

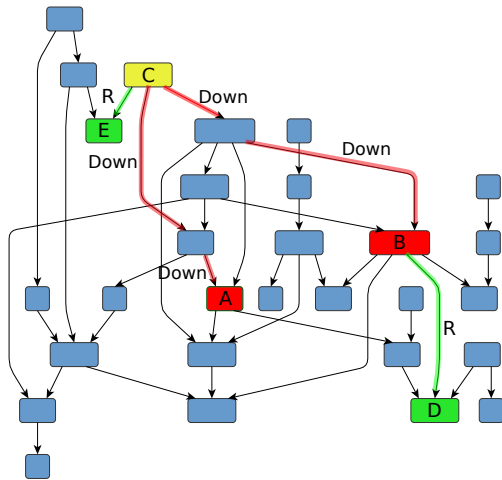
# Multiple-Source Context-Free Path Querying in Terms of Linear Algebra

Arseniy Terekhov, Vlada Pogozhelskaya, Vadim Abzalov,  
Timur Zinnatulin, **Semyon Grigorev**

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

March 24, 2021

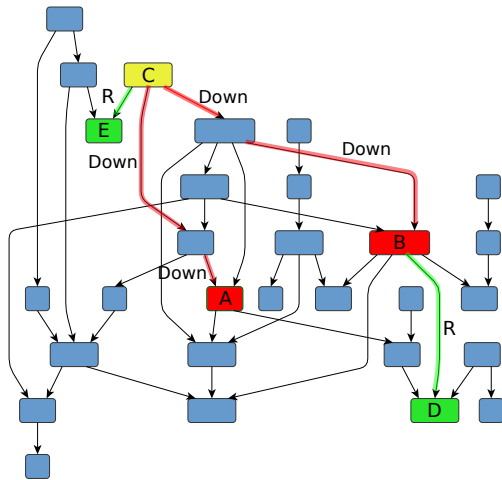
# Formal Language Constrained Path Querying



Navigation through an edge-labeled graph

- **Path** specifies a **word** formed by the labels of the edges
- **Paths constraint** is a **language**: the word specified by the path should be in the given language
- The expressiveness of constraints is related to **formal languages classes**

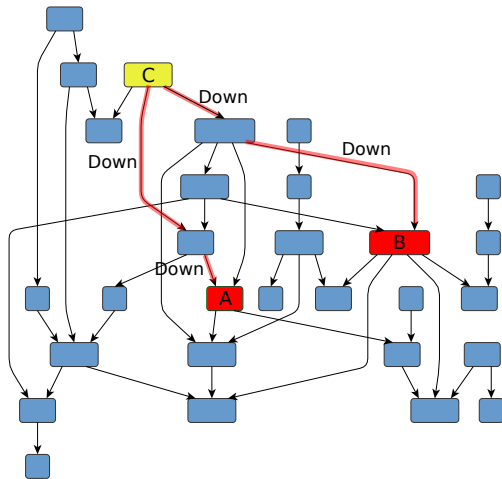
# Regular Path Queries (RPQ)



## Regular languages as constraints

- Which nodes are reachable from **C** by arbitrary number of **R** and **Down** edges?
- Regular language  $\mathcal{L} = (R \mid \text{Down})^*$

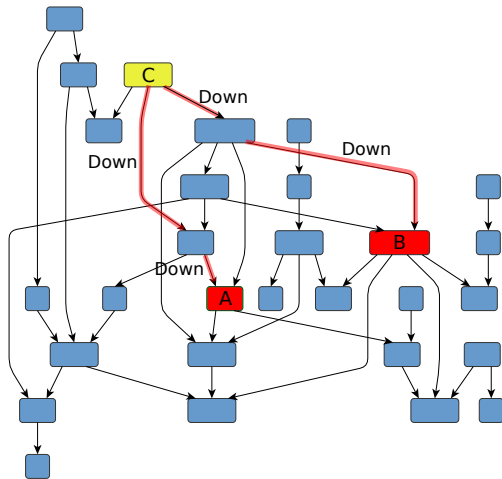
# Context-Free Path Queries (CFPQ)



## Context-free languages as constraints

- Are nodes A and B on the same level of hierarchy?
- Is there a path of form  $\overline{\text{Down}}^n \text{Down}^n$  between A and B?
- Context-free grammar:  
 $\text{sameLvl} \rightarrow \overline{\text{Down}} \text{sameLvl} \text{Down} \mid \varepsilon$

# Context-Free Path Queries (CFPQ)



## Context-free languages as constraints

- Are nodes A and B on the same level of hierarchy?
- Is there a path of form  $\overline{\text{Down}}^n \text{Down}^n$  between A and B?
- Context-free grammar:  
 $\text{sameLvl} \rightarrow \overline{\text{Down}} \text{sameLvl} \text{Down} \mid \varepsilon$

## Applications

- Static code analysis [T. Reps, et al, 1995]
- Graph segmentation [H. Miao, et al, 2019]
- Biological data analysis [P. Sevon, et al, 2008] ...

# Problem Statement

There is no support of CFPQ in real-world graph analysis systems (graph databases)

# Problem Statement

There is no support of CFPQ in real-world graph analysis systems (graph databases)

- J. Kuijpers, et al<sup>1</sup>: existing algorithms are too slow to be used in practical applications (in the context of Neo4j)

---

<sup>1</sup>Jochem Kuijpers, George Fletcher, Nikolay Yakovets, and Tobias Lindaaker. 2019. An Experimental Study of Context-Free Path Query Evaluation Methods.

# Problem Statement

There is no support of CFPQ in real-world graph analysis systems (graph databases)

- J. Kuijpers, et al<sup>1</sup>: existing algorithms are too slow to be used in practical applications (in the context of Neo4j)
- A. Terekhov, et al<sup>2</sup>: linear algebra based CFPQ algorithm can be performant enough

---

<sup>1</sup>Jochem Kuijpers, George Fletcher, Nikolay Yakovets, and Tobias Lindaaker. 2019. An Experimental Study of Context-Free Path Query Evaluation Methods.

<sup>2</sup>Arseniy Terekhov, Artyom Khoroshev, Rustam Azimov, and Semyon Grigorev. 2020. Context-Free Path Querying with Single-Path Semantics by Matrix Multiplication.



# Problem Statement

There is no support of CFPQ in real-world graph analysis systems (graph databases)

- J. Kuijpers, et al<sup>1</sup>: existing algorithms are too slow to be used in practical applications (in the context of Neo4j)
- A. Terekhov, et al<sup>2</sup>: linear algebra based CFPQ algorithm can be performant enough
- There is no full-stack support of CFPQ
  - ▶ Grammars instead of full-featured queries
  - ▶ Custom graph storage instead of a mature graph database

---

<sup>1</sup>Jochem Kuijpers, George Fletcher, Nikolay Yakovets, and Tobias Lindaaker. 2019. An Experimental Study of Context-Free Path Query Evaluation Methods.

<sup>2</sup>Arseniy Terekhov, Artyom Khoroshev, Rustam Azimov, and Semyon Grigorev. 2020. Context-Free Path Querying with Single-Path Semantics by Matrix Multiplication.

# Proposed Solution

- **Multiple-Source CFPQ** to process only the relevant subgraph

# Proposed Solution

- **Multiple-Source CFPQ** to process only the relevant subgraph
- **Cypher** extended with **path patterns**<sup>3</sup> to express context-free constraints

---

<sup>3</sup>Tobias Lindaaker, Path Patterns for Cypher, 2017,  
<https://github.com/thobe/openCypher/blob/rpq/cip/1.accepted/CIP2017-02-06-Path-Patterns.adoc>

# Proposed Solution

- **Multiple-Source CFPQ** to process only the relevant subgraph
- **Cypher** extended with **path patterns**<sup>3</sup> to express context-free constraints
- **RedisGraph** database
  - ▶ Graph storage with matrix-based representation
  - ▶ Linear algebra based query engine (SuiteSparse:GraphBLAS<sup>4</sup> is used)
  - ▶ Cypher for querying (libcypher-parser<sup>5</sup> is used)

---

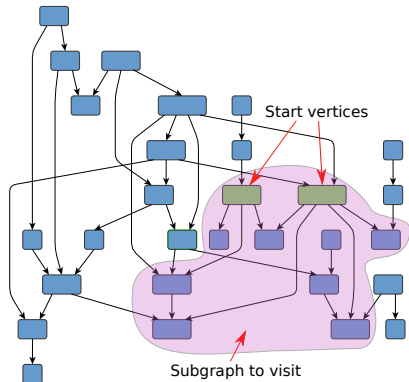
<sup>3</sup>Tobias Lindaaker, Path Patterns for Cypher, 2017,  
<https://github.com/thobe/openCypher/blob/rpq/cip/1.accepted/CIP2017-02-06-Path-Patterns.adoc>

<sup>4</sup>Timothy A. Davis. 2019. Algorithm 1000: SuiteSparse:GraphBLAS: Graph Algorithms in the Language of Sparse Linear Algebra

<sup>5</sup>Chris Leishman, <https://github.com/cleishm/libcypher-parser>

# Multiple-Source CFPQ

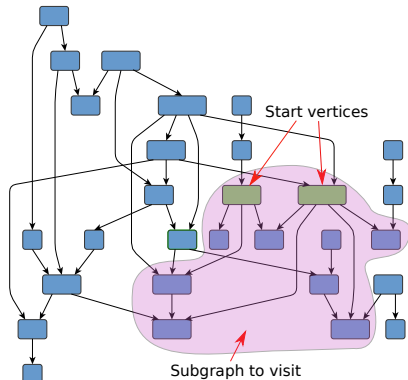
An improved version of Rustam Azimov CFPQ algorithm<sup>6</sup>



<sup>6</sup>Rustam Azimov and Semyon Grigorev. 2018. Context-Free Path Querying by Matrix Multiplication.

# Multiple-Source CFPQ

An improved version of Rustam Azimov CFPQ algorithm<sup>6</sup>

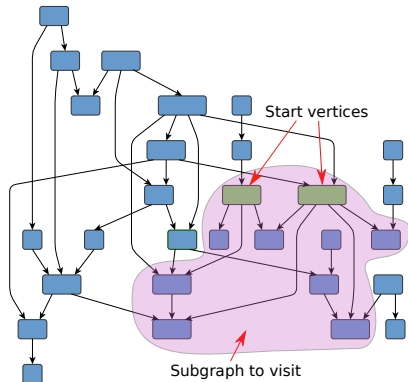


- The set of start vertices can be specified

<sup>6</sup>Rustam Azimov and Semyon Grigorev. 2018. Context-Free Path Querying by Matrix Multiplication.

# Multiple-Source CFPQ

An improved version of Rustam Azimov CFPQ algorithm<sup>6</sup>



- The set of start vertices can be specified
- Only the relevant subgraph will be processed

<sup>6</sup>Rustam Azimov and Semyon Grigorev. 2018. Context-Free Path Querying by Matrix Multiplication.

# Multiple-Source CFPQ

```
1: function MULTISRCCFPQ( $D = (V, E, \Sigma_V, \Sigma_E, \lambda_V, \lambda_E)$ ,  $G = (N, \Sigma, P, S)$ ,  $Src$ )
2:    $T \leftarrow \{T^A \mid A \in N, T^A[i, j] \leftarrow \text{false}, \text{ for all } i, j\}$ 
3:    $TSrc \leftarrow \{TSrc^A \mid A \in N, TSrc^A[i, j] \leftarrow \text{false}, \text{ for all } i, j\}$ 
4:   for all  $v \in Src$  do  $TSrc^S[v, v] \leftarrow \text{true}$ 
5:    $MSrc \leftarrow TSrc^S$ 
6:   for all  $A \rightarrow x \in P \mid x \in \Sigma_E$  do
7:     for all  $(v, to) \in E \mid x \in \lambda_E(v, to)$  do  $T^A[v, to] \leftarrow \text{true}$ 
8:   for all  $A \rightarrow x \in P \mid x \in \Sigma_V$  do
9:     for all  $v \in V \mid x \in \lambda_V(v)$  do  $T^A[v, v] \leftarrow \text{true}$ 
10:  while  $T$  or  $TSrc$  is changing do
11:    for all  $A \rightarrow BC \in P$  do
12:       $M \leftarrow TSrc^A * T^B$ 
13:       $T^A \leftarrow T^A + M * T^C$ 
14:       $TSrc^B \leftarrow TSrc^B + TSrc^A$ 
15:       $TSrc^C \leftarrow TSrc^C + \text{GETDST}(M)$ 
16:  return  $MSrc * T^S$ 
```



# Multiple-Source CFPQ

```
1: function MULTISRCCFPQ( $D = (V, E, \Sigma_V, \Sigma_E, \lambda_V, \lambda_E)$ ,  $G = (N, \Sigma, P, S)$ ,  $Src$ )
2:    $T \leftarrow \{T^A \mid A \in N, T^A[i, j] \leftarrow \text{false, for all } i, j\}$ 
3:    $TSrc \leftarrow \{TSrc^A \mid A \in N, TSrc^A[i, j] \leftarrow \text{false, for all } i, j\}$ 
4:   for all  $v \in Src$  do  $TSrc^S[v, v] \leftarrow \text{true}$ 
5:    $MSrc \leftarrow TSrc^S$ 
6:   for all  $A \rightarrow x \in P \mid x \in \Sigma_E$  do
7:     for all  $(v, to) \in E \mid x \in \lambda_E(v, to)$  do  $T^A[v, to] \leftarrow \text{true}$ 
8:   for all  $A \rightarrow x \in P \mid x \in \Sigma_V$  do
9:     for all  $v \in V \mid x \in \lambda_V(v)$  do  $T^A[v, v] \leftarrow \text{true}$ 
10:  while  $T$  or  $TSrc$  is changing do
11:    for all  $A \rightarrow BC \in P$  do
12:       $M \leftarrow TSrc^A * T^B$ 
13:       $T^A \leftarrow T^A + M * T^C$ 
14:       $TSrc^B \leftarrow TSrc^B + TSrc^A$ 
15:       $TSrc^C \leftarrow TSrc^C + \text{GETDST}(M)$ 
16:  return  $MSrc * T^S$ 
```

# Multiple-Source CFPQ

```
1: function MULTISRCCFPQ( $D = (V, E, \Sigma_V, \Sigma_E, \lambda_V, \lambda_E)$ ,  $G = (N, \Sigma, P, S)$ ,  $Src$ )
2:    $T \leftarrow \{T^A \mid A \in N, T^A[i, j] \leftarrow \text{false, for all } i, j\}$ 
3:    $TSrc \leftarrow \{TSrc^A \mid A \in N, TSrc^A[i, j] \leftarrow \text{false, for all } i, j\}$ 
4:   for all  $v \in Src$  do  $TSrc^S[v, v] \leftarrow \text{true}$ 
5:    $MSrc \leftarrow TSrc^S$ 
6:   for all  $A \rightarrow x \in P \mid x \in \Sigma_E$  do
7:     for all  $(v, to) \in E \mid x \in \lambda_E(v, to)$  do  $T^A[v, to] \leftarrow \text{true}$ 
8:   for all  $A \rightarrow x \in P \mid x \in \Sigma_V$  do
9:     for all  $v \in V \mid x \in \lambda_V(v)$  do  $T^A[v, v] \leftarrow \text{true}$ 
10:  while  $T$  or  $TSrc$  is changing do
11:    for all  $A \rightarrow BC \in P$  do
12:       $M \leftarrow TSrc^A * T^B$ 
13:       $T^A \leftarrow T^A + M * T^C$ 
14:       $TSrc^B \leftarrow TSrc^B + TSrc^A$ 
15:       $TSrc^C \leftarrow TSrc^C + \text{GETDST}(M)$ 
16:  return  $MSrc * T^S$ 
```

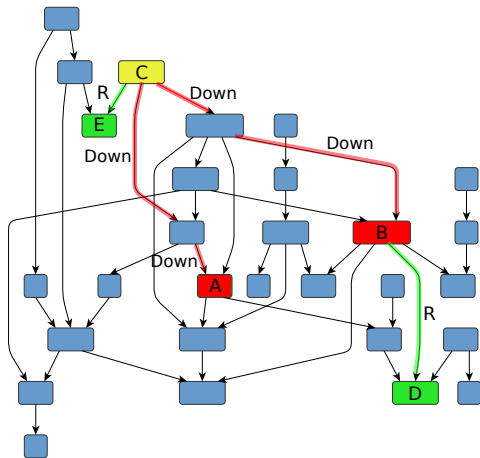
# Multiple-Source CFPQ

```
1: function MULTISRCCFPQ( $D = (V, E, \Sigma_V, \Sigma_E, \lambda_V, \lambda_E)$ ,  $G = (N, \Sigma, P, S)$ ,  $Src$ )
2:    $T \leftarrow \{T^A \mid A \in N, T^A[i, j] \leftarrow \text{false, for all } i, j\}$ 
3:    $TSrc \leftarrow \{TSrc^A \mid A \in N, TSrc^A[i, j] \leftarrow \text{false, for all } i, j\}$ 
4:   for all  $v \in Src$  do  $TSrc^S[v, v] \leftarrow \text{true}$ 
5:    $MSrc \leftarrow TSrc^S$ 
6:   for all  $A \rightarrow x \in P \mid x \in \Sigma_E$  do
7:     for all  $(v, to) \in E \mid x \in \lambda_E(v, to)$  do  $T^A[v, to] \leftarrow \text{true}$ 
8:   for all  $A \rightarrow x \in P \mid x \in \Sigma_V$  do
9:     for all  $v \in V \mid x \in \lambda_V(v)$  do  $T^A[v, v] \leftarrow \text{true}$ 
10:  while  $T$  or  $TSrc$  is changing do
11:    for all  $A \rightarrow BC \in P$  do
12:       $M \leftarrow TSrc^A * T^B$ 
13:       $T^A \leftarrow T^A + M * T^C$ 
14:       $TSrc^B \leftarrow TSrc^B + TSrc^A$ 
15:       $TSrc^C \leftarrow TSrc^C + \text{GETDST}(M)$ 
16:  return  $MSrc * T^S$ 
```

# Multiple-Source CFPQ

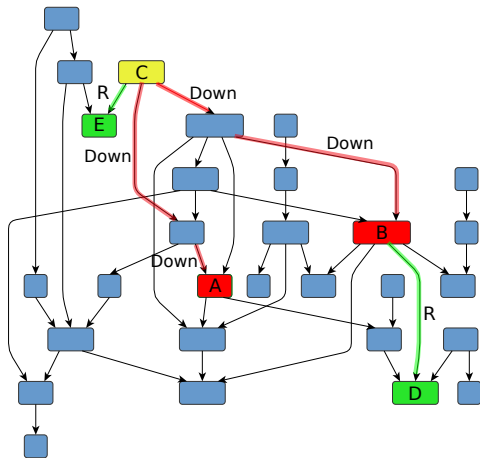
```
1: function MULTISRCCFPQ( $D = (V, E, \Sigma_V, \Sigma_E, \lambda_V, \lambda_E)$ ,  $G = (N, \Sigma, P, S)$ ,  $Src$ )
2:    $T \leftarrow \{T^A \mid A \in N, T^A[i, j] \leftarrow \text{false, for all } i, j\}$ 
3:    $TSrc \leftarrow \{TSrc^A \mid A \in N, TSrc^A[i, j] \leftarrow \text{false, for all } i, j\}$ 
4:   for all  $v \in Src$  do  $TSrc^S[v, v] \leftarrow \text{true}$ 
5:    $MSrc \leftarrow TSrc^S$ 
6:   for all  $A \rightarrow x \in P \mid x \in \Sigma_E$  do
7:     for all  $(v, to) \in E \mid x \in \lambda_E(v, to)$  do  $T^A[v, to] \leftarrow \text{true}$ 
8:   for all  $A \rightarrow x \in P \mid x \in \Sigma_V$  do
9:     for all  $v \in V \mid x \in \lambda_V(v)$  do  $T^A[v, v] \leftarrow \text{true}$ 
10:  while  $T$  or  $TSrc$  is changing do
11:    for all  $A \rightarrow BC \in P$  do
12:       $M \leftarrow TSrc^A * T^B$ 
13:       $T^A \leftarrow T^A + M * T^C$ 
14:       $TSrc^B \leftarrow TSrc^B + TSrc^A$ 
15:       $TSrc^C \leftarrow TSrc^C + \text{GETDST}(M)$ 
16:  return  $MSrc * T^S$ 
```

# Cypher Extension



```
MATCH (u)-/[:R | :Down]*-/-(v)  
RETURN u.name, v.name
```

# Cypher Extension



```
MATCH (u)-/[:R | :Down]*-/>(v)
```

```
RETURN u.name, v.name
```

Named path pattern

$\text{SameLvl} \rightarrow \overline{\text{Down}} \text{ SameLvl} \text{ Down} \mid \varepsilon$

```
PATH PATTERN SameLvl =
```

```
( )-/[:Down ~SameLvl :Down] | ( )/->( )
```

```
MATCH (u)-/ ~SameLvl /->(v)
```

```
RETURN u.name, v.name
```

# Implementation Details

- Linear algebra based multiple-source Context-Free Path Querying is implemented as a part of RedisGraph query engine
- Cypher parser is extended to support path patterns
- Path patterns are partially supported<sup>7</sup> in RedisGreaph query execution workflow

---

<sup>7</sup>Full support is a nontrivial challenge: formal description of the extension is required

## Evaluation Setup

- Ubuntu 18.04, Intel Core i7-6700 CPU, 3.4GHz, DDR4 64Gb RAM
- Graphs are stored in RedisGraph augmented with our extensions
- Queries are generated with template for the given size of the start set
- The union of all start sets is denoted  $V$



## Evaluation Setup

- Ubuntu 18.04, Intel Core i7-6700 CPU, 3.4GHz, DDR4 64Gb RAM
- Graphs are stored in RedisGraph augmented with our extensions
- Queries are generated with template for the given size of the start set
- The union of all start sets is denoted  $V$

Graph	#V	#E	Q
core	1323	4342	$g_1$
pathways	6238	18 598	$g_1$
gohierarchy	45 007	980 218	$g_1$
enzyme	48 815	109 695	$g_1$
eclass_514en	239 111	523 727	$g_1$
geospecies	450 609	2 311 461	$geo$
go	272 770	534 311	$g_1$

## Evaluation Setup

- Ubuntu 18.04, Intel Core i7-6700 CPU, 3.4GHz, DDR4 64Gb RAM
- Graphs are stored in RedisGraph augmented with our extensions
- Queries are generated with template for the given size of the start set
- The union of all start sets is denoted  $V$

Graph	#V	#E	Q
core	1323	4342	$g_1$
pathways	6238	18 598	$g_1$
gohierarchy	45 007	980 218	$g_1$
enzyme	48 815	109 695	$g_1$
eclass_514en	239 111	523 727	$g_1$
geospecies	450 609	2 311 461	$geo$
go	272 770	534 311	$g_1$

PATH PATTERN S =

`()-/[<:SubClassOf [~S | ()] :SubClassOf] | [<:Type [~S | ()] :Type] /->()`

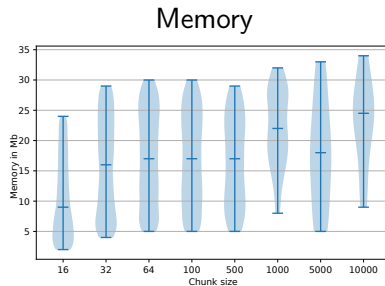
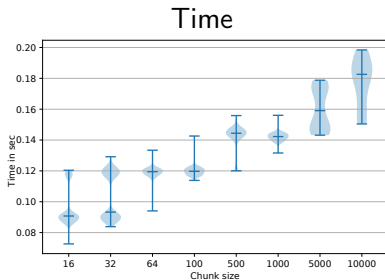
**MATCH** (src)-/ ~S /->()

**WHERE** {id\_from} <= src.id and src.id <= {id\_to}

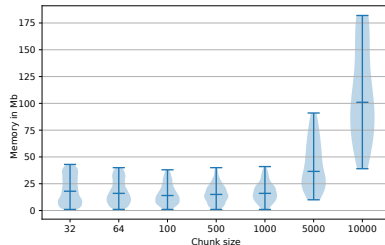
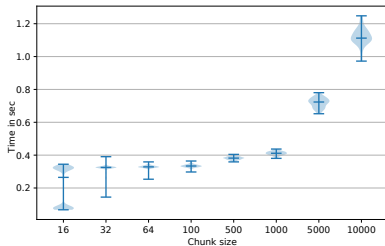
**RETURN** count(\*)

# Evaluation Results

eclass\_514en  
Query: g1



geospecies  
Query: geo



- Full-stack support for CFPQ in real-world applications which use RedisGraph database with Cypher query language
  - ▶ No more context-free grammars
  - ▶ No more custom graph formats and storages
- Reasonable performance of context-free path queries
  - ▶ Multiple-source scenario
  - ▶ Space-time ratio can be tuned
- Context-free path queries can be used in applications with well-established tools

- Mechanization of Cypher semantics in Coq
  - ▶ Semantics which includes path patterns
  - ▶ Goal: prove correctness of translation to linear algebra

- Mechanization of Cypher semantics in Coq
  - ▶ Semantics which includes path patterns
  - ▶ Goal: prove correctness of translation to linear algebra
- Integration of tensor-based CFPQ algorithm<sup>8</sup> to RedisGraph
  - ▶ The algorithm constructs paths, not only reachability facts
  - ▶ The algorithm should be modified to get multiple-source version

---

<sup>8</sup>Egor Orachev, Ilya Epelbaum, R. Azimov and S. Grigorev. 2020. Context-Free Path Querying by Kronecker Product.

- Mechanization of Cypher semantics in Coq
  - ▶ Semantics which includes path patterns
  - ▶ Goal: prove correctness of translation to linear algebra
- Integration of tensor-based CFPQ algorithm<sup>8</sup> to RedisGraph
  - ▶ The algorithm constructs paths, not only reachability facts
  - ▶ The algorithm should be modified to get multiple-source version
- Detailed evaluation
  - ▶ Include more graphs and queries, including RPQs
  - ▶ Evaluate the scalability of the solution
  - ▶ Compare with other graph query engines

---

<sup>8</sup>Egor Orachev, Ilya Epelbaum, R. Azimov and S. Grigorev. 2020. Context-Free Path Querying by Kronecker Product.

## Contact Information

- Try it out (Docker image with extended RedisGraph):  
<https://hub.docker.com/r/simpletondl/redisgraph>
- RedisGraph extended with CFPQ:  
<https://github.com/YaccConstructor/RedisGraph>
- Cypher parser extended with path patterns:  
<https://github.com/YaccConstructor/libcypher-parser>



## Contact Information

- Try it out (Docker image with extended RedisGraph):  
<https://hub.docker.com/r/simpletondl/redisgraph>
- RedisGraph extended with CFPQ:  
<https://github.com/YaccConstructor/RedisGraph>
- Cypher parser extended with path patterns:  
<https://github.com/YaccConstructor/libcypher-parser>
  
- Semyon Grigorev: [s.v.grigoriev@spbu.ru](mailto:s.v.grigoriev@spbu.ru)
- Arseniy Terekhov: [simpletondl@yandex.ru](mailto:simpletondl@yandex.ru)
- Vlada Pogozhelskaya: [pogozhelskaya@gmail.com](mailto:pogozhelskaya@gmail.com)
- Vadim Abzalov: [vadim.i.abzalov@gmail.com](mailto:vadim.i.abzalov@gmail.com)
- Timur Zinnatulin: [teemychteemych@gmail.com](mailto:teemychteemych@gmail.com)

## Contact Information

- Try it out (Docker image with extended RedisGraph):  
<https://hub.docker.com/r/simpletondl/redisgraph>
- RedisGraph extended with CFPQ:  
<https://github.com/YaccConstructor/RedisGraph>
- Cypher parser extended with path patterns:  
<https://github.com/YaccConstructor/libcypher-parser>

Thanks!

- Semyon Grigorev: [s.v.grigoriev@spbu.ru](mailto:s.v.grigoriev@spbu.ru)
- Arseniy Terekhov: [simpletondl@yandex.ru](mailto:simpletondl@yandex.ru)
- Vlada Pogozhelskaya: [pogozhelskaya@gmail.com](mailto:pogozhelskaya@gmail.com)
- Vadim Abzalov: [vadim.i.abzalov@gmail.com](mailto:vadim.i.abzalov@gmail.com)
- Timur Zinnatulin: [teemychteemych@gmail.com](mailto:teemychteemych@gmail.com)