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/

16.07.2021
Formal Language Constrained Path Querying

Navigational queries in edge-labelled graph

![Graph Diagram]

Path to find:

\[ 0 \xrightarrow{a} 1 \xrightarrow{b} 5 \xrightarrow{d} 3 \xrightarrow{c} 2 \xrightarrow{c} 4 \xrightarrow{c} \]

or

\[ 0 \xrightarrow{a} 1 \xrightarrow{b} 5 \xrightarrow{d} 4 \xrightarrow{d} 3 \xrightarrow{c} 2 \xrightarrow{c} 4 \xrightarrow{c} \]

\[ Q = \{ (v_i, v_j) \mid \exists \pi = v_i \rightarrow \ldots \rightarrow v_j; w(\pi) \in \mathcal{L} \} \]

where \( \mathcal{L} \) — formal language

Variations:
- All-pairs
- Multiple source
- Reachability
- All paths

Semyon Grigorev (JBR, SPbSU)
Formal Language Constrained Path Querying

Navigational queries in edge-labelled graph

Path to find:

\[ 0 \xrightarrow{a} v_0 \xrightarrow{b} v_1 \xrightarrow{d} v_2 \xrightarrow{c} v_3 \ldots v_k \xrightarrow{c} v \]

\( c \) or \( d \) in arbitrary order
Navigational queries in edge-labelled graph

Path to find:

\[ 0 \xrightarrow{a} v_0 \xrightarrow{b} v_1 \xrightarrow{d} v_2 \xrightarrow{c} v_3 \ldots v_k \xrightarrow{c} v \]

\(c\) or \(d\) in arbitrary order
Formal Language Constrained Path Querying

Navigational queries in edge-labelled graph

- Path to find:

\[ 0 \xrightarrow{a} v_0 \xrightarrow{b} v_1 \xrightarrow{d} v_2 \xrightarrow{c} v_3 \ldots v_k \xrightarrow{c} v \]

\( c \) or \( d \) in arbitrary order
Navigational queries in edge-labelled graph

Path to find:

\[ 0 \xrightarrow{a} v_0 \xrightarrow{b} v_1 \xrightarrow{d} v_2 \xrightarrow{c} v_3 \ldots v_k \xrightarrow{c} v \]

\[ c \text{ or } d \text{ in arbitrary order} \]

\[ w(v_0 \xrightarrow{l_0} v_1 \xrightarrow{l_1} \ldots \xrightarrow{l_{k-1}} v_k) = l_0l_1 \ldots l_{k-1} \]

\[ Q = \{ (v_i, v_j) \mid \exists \pi = v_i \rightarrow \ldots \rightarrow v_j; w(\pi) \in \mathcal{L} \} \]

where \( \mathcal{L} \) — formal language

- Regular, RPQ (\( ab(c \mid d)^* \))
- Context-Free, CFPQ (\( a^n b^n \))
- Multiple Context-Free (\( a^n c^m b^n d^m \))
Navigational queries in edge-labelled graph

\[ w(v_0 \xrightarrow{l_0} v_1 \xrightarrow{l_1} \ldots \xrightarrow{l_{k-1}} v_k) = l_0 l_1 \ldots l_{k-1} \]

\[ Q = \{(v_i, v_j) \mid \exists \pi = v_i \rightarrow \ldots \rightarrow v_j; w(\pi) \in \mathcal{L}\}, \]

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
- \ldots
Hierarchy analysis: variations of the *same-generation queries* is the essence of CFPQ
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\(^1\)
- Biological data analysis
- Data provenance analysis
- ...

---

\(^1\)Mihalis Yannakakis. 1990. “Graph-theoretic Methods in Database Theory”.
\(^2\)Thomas Reps. 1997. “Program Analysis via Graph Reachability”.
Applications of Context-Free Path Querying

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

<table>
<thead>
<tr>
<th>Graph structured data analysis</th>
<th>Static code analysis</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Introduced by M. Yannakakis in 1990(^1)</td>
<td>• Introduced by T. Reps in 1997(^2)</td>
</tr>
<tr>
<td>• Biological data analysis</td>
<td>• Interprocedural points-to analysis</td>
</tr>
<tr>
<td>• Data provenance analysis</td>
<td>• Interprocedural alias analysis</td>
</tr>
<tr>
<td>• ...</td>
<td>• Type inference related tasks</td>
</tr>
<tr>
<td></td>
<td>• ...</td>
</tr>
</tbody>
</table>

---

\(^1\) Mihalis Yannakakis. 1990. “Graph-theoretic Methods in Database Theory”.  
\(^2\) Thomas Reps. 1997. “Program Analysis via Graph Reachability”.  

Semyon Grigorev (JBR, SPbSU)
Applications of Context-Free Path Querying

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

<table>
<thead>
<tr>
<th>Graph structured data analysis</th>
<th>Static code analysis</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Introduced by M. Yannakakis in 1990(^1)</td>
<td>• Introduced by T. Reps in 1997(^2)</td>
</tr>
<tr>
<td>• Biological data analysis</td>
<td>• Interprocedural points-to analysis</td>
</tr>
<tr>
<td>• Data provenance analysis</td>
<td>• Interprocedural alias analysis</td>
</tr>
<tr>
<td>• ...</td>
<td>• Type inference related tasks</td>
</tr>
<tr>
<td></td>
<td>• ...</td>
</tr>
</tbody>
</table>

Graph databases

\(^1\) Mihalis Yannakakis. 1990. "Graph-theoretic Methods in Database Theory".
\(^2\) Thomas Reps. 1997. "Program Analysis via Graph Reachability".
There is No Unified Infrastructure to Compare CFPQ Solutions

? Which algorithm is better for the specific problem?
? How to assess if a newly developed algorithm is better than the existing ones?
? ...

3Jochem Kuijpers, George Fletcher, Nikolay Yakovets, and Tobias Lindaaker. 2019
There is No Unified Infrastructure to Compare CFPQ Solutions

❔ Which algorithm is better for the specific problem?
❔ How to assess if a newly developed algorithm is better than the existing ones?
❔ ...

💡 Unified benchmarks: dataset, workflow, ...

---

3 Jochem Kuijpers, George Fletcher, Nikolay Yakovets, and Tobias Lindaaker. 2019
There is No Unified Infrastructure to Compare CFPQ Solutions

? Which algorithm is better for the specific problem?
? How to assess if a newly developed algorithm is better than the existing ones?
? ...

💡 Unified benchmarks: dataset, workflow, ...

⚠ The only attempt to compare different algorithms: “An Experimental Study of Context-Free Path Query Evaluation Methods”

---

3Jochem Kuijpers, George Fletcher, Nikolay Yakovets, and Tobias Lindaaker. 2019
There is No Unified Infrastructure to Compare CFPQ Solutions

? Which algorithm is better for the specific problem?
? How to assess if a newly developed algorithm is better than the existing ones?
? ... 

🤔 **Unified benchmarks**: dataset, workflow, ...

❗ The only attempt to compare different algorithms: “An Experimental Study of Context-Free Path Query Evaluation Methods”³

😊 “We conclude that state of the art solutions are not able to cope with large graphs as found in practice.”

---

³Jochem Kuijpers, George Fletcher, Nikolay Yakovets, and Tobias Lindaaker. 2019
There is No Unified Infrastructure to Compare CFPQ Solutions

¿ Which algorithm is better for the specific problem?
¿ How to assess if a newly developed algorithm is better than the existing ones?
¿ …

?? Unified benchmarks: dataset, workflow, …
! The only attempt to compare different algorithms: “An Experimental Study of Context-Free Path Query Evaluation Methods”³

英才 “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, …

³Jochem 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 should you choose?

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 . . .”

---

4H. 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 should you choose?

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 . . .”

Difficulties

How to choose an appropriate algorithm for query engine?

Benchmarks for querying algorithms

How to express context-free constraints in graph query language?

Syntactic features to express context-free language constraints

Semantics of query language

---

4H. 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 should you choose?

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 . . .”

Difficulties

How to choose an appropriate algorithm for query engine?
- Benchmarks for querying algorithms

How to express context-free constraints in graph query language?
- Syntactic features to express context-free language constraints
- Semantics of query language
- GQL standard

---

4H. 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

---

5Arseniy 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

- Full-stack support of CFPQ
  - On top of RedisGraph: query engine is extended with CFPQ algorithm
  - openCypher is extended to support CFPQ

---

5Arseniy 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

✔ Full-stack support of CFPQ
  ▶ On top of RedisGraph: query engine is extended with CFPQ algorithm
  ▶ openCypher is extended to support CFPQ

.Dataset for CFPQ benchmarking: early stages
  ▶ Synthetic graphs
    ★ Theoretical worst case
    ★ Complicated cases
  ▶ Real-world graphs
    ★ Static code analysis
    ★ Biological data analysis
  ▶ Published: https://github.com/JetBrains-Research/CFPQ_Data

---

5Arseniy Terekhov et al. 2019. “Multiple-Source Context-Free Path Querying in Terms of Linear Algebra”.

Semyon Grigorev (JBR, SPbSU)
# 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

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


\(^7\) Arseniy Terekhov et al. 2019. “Multiple-Source Context-Free Path Querying in Terms of Linear Algebra”.

\(^8\) Standalone linear algebra based algorithm on CPU, using SuiteSparse.

\(^9\) 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

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


\(^7\) Arseniy Terekhov et al. 2019. “Multiple-Source Context-Free Path Querying in Terms of Linear Algebra”.

\(^8\) Standalone linear algebra based algorithm on CPU, using SuiteSparse.

\(^9\) 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

<table>
<thead>
<tr>
<th>Graph</th>
<th>#V</th>
<th>#E</th>
<th>Neo4j⁶</th>
<th>RedisGraph⁷</th>
<th>Lin.al. CPU⁸</th>
<th>Lin.al. GPU⁹</th>
</tr>
</thead>
<tbody>
<tr>
<td>geospecies</td>
<td>450 609</td>
<td>2 311 461</td>
<td>6 953.9</td>
<td>80.1</td>
<td>7.1</td>
<td>0.8</td>
</tr>
<tr>
<td>taxonomy</td>
<td>5 728 398</td>
<td>14 922 125</td>
<td>n.a.</td>
<td>1.1</td>
<td>0.7</td>
<td></td>
</tr>
<tr>
<td>crypto</td>
<td>3 464 970</td>
<td>5 976 774</td>
<td>n.a.</td>
<td>84.8</td>
<td>28.1</td>
<td></td>
</tr>
<tr>
<td>drivers</td>
<td>4 273 803</td>
<td>7 415 538</td>
<td>n.a.</td>
<td>269.9</td>
<td>62.5</td>
<td></td>
</tr>
<tr>
<td>fs</td>
<td>4 177 416</td>
<td>7 218 746</td>
<td>n.a.</td>
<td>165.1</td>
<td>47.7</td>
<td></td>
</tr>
</tbody>
</table>

⁷ Arseniy Terekhov et al. 2019. “Multiple-Source Context-Free Path Querying in Terms of Linear Algebra”.
⁸ Standalone linear algebra based algorithm on CPU, using SuiteSparse.
⁹ Standalone linear algebra based algorithm on GPU, using spbla.
Ongoing Research

- 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
Ongoing Research

🎯 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 Should We Do?

- **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 Should We Do?

- 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, ...

- Organize language constrained path querying competition
  - Part of existing competition for graph processing systems
  - Involve static analysis community
What Should We Do?

- 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, ...

- Organize language constrained path querying competition
  - Part of existing competition for graph processing systems
  - Involve static analysis community

- Provide graph database support
  - Different algorithms for different systems
  - Syntax and semantics of query languages