Семинар

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

Предметная область

```
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'а

Структура презентации

- Code2vec: метод и его описание
- Построение embedding'а для ориентированного графа
- Запросы с контекстно-свободными (КС) ограничениями
- Flow2vec: метод и его описание
- Flow2vec: практическое применение
- Наши исследования в области вычисления КС запросов
- Дальнейшее направление исследований

Code2vec: Code Embedding

- Code2vec[1]: построение code embeddings для представление фрагментов кода в виде числовых векторов фиксированной длины
- Идея:
  1. Представить фрагмент кода в виде абстрактного ситнаксического дерева
  2. Сделать случайную выборку путей из дерева
  3. Получить векторное представление путей
  4. Суммировать полученные представления с весами и получить финальный вектор
  5. Использовать данный вектор для анализа программы
  6. Выполнять шаги 3 и 4 одновременно

[1] Code2vec: learning distributed representations of code, Uri Alon, Meital Zilberstein, Omer Levy, and Eran Yahav, https://doi.org/10.1145/3290353

Code2vec: Мотивационный пример

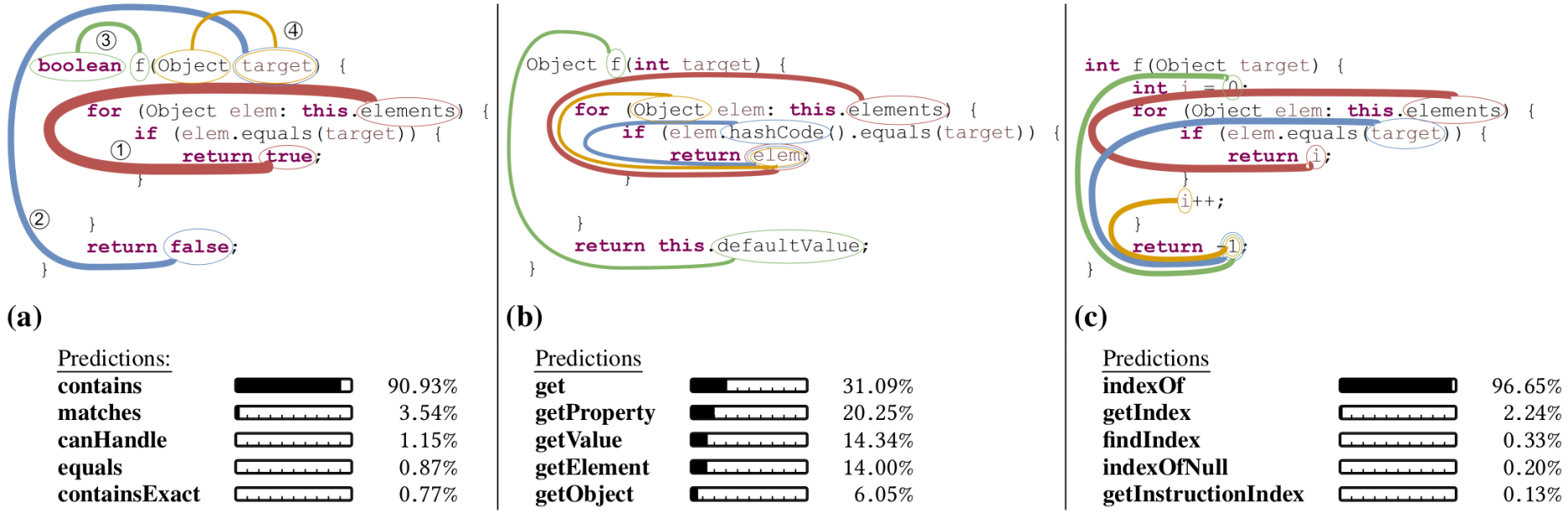


Figure: An example for three methods that albeit having have a similar syntactic structure can be easily distinguished by our model; our model successfully captures the subtle differences between them and manages to predict meaningful names. Each method portrays the top-4 paths that were given the most attention by the model. The widths of the colored paths are proportional to the attention that each path was given.

Code2vec: Терминология

- Абстрактное синтаксическое дерево (AST) $\mathcal{C} = \langle N, T, X, s, \delta, \phi \rangle$, $\delta : N \to (N \cup T)^*$, $\phi : T \to X$
- Путь в дереве $p = n_1 d_1 ... n_k d_k n_{k+1}$, $n_1, n_{k+1} \in T$, $n_2, ..., n_k \in N$, $d_i \in \{\uparrow, \downarrow\}$
- Путь-контекст $\langle x_s, p, x_t \rangle$, $x_s, x_t \in X$, $x_s = \phi(start(p))$, $x_t = \phi(end(p))$, для $x = 7$ путь-контекст $\langle x, NameExpr \uparrow AssignExpr \downarrow IntegerLiteralExpr, 7 \rangle$
- Множество меток Y, P множество AST путей, извлекаемых из датасета

Code2vec: Модель (1)

- Что будем обучать:
  - embedding путей $path\_vocab \in \mathbb{R}^{|P| \times d}$
  - embedding значений $value\_vocab \in \mathbb{R}^{|X| \times d}$
  - полносвязный слой $W \in \mathbb{R}^{d \times 3d}$
  - вектор внимания $\mathbf{a} \in \mathbb{R}^{d}$
  - embedding меток $tags\_vocab \in \mathbb{R}^{|Y| \times d}$
- Параметр $d \in \mathbb{N}$ - размерность embedding'а, подбирается эмпирически

Code2vec: Модель (2)

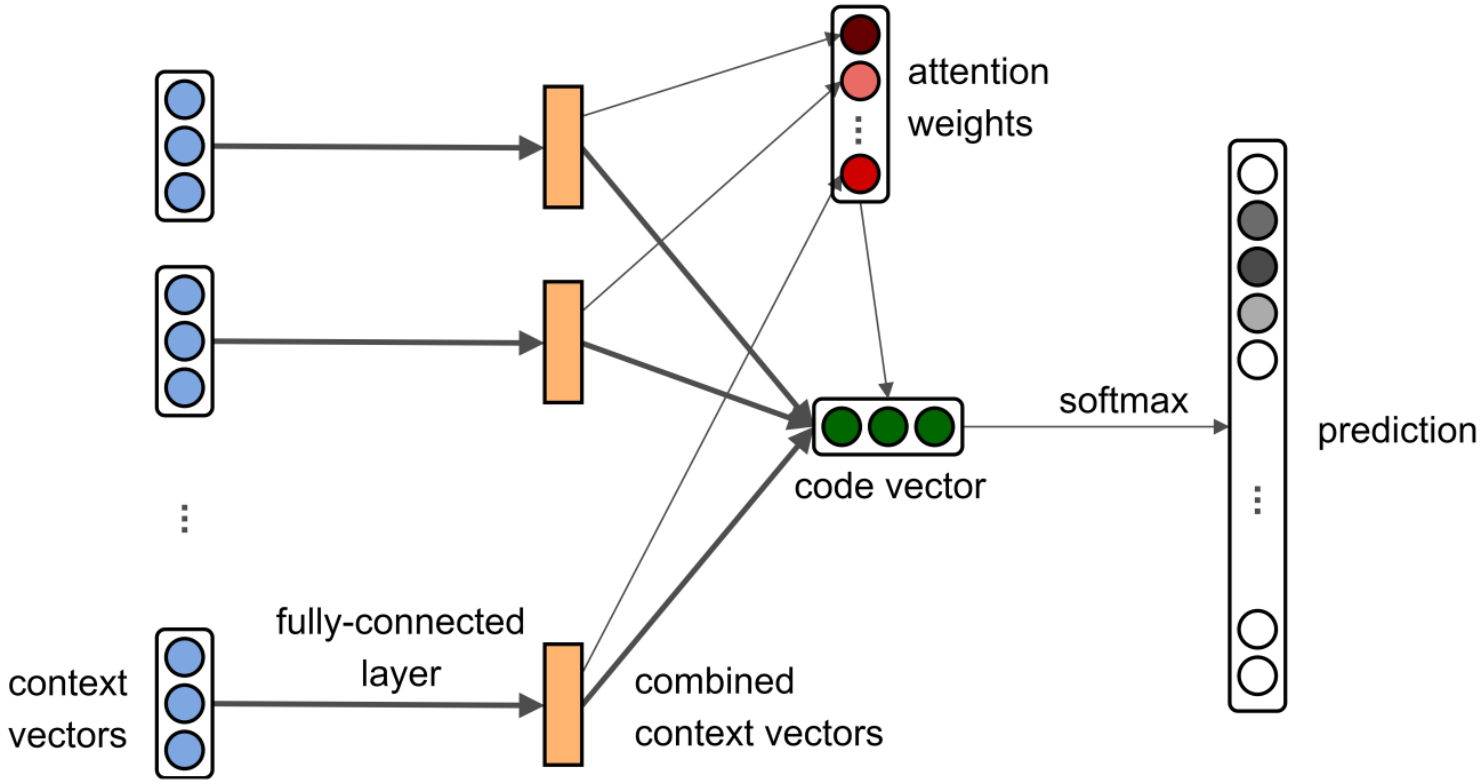


Figure: The architecture of our path-attention network. A full-connected layer learns to combine embeddings of each path-contexts with itself; attention weights are learned using the combined context vectors, and used to compute a code vector. The code vector is used to predicts the label.

Code2vec: Code as Bag of Path-Contexts

- Множество всех путей P
- Фрагмент кода $C = \langle N, T, X, s, \delta, \phi \rangle$
- $T_{pairs} = \{(t_i, t_j) \mid t_i, t_j \in termNodes(C), i \neq j\}$
- $Rep = \{\langle x_s, p, x_t \rangle \mid p \in P$ и $\exists (t_i, t_j) \in T_{pairs}$: $start(p) = x_s, end(p) = x_t$, где $x_s = \phi(t_i), x_t = \phi(t_j)\}$
- Мешок путей-контекстов $\mathcal{B} = \{b_1, ..., b_n\}$, $b_i \in Rep$

Code2vec: Path-Attention Model

- Контекстный вектор $c_i = embedding(\langle x_s, p_j, x_t \rangle) = [value\_vocab_s, path\_vocab_j, value\_vocab_t] \in \mathbb{R}^{3d}$
- Комбинированный контекстный вектор $\tilde{c_i} = tanh(W * c_i) \in \mathbb{R}^d$
- Веса внимания $\alpha_i = \frac{exp({c_i}^T \cdot \mathbf{a})}{\sum_{j=1}^{n} exp({c_j}^T \cdot \mathbf{a})}$
- Вектор фрагмента кода $\mathbf{v} = \sum_{i=1}^{n} \alpha_i \tilde{c_i}$
- Предсказание $y_i \in Y, q(y) = \frac{exp(\mathbf{v}^T \cdot tags\_vocab_i)}{\sum_{i=1}^{|Y|} exp(\mathbf{v}^T \cdot tags\_vocab_j)}$

Code2vec: Резюме

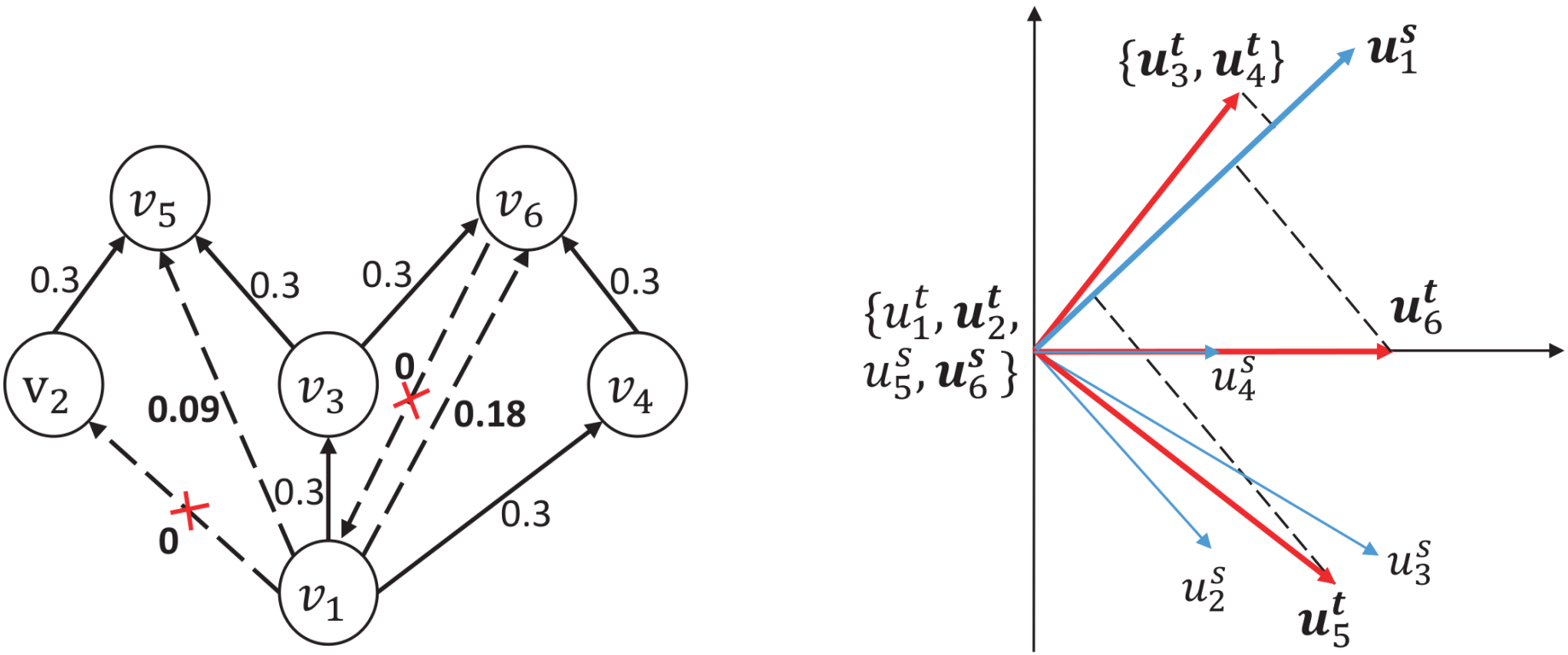
- Можем представить пути в AST дереве фрагмента кода в виде набора векторов
- Используя взвешенную сумму, можем представить весь фрагмент кода как вектор
- Полученный вектор можем использовать в дальнейшем для анализа

HOPE: Graph Embedding

- Построение представления графа в векторном пространстве выбранной размерности
  - Вершины графа — это вектора
  - Можно использовать в задачах реконтсрукции графа, рекомендации соседей, предстказания связей и т.д.
- Проблемы:
  - Необходимо сохранить "важные" свойства графа
  - Реальные графы является ориентированными и обладают ассиметричной транзитивностью
- Решение: Алгоритм High-Order Proximity preserved Embedding (HOPE)[2]

[2] Asymmetric Transitivity Preserving Graph Embedding, Mingdong Ou, Peng Cui, Jian Pei, Ziwei Zhang, and Wenwu Zhu, https://doi.org/10.1145/2939672.2939751

HOPE: Идея



(a) Directed graph

(b) Graph embedding

Figure: HOPE: Asymmetric Transitivity Preserving Graph Embedding

HOPE: Постановка задачи

- Ориентированный граф $G = \langle V, E \rangle$, $V = \{v_1, ..., v_N\}$, $|V| = N$, $e_{ij} = (v_1, v_2) \in E$
- Матрица смежности графа $A \in \mathbb{R}^{N \times N}$, 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 $U = [U^s, U^t]$, $U^s, U^t \in \mathbb{R}^{N \times K}$, $K \in \mathbb{N}$ - размерность emdedding'a, вектора u_i^s, u_i^t соответсвуют вершине графа v_i
- **Хотим приблизить матрицу S оптимально в следующем смысле $min\|S - U^s * U^{t\,T}\|_F^2$, где $\| \bullet \|_F$ - норма Фробениуса**

HOPE: High-order Proximity Matrix

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

HOPE: Approximation of High-Order Proximity

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

HOPE: Резюме

- 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 оценками ошибки приближения. Значение ошибки можно уменьшить, увеличив значение параметра K

Context-free Path Querying

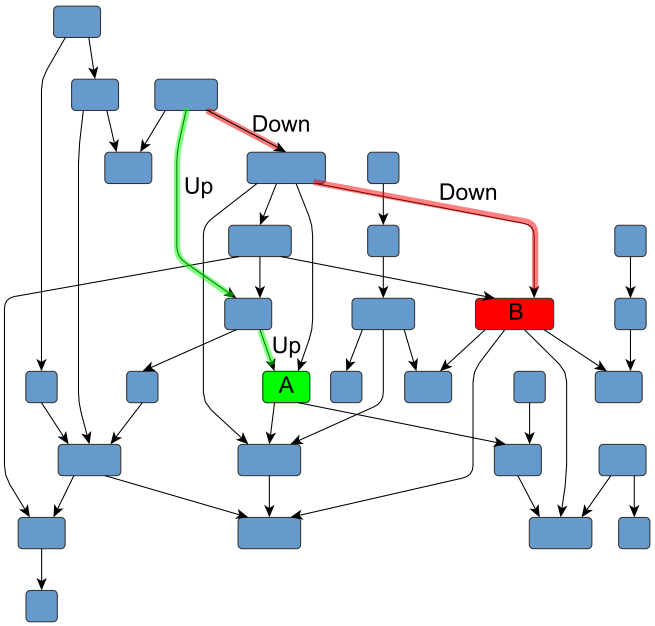


Figure: Example of a graph

Навигация в графе:

- Находятся ли вершины A и B на одном уровне иерархии?
- Существует ли путь вида $\mathbf{Up}^n\,\mathbf{Down}^n$?
- Найти все такие пути $\mathbf{Up}^n\,\mathbf{Down}^n$, которые начинаются в вершине A

Context-free Path Querying: Применение

- Графовые базы данных[3]
- Анализ RDF данных[4]
- Биоинформатика[5]
- Статический анализ кода[6]

---

[3] Querying Graph Databases, Pablo Barceló Baeza, https://doi.org/10.1145/2463664.2465216

[4] Context-Free Path Queries on RDF Graphs, Xiaowang Zhang, Zhiyong Feng, Xin Wang et al., https://arxiv.org/abs/1506.00743

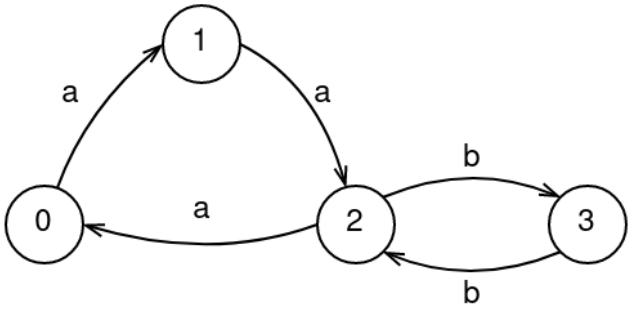
[5] Quantifying variances in comparative RNA secondary structure prediction, James Anderson, Adám Novák, Zsuzsanna Sükösd et al., https://bmcbioinformatics.biomedcentral.com/articles/10.1186/1471-2105-14-149

[6] Fast Algorithms for Dyck-CFL-Reachability with Applications to Alias Analysis, Qirun Zhang, Michael R. Lyu, Hao Yuan, Zhendong Su, https://doi.org/10.1145/2499370.2462159

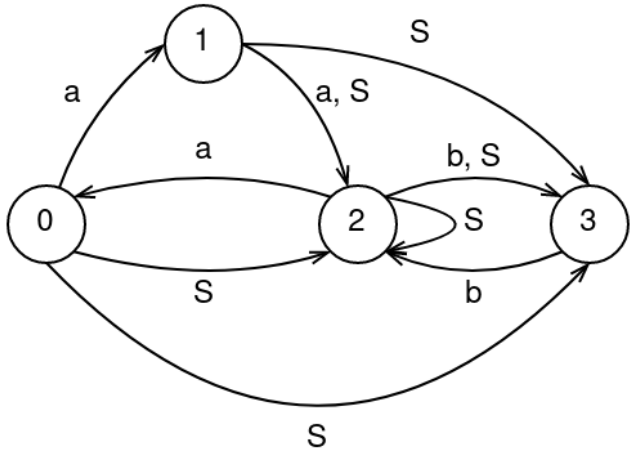
Context-free Path Querying: Терминология

- Ориентированный граф с меткаи $\mathcal{G} = \langle V, E, L \rangle$
- $\omega(\pi) = \omega(v_0 \xrightarrow{l_0} v_1 \xrightarrow{l_1} \cdots \xrightarrow{l_{n-2}} v_{n-1} \xrightarrow{l_{n-1}} v_n) = l_0 l_1 \cdots l_{n-1}$
- КС Грамматика $G = \langle \Sigma, N, P, S \rangle$
- Язык $L(G) = \{w \mid S \to^*_G w\}$
- Семантика достижимости: $R = \{(u, v) \mid \exists u \pi v : \omega(\pi) \in L\}$
- Семантика всех путей: $\Pi = \{u \pi v \mid \omega(\pi) \in L\}$

Context-free Path Querying: Пример



(a) Input graph



(b) Result graph

Figure: CFPQ Example

- Грамматика $S \to aSb \mid ab$
- Результат (Семантика достижимости): ребра с меткой S
- Результат (Семантика всех путей): $\{1 \xrightarrow{a} 3, 0 \xrightarrow{a} 1 \xrightarrow{a} 2 \xrightarrow{b} 3 \xrightarrow{b} 2, 1 \xrightarrow{a} 2 \xrightarrow{a} 0 \xrightarrow{a} 1 \xrightarrow{a} 2 \xrightarrow{b} 3 \xrightarrow{b} 2 \xrightarrow{b} 3 \xrightarrow{b} 2, \ldots\}$

Context-free Path Querying: Существующие алгоритмы

- Алгоритмы, основанные на различных техниках парсинга (CYK, LL, LR, etc.)
- **Алгоритмы, основанные на линейной алгебре**
  - Матричный алгоритм Рустама Азимова[7]
    - семантика: достижимости, одного пути, всех путей
    - платформы: CPU, GPU
  - Алгоритм на основе пересечения графа и рекурсивного автомата через произведения Кронекера[8]
    - семантика: достижимости, одного пути, всех путей
    - платформы: CPU, GPU

---

[7] Context-Free Path Querying with Single-Path Semantics by Matrix Multiplication, Arseniy Terekhov, Artyom Khoroshev, Rustam Azimov, Semyon Grigorev, https://dl.acm.org/doi/10.1145/3398682.3399163

[8] Context-Free Path Querying by Kronecker Product, Egor Orachev, Ilya Epelbaum, Rustam Azimov, Semyon Grigorev, https://link.springer.com/chapter/10.1007/978-3-030-54832-2_6

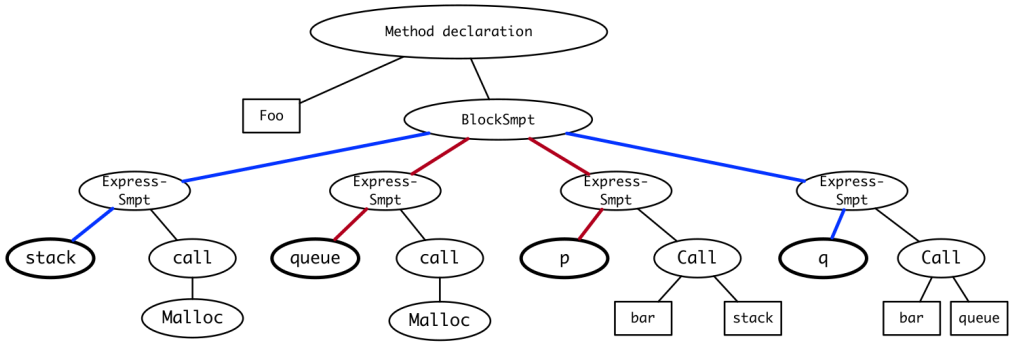
Context-free Path Querying: Резюме

- Для анализа графа можем задавать запросы с КС ограничениями
- Для каждой конкретной задачи можем выбирать различную семантику запроса
- Имеем ээфективные алгоритмы для вычисления запросов на двух основных вычислительных платформах

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

Недостатки существующих интсрументов:

- Не учитывют межпроцедурное взаимодействие
- Не учитывют псевдонимы (ссылки)
- Не учитывют ассиметричную транзитивность программ

Flow2Vec: Value-Flow-Based Precise Code Embedding

- Flow2vec[9]: построение code embeddings для представления фрагмента кода как части всей программы с учетом межпроцедурных потоков данных и множества вызовов функций
- Идея:
  1. Построить промежуточное представление программы в терминах простых операций
  2. Построить граф потоков данных
  3. Используя КС запросы вычислить количество потоков данных между каждым выражение в программе (в ее промежуточном представлении)
  4. Построить матрицу близости высокого порядка
  5. Вычислить на ее основе embedding для графа программы

[9] Flow2Vec: Value-Flow-Based Precise Code Embedding, Yulei Sui, Xiao Cheng, Guanqin Zhang, Haoyu Wang, https://dl.acm.org/doi/10.1145/3428301

Flow2Vec: Обзор

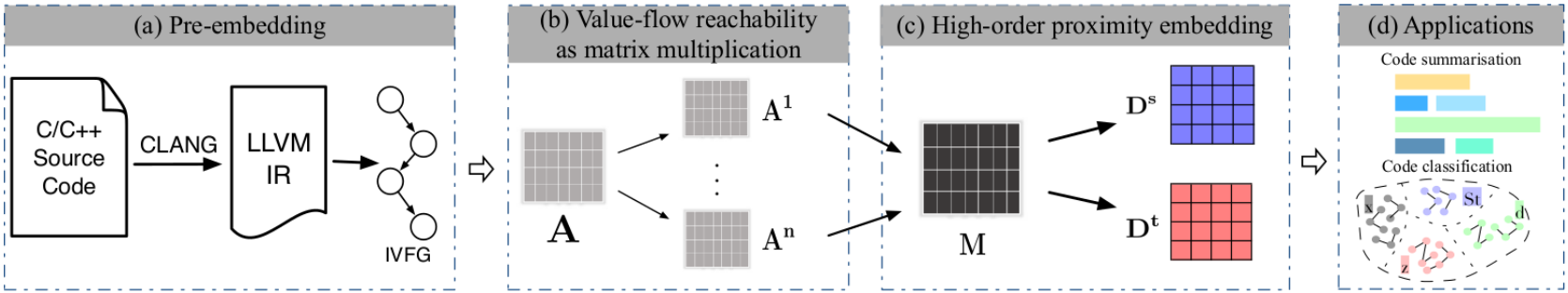


Figure: Overview of the approach

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

Flow2vec: Pre-embedding (LLVM-IR)

- LLVM-IR в качестве промежуочного представления
- $\mathcal{V} = \mathcal{O} \cup \mathcal{P}$, два типа переменных:
  - $\mathcal{O}$ address-taken objects
  - $\mathcal{P}$ top-level variables
- Конвертация в SSA (Static single assignment form)
- Пять типов инструкций: $p = \&o$, $p = q$, $p = \&q \rightarrow f_i$, $p = *q$, $*p = q$, $p, q \in \mathcal{P}, o \in \mathcal{O}$

Flow2vec: Pre-embedding (IVFG)

- Построение Interprocedural value-flow graph (IVFG) $\langle N, E \rangle$ на основе LLVM-IR программы, где $t \xrightarrow{v} t', v \in \mathcal{V}$ def-use отношение, $t \xrightarrow{p} t', p \in \mathcal{P}$ direct value-flow отношение
- Построение
  1. Анализ указателей (points-to информация)
  2. Аннотирование: $a = \chi(a)$, $\mu(a)$
  3. Преобразование аннотаций: $a = \chi(a) \implies$ *def-use*, $\mu(a) \implies$ *use*
  4. Построение def-use цепей для переменных $a \in \mathcal{O}$

Flow2vec: Pre-embedding (Example)

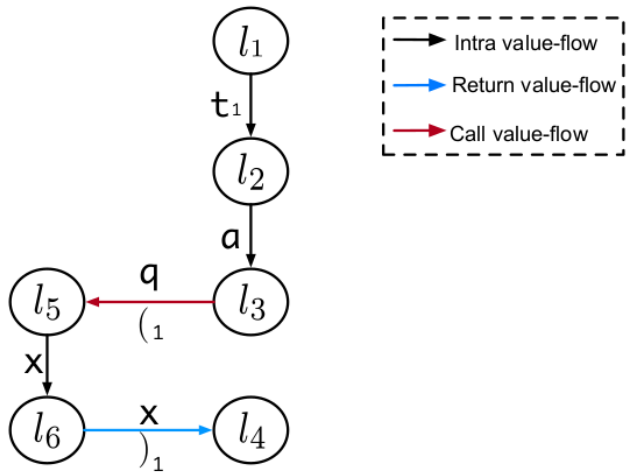
```
foo(){
p = &a;
a = &b;
q = *p;
r=bar(q);
}

bar(x)
{
  return x;
}
```

(a) Source code

```
   foo(){
   p = &a;        //p points to &a
l1 :t = &b;       //t points to &b
l2 :*p = t;       //a1= χ(a0)
l3 :q = *p;       //μ(a1)
l4 :r = bar(q);   // cs1
   }
l5 :bar(x)
   {
l6 :    return x;
   }
```

(b) LLVM-IR



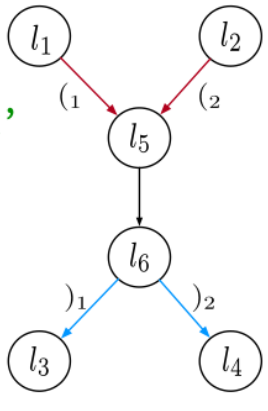
(c) IVFG

Figure: A C code fragment and its LLVM instructions and interprocedural value-flow graph (IVFG)

Flow2vec: Pre-embedding (Symbolic Matrix)

```
   foo(){
l1 :    stack = malloc(..);
l2 :    queue = malloc(..);
l3 :    p = bar(stack);//cs1
l4 :    q = bar(queue);//cs2
        //operations on 'stack'
         via pointer 'p'
        . . .
   }

l5 :bar(x){
   //initialization for
   objects that 'x' points to
l6 :    return x;
   }
```



IVFG

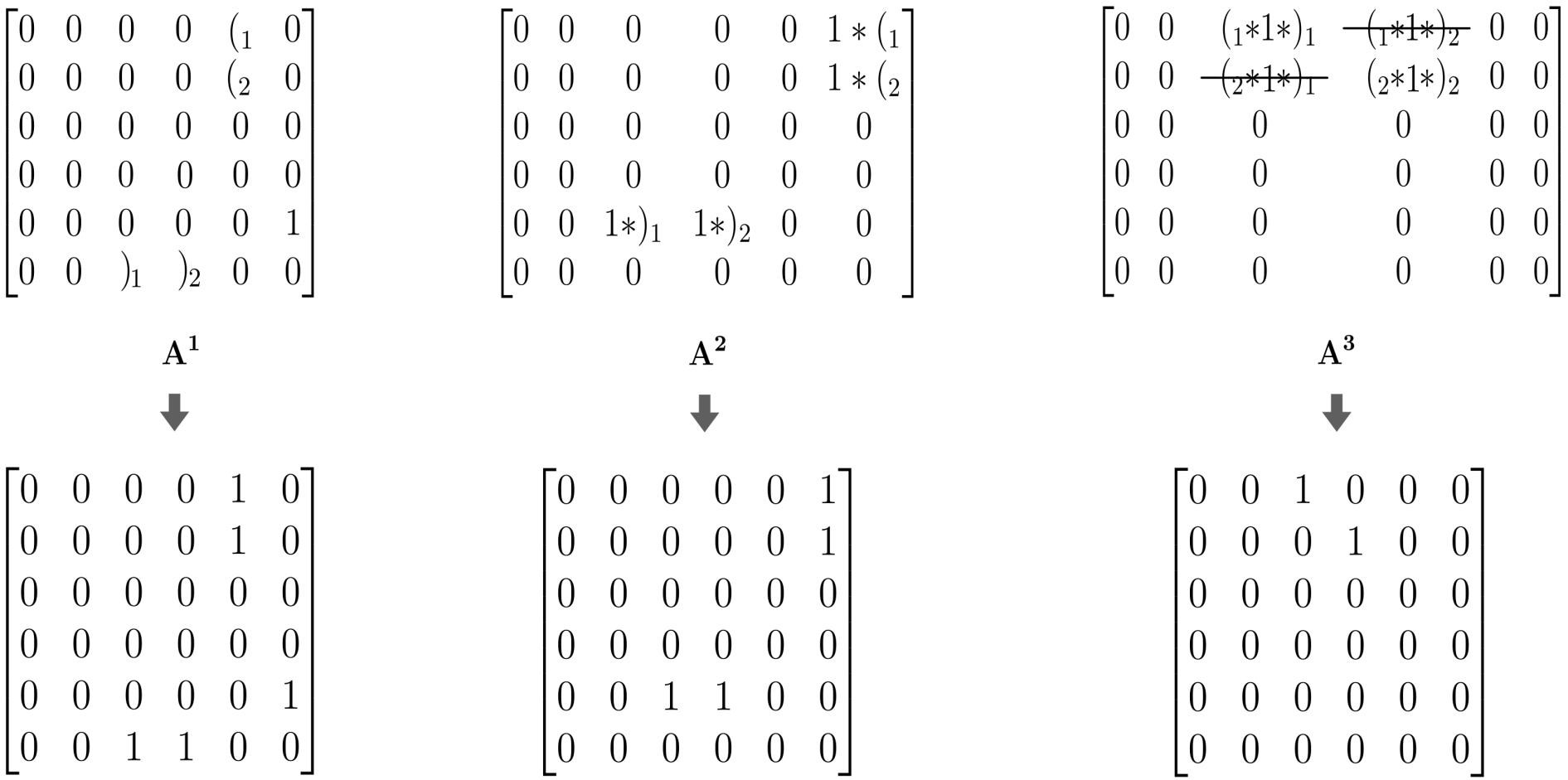
(a) Code fragment and IVFG

$$
\begin{array}{c} \\ l_1 \\ l_2 \\ l_3 \\ l_4 \\ l_5 \\ l_6 \end{array}
\begin{array}{c}
\begin{array}{cccccc} l_1 & l_2 & l_3 & l_4 & l_5 & l_6 \end{array} \\
\left[\begin{array}{cccccc}
0 & 0 & 0 & 0 & (_1 & 0 \\
0 & 0 & 0 & 0 & (_2 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 &)_1 &)_2 & 0 & 0
\end{array}\right]
\end{array}
$$

(b) Call/return and value-flow matrix A

Figure: Pre-embedding example. $A_{ij} = 1$ if there is intraprocedural value-flow between l_i and l_j. $A_{ij} = (_{csId}$, $A_{ij} =)_{csId}$ for interprocedural value-flow between l_i and l_j.

Flow2vec: Value-flow reachability via matrix multiplication



(a) Matrix A^1 (b) Matrix A^2 (c) Matrix A^3

Figure: Context-sensitive value-flow reachability. A^h_{ij} is the number of value-flow paths between l_i and l_j of length h.

Flow2vec: 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

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

$$\left[\begin{array}{ccc:ccc} -0.51 & -0.49 & -0.69 & 0 & 0 & 0 \\ 0.51 & -0.49 & -0.69 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0.33 & -0.79 & 0 & 0 & 0 \\ 0 & 0.71 & -0.58 & 0 & 0 & 0 \end{array}\right] \quad \left[\begin{array}{ccc:ccc} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ -0.51 & 0.49 & -0.69 & 0 & 0 & 0 \\ 0.51 & 0.49 & -0.69 & 0 & 0 & 0 \\ 0 & -0.71 & -0.58 & 0 & 0 & 0 \\ 0 & -0.33 & -0.79 & 0 & 0 & 0 \end{array}\right]$$

$\mathbf{D^s}$ $\mathbf{D^t}$

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

Figure: Embedding step

Flow2vec: Application scenarios

Souce code (A)

```
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;
}
```

Souce code (B)

```
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;
}
```

Ground truth	resize to the minimal	Ground truth	parse and execute command
Code2Vec	isMinimal	Code2Vec	check
Code2Seq	Expand size	Code2Seq	parse
Flow2Vec	reset minimal memory	Flow2Vec	parse command and execute

Figure: Code summarization example

- Code classification
- Code summarization

Flow2vec: Code classification

- Задача: предсказать метки по фрагменту кода
- Цель обучения: предсказывать вероятность $P(y_i \mid x)$
- Контекстный вектор $c_i = embedding(\langle x_i, vfp_n, x_j \rangle) = [value\_vocab_s, d_i^s \cdot {d_j^t}^T, value\_vocab_t] \in \mathbb{R}^{2d+1}$
- Комбинированный контекстный вектор $\tilde{c_i} = tanh(W * c_i) \in \mathbb{R}^d$, $W \in \mathbb{R}^{d \times 2d+1}$
- Веса внимания $\alpha_i = \frac{exp({c_i}^T \cdot \mathbf{a})}{\sum_{j=1}^{n} exp({c_j}^T \cdot \mathbf{a})}$
- Вектор фрагмента кода $\mathbf{v} = \sum_{i=1}^{n} \alpha_i \tilde{c_i}$
- Предсказание $y_i \in Y, q(y) = \frac{exp(\mathbf{v}^T \cdot tags\_vocab_i)}{\sum_{i=1}^{|Y|} exp(\mathbf{v}^T \cdot tags\_vocab_j)}$, Y множество меток

Flow2vec: Code summarization

- Задача: преобразовать фрагмент кода в последовательность слов
- Цель обучения: предсказание вероятност $P(y_1, ..., y_m \mid x)$, Y множество слов
- $P(y_1, ..., y_m \mid x) = \Pi_{t=1}^{m} P(y_t \mid y_1, ..., y_{t-1}, v)$
- Итеративно предсказываем каждое новое слово y_t с учетом уже предсказанных $y_1, ..., y_{t-1}$
- С учетом механизма привлечения внимания к путям

Flow2vec: Резюме

- Используя LLVM-IR, SSA форму и анализ указателей, можем строить IVFG программы
- Используя запросы с KC ограничениями, можем строить матрицу близости высоского порядка, которая учитывает ассиметричные потоки анных как внутри, так и между различных процедур
- Используя результаты работы HOPE, можем построить embedding программы
- Используя path-attention механизм работы code2vec, можем решать задачи обобщения и классификации кода

Заключение

- Code2vec: статический анализ кода с построением embedding'а кода с использованием path-attention механизма на основе методов машинного обучения
- HOPE: построение embedding'а графа с сохранением ассиметричной транзитивности
- CFPQ: применение КС запросов для анализа графа
- Flow2Vec: объединение идей Code2vec, HOPE, CFPQ для построения графа потока данных программы, поиска потоков данных между всеми вершинами, построения embedding'а, и использования этого embedding'а вместе с path-attention механизмом в прикладных задачах

Дополнительно

- Почта: egororachyov@gmail.com
- Материалы презентации:
  - Flow2Vec: Value-Flow-Based Precise Code Embedding, Yulei Sui, Xiao Cheng, Guanqin Zhang, Haoyu Wang, https://dl.acm.org/doi/10.1145/3428301
  - Code2vec: learning distributed representations of code, Uri Alon, Meital Zilberstein, Omer Levy, and Eran Yahav, https://doi.org/10.1145/3290353
  - Asymmetric Transitivity Preserving Graph Embedding, Mingdong Ou, Peng Cui, Jian Pei, Ziwei Zhang, and Wenwu Zhu, https://doi.org/10.1145/2939672.2939751
  - Context-Free Path Querying with Single-Path Semantics by Matrix Multiplication, Arseniy Terekhov, Artyom Khoroshev, Rustam Azimov, Semyon Grigorev, https://dl.acm.org/doi/10.1145/3398682.3399163
  - Context-Free Path Querying by Kronecker Product, Egor Orachev, Ilya Epelbaum, Rustam Azimov, Semyon Grigorev, https://link.springer.com/chapter/10.1007/978-3-030-54832-2_6