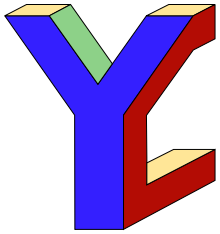


Huawei-SPbSU Open
Day 2020

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

Semyon Grigorev

St Petersburg University

October, 2020

Static Code Analysis

- Important part of development workflow
- The most important case is **interprocedural** code analysis

Static Code Analysis

- Important part of development workflow
- The most important case is **interprocedural** code analysis
  - Computationally hard problem
  - Can be expressed in terms of **language constrained path querying**

Language Constrained Path Querying[1]

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

[1] Or Formal Language Reachability problem

Language Constrained Path Querying[1]

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

[1] Or Formal Language Reachability problem

Language Constrained Path Querying[1]

- 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
- $w(p) = w(v_0 \xrightarrow{l_0} v_1 \xrightarrow{l_1} \cdots v_{n-1} \xrightarrow{l_{n-1}} v_n) = l_0 l_1 \cdots l_{n-1}$

[1] Or Formal Language Reachability problem

Language Constrained Path Querying[1]

- 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

[1] Or Formal Language Reachability problem

Language Constrained Path Querying[1]

- 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:
  $Q = \{(v_0, v_n) \mid \exists p = v_0 \xrightarrow{l_0} \cdots \xrightarrow{l_{n-1}} v_n \ (w(p) \in L(\Sigma))\}$

[1] Or Formal Language Reachability problem

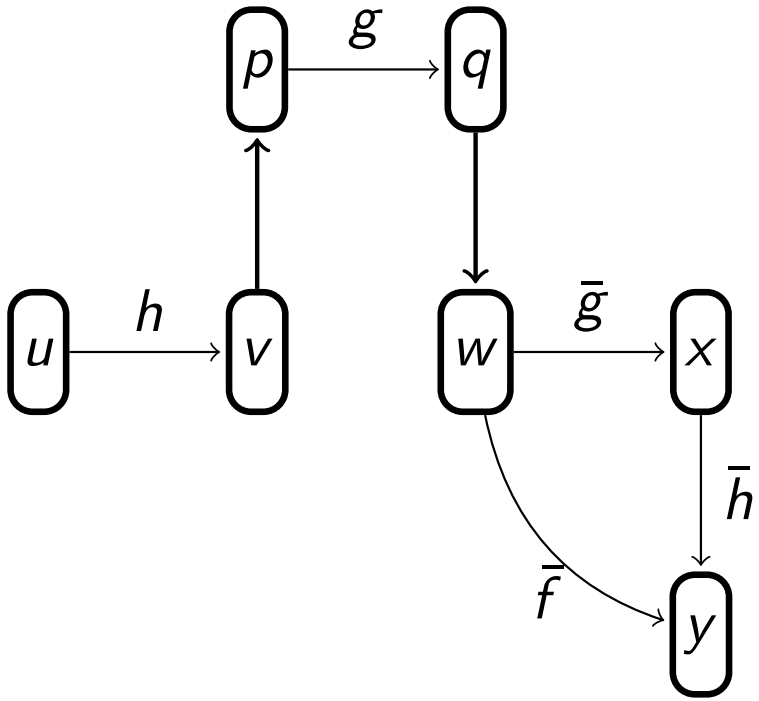
Context-Free Path Querying (CFPQ)

CFPQ is applicable for static code analysis (Language Reachability Framework)

- *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: $hg\bar{f}$

Applicability of CFPQ

- Interprocedural static nullability analysis[2]
  - "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."

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

CFPQ and Linear Algebra

- 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

Evaluation

- PC
  - OS: Ubuntu 18.04
  - CPU: Intel core i7 8700k 3,4GHz
  - RAM: DDR4 64 Gb
- Graphs are generated by LLVM for submodules of Linux core
- Implementation is based on SuiteSparse:GraphBLAS[3]

[3] `https://people.engr.tamu.edu/davis/GraphBLAS.html`

Evaluation: Results[4]

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

[4] Graph analysis only. Graph building time is not included

Conclusion

- Number of tasks of interprocedural static code analysis can be expressed in terms of linear algebra
- Sparse linear algebra can be efficiently implemented for modern parallel hardware
- Not only static code analysis can be reduced to sparse linear algebra