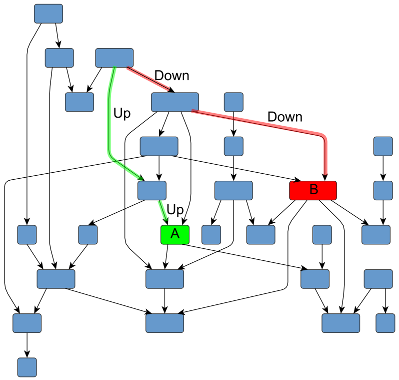
Context-Free Path Querying by Matrix Multiplication

Rustam Azimov, Semyon Grigorev
presents Kate Verbitskaia

JetBrains Research, Programming Languages and Tools Lab
Saint Petersburg University

June 10, 2018

Context-Free Path Querying



Navigation through a graph

- Are nodes A and B on the same level of hierarchy?
- Is there a path of form **Up**^{n} **Down**^{n}?
- Find all paths of form **Up**^{n} **Down**^{n} which start from the node A

Context-Free Path Querying: Relational Query Semantics

- $\mathbb{G} = (\Sigma, N, P)$ — context-free grammar in normal form
  - $A \to BC$, where $A, B, C \in N$
  - $A \to x$, where $A \in N, x \in \Sigma$
  - $L(\mathbb{G}, A) = \{\omega \mid A \to^* \omega\}$
- $G = (V, E, L)$ — directed graph
  - $v \xrightarrow{l} u \in E$
  - $L \subseteq \Sigma$
- $\omega(\pi) = \omega(v_0 \xrightarrow{l_0} v_1 \xrightarrow{l_1} \cdots \xrightarrow{l_{n-2}} v_{n-1} \xrightarrow{l_{n-1}} v_n) = l_0 l_1 \cdots l_{n-1}$
- $R_A = \{(n, m) \mid \exists n \pi m$, such that $\omega(\pi) \in L(\mathbb{G}, A)\}$

Regular Language Constraints

- Widely spread
  - Graph databases query languages (SPARQL, Cypher, PGQL)
  - Network analysis
- Still in active development
  - OpenCypher: `https://goo.gl/5h5a8P`
  - Scalability, huge graphs processing
  - Derivatives for graph querying: *Maurizio Nole and Carlo Sartiani.* Regular path queries on massive graphs. 2016

Context-Free Language Constraints

- Graph databases and semantic networks (Context-Free Path Querying, CFPQ)
  - *Sevon P., Eronen L.* "Subgraph queries by context-free grammars." 2008
  - *Hellings J.* "Conjunctive context-free path queries." 2014
  - *Zhang X. et al.* "Context-free path queries on RDF graphs." 2016
- Static code analysis (Language Reachability Framework)
  - *Thomas Reps et al.* "Precise interprocedural dataflow analysis via graph reachability." 1995
  - *Qirun Zhang et al.* "Efficient subcubic alias analysis for C." 2014
  - *Dacong Yan et al.* "Demand-driven context-sensitive alias analysis for Java." 2011
  - *Jakob Rehof and Manuel Fahndrich.* "Type-base flow analysis: from polymorphic subtyping to CFL-reachability." 2001

Open Problems

- Development of efficient algorithms
- Effective utilization of GPGPU and parallel programming
- Lifting up the limitations on the input graph and the query language

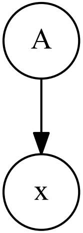
The algorithm

Algorithm Context-free recognizer for graphs

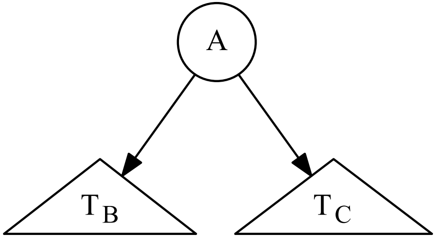
```
1: function CONTEXTFREEPATHQUERYING(D, G)
2:     $n \leftarrow$ the number of nodes in $D$
3:     $E \leftarrow$ the directed edge-relation from $D$
4:     $P \leftarrow$ the set of production rules in $G$
5:     $T \leftarrow$ the matrix $n \times n$ in which each element is $\varnothing$
6:     for all $(i, x, j) \in E$ do                          ▷ Matrix initialization
7:         $T_{i,j} \leftarrow T_{i,j} \cup \{A \mid (A \rightarrow x) \in P\}$
8:     while matrix $T$ is changing do
9:         $T \leftarrow T \cup (T \times T)$          ▷ Transitive closure $T^{cf}$ calculation
10:    return $T$
```

Derivation Step

$A \to x$



$A \to BC$



Matrix Multiplication

- Subset multiplication, $N_1, N_2 \subseteq N$
  - $N_1 \cdot N_2 = \{A \mid \exists B \in N_1, \exists C \in N_2 \text{ such that } (A \to BC) \in P\}$
- Subset addition: set-theoretic union.

- Matrix multiplication
  - Matrix of size $|V| \times |V|$
  - Subsets of N are elements
  - $c_{i,j} = \bigcup_{k=1}^{n} a_{i,k} \cdot b_{k,j}$

Transitive Closure

- $a^{cf} = a^{(1)} \cup a^{(2)} \cup \cdots$
- $a^{(1)} = a$
- $a^{(i)} = a^{(i-1)} \cup (a^{(i-1)} \times a^{(i-1)}),\ i \geq 2$

The algorithm

Algorithm Context-free recognizer for graphs

```
1: function CONTEXTFREEPATHQUERYING(D, G)
2:     $n \leftarrow$ the number of nodes in $D$
3:     $E \leftarrow$ the directed edge-relation from $D$
4:     $P \leftarrow$ the set of production rules in $G$
5:     $T \leftarrow$ the matrix $n \times n$ in which each element is $\varnothing$
6:     for all $(i, x, j) \in E$ do                              ▷ Matrix initialization
7:         $T_{i,j} \leftarrow T_{i,j} \cup \{A \mid (A \rightarrow x) \in P\}$
8:     while matrix $T$ is changing do
9:         $T \leftarrow T \cup (T \times T)$              ▷ Transitive closure $T^{cf}$ calculation
10:    return $T$
```

Algorithm Correctness

Theorem

Let $D = (V, E)$ be a graph and let $G = (N, \Sigma, P)$ be a grammar. Then for any i, j and for any non-terminal $A \in N$, $A \in a^{cf}_{i,j}$ iff $(i, j) \in R_A$.

Theorem

Let $D = (V, E)$ be a graph and let $G = (N, \Sigma, P)$ be a grammar. The Algorithm terminates in a finite number of steps.

Algorithm Complexity

Theorem

Let $D = (V, E)$ be a graph and let $G = (N, \Sigma, P)$ be a grammar. The Algorithm calculates the transitive closure T^{cf} in $O(|V|^2|N|^3(BMM(|V|) + BMU(|V|)))$.

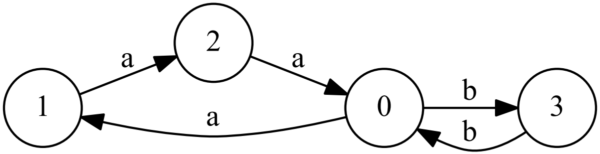
- $BMM(n)$ — number of elementary operations executed by the algorithm of multiplying two $n \times n$ Boolean matrices.
- $BMU(n)$ — number of elementary operations, executed by the matrix union operation of two $n \times n$ Boolean matrices

Algorithm Complexity: the Worst Case

Input graph: two cycles connected via a shared node

- first cycle has $2^k + 1$ edges labeled a
- second cycle has 2^k edges labeled b

$$
\begin{array}{lcl}
S & \to & a\ S\ b \\
  & \mid & a\ b
\end{array}
$$



The Worst Case: Step by Step

$$\begin{array}{lcl} S & \to & A\ B \\ & \mid & A\ S_1 \\ S_1 & \to & S\ B \\ A & \to & a \\ B & \to & b \end{array}$$

$$T_0 = \begin{pmatrix} \varnothing & \{A\} & \varnothing & \{B\} \\ \varnothing & \varnothing & \{A\} & \varnothing \\ \{A\} & \varnothing & \varnothing & \varnothing \\ \{B\} & \varnothing & \varnothing & \varnothing \end{pmatrix}$$

The Worst Case: Step by Step

$$T_0 \times T_0 = \begin{pmatrix} \varnothing & \varnothing & \varnothing & \varnothing \\ \varnothing & \varnothing & \varnothing & \varnothing \\ \varnothing & \varnothing & \varnothing & \{S\} \\ \varnothing & \varnothing & \varnothing & \varnothing \end{pmatrix}$$

$$T_1 = T_0 \cup (T_0 \times T_0) = \begin{pmatrix} \varnothing & \{A\} & \varnothing & \{B\} \\ \varnothing & \varnothing & \{A\} & \varnothing \\ \{A\} & \varnothing & \varnothing & \{\boldsymbol{S}\} \\ \{B\} & \varnothing & \varnothing & \varnothing \end{pmatrix}$$

The Worst Case: Step by Step

$$T_2 = \begin{pmatrix} \varnothing & \{A\} & \varnothing & \{B\} \\ \varnothing & \varnothing & \{A\} & \varnothing \\ \{A, \boldsymbol{S_1}\} & \varnothing & \varnothing & \{S\} \\ \{B\} & \varnothing & \varnothing & \varnothing \end{pmatrix}$$

The Worst Case: Step by Step

$$T_3 = \begin{pmatrix} \varnothing & \{A\} & \varnothing & \{B\} \\ \{\boldsymbol{S}\} & \varnothing & \{A\} & \varnothing \\ \{A, S_1\} & \varnothing & \varnothing & \{S\} \\ \{B\} & \varnothing & \varnothing & \varnothing \end{pmatrix}$$

The Worst Case: Step by Step

$$T_4 = \begin{pmatrix} \varnothing & \{A\} & \varnothing & \{B\} \\ \{S\} & \varnothing & \{A\} & \{\boldsymbol{S_1}\} \\ \{A, S_1\} & \varnothing & \varnothing & \{S\} \\ \{B\} & \varnothing & \varnothing & \varnothing \end{pmatrix}$$

The Worst Case: Step by Step

$$T_5 = \begin{pmatrix} \varnothing & \{A\} & \varnothing & \{B, \boldsymbol{S}\} \\ \{S\} & \varnothing & \{A\} & \{S_1\} \\ \{A, S_1\} & \varnothing & \varnothing & \{S\} \\ \{B\} & \varnothing & \varnothing & \varnothing \end{pmatrix}$$

The Worst Case: Step by Step

$$T_6 = \begin{pmatrix} \{\boldsymbol{S_1}\} & \{A\} & \varnothing & \{B, S\} \\ \{S\} & \varnothing & \{A\} & \{S_1\} \\ \{A, S_1\} & \varnothing & \varnothing & \{S\} \\ \{B\} & \varnothing & \varnothing & \varnothing \end{pmatrix}$$

The Worst Case: Step by Step

$$T_7 = \begin{pmatrix} \{S_1\} & \{A\} & \varnothing & \{B, S\} \\ \{S\} & \varnothing & \{A\} & \{S_1\} \\ \{A, S_1, \boldsymbol{S}\} & \varnothing & \varnothing & \{S\} \\ \{B\} & \varnothing & \varnothing & \varnothing \end{pmatrix}$$

The Worst Case: Step by Step

$$T_8 = \begin{pmatrix} \{S_1\} & \{A\} & \varnothing & \{B, S\} \\ \{S\} & \varnothing & \{A\} & \{S_1\} \\ \{A, S_1, S\} & \varnothing & \varnothing & \{S, \boldsymbol{S_1}\} \\ \{B\} & \varnothing & \varnothing & \varnothing \end{pmatrix}$$

The Worst Case: Step by Step

$$T_9 = \begin{pmatrix} \{S_1\} & \{A\} & \varnothing & \{B, S\} \\ \{S\} & \varnothing & \{A\} & \{S_1, \boldsymbol{S}\} \\ \{A, S_1, S\} & \varnothing & \varnothing & \{S, S_1\} \\ \{B\} & \varnothing & \varnothing & \varnothing \end{pmatrix}$$

The Worst Case: Step by Step

$$T_{10} = \begin{pmatrix} \{S_1\} & \{A\} & \varnothing & \{B, S\} \\ \{S, \boldsymbol{S_1}\} & \varnothing & \{A\} & \{S_1, S\} \\ \{A, S_1, S\} & \varnothing & \varnothing & \{S, S_1\} \\ \{B\} & \varnothing & \varnothing & \varnothing \end{pmatrix}$$

The Worst Case: Step by Step

$$T_{11} = \begin{pmatrix} \{S_1, \boldsymbol{S}\} & \{A\} & \varnothing & \{B, S\} \\ \{S, S_1\} & \varnothing & \{A\} & \{S_1, S\} \\ \{A, S_1, S\} & \varnothing & \varnothing & \{S, S_1\} \\ \{B\} & \varnothing & \varnothing & \varnothing \end{pmatrix}$$

The Worst Case: Step by Step

$$T_{12} = \begin{pmatrix} \{S_1, S\} & \{A\} & \varnothing & \{B, S, \boldsymbol{S_1}\} \\ \{S, S_1\} & \varnothing & \{A\} & \{S_1, S\} \\ \{A, S_1, S\} & \varnothing & \varnothing & \{S, S_1\} \\ \{B\} & \varnothing & \varnothing & \varnothing \end{pmatrix}$$

The Worst Case: Step by Step

$$T_{13} = \begin{pmatrix} \{S_1, S\} & \{A\} & \varnothing & \{B, S, S_1\} \\ \{S, S_1\} & \varnothing & \{A\} & \{S_1, S\} \\ \{A, S_1, S\} & \varnothing & \varnothing & \{S, S_1\} \\ \{B\} & \varnothing & \varnothing & \varnothing \end{pmatrix}$$

Evaluation

- dGPU (dense GPU): row-major matrix representation and a GPU for matrix operation calculation.
- sCPU (sparse CPU): CSR format for sparse matrix representation and a CPU for matrix operation calculation.
- sGPU (sparse GPU): CSR format for sparse matrix representation and a GPU for matrix operation calculation.

Evaluation: Same Generation Queries

Query 1 retrieves the concepts on the same layer

$$\begin{array}{rcl} S & \to & \mathit{subClassOf}^{-1}\ S\ \mathit{subClassOf} \\ & | & \mathit{type}^{-1}\ S\ \mathit{type} \\ & | & \mathit{subClassOf}^{-1}\ \mathit{subClassOf} \\ & | & \mathit{type}^{-1}\ \mathit{type} \end{array}$$

Query 2 retrieves concepts on the adjacent layers

$$\begin{array}{rcl} S & \to & B\ \mathit{subClassOf} \\ & | & \mathit{subClassOf} \\ B & \to & \mathit{subClassOf}^{-1}\ B\ \mathit{subClassOf} \\ & | & \mathit{subClassOf}^{-1}\ \mathit{subClassOf} \end{array}$$

Evaluation: Query 1

Ontology	V	E	#results	GLL	dGPU	sCPU	sGPU
skos	144	323	810	10	56	14	12
generations	129	351	2164	19	62	20	13
travel	131	397	2499	24	69	22	30
univ-bench	179	413	2540	25	81	25	15
atom-prim	291	685	15454	255	190	92	22
biomedical	341	711	15156	261	266	113	20
foaf	256	815	4118	39	154	48	9
people-pets	337	834	9472	89	392	142	32
funding	778	1480	17634	212	1410	447	36
wine	733	2450	66572	819	2047	797	54
pizza	671	2604	56195	697	1104	430	24
g_1	6224	11840	141072	1926	—	26957	82
g_2	5864	19600	532576	6246	—	46809	185
g_3	5368	20832	449560	7014	—	24967	127

time is measured in ms

Evaluation: Query 2

Ontology	V	E	#results	GLL	dGPU	sCPU	sGPU
skos	144	323	1	1	10	2	1
generations	129	351	0	1	9	2	0
travel	131	397	63	1	31	7	10
univ-bench	179	413	81	11	55	15	9
atom-prim	291	685	122	66	36	9	2
biomedical	341	711	2871	45	276	91	24
foaf	256	815	10	2	53	14	3
people-pets	337	834	37	3	144	38	6
funding	778	1480	1158	23	1246	344	27
wine	733	2450	133	8	722	179	6
pizza	671	2604	1262	29	943	258	23
g_1	6224	11840	9264	167	—	21115	38
g_2	5864	19600	1064	46	—	10874	21
g_3	5368	20832	10096	393	—	15736	40

time is measured in ms

Summary

- Algorithm for context-free path querying
- Works on any input graph
- Supports any context-free constraints
- Is independent of matrix representation
- Can utilize GPGPU easily and efficiently

Contact Information

- Semyon Grigorev: s.v.grigoriev@spbu.ru
- Rustam Azimov: st013567@student.spbu.ru

- YaccConstructor: https://github.com/YaccConstructor