



ACM SIGMOD 2020 SRC

# Context-Free Path Querying via Matrix Equations

Yuliya Susanina

JetBrains Research, Programming Languages and Tools Lab  
Saint-Petersburg State University

14.06.2020

# Context-Free Path Querying (CFPQ)

- Context-free grammar  $G = (N, \Sigma, R)$   
 $\mathcal{L}(G_S) = \{\omega \mid S \Rightarrow_G^* \omega\}, S \in N$
- Directed graph  $D = (V, E, \sigma), \sigma \subseteq \Sigma, E \subseteq V \times \sigma \times V$   
 $m\lambda n$  — path from  $m$  to  $n$  in  $D$ ,  $\lambda$  — a unique word of this path
- $R_A = \{(m, n) \mid m\lambda n \text{ — path in } D, \lambda \in \mathcal{L}(G_A)\}$

# Context-Free Path Querying (CFPQ)

- Context-free grammar  $G = (N, \Sigma, R)$   
 $\mathcal{L}(G_S) = \{\omega \mid S \Rightarrow_G^* \omega\}, S \in N$
- Directed graph  $D = (V, E, \sigma), \sigma \subseteq \Sigma, E \subseteq V \times \sigma \times V$   
 $m\lambda n$  — path from  $m$  to  $n$  in  $D$ ,  $\lambda$  — a unique word of this path
- $R_A = \{(m, n) \mid m\lambda n \text{ — path в } D, \lambda \in \mathcal{L}(G_A)\}$
- Application areas:
  - ▶ Graph databases
  - ▶ Bioinformatics
  - ▶ Static code analysis

- Both rich theoretical foundations and constantly improving implementations
- Parallel techniques and GPGPU
- Approximate computational methods

- Both rich theoretical foundations and constantly improving implementations
- Parallel techniques and GPGPU
- Approximate computational methods



Acceleration of CFPQs processing

# Reduction from Solving Boolean Matrix Equations

$$S \rightarrow aSb \mid ab$$

$$T_E \in \mathbb{M}^{|V| \times |V|} : (T_E)_{ij} = 1 \iff (i, j) \in R_E \quad \forall E \in (N \cup \Sigma)$$

# Reduction from Solving Boolean Matrix Equations

$$S \rightarrow aSb \mid ab$$

$$T_E \in \mathbb{M}^{|V| \times |V|} : (T_E)_{ij} = 1 \iff (i, j) \in R_E \quad \forall E \in (N \cup \Sigma)$$

$\Downarrow$

$$\{T_S^k\} : \begin{array}{l} T_S^0 = \mathbf{0} \\ T_S^{k+1} = T_a T_S^k T_b + T_a T_b \end{array}$$

$$T_S^\infty - \text{least solution } T_S = T_a T_S T_b + T_a T_b$$



$$\{\mathcal{T}_S^k\} : \begin{array}{l} \mathcal{T}_S^0 = \mathbf{0} \\ \mathcal{T}_S^{k+1} = \epsilon(T_a \mathcal{T}_S^k T_b + T_a T_b) \end{array}$$

$\mathcal{T}_S^\infty$  – least solution  $\mathcal{T}_S = \epsilon(T_a \mathcal{T}_S T_b + T_a T_b)$ ,  
where  $\epsilon$  such that  $\mathcal{T}_S^k \leq \mathbf{1} \quad \forall k$





$$\{\mathcal{T}_S^k\} : \begin{array}{l} \mathcal{T}_S^0 = \mathbf{0} \\ \mathcal{T}_S^{k+1} = \epsilon(T_a \mathcal{T}_S^k T_b + T_a T_b) \end{array}$$

$\mathcal{T}_S^\infty$  – least solution  $\mathcal{T}_S = \epsilon(T_a \mathcal{T}_S T_b + T_a T_b)$ ,  
where  $\epsilon$  such that  $\mathcal{T}_S^k \leq \mathbf{1} \quad \forall k$

$$(\mathcal{T}_S^{k+1})_{ij} > 0 \iff (T_S^{k+1})_{ij} = 1$$

$$\text{ceil}(\mathcal{T}_S^\infty) = T_S^\infty$$

# Methods for Solving Equations

- Linear equations
  - ▶ Sylvester equations  $AXB + CXD = F$
  - ▶ Linear systems  $Ax = b$
- Nonlinear equations
  - ▶ Newton's method

$$X = G(X) \Rightarrow F(X) = X - G(X) = \mathbf{0}$$

$$X_{i+1} = X_i - (F'(X_i))^{-1}F(X_i) \iff \begin{cases} F'(X_i)H_i = -F(X_i) \\ X_{i+1} = X_i + H_i \end{cases}$$

# First implementation

- *SciPy*
  - ▶ sSLV — to solve as a sparse linear system
  - ▶ dNWT — to find a root of a function with Newton's method
- Comparative analysis of matrix-based approach and equation-based approach (in ms)

Ontology	V	dNWT	sSLV	dGPU	sCPU	sGPU
bio-meas	341	284	35	276	91	24
people-pets	337	73	49	144	38	6
funding	778	502	184	1246	344	27
wine	733	791	171	722	179	6
pizza	671	334	161	943	256	23

- Equation-based approach for CFPQ was proposed
- The possibilities of using both accurate and approximate methods of computational mathematics was reviewed
- The evaluation on a set of conventional benchmarks showed that our approach is comparable with the matrix-based approach and applicable for real-world data processing

- Employ high-performance solvers which utilize GPGPU and distributed computations
- Determine the subclasses of (system of) polynomial equations the solution of which can be reduced to CFPQ
- Try to construct a bidirectional reduction between CFPQ and these subclasses, thereby finding efficient solutions for both these problems