

Parsing Techniques for Context-Free Path Querying

Semyon Grigorev

s.v.grigoriev@spbu.ru

Semen.Grigorev@jetbrains.com

JetBrains Research, Programming Languages and Tools Lab
Saint Petersburg University

April 05, 2019

Formal language constrained path querying

- Finite directed edge-labelled graph $\mathcal{G} = (V, E, L)$
- The path is a word over L
$$\omega(p) = \omega(v_0 \xrightarrow{l_0} v_1 \xrightarrow{l_1} \dots \xrightarrow{l_{n-1}} v_n) = l_0 \cdot l_1 \cdot \dots \cdot l_{n-1}$$
- The language \mathcal{L} (over L)

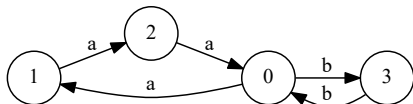
Formal language constrained path querying

- Finite directed edge-labelled graph $\mathcal{G} = (V, E, L)$
- The path is a word over L
$$\omega(p) = \omega(v_0 \xrightarrow{l_0} v_1 \xrightarrow{l_1} \dots \xrightarrow{l_{n-1}} v_n) = l_0 \cdot l_1 \cdot \dots \cdot l_{n-1}$$
- The language \mathcal{L} (over L)
- Reachability problem: $Q = \{(v_i, v_j) \mid \exists p = v_i \dots v_j, \omega(p) \in \mathcal{L}\}$
- Path querying problem: $Q = \{p \mid \omega(p) \in \mathcal{L}\}$
 - ▶ Single path, all paths, shortest path...

Context-Free Path Querying

- \mathcal{L} is a context-free language
- $G_{\mathcal{L}} = (N, \Sigma, R, S)$
- Reachability problem: $Q = \{(v_i, v_j) \mid \exists p = v_i \dots v_j, S \xrightarrow[G_{\mathcal{L}}]{*} \omega(p)\}$
- Path querying problem: $Q = \{p \mid S \xrightarrow[G_{\mathcal{L}}]{*} \omega(p)\}$

Example of CFPQ



Input graph

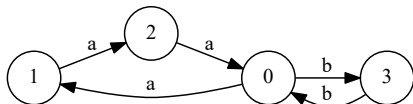
$S \rightarrow a S b$

$S \rightarrow \textit{Middle}$

$\textit{Middle} \rightarrow a b$

Query: language $\{a^n b^n \mid n > 0\}$

Example of CFPQ



Input graph

$S \rightarrow a S b$

$S \rightarrow \text{Middle}$

$\text{Middle} \rightarrow a b$

Query: language $\{a^n b^n \mid n > 0\}$

Paths:

$2 \xrightarrow{a} 0 \xrightarrow{b} 3$

$1 \xrightarrow{a} 2 \xrightarrow{a} 0 \xrightarrow{b} 3 \xrightarrow{b} 0$

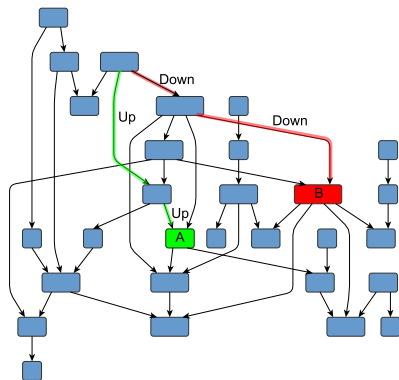
$p_1 = 0 \xrightarrow{a} 1 \xrightarrow{a} 2 \xrightarrow{a} 0 \xrightarrow{b} 3 \xrightarrow{b} 0 \xrightarrow{b} 3$

$p_2 = 0 \xrightarrow{a} 1 \xrightarrow{a} 2 \xrightarrow{a} 0 \xrightarrow{a} 1 \xrightarrow{a} 2 \xrightarrow{a} 0 \xrightarrow{b} 3 \xrightarrow{b} 0 \xrightarrow{b} 3 \xrightarrow{b} 0 \xrightarrow{b} 3 \xrightarrow{b} 0$

...

- Graph databases querying
Mihalis Yannakakis. “Graph-theoretic methods in database theory.” 1990.
- Static code analysis
Thomas Reps et al. “Precise interprocedural dataflow analysis via graph reachability.” 1995
- ...

Graph databases querying



Navigation through a graph

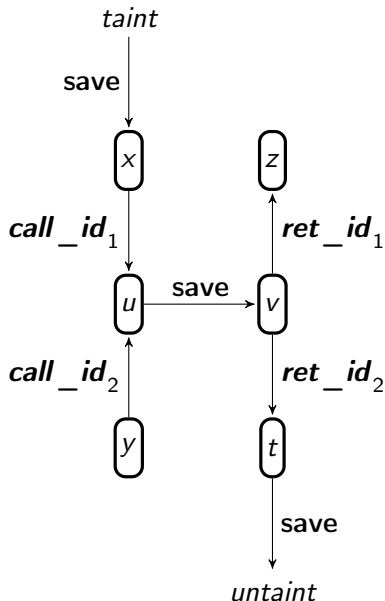
- Are nodes A and B on the same level of hierarchy?
- Is there a path of form $Up^n Down^n$?
- Find all paths of form $Up^n Down^n$ which start from the node A

Context-free path querying

- *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

```
int id(int u)
{
    v = u;
    return v;
}
int main()
{
    //taint
    int x;
    int z, y;
    //untaint
    int t;
    z = id(x);
    t = id(y);
}
```



Static code analysis (Language Reachability Framework)

- *Thomas Reps et al.* “Precise interprocedural dataflow analysis via graph reachability.” 1995
- *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

Static code analysis (Language Reachability Framework)

- *Thomas Reps et al.* “Precise interprocedural dataflow analysis via graph reachability.” 1995
- *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
- *Qirun Zhang and Zhendong Su.* “Context-sensitive data-dependence analysis via linear conjunctive language reachability.” 2017

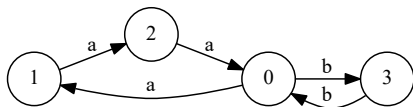
Parsing algorithms for CFPQ

- Structural representation of results
- Number of algorithms with different properties
- Number of theoretical results

Parsing algorithms for CFPQ

- Structural representation of results
- Number of algorithms with different properties
- Number of theoretical results
- Interconnection between different areas

Structural representation of result



Input graph

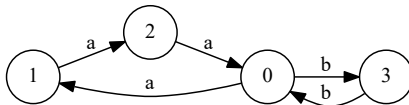
$S \rightarrow a S b$

$S \rightarrow \textit{Middle}$

$\textit{Middle} \rightarrow a b$

Grammar

Structural representation of result



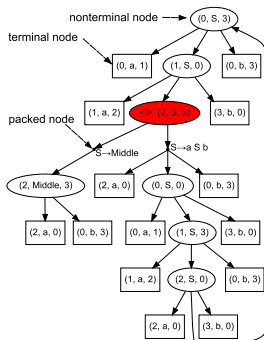
Input graph

$S \rightarrow a S b$

$S \rightarrow \text{Middle}$

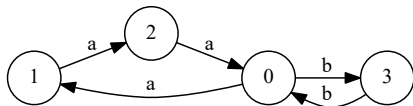
$\text{Middle} \rightarrow a b$

Grammar



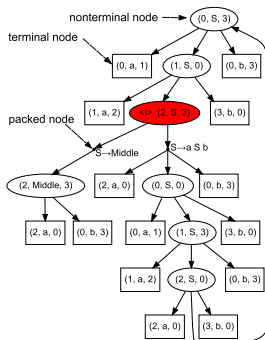
Query result (SPPF)

Structural representation of result

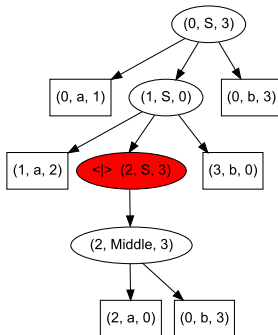


Input graph

$S \rightarrow a S b$
 $S \rightarrow \text{Middle}$
 $\text{Middle} \rightarrow a b$
 Grammar

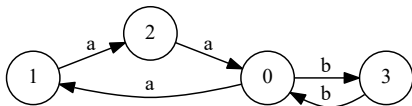


Query result (SPPF)



Tree for p_1

Structural representation of result



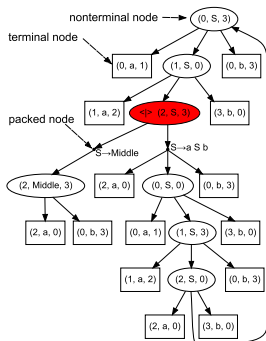
Input graph

$S \rightarrow a S b$

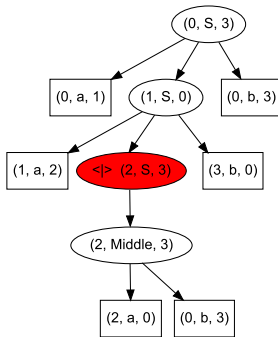
$S \rightarrow \text{Middle}$

$\text{Middle} \rightarrow a b$

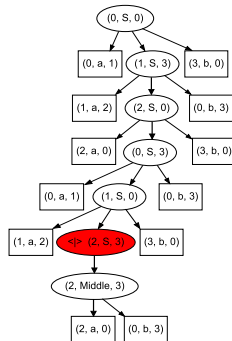
Grammar



Query result (SPPF)

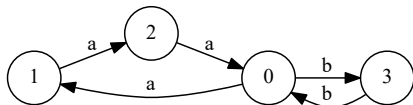


Tree for p_1



Tree for p_2

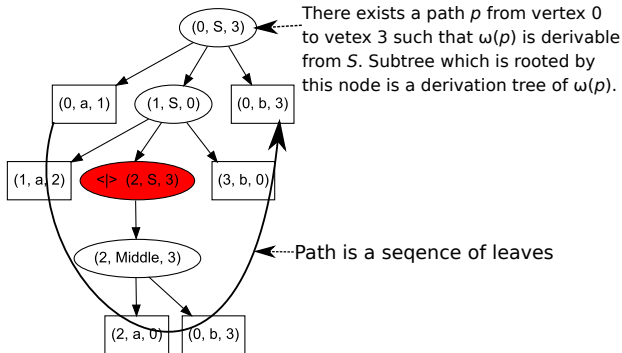
Paths extraction



$S \rightarrow a S b$

$S \rightarrow \text{Middle}$

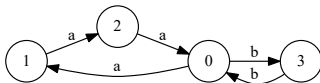
$\text{Middle} \rightarrow a b$



Path: $0 \xrightarrow{a} 1 \xrightarrow{a} 2 \xrightarrow{a} 0 \xrightarrow{b} 3 \xrightarrow{b} 0 \xrightarrow{b} 3$

Bar-Hillel theorem

Context-free languages are closed under intersection with regular languages



Regular language

$S \rightarrow a S b$

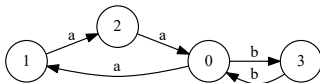
$S \rightarrow \textit{Middle}$

$\textit{Middle} \rightarrow a b$

Context-free language

Bar-Hillel theorem

Context-free languages are closed under intersection with regular languages



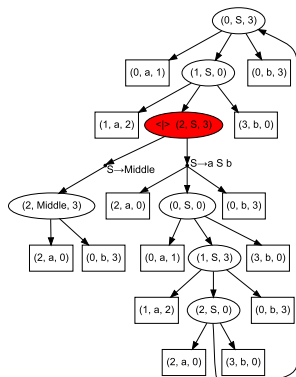
Regular language

$S \rightarrow a S b$

$S \rightarrow \text{Middle}$

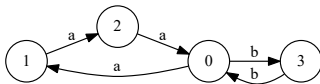
$\text{Middle} \rightarrow a b$

Context-free language



Bar-Hillel theorem

Context-free languages are closed under intersection with regular languages

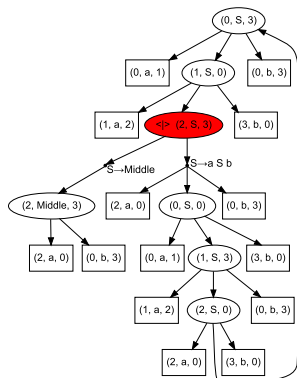


Regular language

$$S \rightarrow a S b$$
$$S \rightarrow Middle$$

Middle $\rightarrow a\ b$

Context-free language


$$(0, S, 3) \rightarrow (0, a, 1) (1, S, 0) (0, b, 3)$$
$$(1, S, 0) \rightarrow (1, a, 2) (2, S, 3) (3, b, 0)$$
$$(2, S, 3) \rightarrow (2, a, 0) (0, S, 0) (0, b, 3)$$
$$(2, S, 3) \rightarrow (2, \textit{Middle}, 3)$$
$$(0, S, 0) \rightarrow (0, a, 1) (1, S, 3) (3, b, 0)$$
$$(1, S, 3) \rightarrow (1, a, 2) (2, S, 0) (0, b, 3)$$
$$(2, S, 0) \rightarrow (2, a, 0) (0, S, 3) (3, b, 0)$$
$$(0, \textit{Middle}, 3) \rightarrow (2, a, 0) \quad (0, b, 3)$$

- Generalized LR for CFPQ
 - ▶ Based on Right Nulled Generalized LR: *Scott E., Johnstone A.* “Right Nulled GLR Parsers”
 - ▶ *Ekaterina Verbitskaia, Semyon Grigorev, and Dmitry Avdyukhin.* “Relaxed Parsing of Regular Approximations of String-Embedded Languages” 2015

Our experiments

- Generalized LR for CFPQ
 - ▶ Based on Right Nulled Generalized LR: *Scott E., Johnstone A.* “Right Nulled GLR Parsers”
 - ▶ *Ekaterina Verbitskaia, Semyon Grigorev, and Dmitry Avdyukhin.* “Relaxed Parsing of Regular Approximations of String-Embedded Languages” 2015
- Generalized LL for CFPQ (**GLL**)
 - ▶ Based on Generalized LL: *Scott E., Johnstone A.* “GLL parsing”
 - ▶ *Semyon Grigorev and Anastasiya Ragozina.* “Context-free path querying with structural representation of result.” 2017

How to integrate query language into a general-purpose programming language?

- Transparency
- Compositionality
- Static error checking

Query language integration

How to integrate query language into a general-purpose programming language?

- Transparency
- Compositionality
- Static error checking
- String-embedded languages
- ORMs
- Combinators

Combinators for CFPQ

- Implemented in Scala
- Based on Meerkat parser combinator library: *Anastasia Izmaylova, Ali Afroozeh, and Tijs van der Storm*. “Practical, general parser combinators” 2016
- *Ekaterina Verbitskaia, Ilya Kirillov, Ilya Nozkin, Semyon Grigorev*. “Parser Combinators for Context-Free Path Querying” 2018

Supported combinators

Combinator	Description
$a \sim b$	sequential parsing: a then b
$a \mid b$	choice: a or b

Supported combinators

Combinator	Description
$a \sim b$	sequential parsing: a then b
$a \mid b$	choice: a or b
$a ?$	optional parsing: a or nothing
$a *$	repetition of zero or more a
$a +$	repetition of at least one a

Supported combinators

Combinator	Description
$a \sim b$	sequential parsing: a then b
$a \mid b$	choice: a or b
$a ?$	optional parsing: a or nothing
$a *$	repetition of zero or more a
$a +$	repetition of at least one a
$a \wedge f$	apply function f to a if a is a token
$a \wedge \wedge$	capture output of a if a is a token
$a \& f$	apply function f to a if a is a parser
$a \& \&$	capture output of a if a is a parser

Supported combinators

Combinator	Description
$a \sim b$	sequential parsing: a then b
$a \mid b$	choice: a or b
$a ?$	optional parsing: a or nothing
$a *$	repetition of zero or more a
$a +$	repetition of at least one a
$a \wedge f$	apply function f to a if a is a token
$a \wedge \wedge$	capture output of a if a is a token
$a \& f$	apply function f to a if a is a parser
$a \& \&$	capture output of a if a is a parser

A set of functions to handle values of edges and vertices

```
def LV(labels: String*) =  
    V(e => labels.forall(e.hasLabel))  
def outLE(label: String) = outE(_.label() == label)  
def inLE (label: String) = inE ( _.label() == label)
```

Basic example

Is there a path from vertex 0 to vertex 3 which has form $a^n b^n$?

```
val Query : Nonterminal
    = syn (LV("0") ~ S ~ LV("3"))
```

```
val S: Nonterminal
    = syn (
        | "a" ~ S ~ "b"
        | "a" ~ "b"
    )
```


Example of generalization

```
def sameGen( brs ) =  
  reduceChoice(  
    brs.map { case ( lbr , rbr ) =>  
      lbr ~ syn( sameGen( brs ).? ) ~ rbr }
```

Example of generalization

```
def sameGen( brs ) =  
  reduceChoice(  
    brs.map { case ( lbr , rbr ) =>  
      lbr ~ syn( sameGen( brs ).? ) ~ rbr } )  
  
val query1 = syn( sameGen( List( ( "a" , "b" ) ) ) ) )  
  
val query2 = syn(  
  sameGen( List( ( p1 , p2 ) , ( "(" , ")" ) ) ) ~ p3 )
```

Example of values handling

Actors who played in some film

In Cypher

```
MATCH (m:Movie {title: 'Forrest_Gump'})  
      <-[:ACTS_IN]-(a:Actor)  
RETURN a.name, a.birthplace;
```

In Meerkat

```
val query =  
  syn((  
    (LV("Movie")::V(_.title == "Forrest_Gump")) ~  
    inLE("ACTS_IN") ~  
    syn(LV("Actor") ^  
      (e => (e.name, e.birthplace)))) &&  
  executeQuery(query, input)
```

- Overhead for the regular constraints
- Not exactly clear how to compute arbitrary semantics for the paths
 - ▶ Paths can be lazily extracted, but in which order?
 - ▶ What kind of semantics can be calculated when there are cycles?

Boolean Matrix Multiplication for CFPQ

- *Rustam Azimov, Semyon Grigorev.* “Context-free path querying by matrix multiplication.” 2017
- *Semyon Grigorev, et. al.* “Evaluation of the Context-Free Path Querying Algorithm Based on Matrix Multiplication” 2019

Transitive closure

- Subset multiplication, $N_1, N_2 \subseteq N$
 - ▶ $N_1 \cdot N_2 = \{A \mid \exists B \in N_1, \exists C \in N_2 \text{ such that } (A \rightarrow BC) \in R\}$
- 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
 - ▶ $a^{cf} = a^{(1)} \cup a^{(2)} \cup \dots$
 - ▶ $a^{(1)} = a$
 - ▶ $a^{(i)} = a^{(i-1)} \cup (a^{(i-1)} \times a^{(i-1)}), \quad i \geq 2$

The algorithm

Algorithm Context-free recognizer for graphs

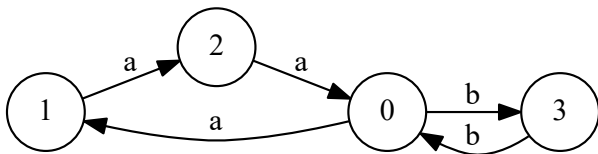
```
1: function CONTEXTFREEPATHQUERYING( $D, G$ )
2:    $n \leftarrow$  the number of nodes in  $D$ 
3:    $E \leftarrow$  the directed edge-relation from  $D$ 
4:    $R \leftarrow$  the set of production rules in  $G$ 
5:    $T \leftarrow$  the matrix  $n \times n$  in which each element is  $\emptyset$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

BMM-based algorithm: 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

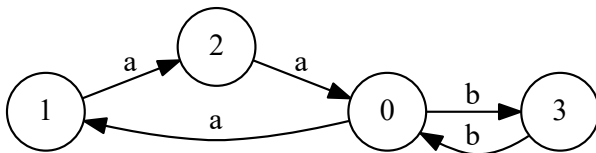
$$\begin{array}{ccc} S & \rightarrow & a S b \\ | & & a b \end{array}$$



The worst case: step by step

$$\begin{array}{lcl} S & \rightarrow & A B \\ & | & A S_1 \\ S_1 & \rightarrow & S B \\ A & \rightarrow & a \\ B & \rightarrow & b \end{array}$$

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



The worst case: step by step

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

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

The worst case: step by step

$$T_2 = \begin{pmatrix} \emptyset & \{A\} & \emptyset & \{B\} \\ \emptyset & \emptyset & \{A\} & \emptyset \\ \{A, \mathbf{S_1}\} & \emptyset & \emptyset & \{S\} \\ \{B\} & \emptyset & \emptyset & \emptyset \end{pmatrix}$$

The worst case: step by step

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

The worst case: step by step

$$T_4 = \begin{pmatrix} \emptyset & \{A\} & \emptyset & \{B\} \\ \{S\} & \emptyset & \{A\} & \{\mathbf{S_1}\} \\ \{A, S_1\} & \emptyset & \emptyset & \{S\} \\ \{B\} & \emptyset & \emptyset & \emptyset \end{pmatrix}$$

The worst case: step by step

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

The worst case: step by step

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

The worst case: step by step

$$T_7 = \begin{pmatrix} \{S_1\} & \{A\} & \emptyset & \{B, S\} \\ \{S\} & \emptyset & \{A\} & \{S_1\} \\ \{A, S_1, \mathbf{S}\} & \emptyset & \emptyset & \{S\} \\ \{B\} & \emptyset & \emptyset & \emptyset \end{pmatrix}$$

The worst case: step by step

$$T_8 = \begin{pmatrix} \{S_1\} & \{A\} & \emptyset & \{B, S\} \\ \{S\} & \emptyset & \{A\} & \{S_1\} \\ \{A, S_1, S\} & \emptyset & \emptyset & \{S, \mathbf{S_1}\} \\ \{B\} & \emptyset & \emptyset & \emptyset \end{pmatrix}$$

The worst case: step by step

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

The worst case: step by step

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

The worst case: step by step

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

The worst case: step by step

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

The worst case: step by step

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

Boolean Matrix Multiplication for CFPQ

- A matrix for a nonterminal is a set of boolean matrices
- Matrix multiplication can be implemented efficiently by using modern hardware and high-performance libraries

Performance comparison setup

We use graphs from the classical set of ontologies: *skos*, *foaf*, *univ-bench*, *wine*, *pizza*, etc.

Queries are classical variants of the same-generation query

$S \rightarrow \text{subClassOf}^{-1} S \text{ subClassOf}$	$S \rightarrow B \text{ subClassOf}$
$S \rightarrow \text{type}^{-1} S \text{ type}$	$S \rightarrow \text{subClassOf}$
$S \rightarrow \text{subClassOf}^{-1} \text{subClassOf}$	$B \rightarrow \text{subClassOf}^{-1} B \text{ subClassOf}$
$S \rightarrow \text{type}^{-1} \text{type}$	$B \rightarrow \text{subClassOf}^{-1} \text{subClassOf}$

Query 1

Query 2

Performance comparison results

№	#V	#E	Query 1 (ms)			Query 2 (ms)	
			CYK ¹	GLL	GPGPU	GLL	GPGPU
1	144	323	1044	10	12	1	1
2	129	351	6091	19	13	1	0
3	131	397	13971	24	30	1	10
4	179	413	20981	25	15	11	9
5	337	834	82081	89	32	3	6
6	291	685	515285	255	22	66	2
7	341	711	420604	261	20	45	24
8	671	2604	3233587	697	24	29	23
9	733	2450	4075319	819	54	8	6
10	6224	11840	–	1926	82	167	38
11	5864	19600	–	6246	185	46	21
12	5368	20832	–	7014	127	393	40

¹Zhang, et al. "Context-free path queries on RDF graphs."

Performance comparison results

- Data from *Zhiwei Fan, et.al.* “Scaling-Up In-Memory Datalog Processing: Observations and Techniques.” 2018.
- Graphs with names of form $Gn - p$: n is a number of vertices, edge between two vertices exists with probability p

Graph	M4RI(sec)	GPU_N(sec)	GPGPU(sec)
G10k-0.01	1.455	0.138	47.525
G10k-0.1	1.050	0.114	395.393
G20k-0.001	11.025	1.274	-
G40k-0.001	97.841	8.393	-
G80k-0.001	1142.959	65.886	-

Directions for research: engineering part

- Develop parallel and distributed algorithms
 - ▶ Distributed GLL(GLR)-based algorithms
 - ▶ Distributed matrix multiplications algorithms
 - ▶ Efficient implementation of sparse boolean matrices multiplication algorithms
- Adopt other parsing algorithms
 - ▶ Brzozowski's derivatives
 - ▶ Derivatives for graph querying: *Maurizio Nole and Carlo Sartiani*. "Regular path queries on massive graphs." 2016
 - ▶ Derivatives for context-free parsing: *Matthew Might, David Darais, and Daniel Spiewak*. "Parsing with derivatives: a functional pearl." 2011.
- Utilize other classes of languages for constraints specification
- Investigate incremental queries evaluation

Directions for research: theoretical part

- Time complexity of GLL(GLR)-based algorithms for different classes of grammars
 - ▶ Current result for GLL-based: $O\left(|V|^3 * \max_{v \in V}(\deg^+(v))\right)$
- Theoretical lower bound for CFPQ
 - ▶ Is it possible to reduce CFPQ to $\tilde{O}(\text{BMM})$?
 - ★ Our result is $O(|V|^2|N|^3(\text{BMM}(|V|) + \text{BMU}(|V|)))$
 - ▶ $\tilde{O}(n^\omega)$ solution for Dyck with one type of brackets: *Phillip G. Bradford*. "Efficient Exact Paths For Dyck and semi-Dyck Labeled Path Reachability." 2018.