



Formal Languages Theory is Not Only a Parsing

Semyon Grigorev

JetBrains Research, Programming Languages and Tools Lab

13.04.2018

Paths in graphs

- Graph analysis
 - ▶ Graph database querying
 - ▶ Network analysis (social networks, Internet, etc)
- Static code analysis
 - ▶ Alias analysis
 - ▶ Taint analysis
 - ▶ Types-related problems
 - ▶ Static analysis of string-embedded languages
- ...

Language constrained path querying

Language-constrained path querying, language reachability

- Σ is a set of terminals
- $L(\Sigma)$ is a language over Σ
- $G = (V, E, D)$ is a directed graph, $E \subseteq V \times D \times V$, $D \subseteq \Sigma$
- $p = v_0 \xrightarrow{l_0} v_1 \xrightarrow{l_1} \dots v_{n-1} \xrightarrow{l_{n-1}} v_n$ is a path in G
- $w(p) = w(v_0 \xrightarrow{l_0} v_1 \xrightarrow{l_1} \dots v_{n-1} \xrightarrow{l_{n-1}} v_n) = l_0 l_1 \dots l_{n-1}$
- $R = \{p \mid w(p) \in L(\Sigma)\}$
 - ▶ **Problem:** R can be an infinite in some cases
- Task may be formulated in other way:
$$Q = \{(v_0, v_n) \mid \exists p = v_0 \xrightarrow{l_0} \dots \xrightarrow{l_{n-1}} v_n (w(p) \in L(\Sigma))\}$$

- !!!
 - ▶ Graph databases query languages (SPARQL, Cypher, PGQL)
 - ▶ OpenCypher: <https://goo.gl/5h5a8P>

- Graph databases and semantic networks (Context-Free Path Querying, CFPQ)
 - ▶ *Sevon P., Eronen L.* “Subgraph queries by context-free grammars.” 2008
 - ▶ *Hellings J.* “Conjunctive context-free path queries.” 2014
 - ▶ *Zhang X. et al.* “Context-free path queries on RDF graphs.” 2016
- Static code analysis
 - ▶ *Thomas Reps et al.* “Precise interprocedural dataflow analysis via graph reachability.” 1995
 - ▶ *Qirun Zhang et al.* “Efficient subcubic alias analysis for C.” 2014
 - ▶ *Dacong Yan et al.* “Demand-driven context-sensitive alias analysis for Java.” 2011
 - ▶ *Jakob Rehof and Manuel Fahndrich.* “Type-base flow analysis: from polymorphic subtyping to CFL-reachability.” 2001

- *Kai Wang et. al.* Graspan: a single-machine disk-based graph system for interprocedural static analyses of large-scale systems code. 2017
 - ▶ “ We have identified a total of 1127 unnecessary NULL tests in Linux, 149 in PostgreSQL, 32 in httpd.”
 - ▶ “Our analyses reported 108 new NULL pointer dereference bugs in Linux, among which 23 are false positives”
 - ▶ “For PostgreSQL and httpd, we detected 33 and 14 new NULL pointer bugs; our manual validation did not find any false positives among them.”

Linear-conjunctive constraints

- $L(\Sigma)$ is a linear-conjunctive language
 - ▶ Interleaving of balanced brackets:
 $L_1 = \{a^n b^n \mid n \geq 0\}; L_2 = \{c^m d^m \mid m \geq 0\}; L_3 = L_1 \odot L_2 = \{ab; acbcdd; cdab; \dots\}$
- Qirun Zhang and Zhendong Su. Context-sensitive data-dependence analysis via linear conjunctive language reachability. 2017

Chelenges for you

- Theoretical open problem
 - ▶ Is there exists an algorithm with time complexity $O(|V|^{3-\epsilon}), \epsilon > 0$?
- Practical utilisation of ideas from “classical” parsing
 - ▶ Algorithms: CYK, (Generalized) LL, (Generalized) LR, Earley, ...
 - ▶ Techniques: parser combinators, parser generators, ...
 - ▶ Advanced techniques: GPGPU utilization, advanced data structures (compact parse forest representation, graph structured stack), ...
- Huge amount of data requires efficien implementation of parallel and/or destributed query processing

Our experiments

- Generalized LL for CFPQ (GLL)
 - ▶ Based on Generalized LL: *Scott E., Johnstone A.* “GLL parsing”
 - ▶ Time complexity: $O\left(|V|^3 * \max_{v \in V} (deg^+(v))\right)$
 - ▶ *Semyon Grigorev and Anastasiya Ragozina.* “Context-free path querying with structural representation of result.” 2017
- GPGPU utilization for CFPQ (GPGPU)
 - ▶ Based on matrix multiplication: *Valiant L.* “General context-free recognition in less than cubic time.” 1974
 - ▶ Time complexity: $O(|V|^2|N|^3(BMM(|V|) + BMU(|V|)))$
 - ▶ *Rustam Azimov, Semyon Grigorev.* “Context-Free Path Querying by Matrix Multiplication.” 2017
- Parser-Combinators for Context-Free Path Querying (in Scala)
 - ▶ Based on Meerkat parser combinator library: *Anastasia Izmaylova, Ali Afroozeh, and Tijs van der Storm.* Practical, general parser combinators. 2016
 - ▶ Work in progress

Performance comparison setup

0 : $\mathbf{S} \rightarrow \text{subClassOf}^{-1} \mathbf{S} \text{ subClassOf}$
1 : $\mathbf{S} \rightarrow \text{type}^{-1} \mathbf{S} \text{ type}$
2 : $\mathbf{S} \rightarrow \text{subClassOf}^{-1} \text{subClassOf}$
3 : $\mathbf{S} \rightarrow \text{type}^{-1} \text{type}$

Query 1

0 : $\mathbf{S} \rightarrow \mathbf{B} \text{ subClassOf}$
1 : $\mathbf{S} \rightarrow \text{subClassOf}$
2 : $\mathbf{B} \rightarrow \text{subClassOf}^{-1} \mathbf{B} \text{ subClassOf}$
3 : $\mathbf{B} \rightarrow \text{subClassOf}^{-1} \text{subClassOf}$

Query 2

Performance comparison result

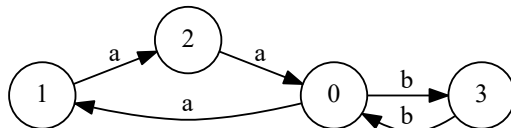
№	#V	#E	Query 1 (ms)			Query 2 (ms)	
			CYK ¹	GLL	GPGPU	GLL	GPGPU
1	144	323	1044	10	12	1	1
2	129	351	6091	19	13	1	0
3	131	397	13971	24	30	1	10
4	179	413	20981	25	15	11	9
5	337	834	82081	89	32	3	6
6	291	685	515285	255	22	66	2
7	341	711	420604	261	20	45	24
8	671	2604	3233587	697	24	29	23
9	733	2450	4075319	819	54	8	6
10	6224	11840	–	1926	82	167	38
11	5864	19600	–	6246	185	46	21
12	5368	20832	–	7014	127	393	40

¹Zhang, et al. "Context-free path queries on RDF graphs."

- E-mail: `semen.grigorev@jetbrains.com`
- GitHub-community YaccConstructor: `https://github.com/YaccConstructor`

Example

Input graph



query is a grammar G which specifies the language $L = \{a^n b^n \mid n \geq 1\}$

0 : $S \rightarrow a S b$

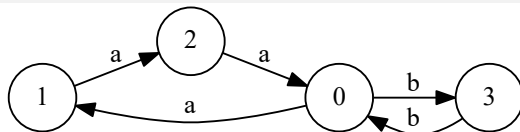
1 : $S \rightarrow \text{Middle}$

2 : $\text{Middle} \rightarrow a b$

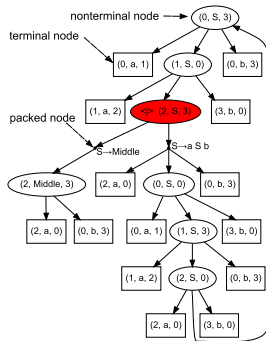
Query result is an infinite set of paths

- $p_1 = 0 \xrightarrow{a} 1 \xrightarrow{a} 2 \xrightarrow{a} 0 \xrightarrow{b} 3 \xrightarrow{b} 0 \xrightarrow{b} 3$
- $p_2 = 0 \xrightarrow{a} 1 \xrightarrow{a} 2 \xrightarrow{a} 0 \xrightarrow{a} 1 \xrightarrow{a} 2 \xrightarrow{a} 0 \xrightarrow{b} 3 \xrightarrow{b} 0 \xrightarrow{b} 3 \xrightarrow{b} 0 \xrightarrow{b} 3 \xrightarrow{b} 0$
- ...

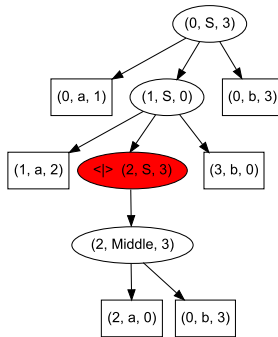
Structural representation of query result



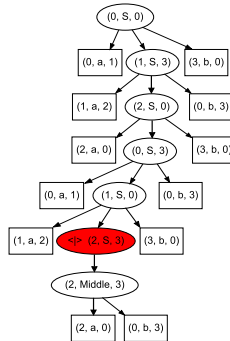
Input graph



Query result (SPPF)

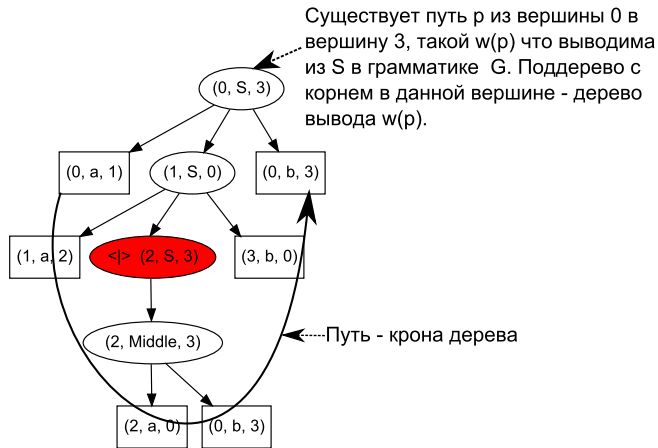
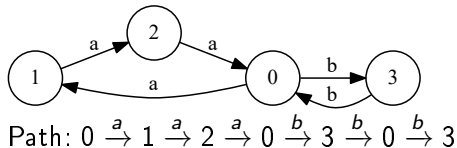


Tree for path p_1



Tree for path p_2

Paths extraction

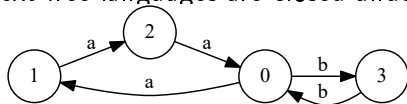


Key idea

Context-free languages are closed under intersection with regular languages

Key idea

Context-free languages are closed under intersection with regular languages



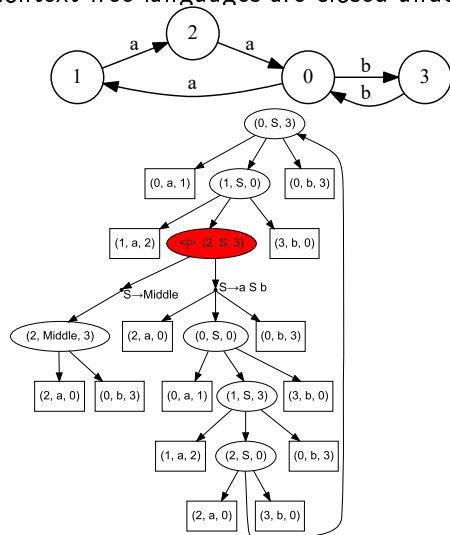
0 : $S \rightarrow a S b$

1 : $S \rightarrow \textit{Middle}$

2 : $\textit{Middle} \rightarrow a b$

Key idea

Context-free languages are closed under intersection with regular languages



0 : $S \rightarrow a S b$

1 : $S \rightarrow \text{Middle}$

2 : $\text{Middle} \rightarrow a b$

$(0, S, 3) \rightarrow (0, a, 1) (1, S, 0) (0, b, 3)$

$(1, S, 0) \rightarrow (1, a, 2) (2, S, 3) (3, b, 0)$

$(2, S, 3) \rightarrow (2, a, 0) (0, S, 0) (0, b, 3)$

$(2, S, 3) \rightarrow (2, \text{Middle}, 3)$

$(0, S, 0) \rightarrow (0, a, 1) (1, S, 3) (3, b, 0)$

$(1, S, 3) \rightarrow (1, a, 2) (2, S, 0) (0, b, 3)$

$(2, S, 0) \rightarrow (2, a, 0) (0, S, 3) (3, b, 0)$

$(0, \text{Middle}, 3) \rightarrow (2, a, 0) (0, b, 3)$