Parser-Combinators for Contex-Free Path Querying

Smolina
Saint Petersburg State University
St. Petersburg, Russia
trovato@corporation.com

Ekaterina Verbitskaia
Saint Petersburg State University
St. Petersburg, Russia
webmaster@marysville-ohio.com

Ilia Kirillov
Saint Petersburg State University
St. Petersburg, Russia
larst@affiliation.org

Semyon Grigorev
Saint Petersburg State University
St. Petersburg, Russia
semen.grigorev@jetbrains.com

ABSTRACT

Aaaaabstract! Abstract, abstract, abstract, abstract, abstract, abstract, abstract, abstract, abstract, abstract, Abstract, abstract, abstract, abstract, abstract, abstract, abstract, abstract, abstract, abstract, Abstract, abstract, abstract, abstract, abstract, abstract, abstract, abstract, abstract, abstract, Abstract, abstract, abstract, abstract, abstract, abstract, abstract, abstract, abstract, Abstract, abstract, abstract, abstract, abstract, abstract, abstract, abstract, abstract, abstract, Abstract, abstract, abstract, abstract, abstract, abstract, abstract, abstract, abstract, Abstract, abstract, abstract, abstract, abstract, abstract, abstract, abstract, abstract, abstract, Abstract, abstract, abstract, abstract, abstract, abstract, abstract, abstract, abstract, abstract, Abstract, abstract, abstract, abstract, abstract, abstract, abstract, abstract, abstract, abstract, abstract, Abstract, abstract, abstract, abstract, abstract, abstract, abstract, abstract, abstract, Abstract, abstract, abstract, abstract, abstract, abstract, abstract, abstract, abstract,

CCS CONCEPTS

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

KEYWORDS

Graph data bases, Language-constrained path problem, Context-Free path querying, Context-Free reachability, Parser Combinators, Generalized LL, GLL, Neo4J, Scala

ACM Reference Format:
Smolina, Ekaterina Verbitskaia, Ilia Kirillov, and Semyon Grigorev. 2018. Parser-Combinators for Contex-Free Path Querying. In *Proceedings of Joint International Workshop on Graph Data Management Experiences & Systems (GRADES) and Network Data Analytics (NDA) 2018 (GRADES-NDA'18)*. ACM, New York, NY, USA, Article 4, 2 pages. https://doi.org/10.475/123_4

1 INTRODUCTION

Graph as data model, Graph data bases. Applications

Navigation queries. Path querying and context-free path querying. Same generation query is not a regular. Static code analysis.

Integration with general purpose programming languages is a problem. String-embedded DSLs, ORMs, etc Strongly-typed integration, an example (string vs linq). Compositionality. Special DSL vs Combinators (LINQ [3, 8], etc) [7]

We propose !!! and we make the following contributions in this paper.

(1) Combinators for CF path querying with structural representation of result
(2) Implementation in Scala. Generalization of linear parsing. Integration with Neo4J graph data base. Available on gitHub:https://github.com/YaccConstructor/Meerkat
(3) Evaluation on realistic data, which shows that it is applicable. Comparison with other tools for CF path querying.

2 RELATED WORK

Language-constrained path querying, Yannakakis [14]. Language reachability framework. Hellings [4, 5], RDF [15], etc [1, 2, 9, 11, 13]

Special graph query languages. SPARQL, cypher

Languge integration problem: special DSLs for SQL, ORM, Linq

Parser-combinators is one of classical approach for parsing!!!.

Scala combinators for graph [7] — one of attempt to adopt combinators technique for graph processing. But language class is not discussed.

Classical combinators has restrictions: left-recursive grammars. GLL [12] can handle arbitrary context-free grammars, SPPF [10]

Parser combinators library bsed on GLL — Meerkat [1] [6].

etc

3 PARSER-COMBITATORS FOR PATH QUERYING

In this section we present our implementation of and describe some details.

Our implenemtation is based on Meerkat library. We need only some steps for generalization.

As far as linear input is a one of case of graph, it is possible to provide input abstraction which make possible to generalize combinators.

The set of implemented combinators:

[1]https://github.com/meerkat-parser/Meerkat

GRADES-NDA'18, June 2018, Houston, Texas USA

ACM ISBN 123-4567-24-567/08/06…$15.00
https://doi.org/10.475/123_4

SPPF may be an arbitrary graph in opposite of linear input parsing.

Let we introduce an example.

Graph.

Grammar.

In terms of combinators from roposed library.

Interface for Neo4J [2] graph bata base

Extensible solution: you need only implement an interface (really?)

An architecture of the solution.

4 EVALUATION

Some experiments on real data and comarison with existing solutions

Comparison with GLL

Comparison with [7]

4.1 Classical RDFs

Query 1 is based on the grammar for retrieving concepts on the same layer (presented in figure 1). For this query our algorithm demonstrates up to 1000 times better performance and provides identical results as compared to the presented in [15] for Q_1.

$$\begin{array}{ll} 0: & S \rightarrow subClassOf^{-1}\ S\ subClassOf \\ 1: & S \rightarrow type^{-1}\ S\ type \\ 2: & S \rightarrow subClassOf^{-1}\ subClassOf \\ 3: & S \rightarrow type^{-1}\ type \end{array}$$

Figure 1: Grammar for query 1

Query 2 is based on the grammar for retrieving concepts on the adjacent layers (presented in figure 2). Note that this query differs from the original query Q_2 from the paper [15] in the following points. First of all, we count only triples for the nonterminal S because only paths derived from it correspond to the paths between concepts on adjacent layers. Algorithm presented in [15] returns triples for all nonterminals. Moreover, the grammar $\mathcal{G}_2$ presented in [15], describes paths not only between concepts on adjacent layers. For example, path "$subClassOf\ subClassOf^{-1}$" can be derived in $\mathcal{G}_2$, but it is a path between concepts on the same layer, not adjacent. We changed the grammar to fit the query to the description provided in the paper [15]. Thus results of our query differs from results for Q_2 which can be found in [15].

$$\begin{array}{ll} 0: & S \rightarrow B\ subClassOf \\ 0: & S \rightarrow subClassOf \\ 1: & B \rightarrow subClassOf^{-1}\ B\ subClassOf \\ 2: & B \rightarrow subClassOf^{-1}\ subClassOf \end{array}$$

Figure 2: Grammar for query 2

Integration with Neo4J

4.2 Static code analysis

Contex-free language reachabiliy framework.

Graph represenataion of program may be stored in graph DB (refs to!!!)

Alias analysis.

5 CONCLUSION

We propose a native way to integrate language for language-constrained path querying into general purpose programming language. We implement it and show that our implementation can be applied for real problems.

Code is available on GitHub:

Future work is

SPPF processing for debugging and results processing

Attributed grammars processing to provide mechanim for semantics calcualtion

REFERENCES

[1] Pablo Barceló, Gaelle Fontaine, and Anthony Widjaja Lin. 2013. Expressive Path Queries on Graphs with Data. In *International Conference on Logic for Programming Artificial Intelligence and Reasoning*. Springer, 71–85.

[2] Chris Barrett, Riko Jacob, and Madhav Marathe. 2000. Formal-language-constrained path problems. *SIAM J. Comput.* 30, 3 (2000), 809–837.

[3] James Cheney, Sam Lindley, and Philip Wadler. 2013. A Practical Theory of Language-integrated Query. *SIGPLAN Not.* 48, 9 (Sept. 2013), 403–416. https://doi.org/10.1145/2544174.2500586

[4] Jelle Hellings. 2014. Conjunctive context-free path queries. (2014).

[5] Jelle Hellings. 2015. Path Results for Context-free Grammar Queries on Graphs. *CoRR* abs/1502.02242 (2015). http://arxiv.org/abs/1502.02242

[6] Anastasia Izmaylova, Ali Afroozeh, and Tijs van der Storm. 2016. Practical, General Parser Combinators. In *Proceedings of the 2016 ACM SIGPLAN Workshop on Partial Evaluation and Program Manipulation (PEPM '16)*. ACM, New York, NY, USA, 1–12. https://doi.org/10.1145/2847538.2847539

[7] Daniel Kröni and Raphael Schweizer. 2013. Parsing Graphs: Applying Parser Combinators to Graph Traversals. In *Proceedings of the 4th Workshop on Scala (SCALA '13)*. ACM, New York, NY, USA, Article 7, 4 pages. https://doi.org/10.1145/2489837.2489844

[8] Erik Meijer, Brian Beckman, and Gavin Bierman. 2006. LINQ: Reconciling Object, Relations and XML in the .NET Framework. In *Proceedings of the 2006 ACM SIGMOD International Conference on Management of Data (SIGMOD '06)*. ACM, New York, NY, USA, 706–706. https://doi.org/10.1145/1142473.1142552

[9] A. Mendelzon and P. Wood. 1995. Finding Regular Simple Paths in Graph Databases. *SIAM J. Computing* 24, 6 (1995), 1235–1258.

[10] Joan Gerard Rekers. 1992. *Parser generation for interactive environments*. Ph.D. Dissertation. Universiteit van Amsterdam.

[11] Juan L Reutter, Miguel Romero, and Moshe Y Vardi. 2015. Regular queries on graph databases. *Theory of Computing Systems* (2015), 1–53.

[12] Elizabeth Scott and Adrian Johnstone. 2010. GLL parsing. *Electronic Notes in Theoretical Computer Science* 253, 7 (2010), 177–189.

[13] Petteri Sevon and Lauri Eronen. 2008. Subgraph queries by context-free grammars. *Journal of Integrative Bioinformatics* 5, 2 (2008), 100.

[14] Mihalis Yannakakis. 1990. Graph-theoretic methods in database theory. In *Proceedings of the ninth ACM SIGACT-SIGMOD-SIGART symposium on Principles of database systems*. ACM, 230–242.

[15] Xiaowang Zhang, Zhiyong Feng, Xin Wang, Guozheng Rao, and Wenrui Wu. 2016. Context-free path queries on RDF graphs. In *International Semantic Web Conference*. Springer, 632–648.

[2] The graph data dase implemented in Java