{s<m\leqslant k \\ p-q\leqslant m}} B(s,m) \cdot \frac{C}{D}(m,k,\ell,t)$$

Hence, in order to compute all elements $\frac{E}{D}\big|_q(s,k,\ell,t)$ with $t-\ell+k=p$, it is needed to iterate over all pairs of non-terminals B, C, such that $E \to BC \in R$, construct the corresponding matrices $U^{p,B}$ and $W^{p,q,C,D}$, and sum up their products. However, elements in each of these matrices indirectly depend on each other, so it is again impossible to build and multiply these matrices in one step. For this reason, matrices $U^{p,B}$ and $W^{p,q,C,D}$ are never built wholly, and their products are gradually calculated by multiplying their submatrices of various size. The higher the value of p, the later the entire product $U^{p,B} \cdot W^{p,q,C,D}$ will be obtained. Let us also mention that both matrices $U^{p,B}$ and $W^{p,q,C,D}$, consist of elements B, $\frac{C}{D}$ and $\frac{E}{D}\big|_q$ or zeros, and so does their product $U^{p,B} \cdot W^{p,q,C,D}$. Hence, there is no need to actually store these matrices – it is only a way to represent computations performed over the named elements.

▶ Remark 8. From the definition of $U^{p,B}$ it follows that an arbitrary element $U^{p,B}[s][m]$ has $m-s$ leaves in its subtree. In other words, the elements with α leaves in their subtrees lay on the α-th diagonal, in the numeration where the main diagonal has number 0.

▶ Remark 9. Similarly, every element $W^{p,q,C,D}[m][k,\ell,t]$ has $\min(k-m+t-\ell, q) = \min(p-m, q)$ leaves in its subtree. In other words, the elements in the α-th row from the bottom of $W^{p,q,C,D}$ have $\min(\alpha, q)$ leaves in their subtrees.

For a fixed value of p, the matrix $U^{p,B}$ is split into disjoint submatrices, which will be multiplied by the algorithm. These matrices are denoted by $L(t,j)$ and $R(t,\ell,j)$, where the number t refers to a large square of size $2^t \times 2^t$, in which the corresponding submatrix is contained. The L-matrices are in the large squares centered at the main diagonal, whereas the R-matrices are in the large squares to the right of the main diagonal. This decomposition is illustrated in Figure 5.

It is important to note that even though the matrices $U^{p,B}$ for different p are corners of the same matrix B, the matrix $U^{p,B}$ is split anew for each p, and there is no relation between the decompositions for different p.

▶ **Definition 10.** *For all $t = 0, \dots \log_2 p - 1$ and $j = 1, \dots, 2^t - 1$, let n_j be the maximal power of two that divides j. Denote by $L(t,j)$ the submatrix of matrix $U^{p,B}$ of size $n_j \times n_j$ with left bottom corner at $(p-2^t-j-1, p-2^t-j)$.*

$$L(t,j) = U^{p,B}[p-2^t-j-n_j, \dots, p-2^t-j-1][p-2^t-j, \dots, p-2^t-j+n_j-1]$$

Each $L(t,j)$ is in the t-th large square, its left corner is next to the main diagonal, and j is the number of the row within the large square, numbered from the bottom.

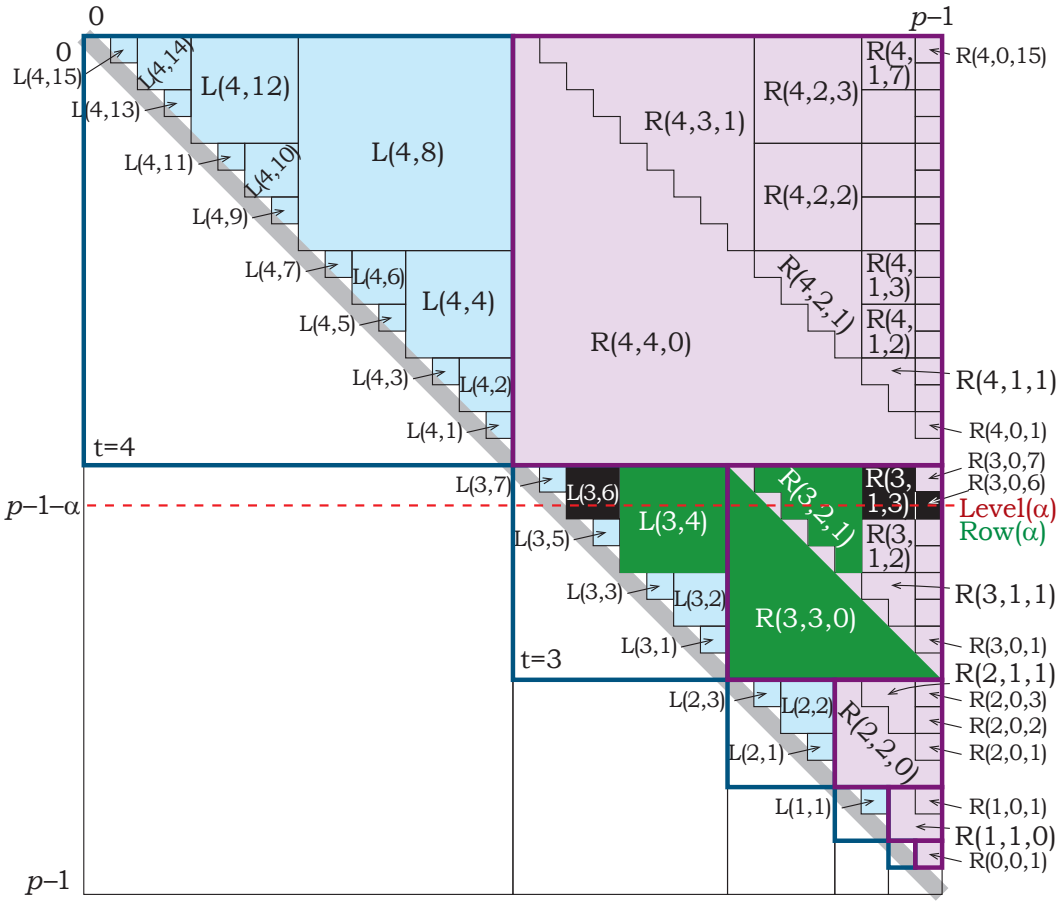

Figure 5 Partition of $U^{p,B}$, with $p = 32$, and the sets $Level(\alpha)$ and $Row(\alpha)$ for $\alpha = 14$.

▶ **Definition 11.** *For all* $t = 0, \dots \log_2 p - 1$, $\ell = 0, \dots t - 1$ *and* $j = 1, \dots, 2^{t-\ell} - 1$ *denote by* $R(t, \ell, j)$ *the submatrix of matrix* $U^{p,B}$ *of size* $2^\ell \times 2^\ell$ *with left bottom corner in* $(p - 2^t - 2^\ell j - 1,\ p - 2^{\ell+1} + 1)$.

$$R(t, \ell, j) = U[p - 2^t - 2^\ell(j+1), \dots, p - 2^t - 2^\ell j - 1][p - 2^{\ell+1} + 1, \dots, p - 2^\ell]$$

Here, if $j = 1$, *then all the entries below the main diagonal in matrix* $R(t, \ell, j)$ *are set to zero.*

In addition to that, for all $t = 0, \dots \log_2 p - 1$ *denote by* $R(t, t, 0)$ *the submatrix of matrix* $U^{p,B}$ *of size* $2^t \times 2^t$ *with left top corner at* $(p - 2^{t+1},\ p - 2^t)$, *where all the entries above the main diagonal are set to zero. If* $0 \leqslant a, b < 2^t$ *are the indices of this submatrix, then*

$$R(t, t, 0)[a][b] = \begin{cases} U[p - 2^{t+1} + a][p - 2^t + b], & a \geqslant b \\ 0, & a < b \end{cases}$$

The lower triangular part of the t-th R-square is $R(t, t, 0)$. An R-square is split into blocks of columns, with each ℓ-th block from the right of width 2^ℓ. Then, a submatrix $R(t, \ell, j)$ is the j-th submatrix down in the ℓ-th block of columns. The bottom submatrix in each block of columns (one with $j = 1$) is upper triangular. All these details are illustrated in the figure. Submatrices $L(\cdot, \cdot)$ and $R(\cdot, \cdot, \cdot)$ form a partition of the upper triangle of matrix U, which is the non-zero part of U.

For each $\alpha = 0, \dots p-1$ denote by $Row(\alpha)$ the set of all submatrices in the partition, which do intersect with the $(p-1-\alpha)$-th row of $U^{p,B}$ (that is, the $(\alpha+1)$-th row from the bottom). The product of each of these matrices by the corresponding stripe of $W^{p,q,C,D}$ is required for the computation of $(p-1-\alpha)$-th row of $U \cdot W$.

▶ **Definition 12.** *Let $\alpha = 2^t + r$ with $0 \leqslant r < 2^t$. Let r be represented as a sequence of binary digits as $r = \sum_{i=1}^{k} 2^{r_i}$, where $r_1 < \ldots < r_k$ are the numbers of all positive bits. Then $Row(\alpha)$ consists of k matrices $L(t, \cdot)$ and $\lfloor \log_2 r \rfloor + 2$ matrices $R(t, \cdot, \cdot)$. To be more precise,*

$$Row(\alpha) = \left\{ L\left(t, \sum_{i=s}^{k} 2^{r_i}\right) \middle| s = 1, \dots, k \right\} \cup \left\{ R\left(t, \ell, \left\lfloor \frac{r}{2^\ell} \right\rfloor\right) \middle| \ell = 0, \dots, \lfloor \log_2 r \rfloor \right\} \cup \{R(t,t,0)\}$$

Let us now denote by $Level(\alpha)$ the set of all matrices from $Row(\alpha)$ whose bottom row intersects with the $(p-1-\alpha)$-th row of $U^{p,B}$. All matrices from $Level(\alpha)$ are to be multiplied by the corresponding stripes of W simulatenously at a certain level of the circuit.

▶ **Definition 13.** *For each $\alpha = 1, \dots p-1$, let $\alpha = 2^t + r$ with $0 \leqslant r < 2^t$. If $r > 0$, then the set of matrices $Level(\alpha)$ consists of the following matrices.*

$$Level(\alpha) = \{L(t,r)\} \cup \left\{ R\left(t, \ell, \frac{r}{2^\ell}\right) \middle| \ell = 0, \dots \lfloor \log_2 r \rfloor, r \vdots 2^\ell \right\}$$

And if $r = 0$, then $Level(\alpha)$ contains only the big lower-triangular matrix R.

$$Level(\alpha) = \{R(t,t,0)\}$$

In Figure 5, the sets $Row(\alpha)$ and $Level(\alpha)$ are illustrated for $\alpha = 14$.

▶ **Lemma 14.** *Every matrix of the partition lies in exactly one set $Level(\alpha)$.*

▶ **Lemma 15.** *Every element of every matrix from $Level(\alpha)$ has at most α leaves.*

▶ **Lemma 16.** *Every matrix from $Level(\alpha)$ is multiplied by some elements of W, each having at most $\min(\alpha, q)$ leaves.*

▶ **Lemma 17.** *For every α, every matrix from $Row(\alpha)$ lies in set $Level(\beta)$, for some $\beta \leqslant \alpha$.*

In other words, in the circuit, every matrix from the partition shall be multiplied by the corresponding stripe of W before their product would be needed.

4.3 Circuit

The circuit is organized into $6\lceil \log_{\frac{3}{2}} n \rceil$ levels, each of depth at most $\log n$. At each group of six levels numbered $6t$, $6t+1$, …, $6t+5$, all elements with α in the range $\left(\frac{3}{2}\right)^t \leqslant \alpha < \left(\frac{3}{2}\right)^{t+1}$ are computed. This is done in six stages, according to the following plan.

0. At the level $\tau = 6t$, elements $A(\cdot, \cdot)$ with α leaves, where $\left(\frac{3}{2}\right)^t \leqslant \alpha < \left(\frac{3}{2}\right)^{t+1}$ are computed directly via summation. These elements depend on B, C and $\frac{A}{E}$ with $\beta < \left(\frac{3}{2}\right)^t$ leaves – each of which was computed at a level $6t'$ or $6t'+3$, where $t' < t$.

1. At the level $\tau = 6t+1$ for all such α that $\left(\frac{3}{2}\right)^t \leqslant \alpha < \left(\frac{3}{2}\right)^{t+1}$ matrices from $Level^{p,B}(\alpha)$ are multiplied by the corresponding strips of the matrix $W^{p,q,C,D}$; this is done separately for each p. As it was mentioned in Lemma 15, all elements of the matrix $U^{p,B}$ involved in this computation have at most $\alpha < \left(\frac{3}{2}\right)^{t+1}$ leaves, so these elements were computed at levels $6t'$ with $t' \leqslant t$. At the same time, as it follows from Lemma 16, the elements

of $W^{p,q,C,D}$ involved in the computation have at most $\min(q,\alpha) < \left(\frac{3}{2}\right)^t$ leaves, so these elements were computed at levels $6t'+3$ with $t' < t$. All in all, the elements computed at level $\tau = 6t+1$ depend only on elements from earlier levels.

2. At the level $\tau = 6t+2$, elements $\frac{E}{D}\big|_q(\cdot,\cdot,\cdot,\cdot)$ with $q < \left(\frac{3}{2}\right)^t$ and with α leaves, where $\left(\frac{3}{2}\right)^t \leqslant \alpha < \left(\frac{3}{2}\right)^{t+1}$, are computed. For each α, the earlier computed products of matrices from the set $Row^{p,B}(\alpha)$ by the corresponding stripes of $W^{p,q,C,D}$ are taken, all their α-th rows are summed up. Next, a sum of the resulting rows is taken over all rules $E \to BC$. Since submatrices from $Row^{p,B}(\alpha)$ form a partition of the non-zero part of α-th row of matrix $U^{p,B}$, the sum is exactly the value of $\frac{E}{D}\big|_q$. As it was proven in Lemma 17, each matrix from $Row^{p,B}(\alpha)$ lies in one of the sets $Level^{p,B}(\beta)$ with $\beta \leqslant \alpha < \left(\frac{3}{2}\right)^{t+1}$. Hence, the elements from level $\tau = 6t+2$ depend on elements from levels $6t'+1$, with $t' \leqslant t$.
3. At the level $\tau = 6t+3$, according to the formulas from Section 4.1, elements $\frac{A}{D}(\cdot,\cdot,\cdot,\cdot)$ with α leaves are computed, where $\left(\frac{3}{2}\right)^t \leqslant \alpha < \left(\frac{3}{2}\right)^{t+1}$.
   To be more precise, for all α in this range and for all $\zeta = 1 \dots n-\alpha$, the following matrix products are computed simultaneously:

$$\left\{\frac{A}{D}[I][K]\right\}_{\substack{|I|=\zeta+\alpha\\|K|=\zeta}} = \sum_{\substack{E\in N\\ \gamma:\frac{2}{3}\alpha+\zeta<\gamma<\zeta+\alpha}} \left\{\frac{A}{E}[I][S]\right\}_{\substack{|I|=\zeta+\alpha\\|S|=\gamma}} \cdot \left\{\frac{E}{D}\Big|_{\frac{2}{3}\alpha}[S][K]\right\}_{\substack{|S|=\gamma\\|K|=\zeta}}$$

   In the left part of the equation there are exactly the elements $\frac{A}{D}$ with α leaves. They depend on elements $\frac{A}{E}$ with at most $\frac{1}{3}\alpha < \left(\frac{3}{2}\right)^t$ leaves, which were computed at some levels $6t'+3$ with $t' < t$. Also there is a dependency on elements $\frac{E}{D}\big|_q$ with fewer than $\alpha < \left(\frac{3}{2}\right)^{t+1}$ leaves, where $q = \frac{2}{3}\alpha < \left(\frac{3}{2}\right)^t$. These elements were computed at levels $\{6t'+2 | t' \leqslant t\} \cup \{6t'+5 | t' < t\}$. Overall, the elements from level $\tau = 6t+3$ depend on elements from levels $6t'+3$ and $6t'+5$ with $t' < t$, as well as on elements from levels $6t'+2$ with $t' \leqslant t$.
4. At the level $\tau = 6t+4$, matrices from the sets $Level^{p,B}(\alpha)$, with $\left(\frac{3}{2}\right)^t \leqslant \alpha < \left(\frac{3}{2}\right)^{t+1}$, are multiplied by the corresponding strips of matrices $W^{p,q,C,D}$, and for all $q \geqslant \left(\frac{3}{2}\right)^t$. The computations are the same as at level $6t+1$, with the only difference that $\min(q,\alpha) < \left(\frac{3}{2}\right)^{t+1}$. So, the elements from level $\tau = 6t+4$ depend on elements from levels $6t'$ and $6t'+3$ with $t' \leqslant t$.
5. At the level $\tau = 6t+5$, the elements $\frac{E}{D}\big|_q$ with $q \geqslant \left(\frac{3}{2}\right)^t$ are computed for all α with $\left(\frac{3}{2}\right)^t \leqslant \alpha < \left(\frac{3}{2}\right)^{t+1}$. This is done similarly to the level $6t+2$, with the computations depending on the elements from levels $6t'+4$ with $t' \leqslant t$.

▶ **Theorem 18.** *For every grammar $G = (\Sigma, N, R, S)$ in the Chomsky normal form and for every string length n, there exists an arithmetic circuit of depth $O((\log n + \log|R|)\log n)$ and with $O(|N|^2 \cdot |R| \cdot n^{\omega+3})$ elements, which computes the number of parse trees of an input string of length n. (where $\omega \geqslant 2$ is a number, such that for every n there is a an arithmetic circuit of depth $O(\log n)$ and with $O(n^\omega)$ elements that computes the product of two $n \times n$ integer matrices)*

Proof.

Depth. Since the elements at every level τ depend only on elements from earlier levels, there are no cycles in the circuit, and the depth of the logical dependencies between elements does not exceed the number of levels, which is $6\lceil \log_{\frac{3}{2}} n \rceil$. At each level, each element is computed either as a sum of at most $n^3|R|$ values, or as a part of a matrix product involving matrices of

size at most $n \times n$. Sums are computed in depth $O(\log n + \log |R|)$ by the standard method. Matrix products are computed in depth $O(\log n)$ using fast matrix multiplication. Hence, the entire circuit has depth $O((\log n + \log |R|) \log n)$.

Number of elements. Elements $B(\cdot, \cdot)$ are computed as sums of $O(|N| \cdot |R| \cdot n^5)$ elements, which is less than $O(|N|^2 \cdot |R| \cdot n^{3+\omega})$.

The computations of all elements $\frac{A}{D}(\cdot, \cdot, \cdot, \cdot)$ are performed at levels $6t + 3$, see stage 3 above. For fixed α and ζ, there are $\alpha + \zeta - (\frac{2}{3}\alpha + \zeta) = \frac{1}{3}\alpha$ different of γ, and for each γ there are $|N|^3$ matrix multiplications corresponding to different nonterminals A, D and E, where each matrix is not larger than $n \times n$. Summing up over all α, ζ, A, D and E gives the following upper bound on the number of operations.

$$const \cdot \sum_{A,D,E \in N} \sum_{\zeta=1}^{n-1} \sum_{\alpha=1}^{n-\zeta} \frac{1}{3} \alpha n^{\omega} = const \cdot |N|^3 \cdot n(n^2 - 1)n^{\omega} = O(|N|^3 \cdot n^{3+\omega})$$

It remains to prove that all matrix multiplications required to compute $\frac{E}{D}\big|_q(\cdot, \cdot, \cdot, \cdot)$ can be done using $O(|N|^2 \cdot |R| \cdot n^{3+\omega})$ elements.

Let firstly fix the parameters p, q, the rule $E \to BC$ and the nonterminal D, and estimate the number of elements used to compute the product of matrices $U^{p,B} \cdot W^{p,q,C,D}$ for these p, q,

Denote the width of the matrix $W^{p,q,C,D}$ by M, it is $O(n^2)$. Fix $t \in \{0, \ldots, \log_2 p\}$ and consider the partition of matrices $U^{p,B}[p - 2^{t+1}, \ldots, p - 2^t - 1][p - 2^{t+1}, \ldots, p - 2^t - 1]$ (the large square of size $2^t \times 2^t$ on the main diagonal in Figure 5) and $U^{p,B}[p - 2^{t+1}, \ldots, p - 2^t - 1][p - 2^t, \ldots, p - 1]$ (the large square of size $2^t \times 2^t$ to the right of the main diagonal in Figure 5). The first matrix has 2^{k-j-1} submatrices $L(t, \cdot)$ of size $2^j \times 2^j$, and the second one has $2^{k-j} - 1$ submatrices $R(t, j, \cdot)$ of size $2^j \times 2^j$. So, among the matrices $\{L(t, \cdot)\} \cup \{\, R(t, \cdot, k) \mid k > 0 \,\}$ there are $2^{t-j-1} + 2^{t-j} - 1$ matrices of size $2^j \times 2^j$, and each of them is multiplied by a strip of size $2^t \times M$ of the matrix $W^{p,q,C,D}$; to be more precise, this product is computed by splitting the stripe into $\frac{M}{2^j}$ square matrices of size $2^t \times 2^t$ each. Each product of square matrices of size $2^j \times 2^j$ is computed using $O(2^{j\omega})$ arithmetic operations, and therefore the total number of operations is bounded by constant times the following expression.

$$\sum_{j=0}^{t-1} (2^{t-j} + 2^{t-j-1} - 1) \frac{M}{2^j} 2^{j\omega} \leq \sum_{j=0}^{t-1} \frac{3}{2} 2^{t-j} \frac{M}{2^j} 2^{j\omega} = \frac{3}{2} M \sum_{j=0}^{t-1} 2^{t-j} 2^{j\omega - j} =$$

$$= \frac{3}{2} 2^t M \sum_{j=0}^{t-1} 2^{j(\omega-2)} \approx const \cdot 2^t M \cdot 2^{t(\omega-2)} = const \cdot 2^{t(\omega-1)} M$$

In addition to that, there is one more matrix $R(t, t, 0)$ of size $2^t \times 2^t$, which is multiplied by the strip of the matrix $W^{p,q,C,D}$ of size $2^t \times M$, implemented as $\frac{M}{2^t}$ multiplications by a square matrix, which gives $O(2^{t\omega}) \cdot \frac{M}{2^t} = O(M 2^{t(\omega-1)})$ additional operations.

All in all, $O(M 2^{t(\omega-1)})$ elements are required for a given t. The sum of the number of elements for $t = 0, \ldots, \log_2 p$ is then bounded by

$$const \cdot \sum_{t=0}^{\log p} M \cdot 2^{t(\omega-1)} \approx const \cdot M \cdot 2^{(\omega-1)\log p} = const \cdot p^{\omega-1} M$$

Hence, for a given p and q all matrices from the partition of $U^{p,B}$ are multiplied using $O(M p^{\omega-1})$ elements, where $M = O(n^2)$. Since $p \leqslant n$, this value is not more than $O(n^{\omega+1})$.

The last step is to sum the number of elements for all $p, q = 1, \ldots, n$ and for all rules $E \to BC$ and nonterminals D, and obtain the desired upper bound $O(|N| \cdot |R| \cdot n^{\omega+3})$ on the number of elements in the circuit. ◀

Like the simple circuit with $O(n^6)$ elements, the new circuit can also be adapted to compute the value of a string in a weighted grammar. Since fast matrix multiplication requires matrices over a ring, this restriction makes its way into the theorem.

▶ **Theorem 19.** *For every weighted grammar $G = (\Sigma, N, R, S)$ in the Chomsky normal form, with weights in a ring $\mathcal{R}$, there is a uniform family of arithmetic circuits computing elements over $\mathcal{R}$, for each length of input strings $n \geqslant 1$, which computes the value of an input string of length n in the ring, has $O(|N|^2 \cdot |R| \cdot n^{\omega+3})$ elements and is of depth $O((\log n + \log |R|) \log n)$. (where $\omega \geqslant 2$ is a number, such that for every n there is a an arithmetic circuit of depth $O(\log n)$ and with $O(n^\omega)$ elements that computes the product of two $n \times n$ matrices over $\mathcal{R}$)*

5 Future work

The smaller of the two proposed circuits for enumerating parse trees uses $O(n^{3+\omega})$ elements. On the other hand, in the Boolean case, where only the existence of a parse tree is determined, it should be sufficient to use only $O(n^{2\omega})$ elements, see Brent and Goldschlager [3, Sect. 6]. Perhaps the enumeration of parse trees might also be done using asymptotically fewer than $n^{\omega+3}$ elements, and this looks like an interesting subject for future research.

References

1 Ekaterina Bakinova, Artem Basharin, Igor Batmanov, Konstantin Lyubort, Alexander Okhotin, and Elizaveta Sazhneva. Formal languages over GF(2). *Inf. Comput.*, 283:104672, 2022. doi:10.1016/j.ic.2020.104672.

2 José-Miguel Benedí and Joan-Andreu Sánchez. Fast stochastic context-free parsing: A stochastic version of the Valiant algorithm. In Joan Martí, José-Miguel Benedí, Ana Maria Mendonça, and Joan Serrat, editors, *Pattern Recognition and Image Analysis, Third Iberian Conference, IbPRIA 2007, Girona, Spain, June 6-8, 2007, Proceedings, Part I*, volume 4477 of *Lecture Notes in Computer Science*, pages 80–88. Springer, 2007. doi:10.1007/978-3-540-72847-4_12.

3 Richard P. Brent and Leslie M. Goldschlager. A parallel algorithm for context-free parsing. *Australian Computer Science Communications*, 6(7):7.1–7.10, 1984.

4 Don Coppersmith and Shmuel Winograd. Matrix multiplication via arithmetic progressions. *J. Symb. Comput.*, 9(3):251–280, 1990. doi:10.1016/S0747-7171(08)80013-2.

5 Hillel Gazit and Gary L. Miller. An improved parallel algorithm that computes the BFS numbering of a directed graph. *Inf. Process. Lett.*, 28(2):61–65, 1988. doi:10.1016/0020-0190(88)90164-0.

6 Alexander Okhotin. Parsing by matrix multiplication generalized to Boolean grammars. *Theor. Comput. Sci.*, 516:101–120, 2014. doi:10.1016/j.tcs.2013.09.011.

7 Wojciech Rytter. Parallel time o(log n) recognition of unambiguous cfls. In Lothar Budach, editor, *Fundamentals of Computation Theory, FCT '85, Cottbus, GDR, September 9-13, 1985*, volume 199 of *Lecture Notes in Computer Science*, pages 380–389. Springer, 1985. doi:10.1007/BFb0028822.

8 Volker Strassen. Gaussian elimination is not optimal. *Numerische Mathematik*, 13:354–356, 1969.

9 Leslie G. Valiant. General context-free recognition in less than cubic time. *J. Comput. Syst. Sci.*, 10(2):308–315, 1975. doi:10.1016/S0022-0000(75)80046-8.