Теория автоматов и формальных языков

Применение запросов с контекстно-свободными ограничениями для решения прикладных задач

Автор: Григорьев Семён

Санкт-Петербургский государственный университет

17 декабря 2020

Пути с контекстно-свободными ограничениями (CFPQ)

- Конечный ориентированный граф с метками на рёбрах $\mathcal{G} = (V, E, L)$
- Путь — это слово в алфавите L
  $\omega(p) = \omega(v_0 \xrightarrow{l_0} v_1 \xrightarrow{l_1} \ldots \xrightarrow{l_{n-1}} v_n) = l_0 \cdot l_1 \cdot \ldots \cdot l_{n-1}$
- $\mathcal{L}$ — контекстно-свободный язык (КС язык)
- $G_{\mathcal{L}} = (N, \Sigma, R, S)$

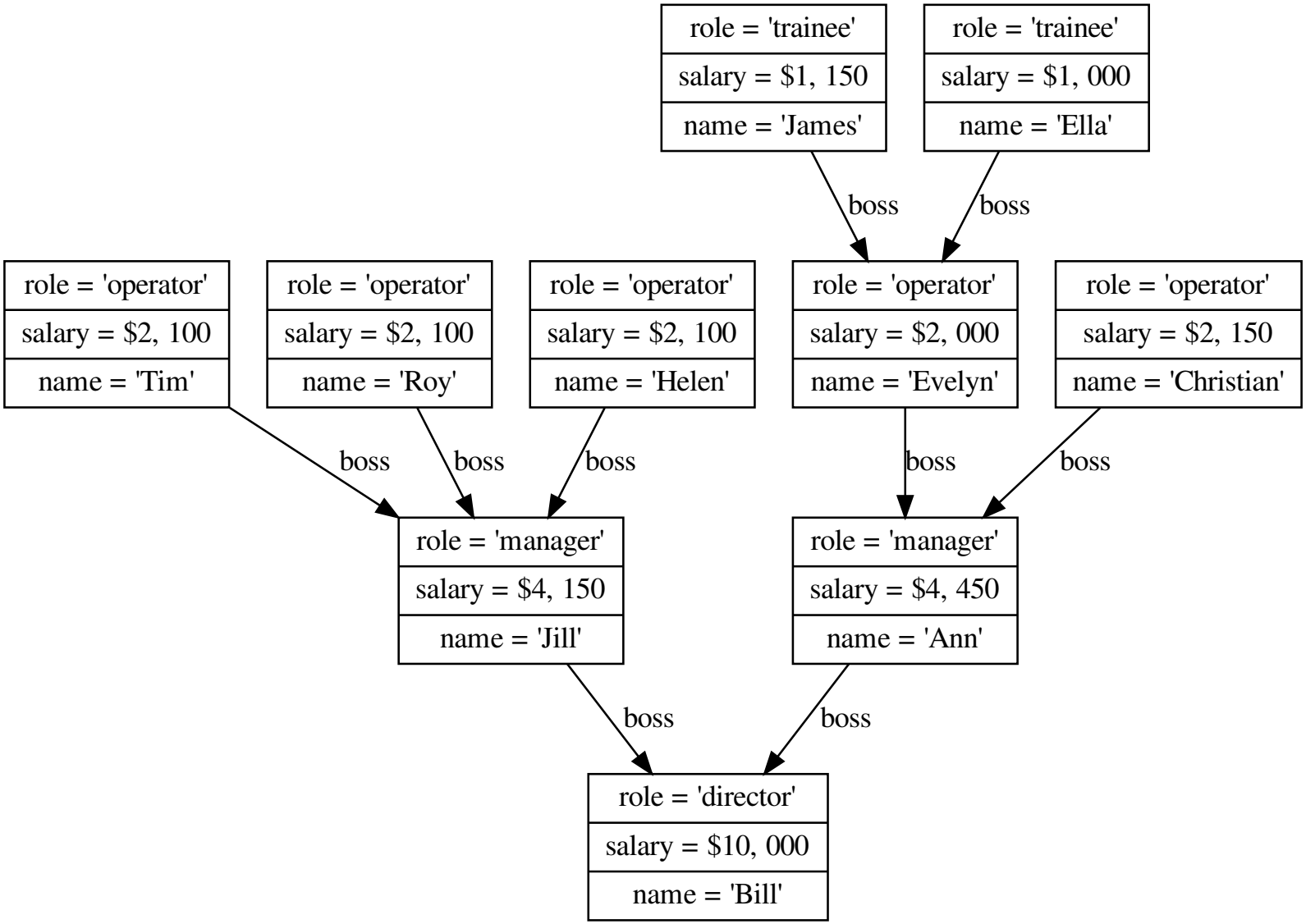
Пути с контекстно-свободными ограничениями (CFPQ)

- Конечный ориентированный граф с метками на рёбрах $\mathcal{G} = (V, E, L)$
- Путь — это слово в алфавите L
  $\omega(p) = \omega(v_0 \xrightarrow{l_0} v_1 \xrightarrow{l_1} \ldots \xrightarrow{l_{n-1}} v_n) = l_0 \cdot l_1 \cdot \ldots \cdot l_{n-1}$
- $\mathcal{L}$ — контекстно-свободный язык (КС язык)
- $G_{\mathcal{L}} = (N, \Sigma, R, S)$
- Задача достижимости: $Q = \{(v_i, v_j) \mid \exists p = v_i \ldots v_j, S \xrightarrow[G_{\mathcal{L}}]{*} \omega(p)\}$
- Задача поиска путей: $Q = \{p \mid S \xrightarrow[G_{\mathcal{L}}]{*} \omega(p)\}$
  - Один путь, все пути, кратчайший путь...

Области применения CFPQ

- Статический анализ кода
  - Межпроцедурный анализ указателей
  - Анализ типов
  - Унификация
  - ...
- Анализ биологических данных
- Обработка грвф-структурированных данных (графовые базы данных)

Пример (База данных)



Пример (Запросы)

- Найти всех, занимающих одинаковое положение в структуре организации, но имеющих разные зарпллаты (на Cypher)

```
PATH PATTERN OnSamePosition  =
  ()-/ :boss> [~OnSamePosition | ()] <:boss /->()
MATCH (a)-/ ~OnSamePosition /->(b)
WHERE a.salary <> b.salary
RETURN a, b
```

Пример (Запросы)

- Найти всех, занимающих одинаковое положение в структуре организации, но имеющих разные зарпллаты (на Cypher)

```
PATH PATTERN OnSamePosition  =
  ()-/ :boss> [~OnSamePosition | ()] <:boss /->()
MATCH (a)-/ ~OnSamePosition /->(b)
WHERE a.salary <> b.salary
RETURN a, b
```

- Кто в структуре ниже Тима, но имеет более высокую зарплату

```
PATH PATTERN OnSamePosition  =
  ()-/ :boss> [~OnSamePosition | ()] <:boss /->()
MATCH
  (a:{name: 'Tim'})-/ ~OnSamePosition /-> ()
   <-[boss*1..]- (b)
WHERE a.salary < b.salary
RETURN b
```

Статический анализ кода

Межпроцедурный анализ кода

- *Thomas Reps et al.* “Precise interprocedural dataflow analysis via graph reachability.” 1995

Статический анализ кода

Межпроцедурный анализ кода

- *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

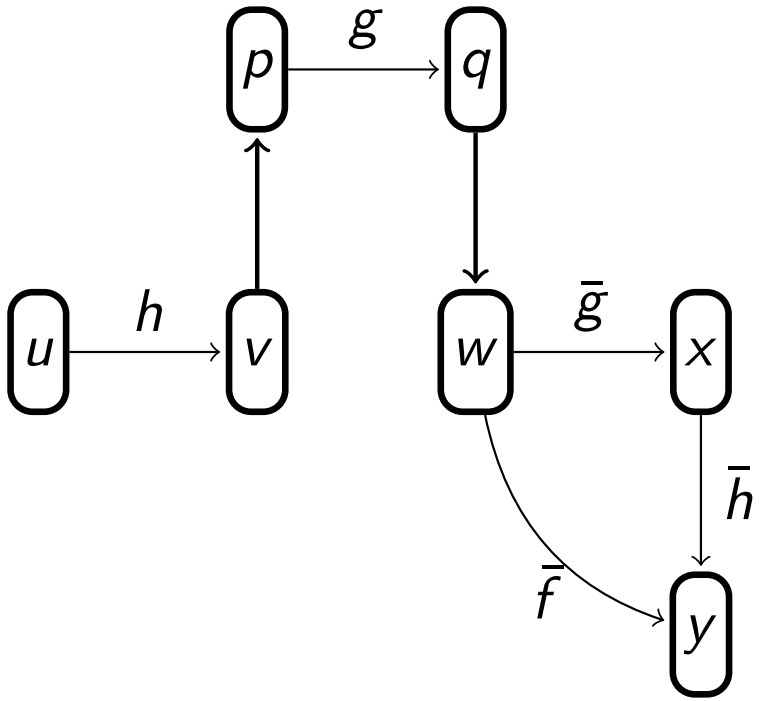
Статический анализ кода

Межпроцедурный анализ кода

- *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
- *Anders Alnor Mathiasen, and Andreas Pavlogiannis.* “The Fine-Grained and Parallel Complexity of Andersen’s Pointer Analysis.“ 2020 (POPL 2021)

Пример

```
v.h = u;
...
p = v;
...
q.g = p;
...
w = q;
...
x = w.g;
if (...) {
  y = w.f;
}
else {
  y = x.h;
}
```



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

Incorrect path: $hg\bar{f}$

Пара слов о практической применимости

- 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

Другие задачи анализа кода

- *Yulei Sui, et al.* “Flow2Vec: value-flow-based precise code embedding.” 2020

[2] Доклад на семинаре:
https://vk.com/ycformallanguagesseminar?w=wall-154415012_63

[3] Доклад на семинаре:
https://vk.com/ycformallanguagesseminar?w=wall-154415012_94

Другие задачи анализа кода

- *Yulei Sui, et al.* “Flow2Vec: value-flow-based precise code embedding.” 2020
- *Osbert Bastani, Saswat Anand, and Alex Aiken.* “Specification inference using context-free language reachability.” 2015
- *Jakob Rehof and Manuel Fahndrich.* “Type-base flow analysis: from polymorphic subtyping to CFL-reachability.” 2001

[2] Доклад на семинаре:
`https://vk.com/ycformallanguagesseminar?w=wall-154415012_63`

[3] Доклад на семинаре:
`https://vk.com/ycformallanguagesseminar?w=wall-154415012_94`

Другие задачи анализа кода

- *Yulei Sui, et al.* “Flow2Vec: value-flow-based precise code embedding.” 2020
- *Osbert Bastani, Saswat Anand, and Alex Aiken.* “Specification inference using context-free language reachability.” 2015
- *Jakob Rehof and Manuel Fahndrich.* “Type-base flow analysis: from polymorphic subtyping to CFL-reachability.” 2001
- *Das, Manuvir, and Thomas Reps.* “BTA termination using CFL-reachability.” 1996[2]

[2] Доклад на семинаре: `https://vk.com/ycformallanguagesseminar?w=wall-154415012_63`

[3] Доклад на семинаре: `https://vk.com/ycformallanguagesseminar?w=wall-154415012_94`

Другие задачи анализа кода

- *Yulei Sui, et al.* "Flow2Vec: value-flow-based precise code embedding." 2020
- *Osbert Bastani, Saswat Anand, and Alex Aiken.* "Specification inference using context-free language reachability." 2015
- *Jakob Rehof and Manuel Fahndrich.* "Type-base flow analysis: from polymorphic subtyping to CFL-reachability." 2001
- *Das, Manuvir, and Thomas Reps.* "BTA termination using CFL-reachability." 1996[2]
- *Venkatesh Choppella, and Christopher T. Haynes.* "Source-tracking unification." 2005[3]

---

[2] Доклад на семинаре:
`https://vk.com/ycformallanguagesseminar?w=wall-154415012_63`

[3] Доклад на семинаре:
`https://vk.com/ycformallanguagesseminar?w=wall-154415012_94`

За пределами КС граммтик

- Пусть $L_1 = D_1(\mathit{deref}, \mathit{ref})$ и $L_2 = D_1(\mathit{call}, \mathit{return})$
- Язык ограничений
  $L_3 = L_1 \odot L_2 = \{\mathit{call\ return}; \mathit{call\ deref\ return\ deref\ ref\ ref}; \dots\}$ — шафл сбалансированных скобочных последовательностей
- L_3 не является контекстно-свободным, но является линейно-конъюктивным

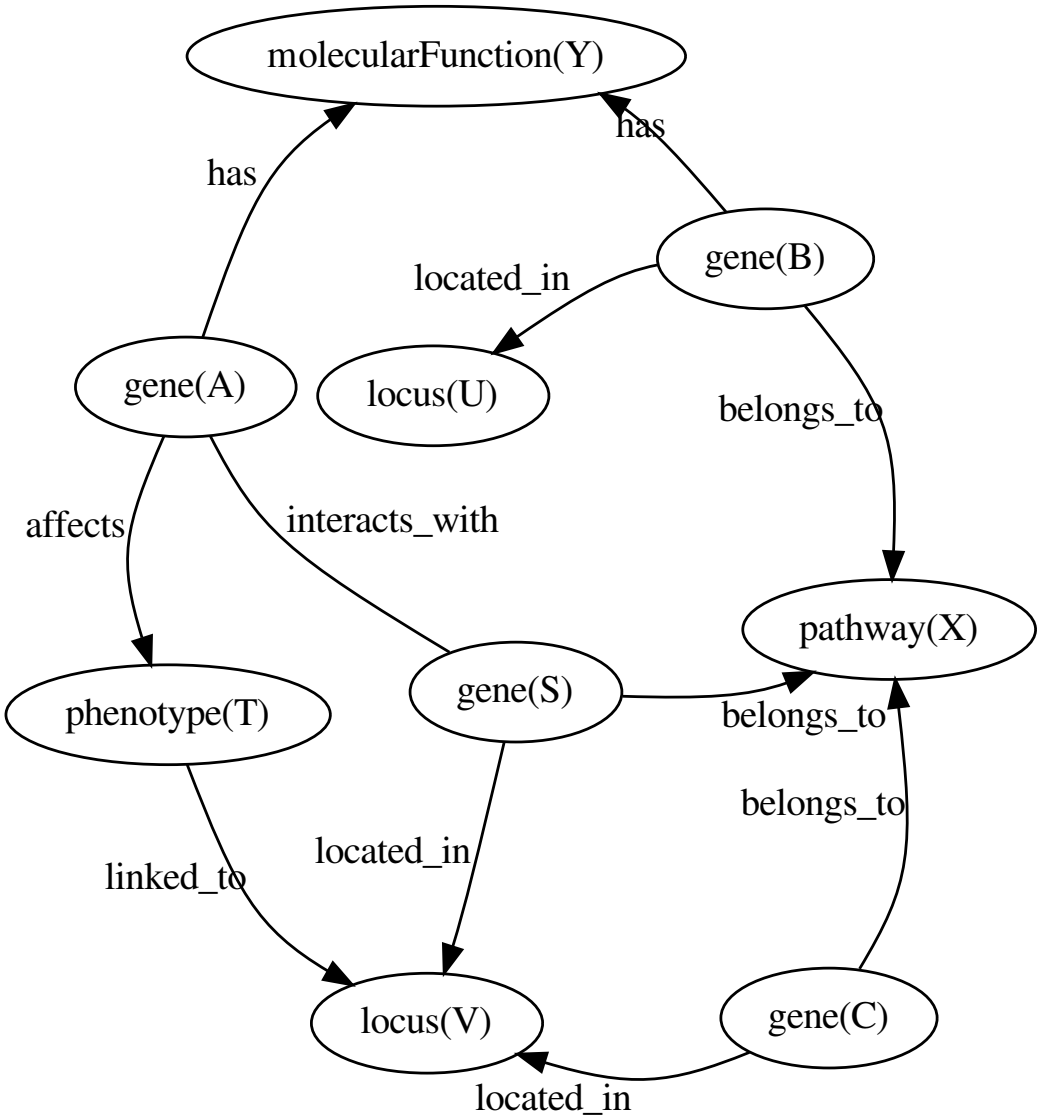
Исследования линейно-конъюнктивных ограничений

- *Qirun Zhang and Zhendong Su.* “Context-sensitive data-dependence analysis via linear conjunctive language reachability.” 2017
- *Yuanbo Li, Qirun Zhang, and Thomas Reps.* “Fast graph simplification for interleaved Dyck-reachability.” 2020.

Графовые базы данных

- *Mihalis Yannakakis.* "Graph-theoretic methods in database theory." 1990
- *Hellings J.* "Conjunctive context-free path queries." 2014
- *J. Kuijpers, G. Fletcher, N. Yakovets, and T. Lindaaker.* "An experimental study of context-free path query evaluation methods." 2019
- *C. M. Medeiros, M. A. Musicante, and U. S. Costa.* "LL-based query answering over rdf databases." 2019

Анализ биологических данных[4]



[4] *Sevon P., Eronen L.* "Subgraph queries by context-free grammars." 2008

Анализ биологических данных (запрос)

```
IS_ASSOCIATED_TO -> -refers_to ARTICLE refers_to

SEQUENCE -> GENE
          | PROTEIN
          | GENE codes_for PROTEIN
          | PROTEIN -codes_for GENE

GENE -> GENE IS_SIMILAR_TO gene | gene
PROTEIN -> PROTEIN IS_SIMILAR_TO protein | protein
PHENOTYPE -> PHENOTYPE IS_SIMILAR_TO phenotype | phenotype

(*for all (edge symbol, vertex non-terminal)-pairs (e, V),
and for all similarity-conferring edge types 'h'
(e.g., is_homologous_to) *)
V -> V IS_SIMILAR_TO v | v
IS_SIMILAR_TO -> e V -e | h
```

Анализ биологических данных (запрос)

```
FROM protein("ACHB3_HUMAN")
TO phenotype("AD")
MATCH SEQUENCE IS_ASSOCIATED_TO PHENOTYPE
```