Formal Languages Theory Is Not Only About 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

- Σ is a set of terminals
- $L(\Sigma)$ is a language over Σ

Language constrained path querying

- Σ is a set of terminals
- $L(\Sigma)$ is a language over Σ
- $G = (V, E, D)$ is a directed graph, $E \subseteq V \times D \times V$, $D \subseteq \Sigma$

Language constrained path querying

- Σ is a set of terminals
- $L(\Sigma)$ is a language over Σ
- $G = (V, E, D)$ is a directed graph, $E \subseteq V \times D \times V$, $D \subseteq \Sigma$
- $p = v_0 \xrightarrow{l_0} v_1 \xrightarrow{l_1} \cdots v_{n-1} \xrightarrow{l_{n-1}} v_n$ is a path in G
- $w(p) = w(v_0 \xrightarrow{l_0} v_1 \xrightarrow{l_1} \cdots v_{n-1} \xrightarrow{l_{n-1}} v_n) = l_0 l_1 \cdots l_{n-1}$

Language constrained path querying

- Σ is a set of terminals
- $L(\Sigma)$ is a language over Σ
- $G = (V, E, D)$ is a directed graph, $E \subseteq V \times D \times V$, $D \subseteq \Sigma$
- $p = v_0 \xrightarrow{l_0} v_1 \xrightarrow{l_1} \cdots v_{n-1} \xrightarrow{l_{n-1}} v_n$ is a path in G
- $w(p) = w(v_0 \xrightarrow{l_0} v_1 \xrightarrow{l_1} \cdots v_{n-1} \xrightarrow{l_{n-1}} v_n) = l_0 l_1 \cdots l_{n-1}$
- $R = \{p \mid w(p) \in L(\Sigma)\}$
  - R can be an infinite set in some cases

Language constrained path querying

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

Regular language constraints

- Widely spread
  - Graph databases query languages (SPARQL, Cypher, PGQL)
  - Network analysis
- Still in active development
  - OpenCypher: `https://goo.gl/5h5a8P`
  - Scalability, huge graphs processing
  - Derivatives for graph querying: *Maurizio Nole and Carlo Sartiani.* Regular path queries on massive graphs. 2016

Context-free language constraints

- 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 (Language Reachability Framework)
  - *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

Context-free language constraints

- Interprocedural static nullability analysis[1]
  - “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.”

[1] *Kai Wang et. al.* Graspan: a single-machine disk-based graph system for interprocedural static analyses of large-scale systems code. 2017

Linear-conjunctive language constraints

- May be useful for more accurate context-sensitive analysis
  - Let $L_1 = \{deref^n ref^n | n \geq 0\}$ and $L_2 = \{call^m return^m | m \geq 0\}$;
  - Constraint is $L_3 = L_1 \odot L_2 = \{ab; acbcdd; cdab; \dots\}$ — interleaving of balanced brackets
  - L_3 is not a context-free language but a linear-conjunctive one
- *Qirun Zhang and Zhendong Su.* "Context-sensitive data-dependence analysis via linear conjunctive language reachability." 2017

Challenges for you

- An open problem
  - Is there an algorithm with time complexity $O(|V|^{3-\varepsilon}), \varepsilon > 0$?
- Practical utilization 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 efficient implementation of parallel and/or distributed 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

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

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 CFPQ
  - 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

We use graphs from the classical set of ontologies: *skos, foaf, univ-bench, wine, pizza, etc.*

Queries are classical variants of the same-generation query

$$
\begin{aligned}
0 &: \quad \mathbf{S} \rightarrow subClassOf^{-1}\ \mathbf{S}\ subClassOf \\
1 &: \quad \mathbf{S} \rightarrow type^{-1}\ \mathbf{S}\ type \\
2 &: \quad \mathbf{S} \rightarrow subClassOf^{-1}\ subClassOf \\
3 &: \quad \mathbf{S} \rightarrow type^{-1}\ type
\end{aligned}
$$

Query 1

$$
\begin{aligned}
0 &: \quad \mathbf{S} \rightarrow \mathbf{B}\ subClassOf \\
1 &: \quad \mathbf{S} \rightarrow subClassOf \\
2 &: \quad \mathbf{B} \rightarrow subClassOf^{-1}\ \mathbf{B}\ subClassOf \\
3 &: \quad \mathbf{B} \rightarrow subClassOf^{-1}\ subClassOf
\end{aligned}
$$

Query 2

Performance comparison results

№	#V	#E	Query 1 (ms)			Query 2 (ms)	
			CYK[2]	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

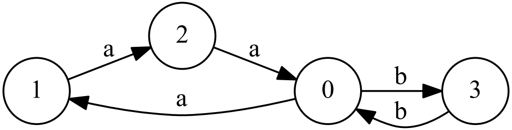
[2] Zhang, et al. "Context-free path queries on RDF graphs."

Thank you!

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

Example

Input graph



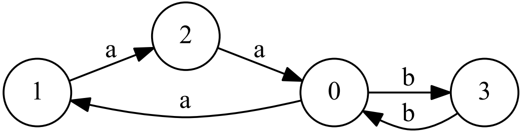
query is a grammar G which specifies the language $L = \{a^n b^n \mid n \geq 1\}$

$$
\begin{array}{ll}
0: & S \to a\ S\ b \\
1: & S \to Middle \\
2: & Middle \to a\ b
\end{array}
$$

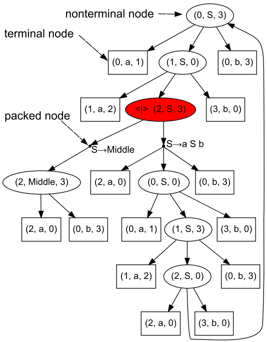
Query result is an infinite set of paths

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

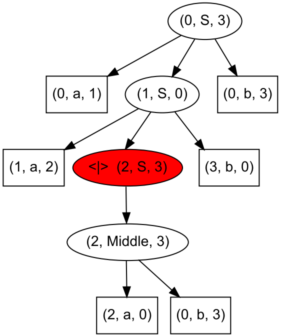
Structural representation of query result



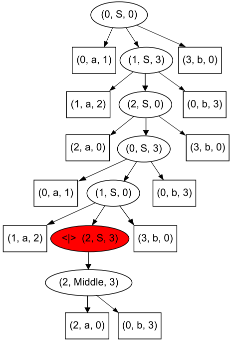
Input graph



Query result (SPPF)

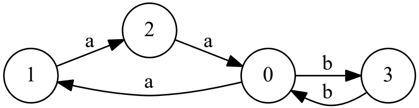


Tree for path p_1



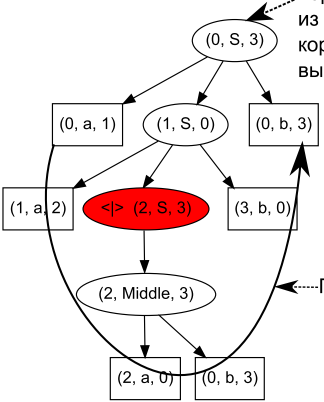
Tree for path p_2

Paths extraction



Path: $0 \xrightarrow{a} 1 \xrightarrow{a} 2 \xrightarrow{a} 0 \xrightarrow{b} 3 \xrightarrow{b} 0 \xrightarrow{b} 3$

Существует путь p из вершины 0 в вершину 3, такой w(p) что выводима из S в грамматике G. Поддерево с корнем в данной вершине - дерево вывода w(p).



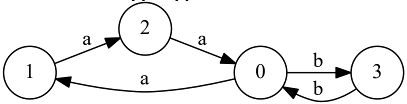
Путь - крона дерева

Key idea

Context-free languages are closed under intersection with regular languages

Key idea

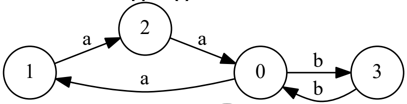
Context-free languages are closed under intersection with regular languages

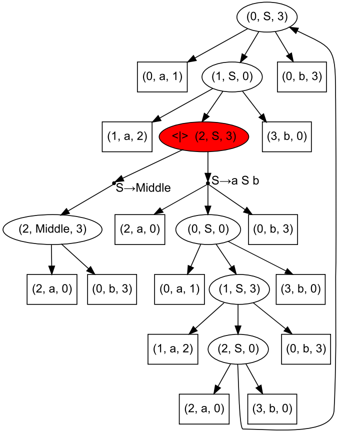


$$
\begin{aligned}
0: &\quad S \to a\ S\ b \\
1: &\quad S \to Middle \\
2: &\quad Middle \to a\ b
\end{aligned}
$$

Key idea

Context-free languages are closed under intersection with regular languages





$$
\begin{aligned}
0 &: \quad S \to a\ S\ b \\
1 &: \quad S \to Middle \\
2 &: \quad Middle \to a\ b
\end{aligned}
$$

$$
\begin{aligned}
(0, S, 3) &\to (0, a, 1)\ (1, S, 0)\ (0, b, 3) \\
(1, S, 0) &\to (1, a, 2)\ (2, S, 3)\ (3, b, 0) \\
(2, S, 3) &\to (2, a, 0)\ (0, S, 0)\ (0, b, 3) \\
(2, S, 3) &\to (2, Middle, 3) \\
(0, S, 0) &\to (0, a, 1)\ (1, S, 3)\ (3, b, 0) \\
(1, S, 3) &\to (1, a, 2)\ (2, S, 0)\ (0, b, 3) \\
(2, S, 0) &\to (2, a, 0)\ (0, S, 3)\ (3, b, 0) \\
(0, Middle, 3) &\to (2, a, 0)\ (0, b, 3)
\end{aligned}
$$