Context-Free Path Querying by Using Kronecker Product*

First Author[1[0000−1111−2222−3333]], Second Author[2,3[1111−2222−3333−4444]], and Third Author[3[2222−−3333−4444−5555]]

[1] Princeton University, Princeton NJ 08544, USA
[2] Springer Heidelberg, Tiergartenstr. 17, 69121 Heidelberg, Germany
`lncs@springer.com`
http://www.springer.com/gp/computer-science/lncs
[3] ABC Institute, Rupert-Karls-University Heidelberg, Heidelberg, Germany
{`abc,lncs`}`@uni-heidelberg.de`

Abstract. Abstact is very abstract.

Keywords: Path querying · Graph database · Context-free grammars · CFPQ · Kronecker product · !!! .

1 Introduction

CFPQ is popular.

Matrices [**?**] — algorithm is fast, but grammar size is problem. Moreover, bad for regualr queryes.

* Supported by organization x.

Following contribution.

1. !!!
2. !!!
3. !!!

2 Recursive State Machines

Or recursive networks [**?**] or resursive finite automata [**?**] or ...

3 Kronecker Product

For graphs, for matrices, for FA intersection.

4 Kronecker Product Based CFPQ Algorithm

The idea of the algorithm is based on generalisation of the finite-state machine intersection for a recursive automata, created from input grammar, and an input graph. The result of the intersection is evaluated as a tensor product of the corresponding adjacency matrices for automata and graph. To solve reachability problem it is enough to represent intersection result as a Boolean matrix, what simplifies algorithm implementation and allows to express it in terms of basic matrix operations. Listing 1. shows main steps of the solution.

As an input algorithm accepts context-free grammar $G = (\Sigma, N, P)$ and graph $\mathcal{G} = (V, E, L)$. Recursive automata R is created from G. The process of the creation is out of the scope of this article. M_1 and M_2 are the adjacency matrices for automata R and graph $\mathcal{G}$ correspondingly. Cell values of this matrices could be represented as sets of elements from $L \cup N \cup \Sigma$.

The algorithm is executed while matrix M_2 is changing. For each iteration tensor product of matrices M_1 and M_2 is evaluated. The result is saved in M_3 as a Boolean matrix. For given M_3 evaluated tC_3 matrix via $transitiveClosure()$ function call. The M_3 could be interpreted as an adjacency matrix for an oriented graph without labels, used to evaluate transitive closure in terms of classical graph definition of this operation. Then the algorithm iterates over cells of the tC_3. For pair of indices (i, j) computes s and f - initial and final states in recursive automata R which relate to the concrete $tC_3[i, j]$ of the tensor matrix. Function $hasPathForNonterminals()$ checks whether for given s and f states automata has at least one non-terminal path. If the conditional statement is *true* then algorithm adds non-terminals of that path via $getNonterminals()$ to the concrete cell of the adjacency matrix M_2 of the graph.

As an result the algorithm returns updated matrix M_2 which contains initial graph $\mathcal{G}$ data and non-terminals from N. If a cell $M_2[i, j]$ for any valid indices i and j contains symbol $S \in N$, therefore, vertex j is reachable from vertex i in grammar G for non-terminal S.

Listing 1 Kronecker product based CFPQ

function CONTEXTFREEPATHQUERYING(G, $\mathcal{G}$)
  $R \leftarrow$ Recursive automata for G
  $M_1 \leftarrow$ Adjacency matrix for R
  $M_2 \leftarrow$ Adjacency matrix for $\mathcal{G}$
  while Matrix M_2 is changing **do**
    $M_3 \leftarrow M_1 \otimes M_2$ ▷ Evaluate tensor product
    $tC_3 \leftarrow \mathit{transitiveClosure}(M_3)$
    $n \leftarrow$ Matrix M_3 dimension ▷ Matrix M_3 size $= n \times n$
    for $i \in 0..n-1$ **do**
      for $j \in 0..n-1$ **do**
        if $tC_3[i,j]$ **then**
          $s \leftarrow$ initial vertex of the edge $tC_3[i,j]$
          $f \leftarrow$ final vertex of the edge $tC_3[i,j]$
          if $hasPathForNonterminals(R, s, f)$ **then**
            $x, y \leftarrow getCoordinates(tC_3, i, j)$
            $M_2[x,y] \leftarrow M_2[x,y] \cup getNonterminals(R, s, f)$
  return M_2

4.1 Remarks

- Mentioned above algorithm description does not take into account the use of ϵ-transitions in the automata R. This transitions might appear in the automata if the grammar allows to derive ϵ-word for some non-terminal. In this case there is required additional step for matrix M_2 before the *while* loop is entered. For each $i \in 0..dim(M_2)-1$ symbol ϵ must be explicitly added for $M_2[i,i]$ as follows: $M_2[i,i] \leftarrow M_2[i,i] \cup \{\epsilon\}$. Here the rule is implied: each vertex of the graph $\mathcal{G}$ is reachable by itself through ϵ-transition.
- The performance-critical part of the algorithm is transitive closure computation. Generally this step requires $O(n^3)$ operations and $O(n^2)$ memory where n is dimension of M_3 what equals $dim(M_1) \times dim(M_2)$.

4.2 Example

This section is intended to provide step-by-step demonstration of the proposed algorithm. As an example query consider the following context-free grammar $G = (\Sigma, N, P)$ for a language $\{a^n b^n | n \geq 1\}$ where:

- Set of terminals $\Sigma = \{a, b\}$.
- Set of non-terminals $V = \{S\}$.
- Set of production rules $P = \{S \rightarrow aSb, S \rightarrow ab\}$.

Since the proposed algorithm processes grammar in form of recursive automata, we first provide automata R in Figure 1. The initial state of the automata is (0), the final state is (3). The notation $\{S\}$ denotes here that non-terminal S could be derived in automata path from vertex (0) to (3).

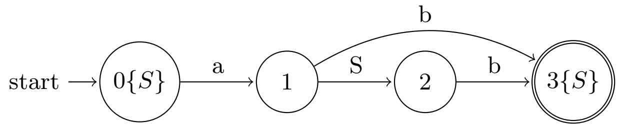

Fig. 1: The recursive automata R of grammar G for example query

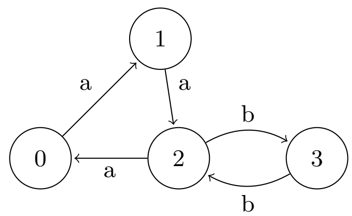

Fig. 2: The input graph $\mathcal{G}$ for example query

For this example we run query on graph $\mathcal{G}$ presented in Figure 2. This graph consists of 4 vertices and 5 edges with labels.

Adjacency matrices M_1 and M_2 for automata R and graph $\mathcal{G}$ respectively are initialised as follows:

$$M_1 = \begin{pmatrix} . & . & \{a\} & . \\ . & . & \{S\} & \{b\} \\ . & . & . & \{b\} \\ . & . & . & . \end{pmatrix}, \qquad M_2^0 = \begin{pmatrix} . & \{a\} & . & . \\ . & . & \{a\} & . \\ \{a\} & . & . & \{b\} \\ . & . & \{b\} & . \end{pmatrix}.$$

Because automata R does not have ε-transitions and ε-word is not included in grammar G language, we can skip additional step for matrix M_2 mentioned in section 4.1.

After all the data is initialised in lines **2–4** of the algorithm, it enters while loop and iterates as long as matrix M_2 is changing. We provide step-by-step evaluation of matrices M_3, tC_3 and updating of matrix M_2. All the matrices are denoted with upper index of the current loop iteration. The first loop iteration is indexed as 1.

For the first while loop iteration the tensor product $M_3^1 = M_1 \otimes M_2^0$ and transitive closure tC_3^1 are evaluated as follows:

$$M_3^1 = \left(\begin{array}{cccc|cccc|cccc|cccc}
\cdot & \cdot & \cdot & \cdot & \cdot & 1 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & 1 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & 1 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
\hline
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & 1 \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & 1 & \cdot \\
\hline
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & 1 \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & 1 & \cdot \\
\hline
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot
\end{array}\right), \quad
tC_3^1 = \left(\begin{array}{cccc|cccc|cccc|cccc}
\cdot & \cdot & \cdot & \cdot & \cdot & 1 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & 1 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \boxed{\mathbf{1}} \\
\cdot & \cdot & \cdot & \cdot & 1 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
\hline
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & 1 \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & 1 & \cdot \\
\hline
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & 1 \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & 1 & \cdot \\
\hline
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot
\end{array}\right).$$

The dimension n of the matrix M_3 equals 16, and this value is constant in time of the algorithm execution.

After the transitive closure evaluation matrix tC_3^1 cell $(1, 15)$ contains non-zero value. It means that vertex with index 15 is accessible from vertex with index 1 in a graph, represented by adjacency matrix M_3^1.

Then the algorithm lines **9-16** are executed. In that section algorithm adds non-terminals to the graph matrix M_2^1. Because this step is additive we are only interested in newly appeared values in matrix tC_3^1 such as value $tC_3^1[1, 15]$.

For the value $tC_3^1[1, 15]$:

- Indices of the automata vertices $s = 0$ and $f = 3$, because value $tC_3^1[1, 15]$ located in upper right matrix block $(0, 3)$.
- Indices of the graph vertices $x = 1$ and $y = 3$ are evaluated as value $tC_3^1[1, 15]$ indices relatively to its block $(0, 3)$.
- Function call $hasPathForNonterminals()$ returns **true** since the automata R has path for non-terminal S from vertex 0 to 3.
- Function call $getNonterminals()$ returns $\{S\}$ since this is the only non-terminal which could be derived in path from vertex 0 to 3.

$$M_2^1 = \begin{pmatrix} . & \{a\} & . & . \\ . & . & \{a\} & \{S\} \\ \{a\} & . & . & \{b\} \\ . & . & \{b\} & . \end{pmatrix}$$

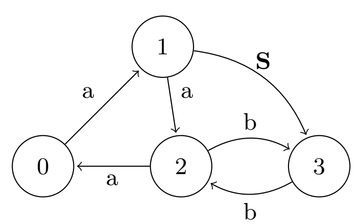

Fig. 3: The updated matrix M_2^1 and graph $\mathcal{G}$ after first loop iteration for example query

After the first loop iteration matrix symbol S is added to the cell $M_2^1[1, 3]$. It is relevant data, because initial graph has path $1 \to 2 \to 3$ which could be derived for S. The updated matrix and graph are depicted in figure 4.

For the second loop iteration matrices M_3^2 and tC_3^2 are evaluated as listed in figure 5. For this iteration in the matrix tC_3^2 appeared new non-zero values in cells with indices $[0, 11]$, $[0, 14]$ and $[5, 14]$. Because only the cell value with index $[0, 14]$ corresponds to the automata path with not empty non-terminal set $\{S\}$ its data affects adjacency matrix M_2. The update matrix and graph $\mathcal{G}$ are depicted in figure 6.

$$M_3^2 = \left(\begin{array}{cccc|cccc|cccc|cccc}
\cdot & \cdot & \cdot & \cdot & \cdot & 1 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & 1 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & 1 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
\hline
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \mathbf{1} & \cdot & \cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & 1 \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & 1 & \cdot \\
\hline
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & 1 \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & 1 & \cdot \\
\hline
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot
\end{array}\right)
\quad
tC_3^2 = \left(\begin{array}{cccc|cccc|cccc|cccc}
\cdot & \cdot & \cdot & \cdot & \cdot & 1 & \cdot & \cdot & \cdot & \cdot & \cdot & \mathbf{1} & \cdot & \cdot & \mathbf{1} & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & 1 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & 1 \\
\cdot & \cdot & \cdot & \cdot & 1 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
\hline
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & 1 & \cdot & \cdot & \mathbf{1} & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & 1 \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & 1 & \cdot \\
\hline
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & 1 \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & 1 & \cdot \\
\hline
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot
\end{array}\right)$$

Fig. 4: The second iteration tensor product and transitive closure evaluation for example query

$$M_2^2 = \begin{pmatrix} . & \{a\} & \{S\} & . \\ . & . & \{a\} & \{S\} \\ \{a\} & . & . & \{b\} \\ . & . & \{b\} & . \end{pmatrix}$$

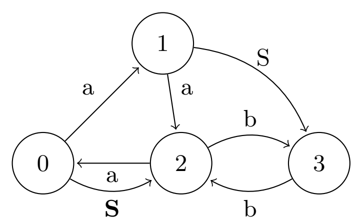

Fig. 5: The updated matrix M_2^2 and graph $\mathcal{G}$ after second loop iteration for example query

The remaining matrices tC_3 and M_2 for the algorithm main loop execution are listed in the figure 7 and figure 8 correspondingly. For the sake of simplicity evaluated matrices M_3 are not included because its computation is a straightforward process. The last loop iteration is 7. Although the matrix M_2^6 is updated with new non-terminal S for the cell $[2, 2]$ after transitive closure evaluation the new values to the matrix M_2 is not added. Therefore matrix M_2 has stopped changing and the algorithm is successfully finished.

For the example query algorithm takes 7 iterations for the $while - loop$. The updated graph $\mathcal{G}$ is depicted in the figure 9.

$$tC_3^3 = \left(\begin{array}{cccc|cccc|cccc|cccc}
. & . & . & . & . & 1 & . & . & . & . & . & 1 & . & . & 1 & . \\
. & . & . & . & . & . & 1 & . & . & . & . & . & . & . & . & 1 \\
. & . & . & . & 1 & . & . & . & . & . & \mathbf{1} & . & . & . & . & \mathbf{1} \\
. & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . \\
\hline
. & . & . & . & . & . & . & . & . & . & 1 & . & . & . & . & \mathbf{1} \\
. & . & . & . & . & . & . & . & . & . & . & 1 & . & . & 1 & . \\
. & . & . & . & . & . & . & . & . & . & . & . & . & . & . & 1 \\
. & . & . & . & . & . & . & . & . & . & . & . & . & . & 1 & . \\
\hline
. & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . \\
. & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . \\
. & . & . & . & . & . & . & . & . & . & . & . & . & . & . & 1 \\
. & . & . & . & . & . & . & . & . & . & . & . & . & . & 1 & . \\
\hline
. & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . \\
. & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . \\
. & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . \\
. & . & . & . & . & . & . & . & . & . & . & . & . & . & . & .
\end{array}\right)
\quad
tC_3^4 = \left(\begin{array}{cccc|cccc|cccc|cccc}
. & . & . & . & . & 1 & . & . & . & . & . & 1 & . & . & 1 & . \\
. & . & . & . & . & . & 1 & . & . & . & . & \mathbf{1} & . & . & \mathbf{1} & 1 \\
. & . & . & . & 1 & . & . & . & . & . & 1 & . & . & . & . & 1 \\
. & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . \\
\hline
. & . & . & . & . & . & . & . & . & . & 1 & . & . & . & . & 1 \\
. & . & . & . & . & . & . & . & . & . & . & 1 & . & . & 1 & . \\
. & . & . & . & . & . & . & . & . & . & . & 1 & . & . & \mathbf{1} & 1 \\
. & . & . & . & . & . & . & . & . & . & . & . & . & . & 1 & . \\
\hline
. & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . \\
. & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . \\
. & . & . & . & . & . & . & . & . & . & . & . & . & . & . & 1 \\
. & . & . & . & . & . & . & . & . & . & . & . & . & . & 1 & . \\
\hline
. & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . \\
. & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . \\
. & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . \\
. & . & . & . & . & . & . & . & . & . & . & . & . & . & . & .
\end{array}\right)$$

$$tC_3^5 = \left(\begin{array}{cccc|cccc|cccc|cccc}
. & . & . & . & . & 1 & . & . & . & . & \mathbf{1} & 1 & . & . & 1 & \mathbf{1} \\
. & . & . & . & . & . & 1 & . & . & . & . & 1 & . & . & 1 & 1 \\
. & . & . & . & 1 & . & . & . & . & . & 1 & . & . & . & . & 1 \\
. & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . \\
\hline
. & . & . & . & . & . & . & . & . & . & 1 & . & . & . & . & 1 \\
. & . & . & . & . & . & . & . & . & . & 1 & 1 & . & . & 1 & \mathbf{1} \\
. & . & . & . & . & . & . & . & . & . & . & 1 & . & . & 1 & 1 \\
. & . & . & . & . & . & . & . & . & . & . & . & . & . & 1 & . \\
\hline
. & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . \\
. & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . \\
. & . & . & . & . & . & . & . & . & . & . & . & . & . & . & 1 \\
. & . & . & . & . & . & . & . & . & . & . & . & . & . & 1 & . \\
\hline
. & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . \\
. & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . \\
. & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . \\
. & . & . & . & . & . & . & . & . & . & . & . & . & . & . & .
\end{array}\right)
\quad
tC_3^6 = \left(\begin{array}{cccc|cccc|cccc|cccc}
. & . & . & . & . & 1 & . & . & . & . & 1 & 1 & . & . & 1 & 1 \\
. & . & . & . & . & . & 1 & . & . & . & . & 1 & . & . & 1 & 1 \\
. & . & . & . & 1 & . & . & . & . & . & 1 & \mathbf{1} & . & . & \mathbf{1} & 1 \\
. & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . \\
\hline
. & . & . & . & . & . & . & . & . & . & 1 & 1 & . & . & \mathbf{1} & 1 \\
. & . & . & . & . & . & . & . & . & . & 1 & 1 & . & . & 1 & 1 \\
. & . & . & . & . & . & . & . & . & . & . & 1 & . & . & 1 & 1 \\
. & . & . & . & . & . & . & . & . & . & . & . & . & . & 1 & . \\
\hline
. & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . \\
. & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . \\
. & . & . & . & . & . & . & . & . & . & . & . & . & . & . & 1 \\
. & . & . & . & . & . & . & . & . & . & . & . & . & . & 1 & . \\
\hline
. & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . \\
. & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . \\
. & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . \\
. & . & . & . & . & . & . & . & . & . & . & . & . & . & . & .
\end{array}\right)$$

Fig. 6: Transitive closure for $3-6$ loop iterations for example query

$$M_2^3 = \begin{pmatrix} . & \{a\} & \{S\} & . \\ . & . & \{a\} & \{S\} \\ \{a\} & . & . & \{b,S\} \\ . & . & \{b\} & . \end{pmatrix}
\; M_2^4 = \begin{pmatrix} . & \{a\} & \{S\} & . \\ . & . & \{a,S\} & \{S\} \\ \{a\} & . & . & \{b,S\} \\ . & . & \{b\} & . \end{pmatrix}$$

$$M_2^5 = \begin{pmatrix} . & \{a\} & \{S\} & \{S\} \\ . & . & \{a,S\} & \{S\} \\ \{a\} & . & . & \{b,S\} \\ . & . & \{b\} & . \end{pmatrix}
\; M_2^6 = \begin{pmatrix} . & \{a\} & \{S\} & \{S\} \\ . & . & \{a,S\} & \{S\} \\ \{a\} & . & \{S\} & \{b,S\} \\ . & . & \{b\} & . \end{pmatrix}$$

Fig. 7: The updated matrix M_2 for $3-6$ loop iterations for example query

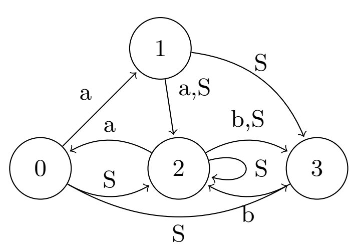

Fig. 8: The result graph $\mathcal{G}$ for example query

5 Evaluation

RedisGraph + CFPQ_Data

Cases, when kronecker should be significantly better that matrix. When grammar is big. When query is regular.

6 Conclusion

Future research. GraphBLAST. Paths, not just reachability.