JBR Interns 2020

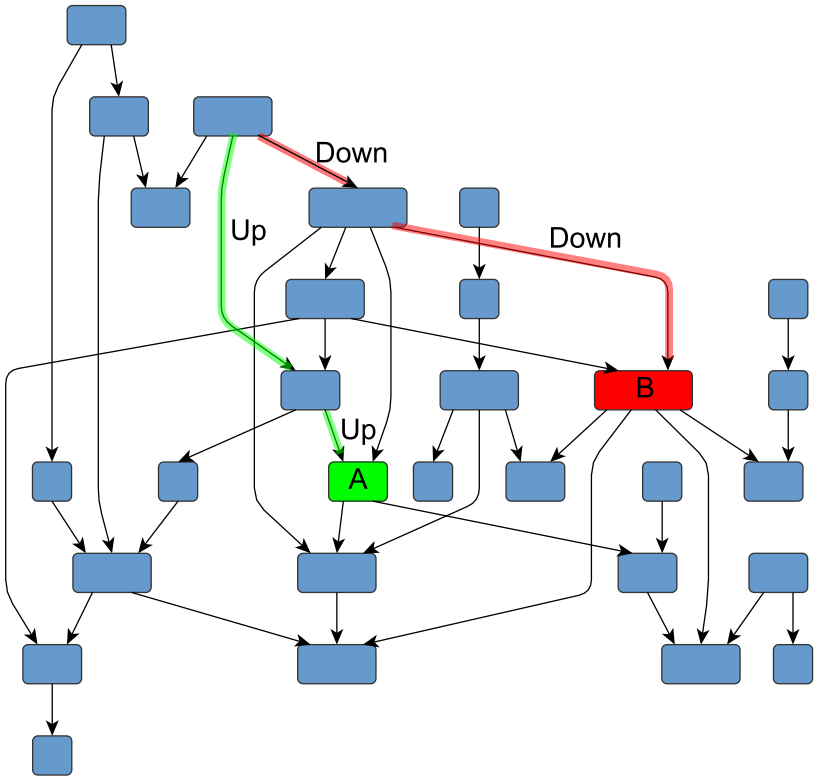
Поиск путей в графе с КС ограничениями через произведение Кронекера

Семен Григорьев, Рустам Азимов, Екатерина Шеметова,
Егор Орачев, **Илья Эпельбаум**

JetBrains Research, Лаборатория языковых инструментов
Санкт-Петербургский Государственный университет

31 Августа 2020

Context-Free Path Querying



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

CFPQ: Семантика запросов

- $\mathbb{G} = (\Sigma, N, P)$ — контекстно-свободная грамматика
  - Σ конечное множество терминалов
  - N конечное множество нетерминалов
  - P конечное множество правил вывода
  - $L(\mathbb{G}, A) = \{\omega \mid A \Rightarrow^* \omega\}, A \in N$
- $G = (V, E, L)$ — ориентированный граф с метками
  - $v \xrightarrow{l} u \in E$
  - $L \subseteq \Sigma$
- $\omega(\pi) = \omega(v_0 \xrightarrow{l_0} v_1 \xrightarrow{l_1} \cdots \xrightarrow{l_{n-2}} v_{n-1} \xrightarrow{l_{n-1}} v_n) = l_0 l_1 \cdots l_{n-1}$
- $R_A = \{(n, m) \mid \exists n \pi m$, такой что $\omega(\pi) \in L(\mathbb{G}, A)\}, A \in N$

Существующие решения

- Решения, основанные на различных техниках парсинга (CYK, LL, LR, etc.)
- Решения, основанные на операциях над матрицами
- Все существующие решения работают только с КС грамматикой в нормальной форме (Нормальная форма Хомского)
- Трансформация требует дополнительного времени на обработку и приводит к *существенному* разрастанию грамматики

Рекурсивные автоматы

- Представляется набором конечных автоматов (компонент) с дополнительными *рекурсивными вызовами*
- Любая КС грамматика может быть представлена рекурсивным автоматом с одной компонентой на каждый нетерминал

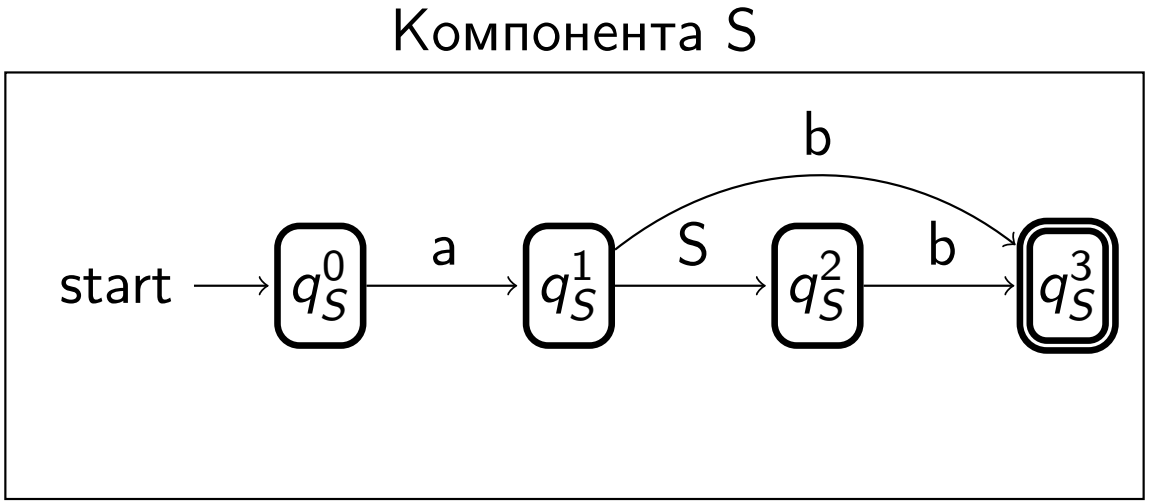
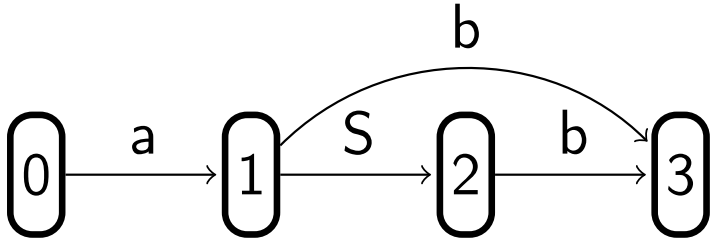


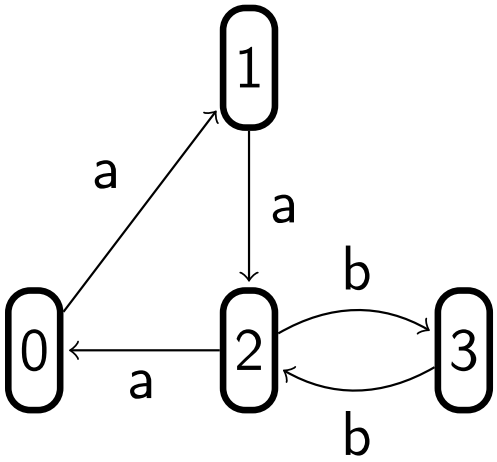
Рис.: Рекурсивный автомат для грамматики вида $S \to aSb \mid ab$

Идея алгоритма

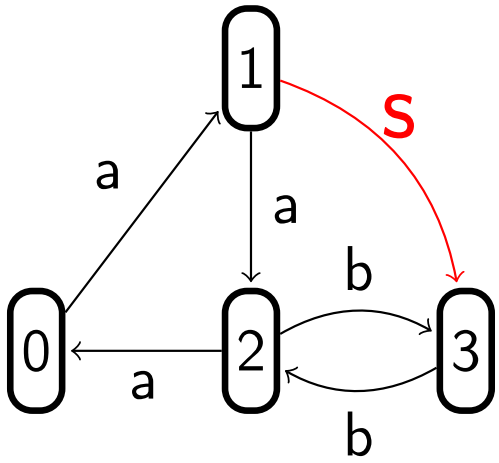
1. Представить грамматику в виде рекурсивного автомата
2. Пересечь рекурсивный автомат и граф используя идею классического алгоритма пересечения двух конечных автоматов, описанного в книге Д.Хопкрофта
3. Использовать транзитивное замыкание, чтобы определить, какие пути выводятся по каким компонентам
4. Добавить найденные ребра с нетерминальными метками в граф
5. Повторять шаги 2 — 4, пока граф меняется

Первая итерация алгоритма

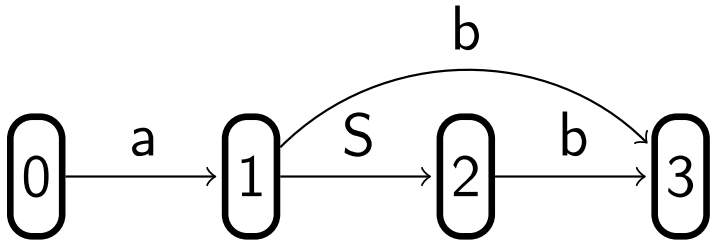


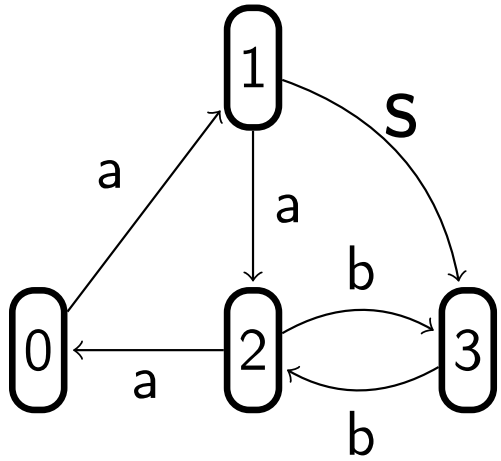


$$
\begin{aligned}
&0,0 \xrightarrow{a} 1,1 \\
&0,\underline{\mathbf{1}} \xrightarrow{\mathbf{a}} 1,2 \xrightarrow{\mathbf{b}} 3,\underline{\mathbf{3}} \\
&0,2 \xrightarrow{a} 1,0 \\
&2,2 \xrightarrow{b} 3,3 \\
&2,3 \xrightarrow{b} 3,2 \\
&1,3 \xrightarrow{b} 3,2
\end{aligned}
$$



Вторая итерация алгоритма





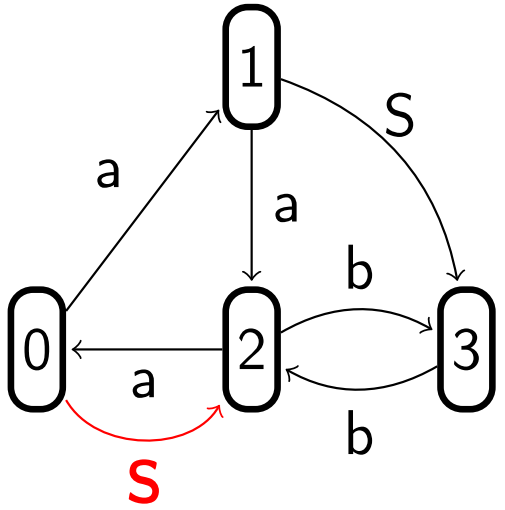
$$0, \underline{\mathbf{0}} \xrightarrow{\mathbf{a}} 1, 1 \xrightarrow{\mathbf{S}} 2, 3 \xrightarrow{\mathbf{b}} 3, \underline{\mathbf{2}}$$

$$0, 1 \xrightarrow{\mathbf{a}} 1, 2 \xrightarrow{\mathbf{b}} 3, 3$$

$$0, 2 \xrightarrow{\mathbf{a}} 1, 0$$

$$2, 2 \xrightarrow{\mathbf{b}} 3, 3$$

$$1, 3 \xrightarrow{\mathbf{b}} 3, 2$$



Произведение Кронекера

- *Пересечение* рекурсивного автомата и графа можно выразить через произведение Кронекера
- В качесве операндов используются матрицы смежности автомата и графа

$$\begin{pmatrix} . & \{a\} & . & . \\ . & . & \{S\} & \{b\} \\ . & . & . & \{b\} \end{pmatrix} \otimes \begin{pmatrix} . & \{a\} & . & . \\ . & . & \{a\} & . \\ \{a\} & . & . & \{b\} \\ . & . & \{b\} & . \end{pmatrix} =$$

	(0,0)	(0,1)	(0,2)	(0,3)	(1,0)	(1,1)	(1,2)	(1,3)	(2,0)	(2,1)	(2,2)	(2,3)	(3,0)	(3,1)	(3,2)	(3,3)
(0,0)	.	.	.	.	.	$\{a\}$	.	.	.	.	.	.	.	.	.	.
(0,1)	.	.	.	.	.	.	**{a}**	.	.	.	.	.	.	.	.	.
(0,2)	.	.	.	.	$\{a\}$	.	.	.	.	.	.	.	.	.	.	.
(0,3)	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
(1,0)	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
(1,1)	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
(1,2)	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	**{b}**
(1,3)	.	.	.	.	.	.	.	.	.	.	.	.	.	.	$\{b\}$	.
(2,0)	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
(2,1)	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
(2,2)	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	$\{b\}$
(2,3)	.	.	.	.	.	.	.	.	.	.	.	.	.	.	$\{b\}$	.
(2,0)	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
(2,1)	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
(2,2)	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
(2,3)	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.

Извлечение путей

- В результате работы алгоритма вычисляется матрица транзитивного замыкания, так называемый *индекс*
- Данная матрица *индекса* может быть использована для извлечения всех путей в графе, которые удовлетворяют достижимости выбранных двух вершин
- Поскольку путей потенциально бесконечное количество, фильтрация и *правильная* нумерация существенны для извлечения

Динамическое транзитивное замыкание

- *Наивная* версия алгоритма предполагает вычисление транзитивного замыкания на каждой итерации
- Транзитивное замыкание *наиболее* сложная часть алгоритма с точки зрения вычислений
- Поскольку транзитивное замыкание *дистрибутивно слева*, мы можем разбить матрицу смежности графа на две части:
  - A добавленные ребра
  - B матрица смежности с прошлой итерации
  - $M_G = A + B$
  - $M_{RSM} \otimes M_G = M_{RSM} \otimes A + M_{RSM} \otimes B$

Реализация

- *Наивная* версия алгоритма без динамического транзитивного замыкания на основе библиотеки **PyGraphBLAS**[1]
- Извлечение путей из *индекса* на основе **PyGraphBLAS**, параметризованное максимальными числом путей для извлечения

[1] PyGraphBLAS — python-обертка для SuiteSparse C — реализации GraphBLAS API для работы с графами в терминах операций линейной алгебры

Замеры

Заключение

- CFQP можно свести к операциям линейной алгебры без трансформации грамматики
- Алгоритм применим для RPQ без значительного *overhead*
- Индекс, построенный в результате работы алгоритма может быть использован для извлечения *всех* путей
- Транзитивное замыкание можно поддерживать за $O(n^3/log\ n)$

Дальнейшие исследования

- Исследование проблемы извлечения всех путей (правильная нумерация путей, временные ограничения)
- Детальное сравнение с классическим матричным алгоритмом
- Исследование вычислительной сложности динамического транзитивного замыкания (возможно ли получить субкубическую сложность)
- Реализация алгоритма на GPGPU с использованием разреженных булевых матриц
- Модификация алгоритма для распределенного вычисления
- Интеграция алгоритма с графовой базой данных (RedisGraph)

Контакты

- Семен Григорьев:
  - s.v.grigoriev@spbu.ru
  - Semen.Grigorev@jetbrains.com
- Рустам Азимов:
  - rustam.azimov19021995@gmail.com
  - Rustam.Azimov@jetbrains.com
- Екатерина Шеметова: katyacyfra@gmail.com
- Егор Орачев: egor.orachev@gmail.com
- Илья Эпельбаум: iliyepelbaun@gmail.com

- Датасет: https://github.com/JetBrains-Research/CFPQ_Data
- Реализация алгоритма: https://github.com/YaccConstructor/RedisGraph