



# ITGM



Теория формальных языков — это не только написание парсеров

Семён Григорьев

JetBrains Research, лаборатория языковых инструментов  
Санкт-Петербургский государственный университет

17.03.2018

- Анализ графов
  - ▶ Запросы к графовым базам данных
  - ▶ Анализ сетей (социальных, интернет и т.д.)
- Статический анализ программ
  - ▶ Анализ алиасов
  - ▶ Taint analysis
  - ▶ Анализ типов
  - ▶ Статический анализ динамически формируемого кода
- ...

- Алфавит — множество символов  $(\Sigma, N, \dots)$
- Язык — множество “слов”
- Язык  $L$  над алфавитом  $\Sigma$ :  $L(\Sigma) = \{w | w \in \Sigma^*\}$
- Классы языков
  - ▶ Регулярные: регулярные выражения (“академические”), конечные автоматы
  - ▶ Контекстно-свободные
  - ▶ ...
- У разных классов разная выразительная сила
  - ▶ Язык “правильных” скобочных последовательностей является контекстно-свободным, но не является регулярным

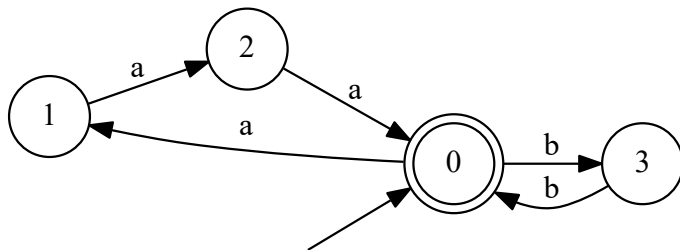
Регулярные языки

$\iff$  регулярные выражения

$\iff$  конечные автоматы

Конечный автомат  $M = (\Sigma, Q, P, S, F)$

- $\Sigma$  — алфавит
- $Q$  — множество состояний
- $P \subseteq Q \times \Sigma \times Q$  — правила перехода из одного состояния в другое
- $S \subseteq Q$  — стартовые состояния
- $F \subseteq Q$  — финальные состояния



- $p = v_0 \xrightarrow{l_0} v_1 \xrightarrow{l_1} \cdots v_{n-1} \xrightarrow{l_{n-1}} v_n$  — путь из стартового состояния в конечное
- $w(p) = w(v_0 \xrightarrow{l_0} v_1 \xrightarrow{l_1} \cdots v_{n-1} \xrightarrow{l_{n-1}} v_n) = l_0 l_1 \cdots l_{n-1}$
- $L(M) = \{w(p) | p \text{ — путь из стартового состояния } M \text{ в конечное}\}$
- $L(M) = \{aaa; bb; aaabb; aaabbbaabbbb, \dots\}$

# Контекстно-свободные языки

Контекстно-свободная грамматика  $\mathbb{G} = (\Sigma, N, P, S)$

- $\Sigma$  — терминальный алфавит
- $N$  — нетерминальный алфавит
- $P$  — правила вывода
- $S \in N$  — стартовый нетерминал

Контекстно свободная грамматика для языка  $L = \{a^n b^n \mid n \geq 1\}$  с явным выделением “середины”

0 :  $S \rightarrow a S b$

1 :  $S \rightarrow \textit{Middle}$

2 :  $\textit{Middle} \rightarrow a b$

Грамматика как правила переписывания:

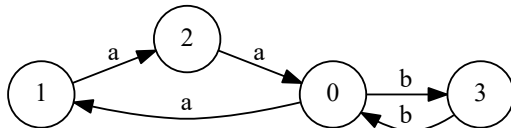
- $S \xrightarrow{1} \textit{Middle} \xrightarrow{2} ab$
- $S \xrightarrow{0} a S b \xrightarrow{0} aa S bb \xrightarrow{1} aa \textit{Middle} bb \xrightarrow{2} aaabbb$

# Поиск путей с контекстно-свободными ограничениями

- $\mathbb{G} = (\Sigma, N, P)$  — контекстно-свободная грамматика
- $G = (V, E, L)$  — ориентированный граф,  $E \subseteq V \times L \times V$ ,  $L \subseteq \Sigma$
- $p = v_0 \xrightarrow{l_0} v_1 \xrightarrow{l_1} \dots v_{n-1} \xrightarrow{l_{n-1}} v_n$  — путь в графе  $G$
- $w(p) = w(v_0 \xrightarrow{l_0} v_1 \xrightarrow{l_1} \dots v_{n-1} \xrightarrow{l_{n-1}} v_n) = l_0 l_1 \dots l_{n-1}$
- $R = \{p \mid \exists N_i \in N (w(p) \in L(\mathbb{G}, N_i))\}$ 
  - ▶ Стартовый нетерминал можно зафиксировать заранее
  - ▶ **Проблема:** множество  $R$  может быть бесконечным
- Задачу можно сформулировать иначе:
$$Q = \{(v_0, N_i, v_n) \mid \exists N_i \in N, \exists p = v_0 \xrightarrow{l_0} \dots \xrightarrow{l_{n-1}} v_n (w(p) \in L(\mathbb{G}, N_i))\}$$

## Пример

Входной граф



Запрос — грамматика  $G$  для языка  $L = \{a^n b^n \mid n \geq 1\}$  с явным выделением середины пути

0 :  $S \rightarrow a S b$

1 :  $S \rightarrow \text{Middle}$

2 :  $\text{Middle} \rightarrow a b$

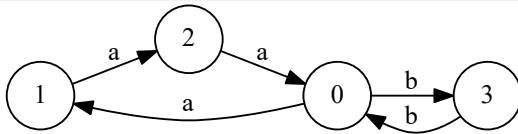
Ответ — бесконечное множество путей

•  $p_1 = 0 \xrightarrow{a} 1 \xrightarrow{a} 2 \xrightarrow{a} 0 \xrightarrow{b} 3 \xrightarrow{b} 0 \xrightarrow{b} 3$

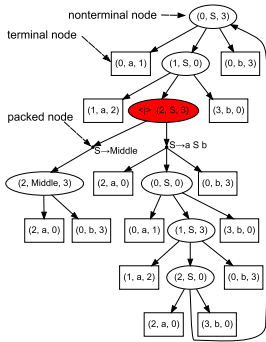
•  $p_2 = 0 \xrightarrow{a} 1 \xrightarrow{a} 2 \xrightarrow{a} 0 \xrightarrow{a} 1 \xrightarrow{a} 2 \xrightarrow{a} 0 \xrightarrow{b} 3 \xrightarrow{b} 0 \xrightarrow{b} 3 \xrightarrow{b} 0 \xrightarrow{b} 3 \xrightarrow{b} 0$



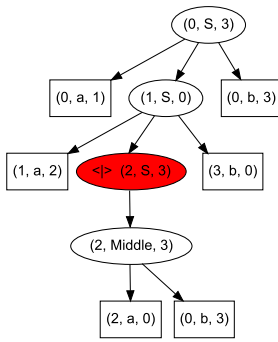
## Структурное представление результата запроса



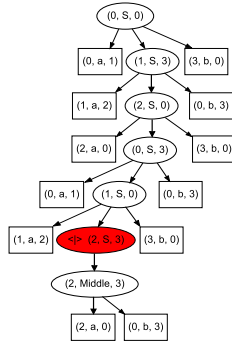
## Входной граф



### Результат (SPPF)

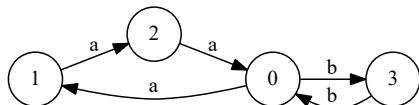


Дерево вывода пути  $p_1$

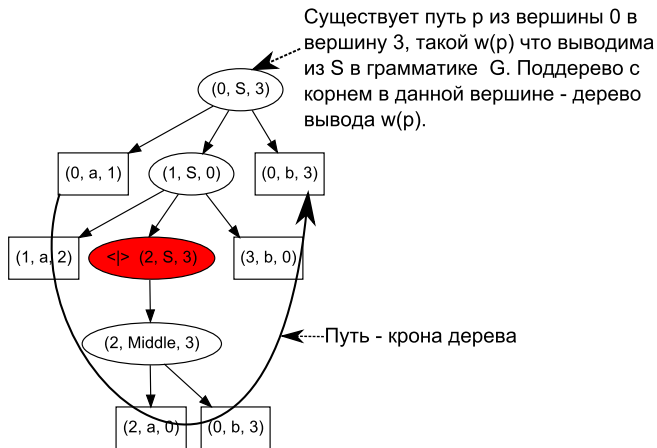


Дерево вывода пути  $p_2$

## Пример: извлечение путей



Путь:  $0 \xrightarrow{a} 1 \xrightarrow{a} 2 \xrightarrow{a} 0 \xrightarrow{b} 3 \xrightarrow{b} 0 \xrightarrow{b} 3$

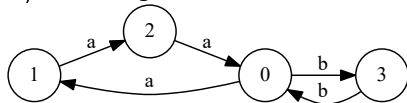


# Почему это работает

Замкнутость КС языков относительно пересечения с регуляными

# Почему это работает

Замкнутость КС языков относительно пересечения с регуляными



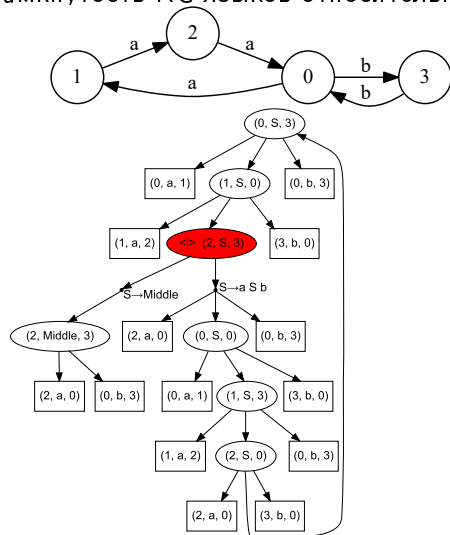
0 :  $S \rightarrow a S b$

1 :  $S \rightarrow \textit{Middle}$

2 :  $\textit{Middle} \rightarrow a b$

# Почему это работает

Замкнутость КС языков относительно пересечения с регуляными



0 :  $S \rightarrow a S b$

1 :  $S \rightarrow \text{Middle}$

2 :  $\text{Middle} \rightarrow a b$

$(0, S, 3) \rightarrow (0, a, 1) (1, S, 0) (0, b, 3)$

$(1, S, 0) \rightarrow (1, a, 2) (2, S, 3) (3, b, 0)$

$(2, S, 3) \rightarrow (2, a, 0) (0, S, 0) (0, b, 3)$

$(2, S, 3) \rightarrow (2, \text{Middle}, 3)$

$(0, S, 0) \rightarrow (0, a, 1) (1, S, 3) (3, b, 0)$

$(1, S, 3) \rightarrow (1, a, 2) (2, S, 0) (0, b, 3)$

$(2, S, 0) \rightarrow (2, a, 0) (0, S, 3) (3, b, 0)$

$(0, \text{Middle}, 3) \rightarrow (2, a, 0) (0, b, 3)$

- Графовые базы данных и семантические сети
  - ▶ Same-generation query (и модификации), similarity query (и модификации)
  - ▶ *Sevon P., Eronen L.* “Subgraph queries by context-free grammars.” 2008
  - ▶ *Zhang X. et al.* “Context-free path queries on RDF graphs.” 2016
  - ▶ *Hellings J.* “Conjunctive context-free path queries.” 2014
- Статический 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
  - ▶ *Jakob Rehof and Manuel Fahndrich.* “Type-base flow analysis: from polymorphic subtyping to CFL-reachability.” 2001

- Попытки расширить OpenCypher: <https://goo.gl/5h5a8P>
  - ▶ Иногда даже похоже на контекстно-свободные запросы: <https://goo.gl/VEpsH5>
- PGQL (Property Graph Query Language) поддерживает регулярные ограничения, но не поддерживает контекстно-свободные

- *Kai Wang et. al.* Graspan: A Single-machine Disk-based Graph System for Interprocedural Static Analyses of Large-scale Systems Code. 2017
  - ▶ “ 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.”



- В данной области существуют открытые проблемы
  - ▶ Например, существует ли алгоритм со сложностью  $O(|V|^{3-\varepsilon})$ ,  $\varepsilon > 0$
- В данной области применимы решения из “классического” синтаксического анализа
  - ▶ Алгоритмы: CYK, (Generalized) LL, (Generalized) LR, Эрли, ...
  - ▶ Техники: комбинаторы, генераторы парсеров, ...
  - ▶ Оптимизации: использование GPGPU, специальные структуры данных (сжатое представление леса разбора, структурированный в виде графа стек), ...
- Из-за существенно бОльших объёмов данных требуются специальные оптимизации (распределённые вычисления, параллельные вычисления, ...)

# Обобщённый LL для выполнения КС запросов к графам

- Основа — обобщённый LL (Generalized GLL, GLL)
  - ▶ *Scott E., Johnstone A.* “GLL parsing”
- Поддерживает произвольные контекстно-свободные грамматики (неоднозначные, леворекурсивные)
- Строит сжатое представление леса разбора (Sharep Packed Parse Forest, SPPF) — конечное представление бесконечного ответа
- *Semyon Grigorev and Anastasiya Ragoza.* “Context-free path querying with structural representation of result.” 2017
- Реализован на F#: <https://github.com/YaccConstructor/YaccConstructor>

Пусть на входе граф  $M = (V, E, L)$ , тогда

- Пространственная сложность предложенного алгоритма  $O(|V|^3 + |E|)$
- Временная сложность предложенного алгоритма  $O\left(|V|^3 * \max_{v \in V} (deg^+(v))\right)$
- Результирующий SPPF имеет размер  $O(|V'|^3 + |E'|)$ , где  $M' = (V', E', L')$  — подграф  $M$ , содержащий только искомые пути

## Экспериментальное исследование: запросы

- 0 :  $\mathbf{S} \rightarrow \text{subClassOf}^{-1} \mathbf{S} \text{ subClassOf}$
- 1 :  $\mathbf{S} \rightarrow \text{type}^{-1} \mathbf{S} \text{ type}$
- 2 :  $\mathbf{S} \rightarrow \text{subClassOf}^{-1} \text{subClassOf}$
- 3 :  $\mathbf{S} \rightarrow \text{type}^{-1} \text{type}$

Грамматика для запроса Query 1

- 0 :  $\mathbf{S} \rightarrow \mathbf{B} \text{ subClassOf}$
- 1 :  $\mathbf{S} \rightarrow \text{subClassOf}$
- 2 :  $\mathbf{B} \rightarrow \text{subClassOf}^{-1} \mathbf{B} \text{ subClassOf}$
- 3 :  $\mathbf{B} \rightarrow \text{subClassOf}^{-1} \text{subClassOf}$

Грамматика для запроса Query 2

## Экспериментальное исследование: результаты

Ontology	#V	#E	Query 1			Query 2	
			CYK <sup>1</sup> (ms)	GLL (ms)	#result	GLL (ms)	#result
skos	144	323	1044	10	810	1	1
generations	129	351	6091	19	2164	1	0
travel	131	397	13971	24	2499	1	63
univ-bench	179	413	20981	25	2540	11	81
people-pets	337	834	82081	89	9472	3	37
atom-primitive	291	685	515285	255	15454	66	122
biomedical- measure-primitive	341	711	420604	261	15156	45	2871
pizza	671	2604	3233587	697	56195	29	1262
wine	733	2450	4075319	819	66572	8	133

<sup>1</sup>Zhang, et al. "Context-free path queries on RDF graphs."

# Использование GPGPU для выполнения КС запросов к графам

- Отправная точка — синтаксический анализ линейного входа через перемножение матриц
  - ▶ *Valiant L.* “General context-free recognition in less than cubic time.” 1974
- Основан на матричных операциях — позволяет использовать GPGPU
  - ▶ Можно использовать разреженное представление и готовые библиотеки для работы с ним
- Применим для других классов грамматик (например, конъюнктивных)
- *Rustam Azimov, Semyon Grigorev.* “Context-Free Path Querying by Matrix Multiplication.” 2017
- Реализован на F#: <https://github.com/YaccConstructor/YaccConstructor>

- Использует GPGPU
  - ▶ .NET(F#) + GPGPU (Managed.CUDA, Alea.CuBase, Brahma.FSharp)
- Основан на стандартных библиотеках (cuSparse)
  - ▶ Интеграция через Managed.CUDA
  - ▶ Вот зачем на провайдер типов для OpenCL
- Нужны булевы матрицы, а не float/double

Пусть на входе граф  $M = (V, E, L)$  и грамматика  $\mathbb{G} = (N, \Sigma, P)$ , тогда

- Пространственная сложность предложенного алгоритма  $O(|N||V|^2)$
- Временная сложность предложенного алгоритма  $O(|V|^2|N|^3(BMM(|V|) + BMU(|V|)))$ 
  - ▶  $BMM(n)$  — время, необходимое для умножения сложения булевых матриц  $n \times n$
  - ▶  $BMU(n)$  — время, необходимое для поэлементного сложения булевых матриц  $n \times n$



## Экспериментальное исследование: результаты

Ontology	#V	#E	Query 1		Query 2	
			GLL (ms)	GPGPU (ms)	GLL (ms)	GPGPU (ms)
skos	144	323	10	12	1	1
generations	129	351	19	13	1	0
travel	131	397	24	30	1	10
univ-bench	179	413	25	15	11	9
people-pets	337	834	89	32	3	6
atom-primitive	291	685	255	22	66	2
biomedical-measure-primitive	341	711	261	20	45	24
pizza	671	2604	697	24	29	23
wine	733	2450	819	54	8	6
$g_1$	6224	11840	1926	82	167	38
$g_2$	5864	19600	6246	185	46	21
$g_3$	5368	20832	7014	127	393	40

- Разработка эффективных “практичных” алгоритмов и реализация библиотек
  - ▶ Интеграция с существующими языками запросов к графам и графовым БД
  - ▶ Интеграция с существующими интерфейсами к граф-структурированным данным
- Формулировка прикладных задач в терминах КС запросов к графам
  - ▶ Сравнение с “классическими” решениями

- Почта: `semen.grigorev@jetbrains.com`
- GitHub-сообщество YaccConstructor: `https://github.com/YaccConstructor`