

Flow2Vec: Value-Flow-Based Precise Code Embedding

Applications of CFPQ analysis

Егор Орачев

JetBrains Research, Лаборатория языковых инструментов
Санкт-Петербургский Государственный университет

22 марта 2021

```
1 int speed(int input) {  
2     int x, y, k;  
3     k = input / 100;  
4     x = 2;  
5     y = k + 5;  
6  
7     while ( x < 10 ) {  
8         x++;  
9         y = y + 3;  
10    }  
11  
12    if ((3*k + 100) > 43) {  
13        y++;  
14        x = x / ( x - y );  
15    }  
16  
17    return x;  
}
```

Figure: Code fragment

Статический анализ кода

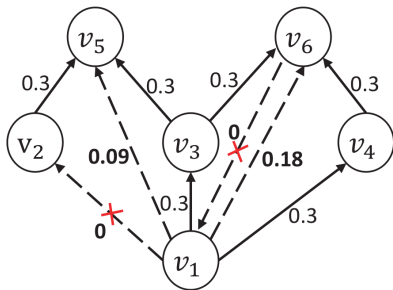
- Interprocedural data flow analysis
- Program slicing
- Pointer analysis
- Shape analysis
- Code classification
- Code summarization

Представление программы

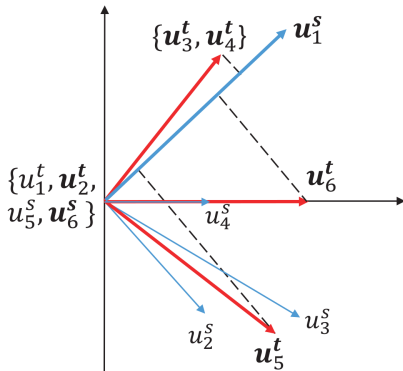
- Проблема: эффективность анализа зависит от того, насколько "хорошим" является используемое представление программы
- Варианты:
 - ▶ Абстрактное синтаксическое дерево
 - ▶ Представление в промежуточном языке
 - ▶ Граф потока данных
 - ▶ Граф потока управления
 - ▶ Граф вызовов
 - ▶ Представление программы в виде embedding'a

- Построение представления графа в векторном пространстве выбранной размерности
 - ▶ Вершины графа — это вектора
 - ▶ Можно использовать в задачах реконструкции графа, рекомендации соседей, предсказания связей и т.д.
- Проблемы:
 - ▶ Необходимо сохранить "важные" свойства графа
 - ▶ Реальные графы являются ориентированными и обладают асимметричной транзитивностью

Graph Embedding: Идея



(a) Directed graph



(b) Graph embedding

Figure: HOPE: Asymmetric Transitivity Preserving Graph Embedding

Graph Embedding: Постановка задачи

- Ориентированный граф $G = \langle V, E \rangle$, $V = \{v_1, \dots, v_N\}$, $|V| = N$, $e_{ij} = (v_1, v_2) \in E$
- Матрица смежности графа A , a_i - i -ая строка матрицы, A_{ij} - элемент матрицы в i -ой строке и j -том столбце
- Матрица $S \in \mathbb{R}^{N \times N}$ - матрица близости графа
- Embedding матрицы $U = [U^s, U^t]$, $U^s, U^t \in \mathbb{R}^{N \times K}$, K - размерность embedding'a, вектора u_i^s, u_i^t соответствуют вершине графа v_i
- Хотим такое U , чтобы минимизировать потери $\min \|S - U^s * U^{tT}\|_F^2$

Graph Embedding: High-order Proximity Matrix

- Построение матрицы близости S , которая сохранит "важные" свойства графа для дальнейшего анализа
- Варианты выбора S :
 - ▶ Индекс Катца (англ. Katz index):
 $S^{Katz} = \sum_{i=1}^{\infty} \beta^i A^i$, где $\beta \in (0, 1)$ - фактор ослабления
 - ▶ Rooted PageRank
 - ▶ Common Neighbors
 - ▶ Adamic-Adar
- Только Katz index и Rooted PageRank сохраняют глобальную ассиметричную транзитивность графа

Graph Embedding: Approximation of High-Order Proximity

- $S = S^{Katz} = \sum_{i=1}^{\infty} \beta^i A^i$
- $S = \sum_{i=1}^N \sigma_i v_i^s v_i^t{}^T$ используя SVD разложение, где $\{\sigma_1, \dots, \sigma_N\}$ сингулярные значения в порядке убывания
- $U^s = [\sqrt{\sigma_1} * v_1^s, \dots, \sqrt{\sigma_K} * v_K^s]$
- $U^t = [\sqrt{\sigma_1} * v_1^t, \dots, \sqrt{\sigma_K} * v_K^t]$
- Ошибка аппроксимации: $\|S - U^s * U^t{}^T\|_F^2 = \sum_{i=K+1}^N \sigma_i^2$
- Относительная ошибка аппроксимации:
$$\frac{\|S - U^s * U^t{}^T\|_F^2}{\|S\|_F^2} = \frac{\sum_{i=K+1}^N \sigma_i^2}{\sum_{i=1}^N \sigma_i^2}$$

- Для ориентированного графа G можем построить embedding, который сохраняет ассиметричную транзитивность графа
- Для построения требуется матрица близости высокого порядка S , которая сохраняет глобальную ассиметричную транзитивность графа
- Полученный embedding $U = [U^s, U^t]$ обладает имеет доказанные оценки ошибки приближения. Значение ошибки можно уменьшить, увеличив значение параметра K

Context-free Path Querying

Context-free Path Querying: Терминология

Context-free Path Querying: Семантика Запросов

Context-free Path Querying: Пример

Code Embedding: Предыстория

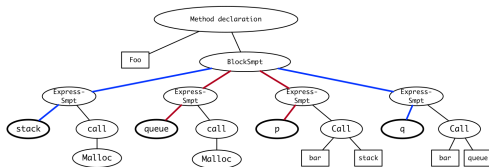
- Развитие техник embedding'а в области обработки и анализа естественных языков (word2vec, и т.д.)
- Развитие техник code embedding'а для построения представления программ (code2vec, и т.д.)

- Развитие техник embedding'а в обработке естественных языков
- Code embedding'и в качестве представления программ
- Методы машинного обучения для анализа программ по новому представлению

Code Embedding: Проблемы

```
foo(){  
    stack = malloc(..);  
    queue = malloc(..);  
    p = bar(stack); //cs1  
    q = bar(queue); //cs2  
}
```

(a) Foo function



(b) Ast for foo

Figure: Spurious paths example

- Существующие инструменты
 - ▶ Code2vec
 - ▶ Code2seq
 - ▶ Word2vec-like ...
- Недостатки
 - ▶ Не учитывают меж-процедурное взаимодействие
 - ▶ Не учитывают псевдонимы (ссылки)
 - ▶ Не учитывают ассиметричную транзитивность программ

- Новый алгоритм, предложенный в статье *Flow2Vec: Value-Flow-Based Precise Code Embedding*¹
- Учитывает ранее указанные недостатки
- Использует Intermediate Representation (IR), Interprocedural Value-Flow Graph (IVFG) и CFPQ для построения value-flow reachability матриц

¹Ссылка: <https://dl.acm.org/doi/10.1145/3428301>

Flow2Vec: Алгоритм

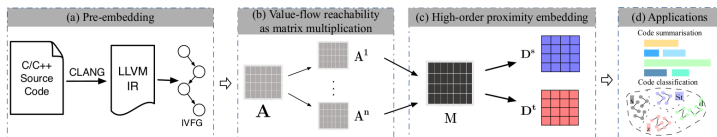


Figure: Flow2vec idea

- Шаг 1: Построение IR, IVFG, исходных матриц с call/return и value-flow информацией
- Шаг 2: Value-flow reachability через CFPQ
- Шаг 3: Построение embedding'a высокого порядка
- Шаг 4: Применение построенного embedding'a в пользовательских приложениях

Шаг 1: Pre-embedding (1)

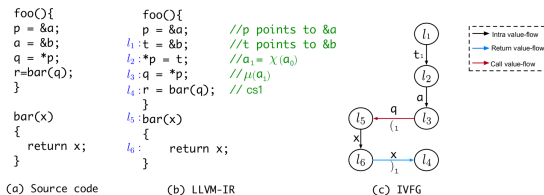
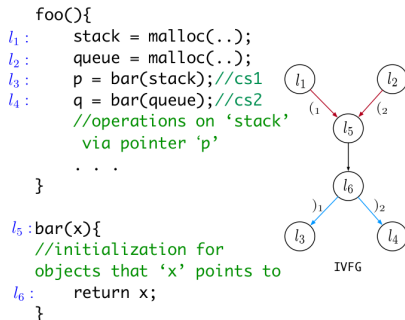


Figure: C Code fragment, its LLVM-IR and IVFG

- LLVM-IR в качестве промежуточного представления
- Построение IVFG на основе LLVM-IR программы
- $\mathcal{V} = \mathcal{O} \cup \mathcal{P}$, два типа переменных
- $t \xrightarrow{v} t', v \in \mathcal{V}$ def-use отношение
- $t \xrightarrow{p} t', p \in \mathcal{P}$ direct value-flow отношение

War 1: Pre-embedding (2)



(a) Code fragment and IVFG

	l_1	l_2	l_3	l_4	l_5	l_6
l_1	0	0	0	0	$(1$	0
l_2	0	0	0	0	$(2$	0
l_3	0	0	0	0	0	0
l_4	0	0	0	0	0	0
l_5	0	0	0	0	0	1
l_6	0	0	$)_1$	$)_2$	0	0

(b) Call/return and value-flow matrix

Figure: Pre-embedding example

Шар 2: Value-flow reachability via matrix multiplication

$$\begin{bmatrix} 0 & 0 & 0 & 0 & (1 & 0) \\ 0 & 0 & 0 & 0 & (2 & 0) \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 &)_1 &)_2 & 0 & 0 \end{bmatrix}$$

A^1



$$\begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 \end{bmatrix}$$

(a) Matrix A^1

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 1 * (1) \\ 0 & 0 & 0 & 0 & 0 & 1 * (2) \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 *)_1 & 1 *)_2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

A^2



$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

(b) Matrix A^2

$$\begin{bmatrix} 0 & 0 & (1*1*)_1 & -(1*1*)_2 & 0 & 0 \\ 0 & 0 & -(2*1*)_1 & (2*1*)_2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

A^3



$$\begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

(c) Matrix A^3

Figure: Context-sensitive value-flow reachability

Шаг 3: High-order proximity embedding

$$\mathbf{M} = \sum_{h=1}^3 (\beta \cdot \mathbf{A})^h$$

$$\begin{bmatrix} 0 & 0 & 0.512 & 0 & 0.8 & 0.64 \\ 0 & 0 & 0 & 0.512 & 0.8 & 0.64 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.64 & 0.64 & 0 & 0.8 \\ 0 & 0 & 0.8 & 0.8 & 0 & 0 \end{bmatrix}$$

$$\mathbf{M}(\beta = 0.8)$$

(a) High-order proximity matrix

$$\mathbf{M} \approx \mathbf{D}^s \cdot \mathbf{D}^{t\top}$$

$$\begin{bmatrix} -0.51 & -0.49 & -0.69 & 0 & 0 & 0 \\ 0.51 & -0.49 & -0.69 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0.33 & -0.79 & 0 & 0 & 0 \\ 0 & 0.71 & -0.58 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ -0.51 & 0.49 & -0.69 & 0 & 0 & 0 \\ 0.51 & 0.49 & -0.69 & 0 & 0 & 0 \\ 0 & -0.71 & -0.58 & 0 & 0 & 0 \\ 0 & -0.33 & -0.79 & 0 & 0 & 0 \end{bmatrix}$$

$$\mathbf{D}^s$$

$$\mathbf{D}^t$$

(b) Embedding vectors (K -factor is 3)

Figure: Embedding step

Шар 4: Application scenarios

Source code (A)		Source code (B)	
<pre>int _____(dict *d){ int minimal; minimal = d->ht[0].used; if (minimal<INITIAL_SIZE) minimal=INITIAL_SIZE; return dictExpand(d, minimal); } int dictExpand(dict *d, unsigned long size){ dictht n; unsigned long realsize = _dictNextPower(size); n.table = zcalloc(realsize*sizeof(dictEntry*)); return DICT_OK; }</pre>		<pre>SIG _____(char* inp){ if (!is_valid(inp)) return FAULT; CMD *cmd = parse(inp); return process(cmd); } SIG process(CMD *cmd){ Executor *e = get_glb_executor(); if(e->is_full) return FULL; e->execute(cmd); return SUCESS; }</pre>	
Ground truth	resize to the minimal	Ground truth	parse and execute command
Code2Vec	<code>isMinimal</code>	Code2Vec	<code>check</code>
Code2Seq	<code>Expand</code> size	Code2Seq	<code>parse</code>
Flow2Vec	<code>reset</code> <code>minimal</code> memory	Flow2Vec	<code>parse</code> <code>command</code> and <code>execute</code>

Figure: Code summarization example

- Code classification
- Code summarization

Что мы можем предложить

- Анализ Java программ
- Инструменты для построения графов
 - ▶ **WALA**²
 - ▶ **Soot**³
- Построение value-flow reachability матриц
- Построение emdebbling'a

²ссылка: <https://github.com/wala/WALA>

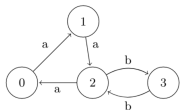
³ссылка: <https://github.com/soot-oss/soot>

- Матричный Алгоритм Рустама Азимова⁴
 - ▶ Семантика достижимости
 - ▶ Семантика одного пути
 - ▶ Семантика всех путей
 - ▶ *Можем модифицировать полукольцо, чтобы считать кол-во уникальных путей*
- Алгоритм на основе произведения Кронекера и PA⁵
 - ▶ Семантика достижимости и всех путей
 - ▶ Итеративное извлечение путей
 - ▶ *Но! На выходе нечто большее, чем просто граф*

⁴ссылка: <https://dl.acm.org/doi/10.1145/3398682.3399163>

⁵ссылка: https://link.springer.com/chapter/10.1007/978-3-030-54832-2_6

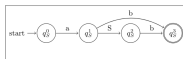
Kronecker CFPQ



(a) Input directed graph \mathcal{G}

$$\begin{pmatrix} \cdot & \{a\} & \cdot & \cdot \\ \cdot & \cdot & \{a\} & \cdot \\ \{a\} & \cdot & \cdot & \{b\} \\ \cdot & \cdot & \{b\} & \cdot \end{pmatrix}$$

(b) \mathcal{G} adjacency matrix



(c) RSA for $S \rightarrow ab \mid aSb$

$$\begin{pmatrix} \cdot & \{a\} & \cdot & \cdot \\ \cdot & \cdot & \{S\} & \{b\} \\ \cdot & \cdot & \cdot & \{b\} \\ \cdot & \cdot & \cdot & \cdot \end{pmatrix}$$

(d) RSA adjacency matrix

$$\left(\begin{array}{ccc|ccc|ccc} \cdot & \cdot & \cdot & \cdot & 1 & \cdot & \cdot & 1 & 1 & \cdot & \cdot & 1 & 1 \\ \cdot & \cdot & \cdot & \cdot & \cdot & 1 & \cdot & \cdot & 1 & \cdot & \cdot & 1 & 1 \\ \cdot & \cdot & \cdot & 1 & \cdot & \cdot & \cdot & \cdot & 1 & \cdot & \cdot & 1 & 1 \\ \hline \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & 1 & 1 & 1 & 1 & 1 \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & 1 & 1 & 1 & 1 & 1 \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & 1 & 1 & 1 & 1 & 1 \\ \hline \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & 1 & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & 1 & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & 1 & \cdot \\ \hline \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \end{array} \right)$$

(e) Result index

Figure: Kronecker CFPQ brief example

- Flow2Vec: связь анализа кода, CFPQ, и методов машинного обучения
- CFPQ для всех путей
- Вычисления на GPU
- Что с этим делать?

- Почта: egororachyov@gmail.com
- Материалы презентации:
 - ▶ Flow2Vec: Value-Flow-Based Precise Code Embedding, Yulei Sui, Xiao Cheng, Guanqin Zhang, Haoyu Wang, ссылка: <https://dl.acm.org/doi/10.1145/3428301>
 - ▶ Context-Free Path Querying with Single-Path Semantics by Matrix Multiplication, Arseniy Terekhov, Artyom Khoroshev, Rustam Azimov, Semyon Grigorev, ссылка: <https://dl.acm.org/doi/10.1145/3398682.3399163>
 - ▶ Context-Free Path Querying by Kronecker Product, Egor Orachev, Ilya Epelbaum, Rustam Azimov, Semyon Grigorev, ссылка: https://link.springer.com/chapter/10.1007/978-3-030-54832-2_6