Theoretical Computer Science 516 (2014) 101–120

Contents lists available at ScienceDirect

Theoretical Computer Science

www.elsevier.com/locate/tcs

Parsing by matrix multiplication generalized to Boolean grammars ☆

Alexander Okhotin [1]

Department of Mathematics and Statistics, University of Turku, Turku FI-20014, Finland

ARTICLE INFO

Article history:
Received 10 October 2012
Received in revised form 11 July 2013
Accepted 13 September 2013
Communicated by Z. Esik

Keywords:
Boolean grammars
Conjunctive grammars
Context-free grammars
Matrix multiplication
Parsing

ABSTRACT

The well-known parsing algorithm for context-free grammars due to Valiant (1975) [25] is analyzed and extended to handle the more general Boolean grammars, which are context-free grammars augmented with conjunction and negation operators in the rules. The algorithm reduces construction of a parsing table to computing multiple products of Boolean matrices of various sizes. Its time complexity on an input string of length n is $O(\mathrm{BMM}(n)\log n)$, where $\mathrm{BMM}(n)$ is the number of operations needed to multiply two Boolean matrices of size $n \times n$, which is $O(n^{\omega})$ with $\omega < 2.373$ as per the current knowledge. A parse tree can be constructed in time $\mathrm{MM}(n)\log^{O(1)} n$ (where $\mathrm{MM}(n)$ is the complexity of multiplying two integer matrices), by applying a known efficient procedure for determining witnesses for Boolean matrix multiplication. The algorithm has a succinct proof of correctness and is ready to be implemented.

1. Introduction

Context-free grammars are the universally accepted mathematical model of syntax, and their status is well justified. On the one hand, their expressive means are *natural*, in the sense that whatever they define is intuitively seen as the syntax of something. On the other hand, they can be implemented in a variety of efficient algorithms, including a straightforward cubic-time parser, as well as many practical parsing algorithms working much faster in special cases.

The main idea of the context-free grammars is *inductive definition* of syntactically correct strings. For example, a grammar $S \to aSb \mid \varepsilon$ represents a definition of the following form: a string has the property S if and only if either it is representable as awb for some string w with the property S, or it is the empty string. Note that the vertical line in the above grammar is essentially a disjunction of two syntactical conditions. *Boolean grammars*, introduced by the author [14], are an extension of the context-free grammars, which maintains the main principle of inductive definition, but allows the use of any Boolean operations to combine syntactical conditions in the rules. This significantly increases the expressive power of the model [8,7, 13,14]. At the same time, Boolean grammars inherit the basic parsing algorithms from the context-free grammars, including the Cocke–Kasami–Younger algorithm [14] along with its variant for unambiguous grammars [17], the Generalized LR [15], as well as the linear-time recursive descent [16]. For more information about Boolean grammars, an interested reader is directed to a recent survey paper [18].

The straightforward upper bound on the complexity of parsing for Boolean grammars is the same as for ordinary context-free grammars: that is, $O(n^3)$, where n is the length of the input string [14]. However, for ordinary grammars, there also

☆ A preliminary version of this paper, entitled "Fast parsing for Boolean grammars: a generalization of Valiant's algorithm", was presented at the 14th International Conference on Developments in Language Theory (DLT 2010) held in London, Ontario, Canada on August 17–20, 2010, and its extended abstract appeared in the conference proceedings.

E-mail address: alexander.okhotin@utu.fi.

[1] Supported by the Academy of Finland under grant 257857.

0304-3975/$ – see front matter
http://dx.doi.org/10.1016/j.tcs.2013.09.011

exists an asymptotically faster parsing algorithm due to Valiant [25]: this algorithm computes the same parsing table as the simple Cocke–Kasami–Younger algorithm, but does so by offloading the most intensive computations into calls to a Boolean matrix multiplication procedure. The latter can be efficiently implemented in a variety of ways. Given two $n \times n$ Boolean matrices, a straightforward calculation of their product uses n^3 conjunctions and $(n-1)n^2$ disjunctions. An improved algorithm by Arlazarov et al. [3] reduces the number of bit operations to $O(\frac{n^3}{\log n})$, which is achieved by pre-computing products of all bit vectors of length $\log n$ with certain submatrices. Another much more sophisticated combinatorial algorithm for Boolean matrix multiplication, due to R. Williams [27], operates in time $O(\frac{n^3}{\log^2 n})$. An asymptotically more significant acceleration is obtained by using fast algebraic algorithms for multiplying $n \times n$ numerical matrices, such as Strassen's [24] algorithm that requires $O(n^{2.81})$ arithmetical operations, or the method of Coppersmith and Winograd [6], which, with the recent improvements by V. Vassilevska Williams [26], achieves theoretical running time $O(n^{2.373})$. These algorithms can be applied to multiplying $n \times n$ Boolean matrices by calculating their product in the ring of residues modulo $n+1$ [1]. Whatever method for Boolean matrix multiplication is used in Valiant's algorithm, as long as matrices are multiplied in time $\mathrm{BMM}(n)$, context-free parsing in time $O(\mathrm{BMM}(n) \log n)$ is obtained.

Valiant's result has inspired some further interesting studies. A total rethinking of the algorithm was undertaken by Rytter [22], who presented its divide-and-conquer approach in terms of parse tree partitions. Benedí and Sánchez [4] implemented the algorithm in the case of stochastic grammars, where Strassen's matrix multiplication can be applied most efficiently, and reported a noticeable performance increase. An extension of the algorithm for context-free grammars over partially computable alphabets was developed by Bertoni et al. [5] and works in time $\Theta(\mathrm{BMM}(n^\alpha))$, where $\alpha \geqslant 1$ is a constant determined by the commutativity relation between the symbols. Rajasekaran and Yooseph [21] extended Valiant's algorithm to the family of tree-adjoining grammars (which generalize context-free grammars by allowing inner rewriting of parse trees), and established a recognition algorithm working in time $\Theta(\mathrm{BMM}(n^2))$, improving over the $\Theta(n^6)$-time direct algorithm. A reverse reduction of parsing to matrix multiplication was first discovered by Satta [23], who demonstrated that any algorithm for constructing a full parsing table for tree-adjoining grammars in time $O(|G|^p \cdot n^q)$ leads to an $O(n^{2p+\frac{q}{6}})$-time algorithm for Boolean matrix multiplication. Satta's idea was later applied by Lee [12] to show that context-free parse table construction in time $O(|G|^p \cdot n^q)$ similarly implies $\mathrm{BMM}(n) = O(n^{2p+\frac{q}{3}})$.

Taking a closer look at Valiant's algorithm, one can see that first the entire grammar is encoded in a certain algebraic structure—basically, a semiring with non-associative multiplication—then the notion of a *transitive closure* of a Boolean matrix is extended to matrices over this semiring, so that the desired parsing table could be obtained as a closure of this kind, and finally it is demonstrated that such a closure can be efficiently computed using Boolean matrix multiplication. This approach essentially relies on having two operations in a grammar, concatenation and union, which give rise to the product and the sum in the semiring. Because of that, Valiant's algorithm in its original presentation cannot be applied to Boolean grammars.

This paper aims at rewriting Valiant's algorithm to make it work in the more general case of Boolean grammars. It is shown that using matrices over a special semiring as an intermediate abstraction is in fact unnecessary, and that matrix multiplication is actually needed to compute the concatenations only, while Boolean logic can be evaluated separately. Furthermore, the proposed algorithm maintains one fixed data structure, the parsing table, and whenever the matrix is to be cut as per Valiant's divide-and-conquer strategy, the new presentation of the algorithm only distributes the ranges of positions in the input string among the recursive calls. This leads to an improved version of the algorithm, which, besides being applicable to a larger family of grammars, is also better understandable than the original Valiant's algorithm, has a succinct proof of correctness and is ready to be implemented.

The resulting improved understanding of how exactly a large bulk of concatenations can be represented by Boolean matrix multiplication has already been used in a recent paper by Okhotin and Reitwießner [20] to obtain a fast recognition algorithm for input strings over a one-symbol alphabet. That algorithm uses Boolean convolution as the underlying algebraic problem, and utilizes fast Fourier transform to calculate it efficiently.

Following a brief introduction to Boolean grammars, given in Section 2, this paper sets off by presenting the simple cubic-time recognition algorithm (Section 3), both in the usual set-theoretic notation and on the level of bit operations. Next, an alternative order of evaluating the operations in this algorithm is illustrated on a small example, where the calculation of the parsing table for a 5-symbol string can use products of 2×2 Boolean matrices. Section 4 presents the desired algorithm for constructing the parsing table, in which the evaluation of logically independent bits in the parsing table is totally reordered to maximize the use of Boolean matrix products. This yields the known Valiant's algorithm, though in the form applicable to Boolean grammars; the algorithm is accompanied with a proof of correctness and an analysis of complexity. Some simple observations on implementing the algorithm are presented in the following Section 5.

The question of constructing a parse tree is handled in Section 6. While the running time is not as issue in the case of ordinary context-free grammars, where, given a parsing table, a tree can be constructed in time $O(n^2)$, in the case of Boolean grammars, this direct approach leads to cubic time. This question is solved by augmenting the parsing table with the data on factorizations of substrings, and modifying the main algorithm to construct such a table, with ordinary Boolean matrix multiplication replaced with the problem of computing integer *witnesses for matrix multiplication*. Applying the known algorithm of Alon and Naor [2] for constructing those witnesses in time $\mathrm{MM}(n) \log^{O(1)} n$, where $\mathrm{MM}(n)$ is the number of