

# Context-Free Path Querying: Obstacles on the Way to Adoption

Semyon Grigorev

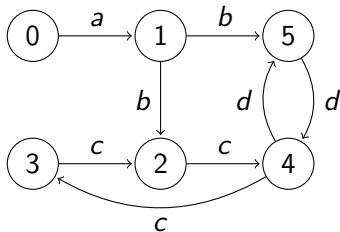
JetBrains Research, Programming Languages and Tools Lab  
St. Petersburg State University

[https://research.jetbrains.org/groups/plt\\_lab/](https://research.jetbrains.org/groups/plt_lab/)

16.07.2021

# Formal language constrained path querying

Navigational queries in edge-labelled graph

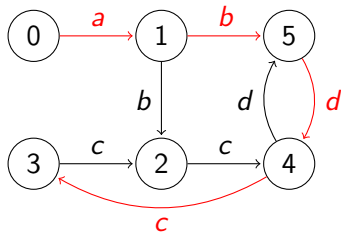


- Path to find:

$$0 \xrightarrow{a} v_0 \xrightarrow{b} v_1 \underbrace{\xrightarrow{d} v_2 \xrightarrow{c} v_3 \dots v_k}_{c \text{ or } d \text{ in arbitrary order}} \xrightarrow{c} v$$

# Formal language constrained path querying

Navigational queries in edge-labelled graph

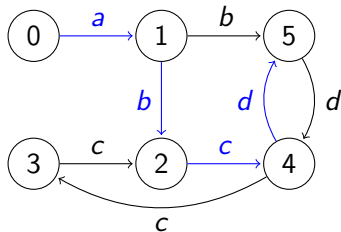


- Path to find:

$$0 \xrightarrow{a} v_0 \xrightarrow{b} v_1 \underbrace{\xrightarrow{d} v_2 \xrightarrow{c} v_3 \dots v_k}_{c \text{ or } d \text{ in arbitrary order}} \xrightarrow{c} v$$

# Formal language constrained path querying

Navigational queries in edge-labelled graph

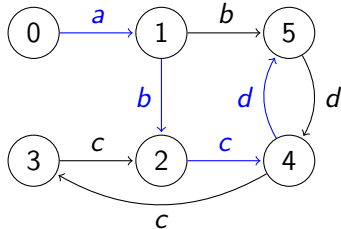


- Path to find:

$$0 \xrightarrow{a} v_0 \xrightarrow{b} v_1 \underbrace{\xrightarrow{d} v_2 \xrightarrow{c} v_3 \dots v_k}_{c \text{ or } d \text{ in arbitrary order}} \xrightarrow{c} v$$

# Formal language constrained path querying

Navigational queries in edge-labelled graph



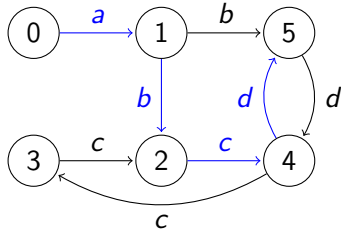
- $w(v_0 \xrightarrow{l_0} v_1 \xrightarrow{l_1} \dots \xrightarrow{l_{k-1}} v_k) = l_0 l_1 \dots l_{k-1}$
- $Q = \{(v_i, v_j) \mid \exists \pi = v_i \rightarrow \dots \rightarrow v_j; w(\pi) \in \mathcal{L}\}$ ,  
where  $\mathcal{L}$  — formal language
  - ✓ Regular, RPQ  $(ab(c \mid d)^*)$
  - ⚙ **Context-Free**, CFPQ  $(a^n b^n)$
  - 🕒 Multiple Context-Free  $(a^n c^m b^n d^m)$

- Path to find:

$$0 \xrightarrow{a} v_0 \xrightarrow{b} v_1 \underbrace{\xrightarrow{d} v_2 \xrightarrow{c} v_3 \dots v_k}_{c \text{ or } d \text{ in arbitrary order}} \xrightarrow{c} v$$

# Formal language constrained path querying

Navigational queries in edge-labelled graph



- Path to find:

$$0 \xrightarrow{a} v_0 \xrightarrow{b} v_1 \xrightarrow{d} v_2 \xrightarrow{c} v_3 \dots v_k \xrightarrow{c} v$$

$\underbrace{\hspace{10em}}_{c \text{ or } d \text{ in arbitrary order}}$

- $w(v_0 \xrightarrow{l_0} v_1 \xrightarrow{l_1} \dots \xrightarrow{l_{k-1}} v_k) = l_0 l_1 \dots l_{k-1}$
- $Q = \{(v_i, v_j) \mid \exists \pi = v_i \rightarrow \dots \rightarrow v_j; w(\pi) \in \mathcal{L}\}$ , where  $\mathcal{L}$  — formal language
  - ✓ Regular, RPQ  $(ab(c \mid d)^*)$
  - ⚙️ **Context-Free**, CFPQ  $(a^n b^n)$
  - ⌚ Multiple Context-Free  $(a^n c^m b^n d^m)$

Variations:

- All-pairs
- Multiple source
- Reachability
- All paths
- ...

# Applications of Context-Free Path Querying

Hierarchy analysis: variations of the *same-generation queries* is the essence of CFPQ

---

<sup>1</sup>Thomas Reps. 1997. “Program Analysis via Graph Reachability”.

<sup>2</sup>Mihalis Yannakakis. 1990. “Graph-theoretic Methods in Database Theory”.

# Applications of Context-Free Path Querying

Hierarchy analysis: variations of the *same-generation queries* is the essence of CFPQ

## Graph structured data analysis

- Introduced by M. Yannakakis in 1990<sup>1</sup>
- Biological data analysis
- Data provenance analysis
- ...

<sup>1</sup>Thomas Reps. 1997. "Program Analysis via Graph Reachability".

<sup>2</sup>Mihalis Yannakakis. 1990. "Graph-theoretic Methods in Database Theory".



# Applications of Context-Free Path Querying

Hierarchy analysis: variations of the *same-generation queries* is the essence of CFPQ

## Graph structured data analysis

- Introduced by M. Yannakakis in 1990<sup>1</sup>
- Biological data analysis
- Data provenance analysis
- ...

## Static code analysis

- Introduced by T. Reps in 1997<sup>2</sup>
- Interprocedural points-to analysis
- Interprocedural alias analysis
- Type inference related tasks
- ...

<sup>1</sup>Thomas Reps. 1997. "Program Analysis via Graph Reachability".

<sup>2</sup>Mihalis Yannakakis. 1990. "Graph-theoretic Methods in Database Theory".

# Applications of Context-Free Path Querying

Hierarchy analysis: variations of the *same-generation queries* is the essence of CFPQ

## Graph structured data analysis

- Introduced by M. Yannakakis in 1990<sup>1</sup>
- Biological data analysis
- Data provenance analysis
- ...

## Static code analysis

- Introduced by T. Reps in 1997<sup>2</sup>
- Interprocedural points-to analysis
- Interprocedural alias analysis
- Type inference related tasks
- ...

## Graph databases

<sup>1</sup>Thomas Reps. 1997. "Program Analysis via Graph Reachability".

<sup>2</sup>Mihalis Yannakakis. 1990. "Graph-theoretic Methods in Database Theory".

# There is no unified infrastructure to compare CFPQ solutions

- ? What the algorithm is better for the specific problem?
- ? How can one detect that a newly developed algorithm is better than existing ones?
- ? ...

---

<sup>3</sup>Jochem Kuijpers, George Fletcher, Nikolay Yakovets, and Tobias Lindaaker. 2019

# There is no unified infrastructure to compare CFPQ solutions

- ? What the algorithm is better for the specific problem?
- ? How can one detect that a newly developed algorithm is better than existing ones?
- ? ...
- 💡 **Unified benchmarks:** dataset, workflow, ...

---

<sup>3</sup>Jochem Kuijpers, George Fletcher, Nikolay Yakovets, and Tobias Lindaaker. 2019

# There is no unified infrastructure to compare CFPQ solutions

- ? What the algorithm is better for the specific problem?
- ? How can one detect that a newly developed algorithm is better than existing ones?
- ? ...
- 💡 **Unified benchmarks:** dataset, workflow, ...
- ! The first and the only attempt to compare different algorithms: “An Experimental Study of Context-Free Path Query Evaluation Methods”<sup>3</sup>

---

<sup>3</sup>Jochem Kuijpers, George Fletcher, Nikolay Yakovets, and Tobias Lindaaker. 2019

# There is no unified infrastructure to compare CFPQ solutions

- ? What the algorithm is better for the specific problem?
- ? How can one detect that a newly developed algorithm is better than existing ones?
- ? ...
- 💡 **Unified benchmarks:** dataset, workflow, ...
- ! The first and the only attempt to compare different algorithms: “An Experimental Study of Context-Free Path Query Evaluation Methods”<sup>3</sup>
  - 😞 “We conclude that state of the art solutions are not able to cope with large graphs as found in practice.”

---

<sup>3</sup>Jochem Kuijpers, George Fletcher, Nikolay Yakovets, and Tobias Lindaaker. 2019

# There is no unified infrastructure to compare CFPQ solutions

- ? What the algorithm is better for the specific problem?
- ? How can one detect that a newly developed algorithm is better than existing ones?
- ? ...
- 💡 **Unified benchmarks:** dataset, workflow, ...
- ! The first and the only attempt to compare different algorithms: “An Experimental Study of Context-Free Path Query Evaluation Methods”<sup>3</sup>
  - 😞 “We conclude that state of the art solutions are not able to cope with large graphs as found in practice.”

## Difficulties

- Data is spread over projects and papers in different communities
- There is a huge number of different subclasses of the problem
  - ▶ all-pairs, single source, multiple source, ...
  - ▶ reachability, single path, all path, ...

---

<sup>3</sup>Jochem Kuijpers, George Fletcher, Nikolay Yakovets, and Tobias Lindaaker. 2019

# There is no support of CFPQ in real-world graph databases

? Which database or graph analysis system one should use?

☹️ H. Miao and A. Deshpande: “Though the problem has been first studied in our community [40], there is little follow up and support in the context of modern graph databases ...”<sup>4</sup>

## Difficulties

? Algorithm for query engine

! Look at the previous slide

? Query language features to express context-free language constraints

? Syntax

? Semantics

---

<sup>4</sup>H. Miao and A. Deshpande, “Understanding Data Science Lifecycle Provenance via Graph Segmentation and Summarization”, 2019



# There is no support of CFPQ in real-world graph databases

? Which database or graph analysis system one should use?

🙄 H. Miao and A. Deshpande: “Though the problem has been first studied in our community [40], there is little follow up and support in the context of modern graph databases ...”<sup>4</sup>

## Difficulties

? Algorithm for query engine

! Look at the previous slide

? Query language features to express context-free language constraints

? Syntax

? Semantics

⚙️ GQL standard

---

<sup>4</sup>H. Miao and A. Deshpande, “Understanding Data Science Lifecycle Provenance via Graph Segmentation and Summarization”, 2019

# Our Results

- ✓ Collection of linear algebra based algorithms for CFPQ
  - ▶ SuiteSparse is utilized for sparse linear algebra subroutines
  - ▶ Published: [https://github.com/JetBrains-Research/CFPQ\\_PyAlgo](https://github.com/JetBrains-Research/CFPQ_PyAlgo)

---

<sup>5</sup>Arseniy Terekhov et al. 2019. "Multiple-Source Context-Free Path Querying in Terms of Linear Algebra".

# Our Results

- ✓ Collection of linear algebra based algorithms for CFPQ
  - ▶ SuiteSparse is utilized for sparse linear algebra subroutines
  - ▶ Published: [https://github.com/JetBrains-Research/CFPQ\\_PyAlgo](https://github.com/JetBrains-Research/CFPQ_PyAlgo)
- ✓ Full-stack support of CFPQ<sup>5</sup>
  - ▶ On top of RedisGraph: query engine is extended with CFPQ algorithm
  - ▶ openCypher is extended to support CFPQ

---

<sup>5</sup>Arseniy Terekhov et al. 2019. "Multiple-Source Context-Free Path Querying in Terms of Linear Algebra".

# Our Results

- ✓ Collection of linear algebra based algorithms for CFPQ
  - ▶ SuiteSparse is utilized for sparse linear algebra subroutines
  - ▶ Published: [https://github.com/JetBrains-Research/CFPQ\\_PyAlgo](https://github.com/JetBrains-Research/CFPQ_PyAlgo)
- ✓ Full-stack support of CFPQ<sup>5</sup>
  - ▶ On top of RedisGraph: query engine is extended with CFPQ algorithm
  - ▶ openCypher is extended to support CFPQ
- ⚙ Collecting of the dataset for CFPQ benchmarking has started
  - ▶ Synthetic graphs
  - ▶ Real-world graphs
    - ★ Static code analysis
    - ★ Biological data analysis
  - ▶ Published: [https://github.com/JetBrains-Research/CFPQ\\_Data](https://github.com/JetBrains-Research/CFPQ_Data)

---

<sup>5</sup>Arseniy Terekhov et al. 2019. "Multiple-Source Context-Free Path Querying in Terms of Linear Algebra".

# Our Results Evaluation

- All-pairs reachability queries
  - *geospecies*, *taxonomy* — biological data
  - *crypto*, *drivers*, *fs* — points-to analysis
  - Time in seconds
- GPU: Geforce GTX 1070, 1.5GHz, 8Gb RAM, 1920 CUDA cores
  - CPU: Intel core i7-6700 CPU, 3.4GHz, DDR4 64Gb RAM

Graph	#V	#E	Neo4j <sup>6</sup>	RedisGraph <sup>7</sup>	Lin.al. CPU <sup>8</sup>	Lin.al. GPU <sup>9</sup>
geospecies	450 609	2 311 461	6 953.9	80.1	7.1	0.8
taxonomy	5 728 398	14 922 125	n.a.	⚙️	1.1	0.7
crypto	3 464 970	5 976 774	n.a.	⚙️	84.8	28.1
drivers	4 273 803	7 415 538	n.a.	⚙️	269.9	62.5
fs	4 177 416	7 218 746	n.a.	⚙️	165.1	47.7

<sup>6</sup>Jochem Kuijpers et al. 2019. “An Experimental Study of Context-Free Path Query Evaluation Methods”.

<sup>7</sup>Arseniy Terekhov et al. 2019. “Multiple-Source Context-Free Path Querying in Terms of Linear Algebra”.

<sup>8</sup>Standalone linear algebra based algorithm on CPU, using SuiteSparse.

<sup>9</sup>Standalone linear algebra based algorithm on GPU, using spbla.

# Our Results Evaluation

- All-pairs reachability queries
  - *geospecies*, *taxonomy* — biological data
  - *crypto*, *drivers*, *fs* — points-to analysis
  - Time in seconds
- GPU: Geforce GTX 1070, 1.5GHz, 8Gb RAM, 1920 CUDA cores
  - CPU: Intel core i7-6700 CPU, 3.4GHz, DDR4 64Gb RAM

Graph	#V	#E	Neo4j <sup>6</sup>	RedisGraph <sup>7</sup>	Lin.al. CPU <sup>8</sup>	Lin.al. GPU <sup>9</sup>
geospecies	450 609	2 311 461	6 953.9	80.1	7.1	0.8
taxonomy	5 728 398	14 922 125	n.a.	⚙️	1.1	0.7
crypto	3 464 970	5 976 774	n.a.	⚙️	84.8	28.1
drivers	4 273 803	7 415 538	n.a.	⚙️	269.9	62.5
fs	4 177 416	7 218 746	n.a.	⚙️	165.1	47.7

<sup>6</sup>Jochem Kuijpers et al. 2019. “An Experimental Study of Context-Free Path Query Evaluation Methods”.





<sup>7</sup>Arseniy Terekhov et al. 2019. “Multiple-Source Context-Free Path Querying in Terms of Linear Algebra”.

<sup>8</sup>Standalone linear algebra based algorithm on CPU, using SuiteSparse.

<sup>9</sup>Standalone linear algebra based algorithm on GPU, using spbla.

# Our Results Evaluation

- All-pairs reachability queries
  - *geospecies*, *taxonomy* — biological data
  - *crypto*, *drivers*, *fs* — points-to analysis
  - Time in seconds
- GPU: Geforce GTX 1070, 1.5GHz, 8Gb RAM, 1920 CUDA cores
  - CPU: Intel core i7-6700 CPU, 3.4GHz, DDR4 64Gb RAM

Graph	#V	#E	Neo4j <sup>6</sup>	RedisGraph <sup>7</sup>	Lin.al. CPU <sup>8</sup>	Lin.al. GPU <sup>9</sup>
geospecies	450 609	2 311 461	6 953.9	80.1	7.1	0.8
taxonomy	5 728 398	14 922 125	n.a.		1.1	0.7
crypto	3 464 970	5 976 774	n.a.		84.8	28.1
drivers	4 273 803	7 415 538	n.a.		269.9	62.5
fs	4 177 416	7 218 746	n.a.		165.1	47.7

<sup>6</sup>Jochem Kuijpers et al. 2019. “An Experimental Study of Context-Free Path Query Evaluation Methods”.

<sup>7</sup>Arseniy Terekhov et al. 2019. “Multiple-Source Context-Free Path Querying in Terms of Linear Algebra”.

<sup>8</sup>Standalone linear algebra based algorithm on CPU, using SuiteSparse.

<sup>9</sup>Standalone linear algebra based algorithm on GPU, using spbla.

- ⚙️ Benchmarking of linear algebra based algorithms
  - ▶ Comparison of different algorithms for different query semantics
  - ▶ Investigation of scalability on multicore machines
  - ▶ Estimation of performance on GPGPU
- ⚙️ Developing and evaluating GLL-based CFPQ algorithm for Neo4j
  - ▶ Multiple-source
  - ▶ All paths and reachability-only



## Benchmarking of linear algebra based algorithms

- ▶ Comparison of different algorithms for different query semantics
- ▶ Investigation of scalability on multicore machines
- ▶ Estimation of performance on GPGPU

## Developing and evaluating GLL-based CFPQ algorithm for Neo4j

- ▶ Multiple-source
- ▶ All paths and reachability-only

## Describing semantics of (subset of) openCypher in terms of linear algebra (in Coq)

## Utilizing multiple context-free languages as path constraints

# What we should to do?

- To publish unified benchmarks for formal language constrained path querying algorithms
  - ▶ Graphs: synthetic and real-world
  - ▶ Queries: templates and real-world queries
  - ▶ Tasks: all-pairs, single source, reachability, ...

# What we should to do?

- To publish unified benchmarks for formal language constrained path querying algorithms
  - ▶ Graphs: synthetic and real-world
  - ▶ Queries: templates and real-world queries
  - ▶ Tasks: all-pairs, single source, reachability, ...
- To organize language constrained path querying competition
  - ▶ Part of existing competition for graph processing systems
  - ▶ Involve static analysis community

# What we should to do?

- To publish unified benchmarks for formal language constrained path querying algorithms
  - ▶ Graphs: synthetic and real-world
  - ▶ Queries: templates and real-world queries
  - ▶ Tasks: all-pairs, single source, reachability, ...
- To organize language constrained path querying competition
  - ▶ Part of existing competition for graph processing systems
  - ▶ Involve static analysis community
- To provide graph database support
  - ▶ Different algorithms for different systems
  - ▶ Syntax and semantics of query languages