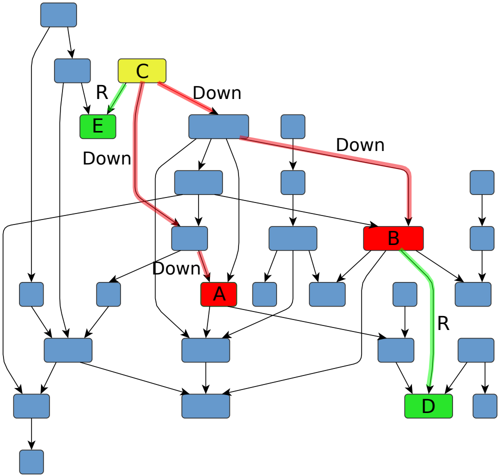
GLL-based Context-Free Path Querying for Neo4j

Vadim Abzalov, Vlada Pogozhelskaya, Vladimir Kutuev,
Olga Bachishche, **Semyon Grigorev**

Saint Petersburg State University

October 31, 2025

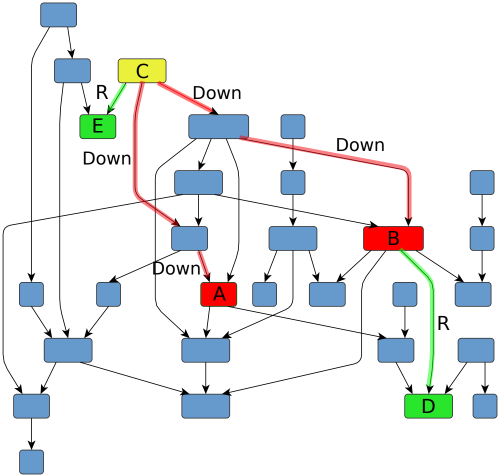
Formal Language Constrained Path Querying



Navigation through an edge-labeled graph

- **Path** specifies a **word** formed by the labels of the edges
- **Paths constraint** is a **language**: the word specified by the path should be in the given language
- The expressiveness of constraints is related to **formal languages classes**

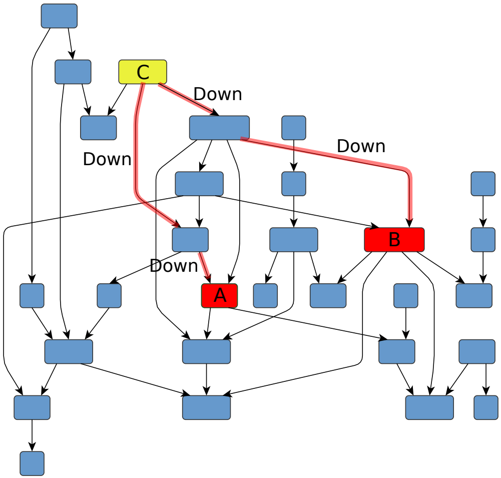
Regular Path Queries (RPQ)



Regular languages as constraints

- Which nodes are reachable from **C** by arbitrary number of **R** and **Down** edges?
- Regular language $\mathcal{L} = (R \mid Down)^*$
- Part of GQL and SQL/PGQ (ISO/IEC 9075-16:2023)

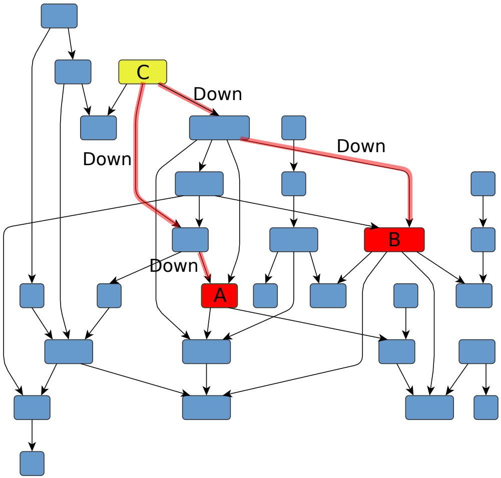
Context-Free Path Queries (CFPQ)



Context-free languages as constraints

- Are nodes A and B on the same level of hierarchy?
- Is there a path of form $\overline{\textbf{Down}}^n\ \textbf{Down}^n$ between A and B?
- Context-free grammar:
  $SameLvl \to \overline{Down}\ SameLvl\ Down \mid \varepsilon$

Context-Free Path Queries (CFPQ)



Context-free languages as constraints

- Are nodes A and B on the same level of hierarchy?
- Is there a path of form $\overline{\textbf{Down}}^n\,\textbf{Down}^n$ between A and B?
- Context-free grammar:
  $SameLvl \to \overline{Down}\ SameLvl\ Down \mid \varepsilon$

Applications

- Static code analysis [T. Reps, et al, 1995]
- Graph segmentation [H. Miao, et al, 2019]
- Bio data analysis [P. Sevon, et al, 2008]
- ...

Problem Statement

- J. Kuijpers, et al[1]: existing algorithms are too slow to be used in practical applications (in the context of Neo4j)
- Reachability in the focus
  - Paths needed in some applications
  - Not for all pairs, but for specified start vertices

? How to create faster multiple source context-free all paths querying algorithm?

[1] Jochem Kuijpers, George Fletcher, Nikolay Yakovets, and Tobias Lindaaker. 2019. An Experimental Study of Context-Free Path Query Evaluation Methods.

Proposed Solution

- **Generalized LL (GLL)**[2] as a base
  - Arbitrary grammars (including left-recursive and ambiguous) without transformations
  - **Shared Packed Parse Forest (SPPF)** is a native representation of all paths
  - Directed — native support of source vertices
- **Recursive State Machine (RSM)** to represent constraints
  - Instead of grammar in (E)BNF[3]

[2] A. Afroozeh, Anastasia Izmaylova. Faster, Practical GLL Parsing. 2015

[3] Right part of the rule is a regular expression over terminals and nonterminals

Definitions

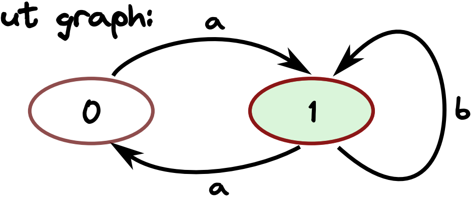
- **Descriptor** $d = (u, q, g)$ — configuration of recognizer/parser
  - u — vertex of the input graph
  - q — state of the RSM (constraint)
  - g — current top of the stack (vertex of GSS)
- **Graph Structured Stack (GSS)** — compact[4] representation of multiple stacks
  - Vertex (M, u) — Handling of nonterminal M is started from vertex u of the input graph
  - Each edge is labelled with p — state of the RSM to continue from after pop
- **Shared Packed Parse Forest (SPPF)** — compact[5] representation of all possible derivation trees
  - **Terminal** and **nonterminal** nodes
  - **Packed, intermediate, etc** — special nodes to reuse subtrees
- **Recursive State Machine (RSM)** — finite-automata-like representation of context-free language
  - DFA representation of right part of rule for each nonterminal

[4] Cubic in worst case

[5] Cubic in worst case

An Example

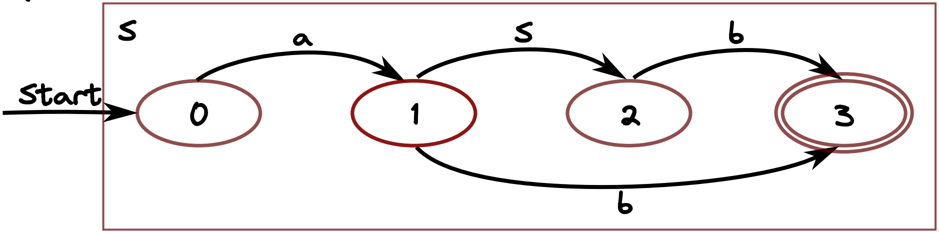
Input graph:



Let 1 be a start vertex

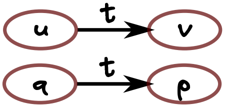
Grammar (query, constarint): S -> a S b | a b

RSM:



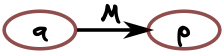
Generalized LL for CFPQ: The Idea

Current descriptor: (u, q, g)

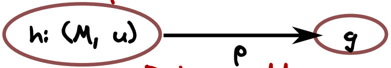


Just read the terminal

New descriptor: (v, p, g)
// For all terminal edges



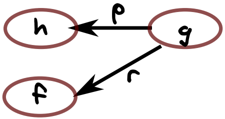
Call: start handling of M on position u



Return address

New descriptor: (u, r, h)
// r: start state for M
// For all Nonterminal edges

q is a final state



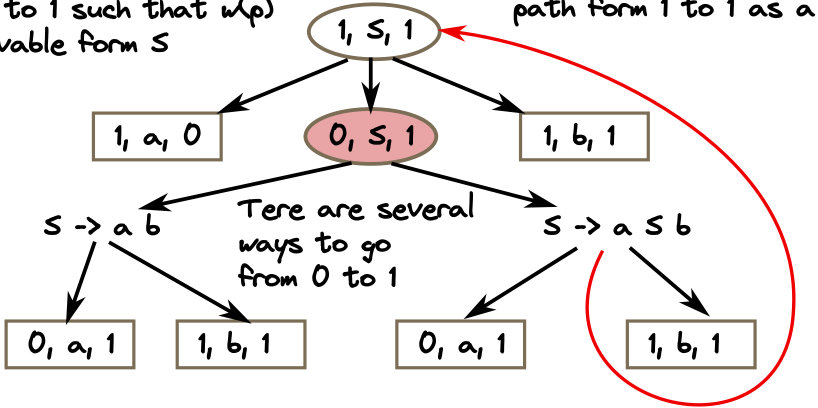
Pop is not destructive.
Just move pointer
alongside outgoing edges

New descriptors: (u, p, h)
(u, r, f)
// For each outgoing edge

SPPF is a Representation of All Paths of Interest

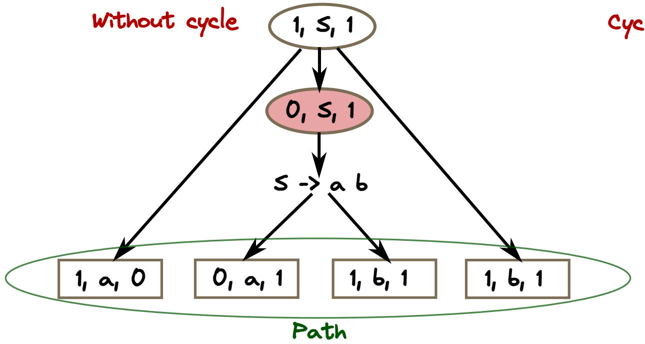
There is at least one path
from 1 to 1 such that w(p)
is derivable form S

Cyclic references:
path from 0 to 1 contains
path form 1 to 1 as a subpath

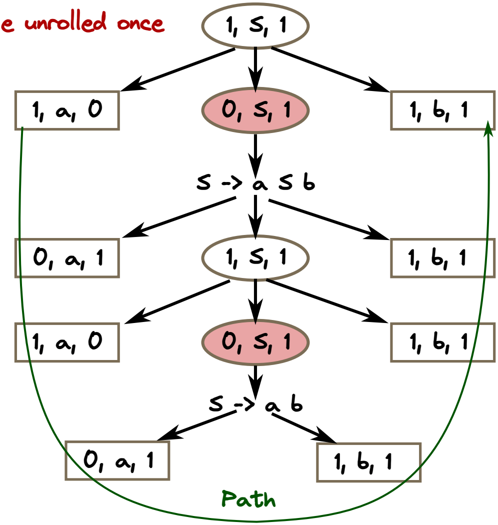


Tere are several
ways to go
from 0 to 1

Trees And Paths



Cycle unrolled once



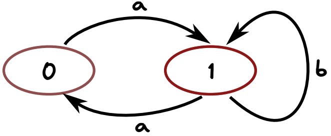
Trees extracted from SPPF for the following paths:

$1 \xrightarrow{a} 0 \xrightarrow{a} 1 \xrightarrow{b} 1 \xrightarrow{b} 1$

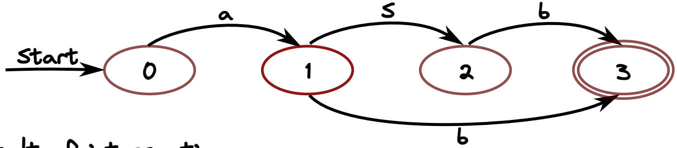
$1 \xrightarrow{a} 0 \xrightarrow{a} 1 \xrightarrow{a} 0 \xrightarrow{a} 1 \xrightarrow{b} 1 \xrightarrow{b} 1 \xrightarrow{b} 1 \xrightarrow{b} 1$

Context-Free Languages Are Closed Under Intersection With Regular Ones

Input graph: regular language

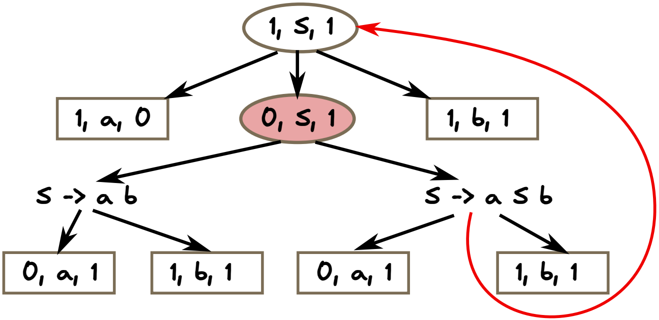


Query: context-free language



Result of intersection

SPPF



Respective context-free grammar

(1, S, 1) -> (1, a, 0) (0, S, 1) (1, b, 1)
(0, S, 1) -> (0, a, 1) (1, b, 1)
(0, S, 1) -> (0, a, 1) (1, S, 1) (1, b, 1)

Nonterminals

Terminals (edges)

Implementation Details

- Generic CFPQ solver (graph is abstraction)
- Neo4j as graph storage (no Cypher extension to support CFPQ)

Evaluation Setup

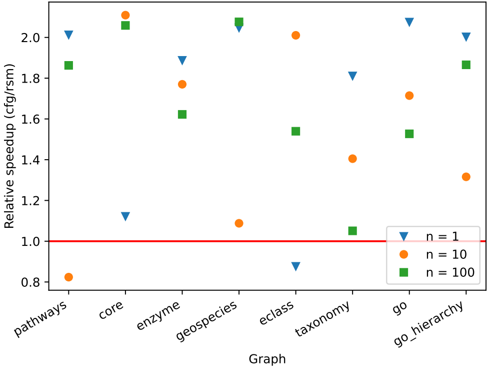
- Ubuntu 20.04, Intel Core i7-6700 CPU, 3.4GHz, DDR4 64Gb RAM
- Neo4j 5.12.0
  - Unlimited memory usage per transaction
- JVM was configured to use 55Gb

Graph name	$\|V\|$	$\|E\|$	#subClassOf	#type
Core	1 323	2 752	178	0
Pathways	6 238	12 363	3 117	3 118
Go_hierarchy	45 007	490 109	490 109	0
Enzyme	48 815	86 543	8 163	14 989
Eclass	239 111	360 248	90 962	72 517
Geospecies	450 609	2 201 532	0	89 065
Go	582 929	1 437 437	94 514	226 481
Taxonomy	5 728 398	14 922 125	2 112 637	2 508 635

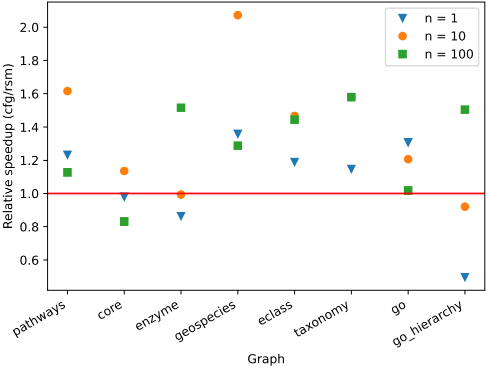
Queries:

$$\begin{aligned}
Q_1 :\quad & S \rightarrow \overline{subClassOf}\ \ S\ subClasOf \mid \overline{type}\ \ S\ type \\
& \quad \mid \overline{subClassOf}\ \ subClassOf \mid \overline{type}\ type \\
Q_2 :\quad & S \rightarrow \overline{subClassOf}\ \ S\ subClasOf \mid subClassOf \\
reg_1 :\quad & S \rightarrow (subClassOf \mid type)^* \\
reg_2 :\quad & S \rightarrow subClassOf^{\,*} \cdot type^*
\end{aligned}$$

CFPQ reachability speedup (RSM over CFG) on RDF graphs



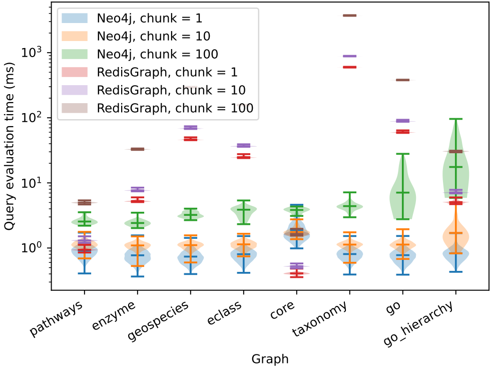
Q_1



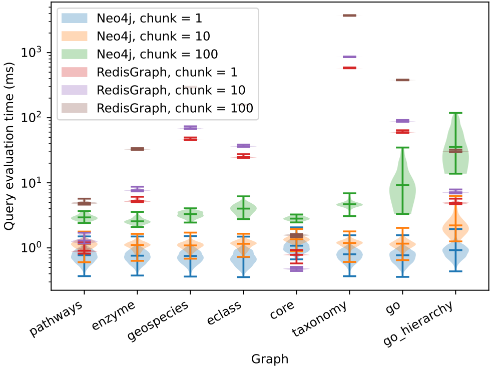
Q_2

CFPQ results (Neo4j vs RedisGraph)

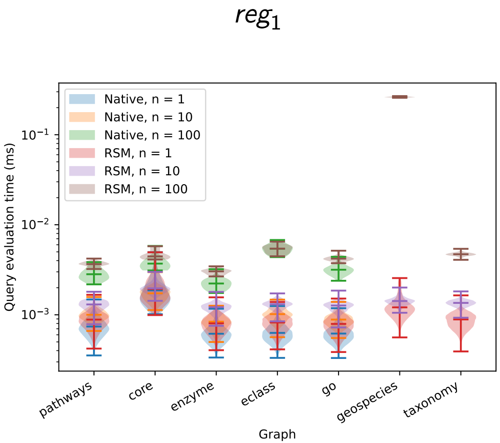
Q_1



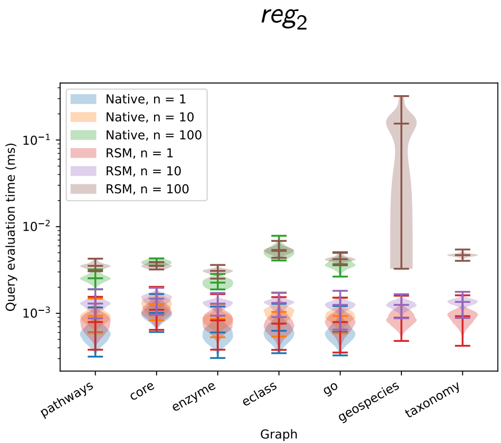
Q_2



RPQ results for Neo4j[6]



reg_1



reg_2

[6] Native solution failed with OOM on last two graphs

Conclusion And Future Research

- ✓ GLL-based CFPQ algorithm
  - RSM is promising representation of CFLs in context of CFPQ
  - In some cases faster than linear-algebra-based approach
  - Comparable with native RPQ (for Neo4j)
- ⚙ Implementation of paths extraction strategies
- ⌛ Parallel version of GLL
  - 💡 All descriptors can be handled independently
  - ☹ Complex global shared structures (GSS, SPPF, etc) — synchronization needed
- ⌛ Incremental version of GLL