**Multiple context-free path querying by matrix multiplication**

**Ilya V. Epelbaum1, Rustam Sh. Azimov2, Semyon V. Grigorev3**

1Querify Labs Ltd, Saint Petersburg, 193313, Russian Federation

2,3St. Petersburg State University (SPbSU), Saint Petersburg, 199034, Russian Federation

1[ilya.epelbaum@gmail.com](mailto:ilya.epelbaum@gmail.com), <https://orcid.org/0000-0003-2660-6193>

2[rustam.azimov19021995@gmail.com](mailto:rustam.azimov19021995@gmail.com), <https://orcid.org/0000-0003-0223-5172>

3[s.v.grigoriev@spbu.ru](mailto:s.v.grigoriev@spbu.ru), <https://orcid.org/0000-0002-7966-0698>

**Abstract**

Many graph analysis problems can be formulated as formal language-constrained path querying problems where the formal languages are used as constraints for navigational path queries. Recently, the context-free language (CFL) reachability formulation has become very popular and can be used in many areas, for example, querying graph databases, Resource Description Framework (RDF) analysis. However, the generative capacity of context-free grammars (CFGs) is too weak to generate some complex queries, for example, from natural languages, and the various extensions of CFGs have been proposed. Multiple context-free grammar (MCFG) is one of such extensions of CFGs. Despite the fact that, to the best of our knowledge, there is no algorithm for MCFL-reachability, this problem is known to be decidable. This paper is devoted to developing the first such algorithm for the MCFL-reachability problem. The essence of the proposed algorithm is to use a set of Boolean matrices and operations on them to find paths in a graph that satisfy the given constraints. The main operation here is Boolean matrix multiplication. As a result, the algorithm returns a set of matrices containing all information needed to solve the MCFL-reachability problem. The presented algorithm is implemented in Python using GraphBLAS API. An analysis of real RDF data and synthetic graphs for some MCFLs is performed. The study showed that using a sparse format for matrix storage and parallel computing for graphs with tens of thousands of edges the analysis time can be performed in 10–20 minutes. The result of the analysis provides tens of millions of reachable vertex pairs. The proposed algorithm can be applied in problems of static code analysis, bioinformatics, network analysis, as well as in graph databases when a path query cannot be expressed using context-free grammars. The provided algorithm is linear algebra-based, hence it allows one to use high-performance libraries and utilize modern parallel hardware.

**Keywords**

path querying, MCFG, graph databases, RDF, Boolean matrix multiplication, GraphBLAS API

**Acknowledgements**

The research was supported by the Russian Science Foundation, Grant 18-11-00100.

**For citation**: Epelbaum I.V., Azimov R.Sh., Grigorev S.V. Multiple context-free path querying by matrix multiplication. *Scientific and Technical Journal of Information Technologies, Mechanics and Optics*, 2023, vol. 23, no. 2, pp. . doi: 10.17586/2226-1494-2023-23-2-

УДК 004.421.2:519.17 004.657

**Решение задачи достижимости в графе с заданными ограничениями в виде многокомпонентной контекстно-свободной грамматики с использованием умножения матриц**

**Илья Владимирович Эпельбаум1, Рустам Шухратуллович Азимов2,**

**Семён Вячеславович Григорьев3**

1ООО «КВЕРИФАЙ ЛАБС», Санкт-Петербург, 193313, Российская Федерация

2,3Санкт-Петербургский государственный университет, Санкт-Петербург, 199034, Российская Федерация

1[ilya.epelbaum@gmail.com](mailto:ilya.epelbaum@gmail.com), <https://orcid.org/0000-0003-2660-6193>

2[rustam.azimov19021995@gmail.com](mailto:rustam.azimov19021995@gmail.com), <https://orcid.org/0000-0003-0223-5172>

3[s.v.grigoriev@spbu.ru](mailto:s.v.grigoriev@spbu.ru), <https://orcid.org/0000-0002-7966-0698>

**Аннотация**

**Предмет исследования.** Многие задачи анализа графов могут быть сформулированы как задачи поиска путей с ограничениями в виде формальных языков. В последнее время задача достижимости в графе с заданными ограничениями в виде контекстно-свободных языков стала очень популярной и используется во многих областях, например, для запросов к графовым базам данных, для анализа RDF (Resource Description Framework) данных. Однако некоторые сложные ограничения на пути в графе не могут быть описаны с помощью контекстно-свободных языков, поэтому были предложены различные расширения. Многокомпонентные контекстно-свободные языки ~~–~~ одно из таких расширений. В данной работе представлены результаты разработки первого алгоритма поиска путей в графе с заданными ограничениями в виде многокомпонентных контекстно-свободных языков. **Метод.** Сущность предложенного алгоритма состоит в использовании набора булевых матриц и операций над ними для поиска путей в графе, удовлетворяющих заданным ограничениям. Основной операцией является умножение булевых матриц. В качестве результата алгоритм возвращает набор матриц, содержащий всю информацию, необходимую для решения задачи достижимости в графе с заданными ограничениями в виде многокомпонентного контекстно-свободного языка. **Основные результаты.** Представленный алгоритм реализован на языке Python с использованием стандарта GraphBLAS. Выполнен анализ реальных RDF данных и синтетических графов для некоторых классических многокомпонентных контекстно-свободных языков. Исследование показало, что при использовании разреженного формата для хранения матриц и параллельных вычислений для графов с десятками тысяч ребер время анализа может составлять 10–20 минут. Результат проведенного анализа представляет десятки миллионов пар достижимых вершин. **Практическая значимость.** Разработанный алгоритм может быть применен в задачах статического анализа программ, в биоинформатике, в сетевом анализе, а также в графовых базах данных, когда ограничения на пути в графе не могут быть выражены с помощью контекстно-свободных грамматик. Алгоритм основан на операциях линейной алгебры, что позволяет использовать высокопроизводительные библиотеки и задействовать современные параллельные вычислительные системы.

**Ключевые слова**

анализ графов, многокомпонентные КС-грамматики, графовые базы данных, RDF, умножение булевых матриц, стандарт GraphBLAS

**Благодарности**

Исследование выполнено при финансовой поддержке Российского научного фонда в рамках научного проекта № 18-11-00100.

**Ссылка для цитирования:** Эпельбаум И.В., Азимов Р.Ш., Григорьев С.В. Решение задачи достижимости в графе с заданными ограничениями в виде многокомпонентной контекстно-свободной грамматики с использованием умножения матриц // Научно-технический вестник информационных технологий, механики и оптики. 2023. Т. 23, № 2. С. (на англ. яз.). doi: 10.17586/2226-1494-2023-23-2-

**Introduction**

Many graph analysis problems can be formulated as formal language-constrained path querying [1] problems where the formal languages are used as constraints for navigational path queries. More precisely, a path in *an edge-labeled graph* is viewed as a word constructed by the concatenation of edge labels, and the formal languages are used to constrain the paths of interest. When answering a query to the graph, some information about paths labeled by words from the given formal language should be found. Recently, the Context-Free Language (CFL) reachability formulation become very popular and can be used in many areas, for example, querying graph databases [2], Resource Description Framework (RDF) analysis [3], static code analysis [4], biological data analysis [5].

However, for some real-world problems the generative capacity of Context-Free Grammars (CFGs) is too weak to describe the necessary path constraints, i.e. natural language path constraints. Thus, the various extensions of CFGs have been proposed to define the syntax of natural languages. *Multiple Context-Free Grammar* (MCFG) is one of such extensions of CFGs. A nonterminal of an MCFG derives tuples of words while the nonterminals of a CFG can only derive words. Using the MCFGs, it is possible to formulate more complex path constraints as, for example, structures involving discontinuous constituents such as “respectively” sentences or inverted sentences in a simple manner.

Such languages allow one to use more complex graph queries that may find application in various areas, for example, in static code analysis. In practice, the Dyck language [6] is the most widely used language in CFL-reachability problem. This language essentially generates the well-matched parentheses. Particularly, many program analyses use the Dyck language to exactly model the *matched-parenthesis* property for *context-sensitivity* or *data-dependence* analysis [7]. Namely, context-sensitivity describes the well-balanced procedure calls and returns using open and close parentheses, respectively. Similarly, the data-dependence represents another well-balanced property among language constructors, for example, field accesses (i.e., reads and writes), pointer indirections (i.e., references and dereferences), etc. However, the precise analysis that captures two or more well-balanced properties is undecidable [7]. For example, the context-sensitive and data-dependence analysis describes an interleaved matched-parenthesis language which is not even context-free. The traditional approach is to approximate the solution using the CFL-reachability algorithms. An interleaved matched-parenthesis language can be viewed as the intersection of two CFLs. However, the CFLs are not closed under intersection [8]. Therefore, the precision of either context-sensitivity or data-dependence must be sacrificed by approximating the corresponding Dyck language using a regular one.

However, for more precise analysis other classes of formal languages can be used. For example, the *linear conjunctive languages* can be applied for context-sensitive data-dependence analysis and demonstrate significant precision and scalability advantages of this approach [9]. Thus, the class of *Multiple Context-Free Languages* (MCFLs) may also contain the formal languages that can be used to increase the precision of the solution for some program analysis problems. One of the candidates for such a language is the language that can model the matched number of opening and closing parentheses for context-sensitive data-dependence analysis. These languages are approximations of interleaved matched-parenthesis languages and are known not to be context-free. Thus, such MCFLs as are of practical interest for static code analysis. For example, is a language of words with equal number of symbols and , and with equal number of symbols and .

Therefore, the creation of the MCFL-reachability algorithms is motivated by real-word problems where CFLs cannot be used. To the best of our knowledge, there is no algorithm for MCFL-reachability.

In practice, the good receipt to achieve high-performance solutions for graph analysis problems is to offload the most critical computations into Linear Algebra (LA) operations, for example matrix operations [10]. Then such algorithm can be effectively implemented using the high-performance LA libraries with wide class of optimizations like parallel computations [11, 12]. There are LA-based efficient MCFL recognition algorithms that use the Boolean matrix multiplications [13, 14] and can form the basis of new MCFL-reachability algorithms. However, the MCFL parsing algorithm in [14] can be applied only for some subclass of MCFGs called *unbalanced*.

To sum up, we make the following contributions in this paper.

* We provide the first MCFL-reachability algorithm by extending the MCFL parsing algorithm from [13]. Our algorithm is LA-based, hence it allows one to use high-performance libraries and utilize modern parallel hardware.
* We implement the proposed algorithm using the pygraphblas[[1]](#footnote-1) implementation of the GraphBLAS API [10] and evaluate it on some real RDFs and synthetic graphs using some classical MCFLs.

**Problem statement**

In this section, we introduce some definitions in graph theory and formal language theory which are used in this paper. Also, we provide the definition of the MCFL-reachability problem.

In this paper, we use an edge-labeled directed graph as a data model and define it as follows.

**Definition 1.** *An edge-labeled directed graph* is a tuple where

* is a finite set of vertices. For simplicity, we assume that the vertices are natural numbers ranging from 1 to ,
* is a set of edge labels,
* is a set of labeled edges.

We define MCFLs and MCFGs as follows.

**Definition 2.** An MCFG is a tuple where

* is a finite set of nonterminals,
* is the start nonterminal.
* For each a natural number , called the dimension of , is defined. In particular, we assume . Sometimes is written as , where each is called a component symbol of . Let be the set of all component symbol of all nonterminals from the set .
* is a finite set of terminals, .
* is a finite set of production rules. has a form of , where and . Each rule satisfies the following condition.
  + **Right-linearity**: , appears in the right-hand side (rhs) of at most once.

We use the conventional notation , where , to denote that a tuple of words can be derived from a nonterminal by some sequence of production rule applications from in the grammar if can be derived from -th component of the nonterminal *A*.

**Definition 3.** An MCFL is a language generated by an MCFG where .

Now, we can define the MCFL-reachability problem using following definitions.

**Definition 4.** Let be a labeled graph, be an MCFL. Then a *multiple context-free relation* with the language on the labeled graph is the relation :

, where is a graph vertex for all .

**Definition 5.** *MCFL-reachability problem* is the problem of finding multiple context-free relation for a given directed labeled graph and an MCFL .

In other words, the result of MCFL-reachability evaluation is a set of vertex pairs such that there is a path between them that forms a word from the given MCFL. This problem is known to be decidable because the MCFL are closed under intersection with regular languages (i.e. with graphs) and the reachability information can be computed by checking the resulting language for emptiness, which is a decidable problem.

**Matrix-based MCFL-reachability algorithm**

In this section, we introduce the matrix-based algorithm for the MCFL-reachability problem. Firstly, we introduce the following normal form for the MCFGs that allow us to solve the MCFL-reachability problem using the LA operations.

**Definition 6.** An MCFG is in *normal form* if every production rule satisfies one of the following two forms.

* and . In this case, we call the rule *p terminating*.
* , i.e. no terminal symbol appears in the rhs of *p*. For simplicity of notation, we denote , where . It is necessary that . To sum up, , where . In this case, we call the rule *p* *nonterminating*. Any rule in this form must satisfy the following conditions.
  + **Non-erasing condition**: appears in for some *k*.
  + No pair of component symbols of the same nonterminal appear adjacently in the rhs in one component, i.e. component symbols of appear alternately in one component.
  + .

According to [13], the following theorem holds.

**Theorem 1.** Let be an MCFG. An MCFG can be constructed from such that and satisfies described normal form.

For example, for the normal form defined above is the same as the weak Chomsky normal form for context-free grammars described in [11].

Next, we define the following sets that describe the production rules of the given MCFG.

**Definition 7.** Let be an MCFG in the described normal form. Then we define:

* , each of the sets is considered ordered by the components of the and inside the component from left to right;
* defined similarly with but with an offset of , that is, will be of the form and ;
* is ordered set of pairs where the first element of the pair corresponds to the leftmost nonterminal of some component in the rule and the second element – to the rightmost, i.e. the elements of pairs are elements of the sets and defined above;
* iff and iff , ;
* defined similarly with but with an offset of .

Note that for grammars in the described normal form.

Let be an MCFG, be a labeled directed graph, where . The proposed algorithm is presented in Listing 1. The *MCFL rechability* procedure takes as input a graph and an MCFG in normal form.

1. *G = (N, , P, S) — MCFG, D = (V, E,) — directed edge-labeled graph*
2. procedure *MCFL\_reachability(G, D)*
3. for alldo
4. for alldo
6. // add information about terminating rules
7. for alldo
8. // all values are new for the first iteration
9. whiledo
10. for all do // consider nonterminating rules
11. // use only new values
13. for alldo

16. // add new information for nonterminal A
18. for alldo// collect all information for the startnonterminal S
19. for alldo
21. // size of is equal to 2 since
22. return

*Listing 1.* A matrix-based MCFL-reachability algorithm

At the first stage, the algorithm processes rules where there are only terminals on their rhs. Thus, the algorithm restores the paths that can be obtained in one application of the rule. The *update* procedure is presented in Listing 2. It is used to update all necessary matrices for rules with nonterminal in rhs with a new value according to the new paths found. In the update procedure, only the index of the value is recalculated, taking into account the sets, , and and the value *True* is added according to the calculated index.

In the line 6 of Listing 1 the paths of length 1 or 0 corresponding to the components of the rule are found. That is, each is a pair of vertices between which there is a path derived from the -component of the rule. We note two facts about this index. First of all, there are pairs in this index, that is, the number of elements in it is even. Second, such index can be encoded as a -ary number. The second fact allows us to use the *FromIndex* algorithm (*ToIndex* inverse to it) to convert such a number to the -ary numeral system. The parity of the number of elements allows us to divide the index in half and write the first part in the row number of the matrix and the second part in the column number. Thus, we write the fact of the restored paths for the nonterminal into a square Boolean matrix by dividing the index in half and translating each part into the desired numeral system (let these numbers be and ), and then put *True* value in the cell .

1. procedure *update(B, index)* // update matrices for all rules with B in the rhs
2. for alldo
4. for all do
6. for all do

9. for alldo
11. for all do
13. for all do

*Listing 2.* The procedure for updating matrix values

Further, the algorithm uses five matrices for each nonterminal rule. Namely, for a rule of the form , the algorithm supports the matrix in which the result is stored, taking into account the sets in Definition 7 for the nonterminal , as well as the matrix, which stores the result, taking into account the sets, which was obtained only at the previous step. Similarly for the nonterminal . Also, the information about found paths corresponding to this rule is stored in matrix , taking into account the set . And after processing the terminating rules, it is necessary to add new results to the supported matrices. This is exactly what the algorithm in lines 8–9 does.

Next, the algorithm proceeds to the consideration of nonterminating rules. Namely, in the line 12, the algorithm calculates new paths in the graph using four matrices for nonterminals from the rhs of the rule. Further, the algorithm updates the matrices and to store only new values that was added at this iteration. Also, algorithm writes only new values to the matrix for the nonterminal and propagates new results among all matrices for the nonterminal in the rhs of other rules.

The algorithm works while at least one value has appeared on the current iteration using the loop in lines 10–17. As the last step, the algorithm collects values from all rules where there is the starting nonterminal on the left-hand side (lhs) and puts these values into the matrix *Res* for the MCFL-reachability result. The indices must be recalculated using the *transform\_index* procedure presented in Listing 3.

Similarly to the proof of the correctness of the matrix-based CFL-reachability algorithm from [12], it can be shown that the following theorem holds by the induction on the iteration number and on the height of derivation trees.

**Theorem 2.** Let be an MCFG in *normal form*, be an labeled directed graph and *Res* be the matrix obtained as a result of the algorithm in Listing 1. Then iff .

1. procedure *transform\_index(i, j, p)* // transform the indices *i* and *j* taking into account the set


5. for alldo
6. if then

9. else

12. if then

15. else

18. return

*Listing 3*. The procedure for index transformation

**Evaluation**

We provide a prototype implementation of the proposed MCFL-reachability algorithm using the pygraphblas implementation of the GraphBLAS API. The source code is available on GitHub[[2]](#footnote-2). For evaluation, we use a PC with Ubuntu 18.04 installed. It has Intel core i7-6700 CPU, 3.4GHz, and DDR4 64Gb RAM.

We evaluate our implementation on some real-world RDFs and synthetic graphs using following classical MCFLs. The query corresponds to the MCFL , and the query corresponds to the MCFL . These languages are known to be not context-free [15]. We use some RDFs from the CFPQ\_Data dataset[[3]](#footnote-3) provided in [2]. Also, we generate some synthetic graphs that describe network structures using the LFR (Lancichinetti–Fortunato–Radicchi) graph generator from the NetworkX [16] Python package.

*Table 1.* MCFL-reachability execution time of queriesand

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Graph type | Graph | Number of elements | | Execution time, s | |
| |V| | |E| |  |  |
| RDF | skos | 144 | 323 | 0.07 | 0.08 |
| pizza | 671 | 2604 | 3.75 | 2.06 |
| wine | 733 | 2450 | 4.55 | 4.41 |
| funding | 778 | 1480 | 1.68 | 1.50 |
| core | 1323 | 8684 | 10.77 | 9.93 |
| pathways | 6238 | 37,196 | 533.73 | 148.62 |
| LFR |  | 100 | 210 | 0.12 | 0.06 |
|  | 500 | 970 | 2.55 | 1.47 |
|  | 1000 | 2100 | 13.50 | 6.10 |
|  | 10,000 | 21,005 | 1261.97 | 656.28 |

The results of the MCFL-reachability evaluation for queries and are presented in Table 1. We can see, that while the execution time for small graphs is decent, our prototype implementation is underperforming for graphs with thousands of vertices. To show the reason for such behavior, we also present the number of non-zero elements in matrices and added by the MCFL-reachability algorithm in Table 2. The is a sum of non-zero elements in these matrices for the query , and the – for the query . This information describes the big amount of consumed memory and the complexity of the used matrix operations. Our implementation is underperforming for used graphs and queries because there is a big number of combinations of paths that are relevant to the queryand was founded by our algorithm. The used graphs are composed entirely of edges that are relevant to the query, and they form such a large number of combinations. For more practical use of our algorithm, the huge graphs can contain only a small part of the edges that are relevant to the query. This will lead to a decent number of the non-zero matrix elements, to a small number of used memories, and to a small running time. Also, for big graphs our algorithm constructs matrices of huge sizes that can exceed the numeric range of integer data types. However, this problem can be solved since there are formats like COO (coordinate list) for storing the sparse matrices that stores only non-zero values and does not depend on the matrix size.

*Table 2.* The number of non-zero elements in matrices after the MCFL-reachability evaluation

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Graph type | Graph | Number of elements | | |
| |E| |  |  |
| RDF | skos | 323 | 5043 | 5312 |
| pizza | 2604 | 201,554 | 134,993 |
| wine | 2450 | 252,323 | 241,485 |
| funding | 1480 | 101,304 | 94,401 |
| core | 8684 | 531,900 | 503,943 |
| pathways | 37,196 | 19,452,226 | 9,734,396 |
| LFR |  | 210 | 5687 | 2851 |
|  | 970 | 118,449 | 63,786 |
|  | 2100 | 553,002 | 276,299 |
|  | 21,005 | 55,172,204 | 27,349,209 |

We conclude that we should improve our implementation to achieve better performance and find the graphs and queries for more practical use, while the algorithm idea is viable.

**Conclusion**

In this paper, we propose the first MCFL-reachability algorithm by extending theMCFL parsing algorithm from [13]. Thus, we show how the MCFL-reachabilityproblem can be solved using the LA operations. We implement the proposed algorithm using the GraphBLAS API. We should improve our prototype implementationusing various high-performance libraries for the LA operations,distributed computations, and modern parallel hardware like GPU. Also, weshould find the real graphs and queries for more practical use of our algorithm,for example, from the area of static code analysis.

**References**

1. Barrett C., Jacob R., Marathe M. Formal-language-constrained path problems // SIAM Journal on Computing. 2000. V. 30(3). P. 809–837. https://doi.org/10.1137/S0097539798337716
2. Terekhov A., Khoroshev A., Azimov R., Grigorev S. Context-free path querying with single-path semantics by matrix multiplication // In: Proceedings of the 3rd Joint International Workshop on Graph Data Management Experiences & Systems (GRADES) and Network Data Analytics (NDA). GRADES-NDA’20, Association for Computing Machinery, New York, NY, USA (2020). https://doi.org/10.1145/3398682.3399163
3. Zhang X., Feng Z., Wang X., Rao G., Wu W. Context-free path queries on rdf graphs. In: Groth, P., Simperl, E., Gray, A., Sabou, M., Kr ̈otzsch, M., Lecue, F., Fl ̈ock, F., Gil, Y. (eds.) The Semantic Web – ISWC 2016. P. 632–648. Springer International Publishing, Cham, 2016.
4. Zheng X., Rugina R. Demand-driven alias analysis for c // In: Proceedings of the 35th Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages. P. 197–208. POPL ’08, ACM, New York, NY, USA (2008). https://doi.org/10.1145/1328438.1328464
5. Sevon P., Eronen L. Subgraph queries by context-free grammars // Journal of Integrative Bioinformatics. 2008. V. 5(2). P. 157–172. https://doi.org/https://doi.org/10.1515/jib-2008-100
6. Zhang Q., Lyu M.R., Yuan H., Su Z. Fast algorithms for dyck-cfl-reachability with applications to alias analysis // In: Proceedings of the 34th ACM SIGPLAN Conference on Programming Language Design and Implementation. 2013. P. 435–446.
7. Reps T. Undecidability of context-sensitive data-dependence analysis // ACM Transactions on Programming Languages and Systems (TOPLAS). 2000. V. 22(1). P. 162–186.
8. Hopcroft J.E., Motwani R., Ullman J.D. Introduction to automata theory, languages, and computation // Acm Sigact News. 2001. V. 32(1). P. 60–65.
9. Zhang Q., Su Z. Context-sensitive data-dependence analysis via linear conjunctive language reachability // In: Proceedings of the 44th ACM SIGPLAN Symposium on Principles of Programming Languages. 2017. P. 344–358.
10. Kepner J., Aaltonen P., Bader D., Bulu ̧c A., Franchetti F., Gilbert J., Hutchison D., Kumar M., Lumsdaine A., Meyerhenke,H., et al. Mathematical foundations of the graphblas // In: 2016 IEEE High Performance Extreme Computing Conference (HPEC). P. 1–9. IEEE (2016).
11. Azimov R., Epelbaum I., Grigorev S. Context-free path querying with all-path semantics by matrix multiplication // In: Proceedings of the 4th ACM SIGMOD Joint International Workshop on Graph Data Management Experiences & Systems (GRADES) and Network Data Analytics (NDA). 2021. P. 1–7.
12. Azimov R., Grigorev S. Context-free path querying by matrix multiplication // In: Proceedings of the 1st ACM SIGMOD Joint International Workshop on Graph Data Management Experiences & Systems (GRADES) and Network Data Analytics (NDA). 2018. P. 1–10.
13. Nakanish, R., Takada K., Seki H. An efficient recognition algorithm for multiple context-free languages //In: In Proceedings of the Fifth Meeting on Mathematics of Language, MOL5. Citeseer. 1997.
14. Cohen S.B., Gildea D.: Parsing linear context-free rewriting systems with fast matrix multiplication // Computational Linguistics. 2016. V. 42(3). P. 421–455.
15. Seki H., Matsumura T., Fujii M., Kasami T. On multiple context-free grammars // Theoretical Computer Science. 1991. V. 88(2). P. 191–229.
16. Hagberg A., Swart P., S Chult D. Exploring network structure, dynamics, and function using networkx. Tech. rep., Los Alamos National Lab.(LANL), Los Alamos, NM (United States) (2008).

|  |  |  |
| --- | --- | --- |
| **Ilya V. Epelbaum** | – | Student, Software Developer, Querify Labs Ltd., Saint Petersburg, 193313, Russian Federation, sc 57218700891, https://orcid.org/0000-0003-2660-6193, ilya.epelbaum@gmail.com |
| **Rustam Sh. Azimov** | – | Senior Lecturer, St. Petersburg State University (SPbSU), Saint Petersburg, 199034, Russian Federation, sc 57203022098, https://orcid.org/0000-0003-0223-5172, rustam.azimov19021995@gmail.com |
| **Semyon V. Grigorev** | – | PhD (Physics & Mathematics), Associate Professor, St. Petersburg State University (SPbSU), Saint Petersburg, 199034, Russian Federation, sc 56047575300, https://orcid.org/0000-0002-7966-0698, s.v.grigoriev@spbu.ru |

*Received*

*Approved after reviewing*

*Accepted*

|  |  |  |
| --- | --- | --- |
| **Эпельбаум Илья Владимирович** | – | студент, программист, ООО «КВЕРИФАЙ ЛАБС», Санкт-Петербург, 193313, Российская Федерация, sc 57218700891,  <https://orcid.org/0000-0003-2660-6193>, [ilya.epelbaum@gmail.com](mailto:ilya.epelbaum@gmail.com) |
| **Азимов Рустам Шухратуллович** | – | старший преподаватель, Санкт-Петербургский государственный университет, Санкт-Петербург, 199034, Российская Федерация, sc 57203022098, <https://orcid.org/0000-0003-0223-5172>, [rustam.azimov19021995@gmail.com](mailto:rustam.azimov19021995@gmail.com) |
| **Григорьев Семён Вячеславович** | – | кандидат физико-математических наук, доцент, Санкт-Петербургский государственный университет, Санкт-Петербург, 199034, Российская Федерация, sc 56047575300, <https://orcid.org/0000-0002-7966-0698>, [s.v.grigoriev@spbu.ru](mailto:s.v.grigoriev@spbu.ru) |

*Статья поступила в редакцию*

*Одобрена после рецензирования*

*Принята к печати*

1. The pygraphblas GitHub repository. Available at: <https://github.com/Graphegon/pygraphblas> (accessed: 01.09.2022). [↑](#footnote-ref-1)
2. Sources of the prototype implementation of the proposed MCFL-reachability algorithm. Available at: <https://github.com/FormalLanguageConstrainedPathQuerying/CFPQ_PyAlgo> (accessed: 09.12.2022). [↑](#footnote-ref-2)
3. CFPQ\_Data dataset GitHub repository. Available at: <https://github.com/FormalLanguageConstrainedPathQuerying/CFPQ_Data> (accessed: 09.12.2022). [↑](#footnote-ref-3)