



Huawei-SPbSU Open  
Day 2020



# Fast and Scable Static Code Analysis Requires Fast and Scable Linear Algebra

Semyon Grigorev

Saint Petersburg State University

October 28, 2020

- It plays important role in the development workflow
- **Interprocedural** code analysis is the most important
  - ▶ Computationally hard problem
  - ▶ Can be expressed in terms of **language constrained path querying**

# Language Constrained Path Querying<sup>1</sup>

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

---

<sup>1</sup>Or Formal Language Reachability problem

# Language Constrained Path Querying<sup>1</sup>

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

---

<sup>1</sup>Or Formal Language Reachability problem

# Language Constrained Path Querying<sup>1</sup>

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

---

<sup>1</sup>Or Formal Language Reachability problem

# Language Constrained Path Querying<sup>1</sup>

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

---

<sup>1</sup>Or Formal Language Reachability problem

# Language Constrained Path Querying<sup>1</sup>

- $\Sigma$  is a set of terminals
- $L(\Sigma)$  is a language over  $\Sigma$
- $G = (V, E, D)$  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  $G$
- $w(p) = w(v_0 \xrightarrow{l_0} v_1 \xrightarrow{l_1} \dots v_{n-1} \xrightarrow{l_{n-1}} v_n) = l_0 l_1 \dots l_{n-1}$
- $R = \{p \mid w(p) \in L(\Sigma)\}$ 
  - ▶  $R$  can be an infinite set
- Alternative formulation:  
$$Q = \{(v_0, v_n) \mid \exists p = v_0 \xrightarrow{l_0} \dots \xrightarrow{l_{n-1}} v_n (w(p) \in L(\Sigma))\}$$

---

<sup>1</sup>Or Formal Language Reachability problem

# Context-Free Path Querying (CFPQ)

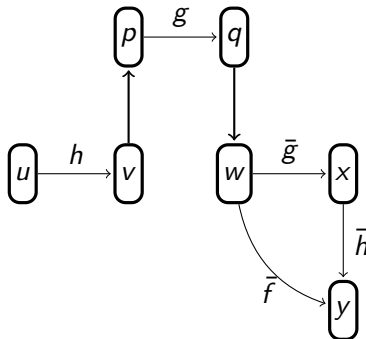
CFPQ is applicable for static code analysis

- *Thomas Reps et al.* “Precise interprocedural dataflow analysis via graph reachability.” 1995
- *Jakob Rehof and Manuel Fahndrich.* “Type-base flow analysis: from polymorphic subtyping to CFL-reachability.” 2001
- *Dacong Yan et al.* “Demand-driven context-sensitive alias analysis for Java.” 2011
- *Qirun Zhang et al.* “Efficient subcubic alias analysis for C.” 2014
- ...



## Example: Field-Sensitivity

```
1   v.h = u;  
2   ...  
3   p = v;  
4   ...  
5   q.g = p;  
6   ...  
7   w = q;  
8   ...  
9   x = w.g;  
10  if (...) {  
11    y = w.f;  
12  }  
13  else {  
14    y = x.h;  
15  }
```



Correct path:  $hg\bar{g}\bar{h}$

Incorrect path:  $hg\bar{f}$

## Interprocedural static nullability analysis<sup>2</sup>

- “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.”

---

<sup>2</sup>*Kai Wang et. al.* Graspan: a single-machine disk-based graph system for interprocedural static analyses of large-scale systems code. 2017

- CFQP can be formulated in terms of linear algebra: *Rustam Azimov, Semyon Grigorev* “Context-free path querying by matrix multiplication.” 2018
- CFQP can be efficiently implemented using sparse linear algebra and modern parallel hardware: *Arseniy Terekhov, Artyom Khoroshev, Rustam Azimov, Semyon Grigorev* “Context-Free Path Querying with Single-Path Semantics by Matrix Multiplication.” 2020

- PC
  - ▶ OS: Ubuntu 18.04
  - ▶ CPU: Intel core i7 8700k 3,4GHz
  - ▶ RAM: DDR4 64 Gb

---

<sup>3</sup><https://people.engr.tamu.edu/davis/GraphBLAS.html>

- PC
  - ▶ OS: Ubuntu 18.04
  - ▶ CPU: Intel core i7 8700k 3,4GHz
  - ▶ RAM: DDR4 64 Gb
- Points-to analysis
- Graphs are generated by LLVM for submodules of Linux core
- Implementation is based on SuiteSparse:GraphBLAS<sup>3</sup>

---

<sup>3</sup><https://people.engr.tamu.edu/davis/GraphBLAS.html>

## Evaluation: Results<sup>4</sup>

Name	#V	#E	Time (min)
arch	3 448 422	5 940 484	3.25
crypto	3 464 970	5 976 774	3.25
drivers	4 273 803	7 415 538	17.5
fs	4 177 416	7 218 746	6.2

---

<sup>4</sup>We include only time spent analyzing graph. Graph building time is not included

# Conclusion

- Many interprocedural static code analysis tasks can be expressed in terms of linear algebra
- Sparse linear algebra can be efficiently implemented on modern parallel hardware
- Database querying, social network analysis and other problems can be reduced to sparse linear algebra