Приложения теории формальных языков и синтаксического анализа

Семён Григорьев

27 августа 2019 г.

Содержание

1 Введение

Одна из классических задач, связанных с анализом графов — это поиск путей в графе. Возможны различные формулировки этой задачи. В некоторых случайх необходимо выяснить, существует ли путь с определёнными свойствами между двумя выбранными вершинами. В других же ситуациях необходимо найти все пути в графе, удовлетворяющие некоторым свойствам.

Так или иначе, на практике часто требуется указать, что интересующие нас пути должны обладать каким-либо специальными свойствами. Иными словами, наложить на пути некоторые ограничения. Например, указать, что искомый путь должен быть простым, кратчайшим, гамильтоновым и так далее.

Один из способов задавать ограничения на пути в графе основан на использовании формальных языков. Базоваое определение языка говорит нам, что язык — это множество слов над некоторым алфавитом. Если рассмотреть граф, рёбра которого помечены символами из алфавита, то путь в графе будет задавать слово: достаточно соединить последовательно символы, лежащие на рёбрах пути. Множество же таких путей будет задавать множество слов или язык. Таким образом, если мы хотим найти некоторое множество путей в графе, то в качестве ограничения можно описать язык, который должно задачать это множество. Иными словами, задача поиска путей может быть сформулирована следующим образом: необходимо найти такие пути в графе, что слова, получаемые конкатенацей меток их рёбер, принадлежат заданному языку. Такой класс задач будем называть задачами поиска путей с ограничениям в теринах формальных языков.

Рассмотрим различные варианты постановки задачи.

Различные алгоритмы решения.

2 Постановка задачи

Пусть $\mathcal{G} = \langle V, E, L \rangle$ — конечный ориентрованный граф с метками на рёбрах, где V — конечное множество вершин, E — конечное множество рёбер, L — конечное множество или алфавит меток. Рёбра будем представлять в виде упорядоченных троек из $V \times L \times V$. Путём π в графе $\mathcal{G}$ будем называть последовательность рёбер такую, что для любых двух последовательных рёбер $e_1 = (u_1, l_1, v_1)$ и $e_2 = (u_2, l_2, v_2)$ в этой последовательности, конечная вершина первого ребра является начальной вершиной второго, то есть $v_1 = u_2$. Будем обобзначать путь из вершины v_0 в вершину v_n как $v_0 \pi v_n = e_0, e_1, \ldots, e_{n-1} = (v_0, l_0, v_1), (v_1, l_1, v_2), \ldots, (v_{n-1}, l_n, v_n)$.

Функция $\omega(\pi) = \omega((v_0, l_0, v_1), (v_1, l_1, v_2), \ldots, (v_{n-1}, l_n, v_n)) = l_0 \cdot l_1 \cdot \ldots \cdot l_n$ строит слово по пути посредством конкатенации меток рёбер вдоль этого пути. Очевидно, для пустого пути данная функция будет возвращать пустое слово, а для пути длины $n > 0$ — непустое слово длины n.

Путь $G = \langle \Sigma, N, P \rangle$ — контекстно-свободная грамматика. Будем считать, что $L \subseteq \Sigma$. Мы не фиксируем стартовый нетерминал в определении грамматики, поэтому, чтобы описать язык, задаваемый ей, нам необходимо отдельно зафиксировать стартовый нетерминал. Таким образом, будем говорить, что $L(G, N_i) = \{w | N_i \overset{*}{\underset{G}{\Rightarrow}} w\}$ — это язык задаваемый граммтикой G со стартовым нетерминалом N_i.

Задача достижимости:

Задача поиска путей:

3 О разрешимости задачи поиска путей с ограничениями в терминах языков

Сведение к задаче о пересечениии с регулярным.

Замкнутость регулярных.

Проверка пустоты.

Замкнутость контекстно-свободных. Проверка пустоты.

Про другие классы языков: конъюнктивные, булевы, многокомпонентные.

4 Алгоритм на матричных опреациях

4.1 Транзитивне замыкание графа

Флойд-Уоршал, матрицы.

Рассуждения про субкубичность. Про то, что булево полукольцо.

4.2 Нормальная форма Хомского

Данный алгоритм накладывает ограничение на форму грамматики: грамматика должна быть в "ослабленной" нормальной форме Хомского.

Определение НФХ.

Любую КС граммтику можно преобразовать в НФХ.

4.3 Алгоритм CYK

Построение и заполнение таблицы.

4.4 Алгоритм для графов на основе CYK

Обобщение. Смотрим на транзитивное замыкание.

4.5 Алгоритм на основе матриц

Ссылка на Валианта [?].

Оригинальные матрицы (Рустам) [1]

Кратчайшие.

Одно дерево?

4.6 Конъюнктивные и булевы граммтики

4.6.1 Определения

Охотин [?].

4.6.2 Для графов

Классическая семантика — работает для конъюнктивных для любых графов. Для Булевых и конъюнктивных только для графов без циклов.

Альтернативная семантика (DataLog). Трактуем конъюнкцию как в даталоге. Тогда всё хорошо. Это похоже на даталог через линейную алгебру [?]

4.7 Особенности реализации матричного алгоритма

Кое-что про наши реализации [6]

Разреженные матрицы, плотные матрицы, GraphBLAS[1], GPGPU, CUTLASS[2].

Расположенеи в памяти: хорошо, когда всё влезло на одну карту.

Распределённые решения. Через GraphBLAS

4.8 Обзор

Китайцы [?], Брэдфорд [?], Ли [?], Хеллингс [?], OpenCypher [4].

Рассуждения про ассимптотику.

Субкубический для частного случая (Брэдфорд) [2].

4.9 Вопросы и задачи

1. !!!

5 Через тензорное произведение

5.1 Рекурсивные автоматы

Определение, примеры.

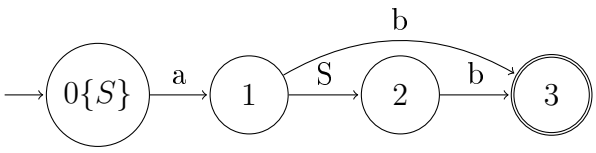

Рис. 1: Рекурсивный автомат для грамматики!!!

[1] !!!

[2] Репозиторий библиотеки CUTLASS: `https://github.com/NVIDIA/cutlass`

5.2 Тензорное произведение

Матриц и графов

Сперва дадим классическое определение тензорного произведения двух неориентированных графов.

Определение 5.1. Пусть даны два графа: $\mathcal{G}_1 = \langle V_1, E_1 \rangle$ и $\mathcal{G}_2 = \langle V_2, E_2 \rangle$. Тензорным произведением этих графов будем называть граф $\mathcal{G}_3 = \langle V_3, E_3 \rangle$, где $V_3 = V_1 \times V_2$, $E_3 = \{((v_1, v_2), (u_1, u_2)) \mid (v_1, u_1) \in E_1 \text{ и } (v_2, u_2) \in E_2\}$.

Иными словами, тензорным произведением двух графов является граф, такой что:

1. множество вершин — это прямое произведение множемтв вершин исходных графов;
2. ребро между вершинами $v = (v_1, v_2)$ и $u = (u_1, u_2)$ существует тогда и только тогда, когда существуют рёбра между парами вершин v_1, u_1 и v_2, u_2 в соответсвующих графах.

Для того, чтобы построить тензорное произведение ориентированных графов, необходимо в предыдущем определении, в условии существования реба в результирующем графе, дополнительно потребовать, чтобы направления рёбер совпадали. Данное требование получается естесвенным образом, если считать, что пары вершин, задающие ребро упорядочены, поэтому формальное определение отличаться не будет.

Нетрудно заметить, что матрица смежности графа $\mathcal{G}_3$ равна тенорному произведению матриц смежностей исходных графов.

Осталось добавить метки к рёбрам. Это приведёт к логичному усилению требованя к существованию ребра: метки рёбер в исходных графах должны совпадать. Таким образом, мы получаем следующее определение тензорного произведения ориентированных графов с метками на рёбрах.

Определение 5.2. Пусть даны два ориентированных графа с метками на рёбрах: $\mathcal{G}_1 = \langle V_1, E_1, L_1 \rangle$ и $\mathcal{G}_2 = \langle V_2, E_2, L_2 \rangle$. Тензорным произведением этих графов будем называть граф $\mathcal{G}_3 = \langle V_3, E_3, L_3 \rangle$, где $V_3 = V_1 \times V_2$, $E_3 = \{((v_1, v_2), l, (u_1, u_2)) \mid (v_1, l, u_1) \in E_1 \text{ и } (v_2, l, u_2) \in E_2\}$, $L_3 = L_1 \cap L_2$.

Рассмотрим пример. В качестве одного из графов возьмём рекурсивный автомат, построенный ранее!!!. Его матрица смежности выглядит следующим образом.

$$M_1 = \begin{pmatrix} . & [a] & . & . \\ . & . & [S] & [b] \\ . & . & . & [b] \\ . & . & . & . \end{pmatrix}$$

Второй граф представлен на изображении 2. Его матрица смежности имеет следующий вид.

$$M_2 = \begin{pmatrix} . & [a] & . & . \\ . & . & [a] & . \\ [a] & . & . & [b] \\ . & . & [b] & . \end{pmatrix}$$

Вычислим $M_1 \otimes M_2$.

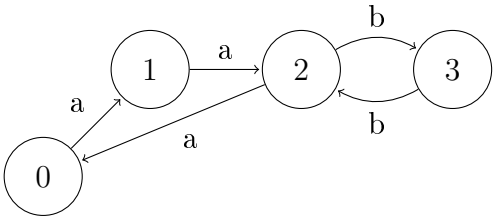

Рис. 2: The input graph

$$M_1 \otimes M_2 = \begin{pmatrix} . & [a] & . & . \\ . & . & [S] & [b] \\ . & . & . & [b] \\ . & . & . & . \end{pmatrix} \otimes \begin{pmatrix} . & [a] & . & . \\ . & . & [a] & . \\ [a] & . & . & [b] \\ . & . & [b] & . \end{pmatrix} = \begin{pmatrix} . & . & . & . & . & [a] & . & . & . & . & . & . & . & . & . & . \\ . & . & . & . & . & . & [a] & . & . & . & . & . & . & . & . & . \\ . & . & . & . & . & [a] & . & . & . & . & . & . & . & . & . & . \\ . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . \\ . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . \\ . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . \\ . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & [b] \\ . & . & . & . & . & . & . & . & . & . & . & . & . & [b] & . & . \\ . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . \\ . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . \\ . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & [b] \\ . & . & . & . & . & . & . & . & . & . & . & . & . & [b] & . & . \\ . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . \\ . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . \\ . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . \\ . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . \end{pmatrix}$$

asdasd

5.3 Алгоритм

По грамматике строим автомат.

В цикле: пересекли через тензорное произведениеб замкнули, чтобы нацти пути из начальной в конечную в граммтике, поставили туда нетерминалы.

Можно вычислять только разницу. Для этого, правда, потребуется держать ещё одну матрицу. И надо проверять, что вычислительно дешевле: поддерживать разницу и потом каждый раз поэлементно складывать две матрицы или каждый раз вычислять полностью произведение.

Всего несколько матриц. Разреженные. Необходимо отметить, что для реальных графов и запросов результат тензорного произведения будет очень разрежен. На готовых либах должно быть быстро.

5.4 Вопросы и задачи

1. !!!
2. !!!

6 Сжатое представление леса разбора

Матричный алгоритм даёт нам ответ на вопрос о достижимости, но не предоставляет самих путей. Что делать, если мы хотим построить все пути, удовлетворяющие ограичениям?

Проблема в том, что множество путей может быть бесконечным. Можем ли мы предложить конечную структуру, однозначно описывающую такое множество? Вспомним, что пересечение контекстно-свободного языка с регулярным — это контекстно-свободный язык. Мы знаем, что конекстно-свободный язык можно описать коньекстно-своюодной граммтикой, которая конечна. Это и есть решение нашего вопроса. Осталось толко научиться строить такую грамматику.

6.1 Дерево вывода

Дерево вывода цепочки в граммтике

- Корневое. Корень помечен стартовым нетерминалом.
- Упорядоченное
- Соотношение узлов и правил
- Крона — цепочка

6.2 Неоднозначные граммтики

Левосторонний и правосторонний выводы.

Существенно неоднозначные языки

6.3 Лес разбора как представление контекстно-свободной грамматики

Добавление информации в классическое дерево разбора. Координаты и промежуточные узлы.

6.4 Вопросы и задачи

1. Постройте дерево вывода цепочки $w = aababb$ в грамматике $G = \langle\{a, b\}, \{S\}, \{S \to \varepsilon \mid a\ S\ b\ S\}, S\rangle$.
2. Постройте все левосторонние выводы цепочки $w = ababab$ в грамматике $G = \langle\{a, b\}, \{S\}, \{S \to \varepsilon \mid a\ S\ b \mid S\ S\}, S\rangle$.
3. Постройте все правосторонние выводы цепочки $w = ababab$ в грамматике $G = \langle\{a, b\}, \{S\}, \{S \to \varepsilon \mid a\ S\ b \mid S\ S\}, S\rangle$.
4. Постройте все деревья вывода цепочки $w = ababab$ в грамматике $G = \langle\{a, b\}, \{S\}, \{S \to \varepsilon \mid a\ S\ b \mid S\ S\}, S\rangle$, соответствующие левосторонним выводам.

5. Постройте все деревья вывода цепочки $w = ababab$ в грамматике $G = \langle\{a, b\}, \{S\}, \{S \to \varepsilon \mid a\ S\ b \mid S\ S\}, S\rangle$, соответствующие правосторонним выводам.
6. Как связаны между собой леса, полученные в предыдущих двух задачах (4 и 5)? Какие выводы можно сделать из такой связи?
7. Постройте сжатое представление леса разбора, полученного в задаче 4.
8. Постройте сжатое представление леса разбора, полученного в задаче 5.
9. Предъявите контекстно-свободную граммтику существенно неоднозначного языка. Возьмите цепочку длины болше пяти, при надлежащую этому языку, и постройте все деревья вывода этой цепочки в предъявленной граммтике.
10. Постройте сжатое представление леса, полученного в задаче 9.

7 Алгоритм на основе восходящего анализа

Наша реализация [?]

7.1 Восходящий синтаксический анализ

Основы LR-анализа.

Таблицы, конфликты.

Пример автомата и таблиц.

7.2 Вопросы и задачи

1. Постройте автомат для грамматики
2. Постройте таблицу для автомата из задачи
3. В том числе дать неоднозначную грамматику
4. Запустить, постоить деревья, стеки и т.д.

8 Алгоритм на основе нисходящего анализа

Комбинаторы [7] GLL [3]

Другие реализации [5]

8.1 Нисходящий синтаксический анализ

9 От CFPQ к вычислению Datalog-запросов

9.1 Datalog

Конечные Эрбрановы модели. Наименьшая неподвижная точка. $C :- d$

9.2 КС-запрос как запрос на Datalog

Покажем, что для данного графа и КС-запроса можно построить эквивалентный запрос на Datalog.

Пусть дан граф Граф преобразуется в набор фактов (базу данных).

Пусть есть грамматика G: $S \to a\ b \mid a\ S\ b$. Она может быть преобразована в запрос следующего вида. $s(X,Y) :- a(X,Z), b(Z,Y)$. $s(X,Y) :- a(X,Z), s(Z,W)b(W,Y)$. $? :- s(X,Y)$

Наблюдения: появились пременные, есть порядок у конъюнктов, который задаёт порядок связывания.

9.3 Обобщение GLL для вычисления Datalog-запросов

Дескриптор — состояние процесса: состояние автомата, результат проделанной работы, подстановка. Задача — найти подстановки. На каждом шаге есть набор подстановок.

Список литературы

[1] R. Azimov and S. Grigorev. Context-free path querying by matrix multiplication. In *Proceedings of the 1st ACM SIGMOD Joint International Workshop on Graph Data Management Experiences & Systems (GRADES) and Network Data Analytics (NDA)*, GRADES-NDA '18, pages 5:1–5:10, New York, NY, USA, 2018. ACM.

[2] P. G. Bradford. Efficient exact paths for dyck and semi-dyck labeled path reachability (extended abstract). In *2017 IEEE 8th Annual Ubiquitous Computing, Electronics and Mobile Communication Conference (UEMCON)*, pages 247–253, Oct 2017.

[3] S. Grigorev and A. Ragozina. Context-free path querying with structural representation of result. In *Proceedings of the 13th Central & Eastern European Software Engineering Conference in Russia*, CEE-SECR '17, pages 10:1–10:7, New York, NY, USA, 2017. ACM.

[4] J. Kuijpers, G. Fletcher, N. Yakovets, and T. Lindaaker. An experimental study of context-free path query evaluation methods. In *Proceedings of the 31st International Conference on Scientific and Statistical Database Management*, SSDBM '19, pages 121–132, New York, NY, USA, 2019. ACM.

[5] C. M. Medeiros, M. A. Musicante, and U. S. Costa. Ll-based query answering over rdf databases. *Journal of Computer Languages*, 51:75 – 87, 2019.

[6] N. Mishin, I. Sokolov, E. Spirin, V. Kutuev, E. Nemchinov, S. Gorbatyuk, and S. Grigorev. Evaluation of the context-free path querying algorithm based on matrix multiplication. In *Proceedings of the 2Nd Joint International Workshop on Graph Data Management Experiences & Systems (GRADES) and Network Data Analytics (NDA)*, GRADES-NDA'19, pages 12:1–12:5, New York, NY, USA, 2019. ACM.

[7] E. Verbitskaia, I. Kirillov, I. Nozkin, and S. Grigorev. Parser combinators for context-free path querying. In *Proceedings of the 9th ACM SIGPLAN International Symposium on Scala*, Scala 2018, pages 13–23, New York, NY, USA, 2018. ACM.