

On Combinators and Single Source Context-Free Path Querying

Mikhail Nikilukin
Inria Paris-Rocquencourt
Rocquencourt, France
trovato@corporation.com

Ekaterina Verbitskaia
The Thørvöld Group
Hekla, Iceland
larst@affiliation.org

Semyon Grigorev
Rajiv Gandhi University
Doimukh, Arunachal Pradesh, India
larst@affiliation.org

ABSTRACT

A clear and well-documented \LaTeX document is presented as an article formatted for publication by ACM in a conference proceedings or journal publication. Based on the “acmart” document class, this article presents and explains many of the common variations, as well as many of the formatting elements an author may use in the preparation of the documentation of their work.

CCS CONCEPTS

• **Information systems** → **Graph-based database models; Query languages for non-relational engines;** • **Theory of computation** → *Grammars and context-free languages;* • **Software and its engineering** → *Functional languages.*

KEYWORDS

Graph Database, Context-Free Path Querying, Parser Combinators, Single-Source Path Querying, CFPQ, Language Constrained Path Querying

ACM Reference Format:

Mikhail Nikilukin, Ekaterina Verbitskaia, and Semyon Grigorev. 2018. On Combinators and Single Source Context-Free Path Querying. In *Woodstock '18: ACM Symposium on Neural Gaze Detection, June 03–05, 2018, Woodstock, NY*. ACM, New York, NY, USA, 3 pages. <https://doi.org/10.1145/1122445.1122456>

1 INTRODUCTION

Context-Free path querying (CFPQ) is an actively developed area in graph database analysis. CFPQ is widely used for static code analysis [?], RDF querying [?], biological data analysis [?].

While lots of research aimed to CFPQ evaluation algorithm development [?], languages which supports context-free constraints specification are not investigated enough. In our knowledge, only extension for Sparql Cfsparql [?] supports context-free constraints. There is also proposal for Cypher¹ which is not implemented yet. So, ways to involve context-free constraints for graph querying should be investigated.

Note, that graph analysis often is only a part of more complex solution. So, graph query languages should be integrated with general-purpose programming languages. Typing [1].

¹!!!!

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

Woodstock '18, June 03–05, 2018, Woodstock, NY

© 2018 Association for Computing Machinery.

ACM ISBN 978-1-4503-XXXX-X/18/06...\$15.00

<https://doi.org/10.1145/1122445.1122456>

Combinators can solve these problems EV!!! [2].

Single source scenario instead of traditional all pairs. Also useful. For manual data analysis. Some of algorithms inherently calculate only all pairs reachability.

In this paper we make the following contributions.

- Introduce example and show how to use combinators for context-free path querying. We demonstrate main features of combinator-based approach such as type-safety, flexibility (compositionality and generics), IDE support and user-defined actions.
- We evaluate single source context-free path querying on some real-world RDFs. We find that the case when number of paths in answer is big, but length of these paths is relatively small is the main case in classical RDF context-free queries. And we show that in this case single-source CFPQ can be evaluated in reasonable time and space. Also our evaluation demonstrates that detailed analysis of theoretical time and space complexity of CFPQ algorithms is required.

2 COMBINATORS FOR CONTEXT-FREE PATH QUERYING

In this section we demonstrate main features of combinators in the context of context-free path querying and integration with general-purpose programming languages. To do it we first introduce a simple graph analysis problem and then show how to solve it by using parser combinators. In our work we use Merrkst.Graph combinators library.

2.1 Problem Statement

Suppose we have an RDF graph and want to analyze hierarchical dependencies over different types of relations. Our goal is for the given object to find all objects which lie on the same level of hierarchy!!!!

2.2 Graph Schema

After graph loaded to the database first we should do is to create typed schema: specify types of nodes and edges.

2.3 Compositionality

same generation query for each type of relation?

We can create generic function!

2.4 Type Safety

Static type checking

2.5 User-Defined Actions

We want to get all reachable vertices and collect information about paths as a result of a query. To do it we equip query with additional user-defined actions.

2.6 IDE Support

Screens!!!!

3 EVALUATION

We evaluate Meerkat.Graph on single source context-free path querying scenario. For evaluation we use Neo4j graph database which was run on PC with the following configuration.

- CPU
- RAM
- OS
- JVM

Neo4j is integrated into application !!!!

Dataset contains two real-world RDFs: Geospecies which contains information about biological hierarchy² and Enzyme which is a part of UniProt database³. Detailed description of these graphs is presented in table 1. Note, that graphs was loaded into database fully, not only edges which labelled by relations used in queries.

Graph	#Vertices	#Edges	#NT	#BT
Enzyme				
Geospecies				

Table 1: Details of graphs

Queries for evaluation are versions of same-generation query – classical context-free query which is useful for hierarchy analysis. We equip queries with user-defined actions for end vertices saving, paths length calculation and unique path counting. To demonstrate power of combinators, we use the function !!! defined above to create queries.

For each graph and each query we run this query from each vertex from graph and measure elapsed time and required memory by using !!! tool. Note, that measured memory is allocated by JVM, not really used.

Enzyme RDF querying. We evaluate two queries: Q_1 – same generation over !!!! relation

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

and Q_2 – same generation over !!!

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

²<https://old.datahub.io/dataset/geospecies>. Access date: 12.11.2019.

³Protein sequences data base: <https://www.uniprot.org/>. RDFs with data are available here: ftp://ftp.uniprot.org/pub/databases/uniprot/current_release/rdf. Access date: 12.11.2019

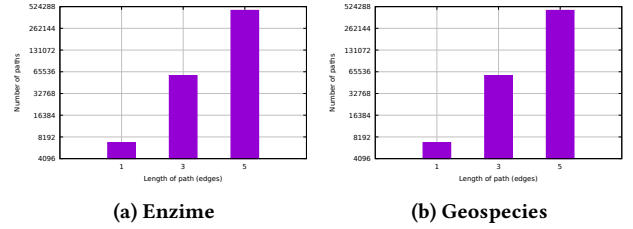


Figure 1: Paths length distribution

Results of evaluation are presented in figures 2 and 3. Also we collect paths length distribution which is showed in figure 1. We can see that provided datasets contain relatively short paths which satisfy queries.

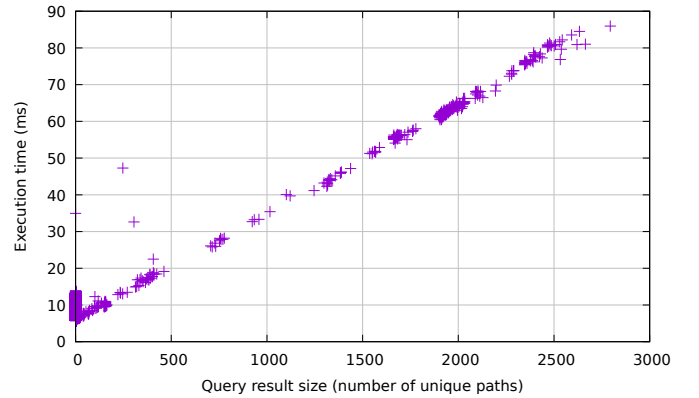


Figure 2: Query execution time for Enzyme dataset and queries Q_1 and Q_2

Figure 2 shows dependency of query evaluation time on query answer size in terms of number of edge-different !!! paths. First of all, we can see that evaluation time is linear on answer size. Also we can see, that time which required to evaluate query for one specific vertex is relatively small. In our case it is less than 90ms.

Figure 3 shows dependency of memory required to evaluate query on query answer size in terms of number of unique paths.

Geospecies RDF querying.

Here we can see !!!!

Finally, we can conclude that context-free path querying in single source scenario can be efficiently evaluated by using !!! in case when number of paths in answer is big but its length is relatively small. While all pairs scenario is still hard [?], single source scenario, which is useful for manual or interactive data analysis, can be !!! Also we can see that while theoretical time and space complexity of CFPQ algorithms at least cubic, in demonstrated scenario real execution time and required memory is linear. So, it is necessary to provide detailed time and space complexity analysis of algorithms.

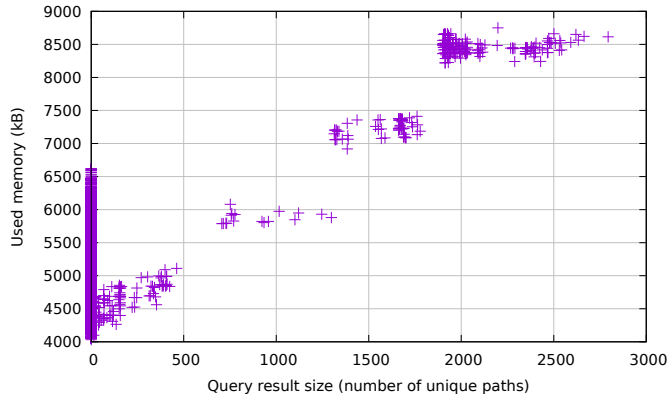


Figure 3: Query required memory for Enzyme dataset and queries Q_1 and Q_2

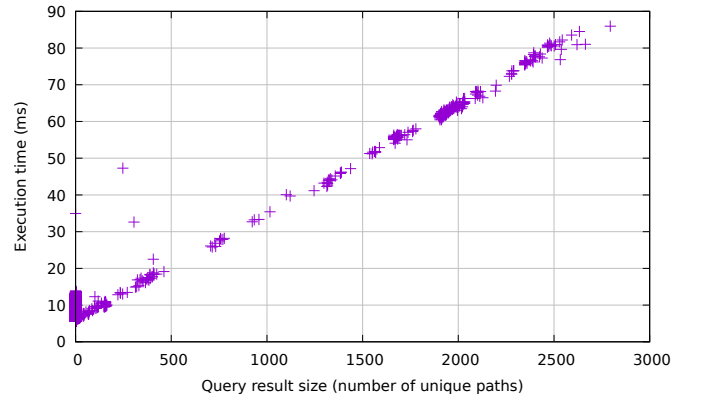


Figure 5: Query execution time for Enzyme dataset and queries Q_3 and Q_4

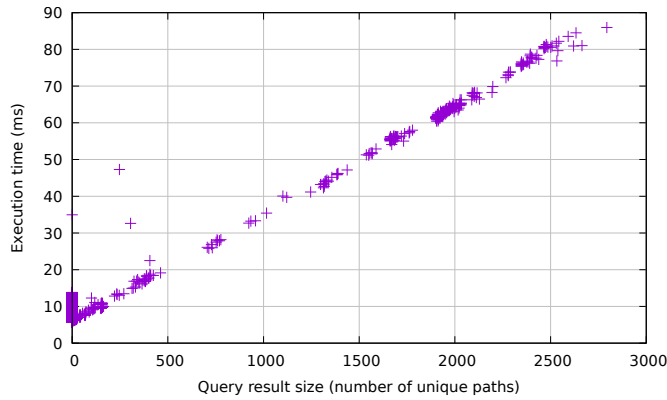


Figure 4: Query execution time for Enzyme dataset and queries Q_3 and Q_4

4 CONCLUSION AND FUTURE WORK

We show that single-source context-free path querying can be !!! We demonstrate a combinator-based approach implemented in Meerkat.Graph Scala library, but this approach can be implemented in almost any high-level programming language. While combinators is a very powerful way to specify context-free queries, it may seem hard to understand for many users. There are other algorithms for context-free path queries which should be applicable for single-source path querying and we hope that they can be integrated with the existing graph database in a more convenient way. But it is necessary more research in this direction.

We should investigate more datasets to detect other shapes of query results. For example, we should investigate the behavior of single-source querying in the case when a number of resulting paths is small, but paths are relatively long. And the first question is which data analysis tasks lead to this scenario.

One of important direction of the future research is to optimize performance of proposed solution. One of possible solution is deep integration with Neo4j infrastructure to utilize cache system.

Another direction is combinators library improvement. First of all, it is necessary to make combinators syntax more user-friendly. Also, it is necessary to create set of query templates (see same-generation template).

ACKNOWLEDGMENTS

To Robert, for the bagels and explaining CMYK and color spaces.

REFERENCES

- [1] Norbert Tausch, Michael Philippsen, and Josef Adersberger. 2011. A Statically Typed Query Language for Property Graphs. In *Proceedings of the 15th Symposium on International Database Engineering & Applications (Lisboa, Portugal) (IDEAS '11)*. Association for Computing Machinery, New York, NY, USA, 219–225. <https://doi.org/10.1145/2076623.2076653>
- [2] Ekaterina Verbitskaia, Ilya Kirillov, Ilya Nozkin, and Semyon Grigorev. 2018. Parser Combinators for Context-Free Path Querying. In *Proceedings of the 9th ACM SIGPLAN International Symposium on Scala (St. Louis, MO, USA) (Scala 2018)*. Association for Computing Machinery, New York, NY, USA, 13–23. <https://doi.org/10.1145/3241653.3241655>