

# Bring Graph Querying, Formal Language Theory, and Linear Algebra Together to Make It Better

Semyon Grigorev  
Institute for Clarity in Documentation  
Dublin, Ohio, USA  
!!!@corporation.com

Egor Orachev  
The Thørvöld Group  
Hekla, Iceland  
larst@affiliation.org

Vadim Abzalov  
The Thørvöld Group  
Hekla, Iceland  
larst@affiliation.org

Rustam Azimov  
The Thørvöld Group  
Hekla, Iceland  
larst@affiliation.org

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

## ABSTRACT

A clear and well-documented  $\text{\LaTeX}$  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

• **Computer systems organization** → **Embedded systems**; *Redundancy*; Robotics; • **Networks** → Network reliability.

## KEYWORDS

Graph querying, formal language theory, formal language constrained path querying, education

### ACM Reference Format:

Semyon Grigorev, Egor Orachev, Vadim Abzalov, Rustam Azimov, and Ekaterina Shemetova. 2018. Bring Graph Querying, Formal Language Theory, and Linear Algebra Together to Make It Better. In *Woodstock '18: ACM Symposium on Neural Gaze Detection, June 03–05, 2018, Woodstock, NY*. ACM, New York, NY, USA, 2 pages. <https://doi.org/10.1145/1122445.1122456>

## 1 INTRODUCTION

Bridging the gap between fundamental disciplines and application is one of the important problems of education in software engineering. At the same time, real-world problems requires huge amount of preparatory work before the !!!

Data analysis involves broad range of !!! . Graph analysis. Application of fundamental ones. Motivation for students which focused on applied areas to study fundamental disciplines

Integration: Formal languages + HPC + linear algebra + graphs  
To show interconnection between different areas. How concepts and ideas from one area can be applied in another one.

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](mailto: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>

For engineers, so it is important to equip theory with practical tasks to !!!motivate!!!. Historically, programming languages and natural language processing form an area of formal languages theory application

Our experience. Third-year bachelor students. Software engineering.

- Motivation
- Course structure: technical environment, Exercises and how they are related to !!!
- Discussion of !!!

## 2 MOTIVATION

We are aimed to !!!! !!! !!!

Why formal language constrained path querying. Formal languages is not only parsing.

Immediate and direct usage of theory to solve practical tasks: Closure properties. Languages intersection. Results representation — languages representation. Complexity analysis.

Parsing algorithms. New old problems: incremental parsing, parallel parsing.

HPC, matrices, easy to abstract. As a result, easy to apply for real-world data. Which also helps to solve a motivation problem (small synthetic tasks looks disconnected with real-world).

Selected algorithms can be expressed in terms of boolean matrices and vectors: easy to start. No need to create custom semirings. At the same time, native description requires custom semirings. Conversion techniques.

Relations with other areas such as graph theory, dynamic graph problems, algebra.

Open problems (truly subcubic CFPQ).

## 3 COURSE

Actually, our course is a practical part of formal language theory course.

Prerequisites: linear algebra, Python programming.

Formal language theory,

Basic graph theory can be introduced on the fly,

Environment. Automatization. Python programming language, PyFormLang<sup>1</sup> [1] for basis formal languages concepts: regular expressions, finite automata, context-free grammars, recursive automata, and operations over them (automata minimization, regular expression to finite automata conversion), CFPQ-Data<sup>2</sup> as a collection of graphs and queries, sciPy<sup>3</sup> for sparse linear algebra, pyGraphBlas<sup>4</sup> for advanced sparse linear algebra, pyCubool<sup>5</sup> [?], Google colab. Testing (unit tests), code style guide checkers, automation using GitHub actions.

Structure

- (1) Introduction to formal languages, problem statement (FLPQ), different semantics. Problems (infinite number of paths). String to graph generalization.
- (2) Introduction to graphs and linear algebra. Matrices, edge-labelled graphs.
- (3) Regular languages, queries. Closure properties, languages representation.
- (4) Context-Free languages, queries. Closure properties. Language representation.

Exercises. Focused on reachability problem.

- (1) RPQ tensors. FA intersection. (commutativity of Kronecker product)
- (2) RPQ multiple-source BFS-based. Another algorithm for automata intersection. Multiple-source BFS.
- (3) RPQ evaluation and performance analysis. Different matrices formats etc. All-pairs vs multiple sources. Advanced: GPU or GraphBLAS
- (4) CFPQ Hellings
- (5) CFPQ matrices (associativity and commutativity of operations)
- (6) CFPQ tensors (unification of RPQ and CFPQ), RSF
- (7) CFPQ evaluation and performance analysis. Different algorithms comparison. Advanced: GPU or GraphBLAS
- (8) Query language design. Introduction of GQL and other real-world languages.
- (9) Query language implementation

Can be splitted in subtasks or equipped with additional intros (for example, simple tasks aimed to introduce new library).

Outcomes FLPQ HPC

## 4 DISCUSSION

Motivation to study formal languages, refresh algebra, HPC.... Why matrices: pretty simple ideas and algorithms, rather than LR, hides implementation complexity, clear abstraction (it is easy to realize that one library of linear algebra operations can be replaced with another one). More over, students can do it yourself.

Also we want to highlight some drawbacks and weakness of our course. First is that non-linear-algebra-based algorithm for FLPQ (and parsing, respectively) are missed. Based on GLR, GLL. These algorithms are Powerful (can natively solve all-paths queries), but require special techniques that cannot be natively inferred

from linear-algebra-based algorithms. Especially high-performance solution. Looks like advanced block.

Hides basics of some concepts. But learn to use existing libraries that useful skill and allows students to touch real-world problems and tasks.

Directions Multiple sources versions of algorithms for CFL-r. But it should be simplified first. A bit more concepts required. Non matrix based algorithms. Beyond context-free (MCFG). Desidability problems, so on.

## REFERENCES

- [1] Julien Romero. 2021. Pyformlang: An Educational Library for Formal Language Manipulation. In *Proceedings of the 52nd ACM Technical Symposium on Computer Science Education* (Virtual Event, USA) (*SIGCSE '21*). Association for Computing Machinery, New York, NY, USA, 576–582. <https://doi.org/10.1145/3408877.3432464>

<sup>1</sup>!!!

<sup>2</sup>!!!

<sup>3</sup>!!!

<sup>4</sup>!!!

<sup>5</sup>!!!