НАУЧНО-ТЕХНИЧЕСКИЙ ВЕСТНИК ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ, МЕХАНИКИ И ОПТИКИ
март–апрель 2023 Том 23 № 2 http://ntv.ifmo.ru/
SCIENTIFIC AND TECHNICAL JOURNAL OF INFORMATION TECHNOLOGIES, MECHANICS AND OPTICS
March–April 2023 Vol. 23 No 2 http://ntv.ifmo.ru/en/
ISSN 2226-1494 (print) ISSN 2500-0373 (online)

doi: 10.17586/2226-1494-2023-23-2-271-278

Multiple context-free path querying by matrix multiplication

Ilya V. Epelbaum[1], Rustam Sh. Azimov[2]✉, Semyon V. Grigorev[3]

[1] Querify Labs Ltd, Saint Petersburg, 193313, Russian Federation
[2,3] St. Petersburg State University (SPbSU), Saint Petersburg, 199034, Russian Federation

[1] ilya.epelbaum@gmail.com, https://orcid.org/0000-0003-2660-6193
[2] rustam.azimov19021995@gmail.com✉, https://orcid.org/0000-0003-0223-5172
[3] 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 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. 271–278. doi: 10.17586/2226-1494-2023-23-2-271-278

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 Эпельбаум[1], Рустам Шухратуллович Азимов[2]✉, Семён Вячеславович Григорьев[3]

[1] ООО «КВЕРИФАЙ ЛАБС», Санкт-Петербург, 193313, Российская Федерация
[2,3] Санкт-Петербургский государственный университет, Санкт-Петербург, 199034, Российская Федерация

[1] ilya.epelbaum@gmail.com, https://orcid.org/0000-0003-2660-6193
[2] rustam.azimov19021995@gmail.com✉, https://orcid.org/0000-0003-0223-5172
[3] 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. 271–278 (на англ. яз.). doi: 10.17586/2226-1494-2023-23-2-271-278

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 allows 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 O_n 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 O_n are of practical interest for static code analysis. For example, $O_2 = \{w \in \{a, a', b, b'\}^* | |w|_a = |w|_{a'} \wedge |w|_b = |w|_{b'}\}$ is a language of words with equal number of symbols a and a', and with equal number of symbols b and b'.

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] 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 $D = (V, E, \Sigma)$ where

— V is a finite set of vertices. For simplicity, we assume that the vertices are natural numbers ranging from 1 to $|V|$,
— Σ is a set of edge labels,
— $E \subseteq V \times \Sigma \times V$ is a set of labeled edges.

We define MCFLs and MCFGs as follows.

Definition 2. An MCFG G is a tuple (N, Σ, P, S, d) where

— N is a finite set of nonterminals,
— $S \in N$ is the start nonterminal.
— For each $A \in N$ a natural number $d(A)$, called the dimension of A, is defined. In particular, we assume $d(S) = 1$. Sometimes A is written as $(A^1, \ldots, A^{d(A)})$, where each A^i is called a component symbol of A. Let N_c be the set of all component symbol of all nonterminals from the set N.
— Σ is a finite set of terminals, $N \cap \Sigma = \emptyset$.
— P is a finite set of production rules. $p \in P$ has a form of p: $(A^1, \ldots, A^{d(A)}) \to (\gamma_1, \ldots, \gamma_{d(A)})$ where $A \in N$ and $\gamma_i \in (N_c \cup \Sigma)^*$. Each rule p satisfies the following condition.

Right-linearity: $\forall A^i \in N_c$, A^i appears in the right-hand side (rhs) of p at most once.

We use the conventional notation $A \to^*_G (w_1, \ldots, w_{d(A)})$, where $w^i \in \Sigma^*$, to denote that a tuple of words can be derived from a nonterminal A by some sequence of production rule applications from P in the grammar G if $\forall i$: w_i can be derived from i-th component of the nonterminal A.

Definition 3. An MCFL is a language generated by an MCFG $G = (N, \Sigma, P, S, d)$ where $L(G) = \{w \in \Sigma^* | S \to^*_G w\}$.

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

Definition 4. Let $D = (V, E, \Sigma)$ be a labeled graph, L be an MCFL. Then a *multiple context-free relation* with the language L on the labeled graph D is the relation $R_{D,L} \subseteq V \times V$:

$R_{D,L} = \{(v_0, v_n) \in V \times V | \exists \pi = ((v_0, l_1, v_1), \ldots, (v_{n-1}, l_n, v_n)) \in \pi$: $l(\pi) \in L\}$, where v_i is a graph vertex for all $0 \le i \le n$.

Definition 5. *MCFL-reachability problem* is the problem of finding multiple context-free relation $R_{D,L}$ for a given directed labeled graph D and an MCFL L.

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 $G = (N, \Sigma, P, S, d)$ is in *normal form* if every production rule p: $A \to (\gamma_1, \ldots \gamma_{d(A)}) \in P$ satisfies one of the following two forms.

— $\forall i$: $1 \le i \le d(A)$, $|\gamma_i| = 1$ and $\gamma_i \in \Sigma \cup \{\varepsilon\}$. In this case, we call the rule *p terminating*.
— $\forall i$: $1 \le i \le d(A)$, $\gamma_i \in N_c^*$, i.e., no terminal symbol appears in the rhs of p. For simplicity of notation, we denote

[1] The pygraphblas GitHub repository. Available at: https://github.com/Graphegon/pygraphblas (accessed: 01.09.2022).

p: $A \to f(B_1, \ldots, B_n)$, where $B_k \in N$. It is necessary that $n = 2$. To sum up, p: $A \to f(B_1, B_2)$, where $B_1, B_2 \in N$. In this case, we call the rule *p nonterminating*. Any rule in this form must satisfy the following conditions.

— **Non-erasing condition:** $\forall i \in \{1, 2\}$, $1 \le j \le d(B_i)$, B_i^j appears in γ_k for some k.
— No pair B^i, B^k of component symbols of the same nonterminal B appear adjacently in the rhs in one component, i.e., component symbols of B_1, B_2 appear alternately in one component.
— $\exists i$: $1 \le i \le d(A)$, $|\gamma_i| \ge 2$.

According to [13], the following theorem holds.

Theorem 1. Let G be an MCFG. An MCFG G' can be constructed from G such that $L(G') = L(G)$ and G' satisfies described normal form.

For example, for $m = 1$ 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 $G = (N, \Sigma, P, S, d)$ be an MCFG in the described normal form. Then $\forall p$: $A \to f(B, C) \in P$ we define:

— $end_{B(p)} = \{2(i-1)|$ iff B^i is the leftmost in the component of rhs of $p\} \cup \{2(i-1)+1)|$ iff B^i, each of the sets is considered ordered by the components of the A and inside the component from left to right;
— *end_C*(p) defined similarly with *end_B*(p) but with an offset of $2d(B)$, that is *end_C*(p), will be of the form $2(i-1) + 2d(B)$ and $2(i-1) + 1 + 2d(B)$;
— *end_A*(p) 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 *end_B*(p) and *end_C*(p) defined above;
— $(2(i-1)) \in$ *alter_B*(p) iff $(2(i-1)) \notin$ *end_B*(p) and $(2(i-1)+1) \in$ *alter_B*(p) iff $(2(i-1)+1) \notin$ *end_B*(p), $\forall i$: $1 \le i \le d(B)$;
— *alter_C*(p) defined similarly with *alter_B*(p) but with an offset of $2d(B)$.

Note that $|$*alter_B*(p)$| = |$*alter_C*(p)$|$ for grammars in the described normal form.

Let $G = (N, \Sigma, P, S, d)$ be an MCFG, $D = (V, E, \Sigma)$ be a labeled directed graph, where $|V| = n$. The proposed algorithm is presented in Listing 1. The MCFL *rechability* procedure takes as input a graph and an MCFG in normal form.

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 B 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 *end_B*(p), *end_C*(p), *alter_B*(p) and *alter_C*(p) and the value *True* is added according to the calculated index.

In the line 6 of Listing 1 the $d(A)$ paths of length 1 or 0 corresponding to the components of the rule $A \to (a_1, \ldots, a_{d(A)})$ are found. That is, each (l_i, r_i) is a pair of vertices between which there is a path derived from the i-component of the rule. We note two facts about this index. First of all, there are $d(A)$ pairs in this index, that is, the number of elements in it is even. Second, such index can be encoded as a $(n + 1)$-ary number. The second fact allows us to use the *FromIndex* algorithm (*ToIndex* inverse to it) to convert such a number to the $(n + 1)$-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 i and j), and then put *True* value in the cell (i, j).

Further, the algorithm uses five matrices for each nonterminal rule. Namely, for a rule p of the form $A \to f(B, C)$, the algorithm supports the matrix B_p in which the result is stored, taking into account the sets in Definition 7 for the nonterminal B, as well as the matrix B_{p_new}, which stores the result, taking into account the sets which was obtained only at the previous step. Similarly for the nonterminal C. Also, the information about found paths corresponding to this rule is stored in matrix A_p, taking into account the set *end_A*(p). 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 update the matrices B_{p_new} and C_{p_new} to store only new values that was added at this iteration. Also, algorithm writes only new values to the matrix A_p for the nonterminal A and propagate new results among all matrices for the nonterminal A 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 $G = (N, \Sigma, P, S, d)$ be an MCFG in *normal form*, $D = (V, E, \Sigma)$ be an labeled directed graph and *Res* be the matrix obtained as a result of the algorithm in Listing 1. Then $(v_0, v_n) \in R_{D,L(G)}$ iff $Res[v_0, v_n] = True$.

Evaluation

We provide a prototype implementation of the proposed MCFL-reachability algorithm using the pygraphblas

```
G = (N, Σ, P, S) – MCFG, D = (V, E, Σ) – directed edge-labeled graph
A_p, B_p, B_p_new, C_p, C_p_new ← empty Boolean matrices
procedure MCFL_reachability(G, D)
    for all p ∈ P : p = A → (a_1, …, a_d(A)) do
        for all (l_1, a_1, r_1), …, (l_d(A), a_d(A), r_d(A)) ∈ E do
            index → (l_1, r_1, l_2, …, r_d(A))
            update(A, index)     // add information about terminating rules
    for all p ∈ P : p = A → f(B, C) do
        B_p_new, C_p_new ← B_p, C_p     // all values are new for the first iteration
    while matrices B_p, C_p are changing do
        for all p ∈ P : p = A → f(B, C) do     // consider nonterminating rules
            A_p_new ← B_p_new × C_p + B_p × C_p_new     // use only new values
            B_p_new, C_p_new ← empty Boolean matrices
            for all (i, j) : A_p_new[i, j] = True ∧ A_p[i, j] = False do
                A_p[i, j] ← True
                index ← transform_index(ToIndex(i), ToIndex(j), p)
                update(A, index)     // add new information for nonterminal A
    Res ← empty Boolean matrix of proper size
    for all p ∈ P : p = S → f(B, C) do     // collect all information for the start nonterminal S
        for all (i, j) : S_p[i, j] = True do
            index ← transform_index(ToIndex(i), ToIndex(j), p)
            Res[index[0], index[1]] ← True     // size of index is equal to 2 since d(S) = 1
    return Res
```

Listing 1. A matrix-based MCFL-reachability algorithm

```
procedure update(B, index)     // update matrices for all rules with B in the rhs
    for all p ∈ P : p = A → f(B, C) do
        i_B, j_B ← empty lists
        for all end ∈ end_B(p) do
            i_B.append(index[end])
        for all alter ∈ alter_B(p) do
            j_B.append(index[alter])
        B_p[FromIndex(i_B), FromIndex(j_B)] ← True
    for all p ∈ P : p = A → f(C, B) do
        i_B, j_B ← empty lists
        for all alter ∈ alter_B(p) do
            i_B.append(index[alter − 2d(C)])
        for all end ∈ end_B(p) do
            j_B.append(index[end − 2d(C)])
```

Listing 2. The procedure for updating matrix values

```
1  procedure transform_index(i, j, p)    // transform the indices i
   and j taking into account the set end_A(p)
2      A ← the nonterminal in the lhs of p
3      B, C ← the nonterminals in the rhs of p
4      index ← empty list
5      for all (end_l, end_r) ∈ end_A(p) do
6          if  end_l < 2d(B) then
7              pos ← position end_l in end_B(p)
8              index.append(i[pos])
9          else
10             pos ← position end_l in end_C(p)
11             index.append(j[pos])
12         if  end_r < 2d(B) then
13             pos ← position end_r in end_B(p)
14             index.append(i[pos])
15         else
16             pos ← position end_r in end_C(p)
17             index.append(j[pos])
18     return index
```

Listing 3. The procedure for index transformation

implementation of the GraphBLAS API. The source code is available on GitHub[1]. For evaluation, we use a PC with Ubuntu 18.04 installed. It has Intel core i7-6700 CPU, 3.4 GHz, and DDR4 64 Gb RAM.

We evaluate our implementation on some real-world RDFs and synthetic graphs using following classical MCFLs. The query Q_1 corresponds to the MCFL $L_1 = \{a^n b^m c^n d^m | n, m \in \mathbb{N}\}$ and the query Q_2 corresponds to the MCFL $L_2 = \{b^n a^m b^n | n, m \in \mathbb{N}\}$. These languages are known to be not context-free [15]. We use some RDFs from the CFPQ_Data dataset[2] 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.

The results of the MCFL-reachability evaluation for queries Q_1 and Q_2 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 B_p and C_p added by the MCFL-

[1] Sources of the prototype implementation of the proposed MCFL-reachability algorithm. Available at: https://github.com/FormalLanguageConstrainedPathQuerying/CFPQ_PyAlgo (accessed: 09.12.2022).

[2] CFPQ_Data dataset GitHub repository. Available at: https://github.com/FormalLanguageConstrainedPathQuerying/CFPQ_Data (accessed: 09.12.2022).

Table 1. MCFL-reachability execution time of queries Q_1 and Q_2

Graph type	Graph	Number of elements		Execution time, s	
		$\|V\|$	$\|E\|$	Q_1	Q_2
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	LFR_{100}	100	210	0.12	0.06
	LFR_{500}	500	970	2.55	1.47
	LFR_{1000}	1000	2100	13.50	6.10
	LFR_{1000}	10,000	21,005	1261.97	656.28

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

Graph type	Graph	Number of elements		
		$\|E\|$	NNZ_1	NNZ_2
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	LFR_{100}	210	5687	2851
	LFR_{500}	970	118,449	63,786
	LFR_{1000}	2100	553,002	276,299
	LFR_{1000}	21,005	55,172,204	27,349,209

reachability algorithm in Table 2. The NNZ_1 is a sum of non-zero elements in these matrices for the query Q_1, and the NNZ_2 — for the query Q_2. 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 query, and 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 memory, 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.

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 the MCFL parsing algorithm from [13]. Thus, we show how the MCFL-reachability problem can be solved using the LA operations. We implement the proposed algorithm using the GraphBLAS API. We should improve our prototype implementation using various high-performance libraries for the LA operations, distributed computations, and modern parallel hardware like GPU. Also, we should 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, vol. 30, no. 3, pp. 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. *Proc. of the 3rd Joint International Workshop on Graph Data Management Experiences & Systems (GRADES) and Network Data Analytics (NDA)*, 2020, pp. 1–12. 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. *Lecture Notes in Computer Science*, 2016, vol. 9981, pp. 632–648. https://doi.org/10.1007/978-3-319-46523-4_38
4. Zheng X., Rugina R. Demand-driven alias analysis for C. *Proc. of the 35th Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, 2008, pp. 197–208. https://doi.org/10.1145/1328438.1328464
5. Sevon P., Eronen L. Subgraph queries by context-free grammars. *Journal of Integrative Bioinformatics*, 2008, vol. 5, no. 2, pp. 157–172. 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. *Proc. of the 34th ACM*

Литература

1. Barrett C., Jacob R., Marathe M. Formal-language-constrained path problems // SIAM Journal on Computing. 2000. V. 30. N 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 // Proc. of the 3rd Joint International Workshop on Graph Data Management Experiences & Systems (GRADES) and Network Data Analytics (NDA). 2020. P. 1–12. 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 // Lecture Notes in Computer Science. 2016. V. 9981. P. 632–648. https://doi.org/10.1007/978-3-319-46523-4_38
4. Zheng X., Rugina R. Demand-driven alias analysis for C // Proc. of the 35th Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages. 2008. P. 197–208. 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. N 2. P. 157–172. 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 // Proc. of the 34th ACM SIGPLAN Conference on Programming Language Design and

SIGPLAN Conference on Programming Language Design and Implementation, 2013, pp. 435–446. https://doi.org/10.1145/2491956.2462159
7. Reps T. Undecidability of context-sensitive data-dependence analysis. *ACM Transactions on Programming Languages and Systems*, 2000, vol. 22, no. 1, pp. 162–186. https://doi.org/10.1145/345099.345137
8. Hopcroft J.E., Motwani R., Ullman J.D. Introduction to automata theory, languages, and computation. *ACM Sigact News*, 2001, vol. 32, no. 1, pp. 60–65. https://doi.org/10.1145/568438.568455
9. Zhang Q., Su Z. Context-sensitive data-dependence analysis via linear conjunctive language reachability. *Proc. of the 44th ACM SIGPLAN Symposium on Principles of Programming Languages*, 2017, pp. 344–358. https://doi.org/10.1145/3009837.3009848
10. Kepner J., Aaltonen P., Bader D., Buluç A., Franchetti F., Gilbert J., Hutchison D., Kumar M., Lumsdaine A., Meyerhenke H., McMillan S., Yang C., Owens J.D., Zalewski M., Mattson T., Moreira J. Mathematical foundations of the graphblas. *Proc. of the 2016 IEEE High Performance Extreme Computing Conference (HPEC)*, 2016, pp. 1–9. https://doi.org/10.1109/hpec.2016.7761646
11. Azimov R., Epelbaum I., Grigorev S. Context-free path querying with all-path semantics by matrix multiplication. *Proc. of the 4th ACM SIGMOD Joint International Workshop on Graph Data Management Experiences & Systems (GRADES) and Network Data Analytics (NDA)*, 2021, pp. 1–7. https://doi.org/10.1145/3461837.3464513
12. Azimov R., Grigorev S. Context-free path querying by matrix multiplication. *Proc. of the 1st ACM SIGMOD Joint International Workshop on Graph Data Management Experiences & Systems (GRADES) and Network Data Analytics (NDA)*, 2018, pp. 1–10. https://doi.org/10.1145/3210259.3210264
13. Nakanish R., Takada K., Seki H. An efficient recognition algorithm for multiple context-free languages. *Proc. of the 5th Meeting on the Mathematics of Language (MOL5)*, 1997, pp. 119–123.
14. Cohen S.B., Gildea D. Parsing linear context-free rewriting systems with fast matrix multiplication. *Computational Linguistics*, 2016, vol. 42, no. 3, pp. 421–455. https://doi.org/10.1162/coli_a_00254
15. Seki H., Matsumura T., Fujii M., Kasami T. On multiple context-free grammars. *Theoretical Computer Science*, 1991, vol. 88, no. 2, pp. 191–229. https://doi.org/10.1016/0304-3975(91)90374-B
16. Hagberg A., Schult D., Swart P. Exploring network structure, dynamics, and function using network, *Proc. of the 7th Python in Science Conference (SciPy 2008)*, 2008, pp. 11–15.

Implementation. 2013. P. 435–446. https://doi.org/10.1145/2491956.2462159
7. Reps T. Undecidability of context-sensitive data-dependence analysis // ACM Transactions on Programming Languages and Systems. 2000. V. 22. N 1. P. 162–186. https://doi.org/10.1145/345099.345137
8. Hopcroft J.E., Motwani R., Ullman J.D. Introduction to automata theory, languages, and computation // ACM Sigact News. 2001. V. 32. N 1. P. 60–65. https://doi.org/10.1145/568438.568455
9. Zhang Q., Su Z. Context-sensitive data-dependence analysis via linear conjunctive language reachability // Proc. of the 44th ACM SIGPLAN Symposium on Principles of Programming Languages. 2017. P. 344–358. https://doi.org/10.1145/3009837.3009848
10. Kepner J., Aaltonen P., Bader D., Buluç A., Franchetti F., Gilbert J., Hutchison D., Kumar M., Lumsdaine A., Meyerhenke H., McMillan S., Yang C., Owens J.D., Zalewski M., Mattson T., Moreira J. Mathematical foundations of the graphblas // Proc. of the 2016 IEEE High Performance Extreme Computing Conference (HPEC). 2016. P. 1–9. https://doi.org/10.1109/hpec.2016.7761646
11. Azimov R., Epelbaum I., Grigorev S. Context-free path querying with all-path semantics by matrix multiplication // Proc. of the 4th ACM SIGMOD Joint International Workshop on Graph Data Management Experiences & Systems (GRADES) and Network Data Analytics (NDA). 2021. P. 1–7. https://doi.org/10.1145/3461837.3464513
12. Azimov R., Grigorev S. Context-free path querying by matrix multiplication // Proc. of the 1st ACM SIGMOD Joint International Workshop on Graph Data Management Experiences & Systems (GRADES) and Network Data Analytics (NDA). 2018. P. 1–10. https://doi.org/10.1145/3210259.3210264
13. Nakanish, R., Takada K., Seki H. An efficient recognition algorithm for multiple context-free languages // Proc. of the 5th Meeting on the Mathematics of Language (MOL5). 1997. P. 119–123.
14. Cohen S.B., Gildea D. Parsing linear context-free rewriting systems with fast matrix multiplication // Computational Linguistics. 2016. V. 42. N 3. P. 421–455. https://doi.org/10.1162/coli_a_00254
15. Seki H., Matsumura T., Fujii M., Kasami T. On multiple context-free grammars // Theoretical Computer Science. 1991. V. 88. N 2. P. 191–229. https://doi.org/10.1016/0304-3975(91)90374-B
16. Hagberg A., Schult D., Swart P. Exploring network structure, dynamics, and function using network // Proc. of the 7th Python in Science Conference (SciPy 2008). 2008. P. 11–15.

Authors

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

Авторы

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

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

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

Received 30.09.2022
Approved after reviewing 09.12.2022
Accepted 14.03.2023

Статья поступила в редакцию 30.09.2022
Одобрена после рецензирования 09.12.2022
Принята к печати 14.03.2023