Приложения теории формальных языков и синтаксического анализа

Семён Григорьев

13 сентября 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.

Сперва кратко базовые вещи из теории графов и теории формальных языков.

Далее рассмотрим различные варианты постановки задачи.

И различные алгоритмы решения.

2 Общие сведения теории графов

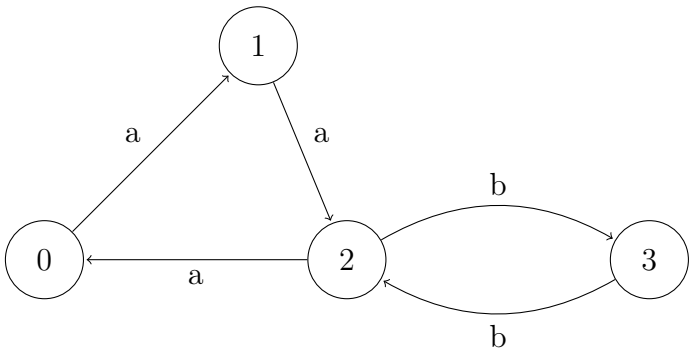
В данном разделе мы дадим определения базовым понятиям из теории графов, рассмотрим несколько классических задач из области анализа графов и алгоритмы их решения. Всё это понадобится нам при последующёй работе.

В дальнейшем нам будут нужны конечные ориентированные помеченные графы. Мы будем использовать термин *граф* подразумевая именно конечный ориентированный помеченный граф, если только не оговорено противное.

Определение 2.1. *Граф* $\mathcal{G} = \langle V, E, L \rangle$, где V — конечное множество вершин, E — конечное множество рёбер, L — конечное множество меток рёбер.

Мы будем считать, что все вершины занумерованы подряд с нуля. То есть можно считать, что V — это отрезок неотрицательный целых чисел.

Пример 2.1. Пример графа и его графического представления. Пусть граф $\mathcal{G}_1 = \langle \{0, 1, 2, 3\}, \{(0, a, 1)$
Графическое представление графа $\mathcal{G}_1$:



Определение 2.2. *Ребро* ориентированного помеченного графа $\mathcal{G} = \langle V, E, L \rangle$ это упорядоченная тройка из $V \times L \times V$.

Пример 2.2. $(0, a, 1)$ и $(3, b, 2)$ — это рёбра графа $\mathcal{G}_1$.При этом, $(3, b, 2)$ $(2, b, 3)$ — это разные рёбра, что видно из рисунка.

Определение 2.3. *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 $v_0 \pi v_n = e_0, e_1, \ldots, e_{n-1} = (v_0, l_0, v_1), (v_1, l_1, v_2), \ldots, (v_{n-1}, l_n, v_n)$.

Пример 2.3. $(0, a, 1)(1, a, 2) = 1\pi_1 2$ — путь из вершины 0 в вершину 2 в графе $\mathcal{G}_1$. При этом, $(0, a, 1)(1, a, 2)(2, b, 3)(3, b, 2) = 1\pi_2 2$ — это тоже путь из вершины 0 в вершину 2 в графе $\mathcal{G}_1$, но он не равен $0\pi_1 2$.

Нам потребуется также отношение, отражающее факт существования пути между двумя вершинами.

Определение 2.4. Отношение достижимости в графе $\mathcal{G} = \langle V, E, L \rangle$: $(v_i, v_j) \in P \iff \exists v_i \pi v_j$.

Отметим, что рефлексивность этого отношения часто зависит от контекста. В некоторых задачах по-умолчанию $(v_i, v_i) \notin P$, а чтобы это было верно, требуется явное наличие ребра-петли.

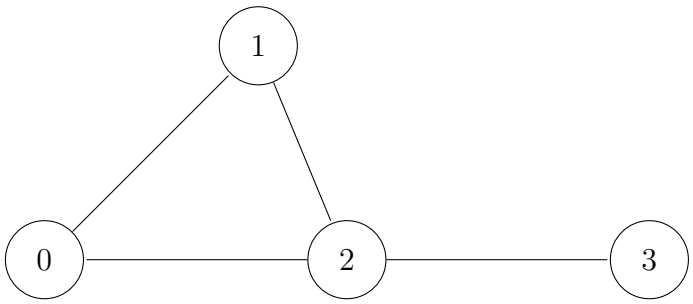
Один из способов задать граф — это задать его *матрицу смежности.*

Определение 2.5. *Матрица смежности* графа $\mathcal{G} = \langle V, E, L \rangle$ — это квадратная матрица M размера $n \times n$, где $|V| = n$ ячейки которой содержат множества. При этом $l \in M[i, j] \iff \exists e = (i, l, j) \in E$.

Заметим, что наше определение матрицы смежности отличается от классического, в котором матрица отражает лишь факт наличия хотя бы одного ребра и, соответственно, является булевой. То есть $M[i, j] = 1 \iff \exists e = (i, \_, j) \in E$. Также можно встретить матрицы смежности в ячейках которых всё же хранится некоторая информация, однако, в единственном экземпляре. То есть запрещены параллельные рёбра. Такой подход часто можно встретить в задачах о кратчайших путях: в этом случае в ячейке хранится расстояние между двумя вершинами. При этом, так как в качестве весов часто рассматривают произвольные (в том числе отрицательные) числа, то в задачах о кратчайших путях отдельно вводят значение "бесконечность" для обозначения ситуации, когда между двумя вершинами нет пути или его длина ещё не известна.

Пример 2.4. Вариации на тему матриц смежности.

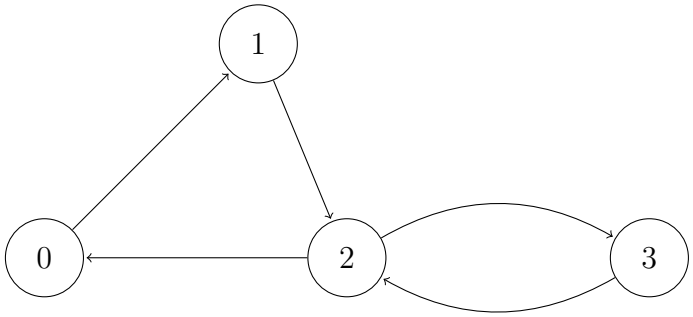
Неориентированный граф:



И его матрица смежности:

$$\begin{pmatrix} 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 \end{pmatrix}$$

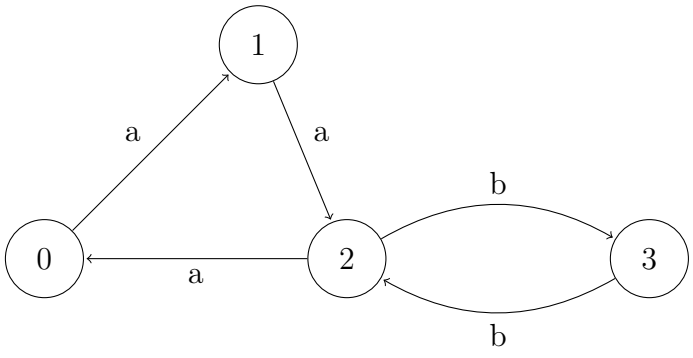
Ориентированный граф:



И его матрица смежности:

$$\begin{pmatrix} 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 \end{pmatrix}$$

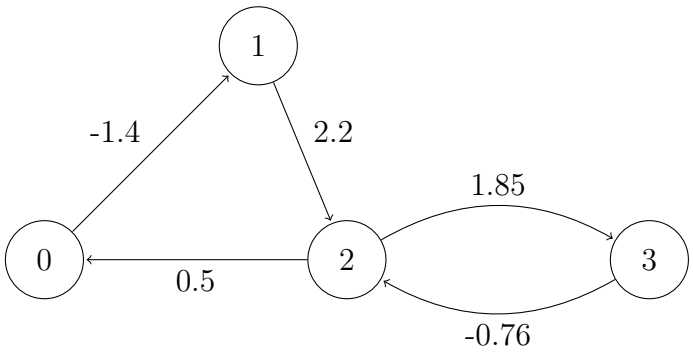
Помеченный граф:



И его матрица смежности:

$$\begin{pmatrix} \varnothing & \{a\} & \varnothing & \varnothing \\ \varnothing & \varnothing & \{a\} & \varnothing \\ \{a\} & \varnothing & \varnothing & \{b\} \\ \varnothing & \varnothing & \{b\} & \varnothing \end{pmatrix}$$

Взвешенный граф для задачи о кратчайших путях:



И его матрица смежности (для задачи о кратчайших путях):

$$\begin{pmatrix} 0 & -1.4 & \infty & \infty \\ \infty & 0 & 2.2 & \infty \\ 0.5 & \infty & 0 & 1.85 \\ \infty & \infty & -0.76 & 0 \end{pmatrix}$$

Мы ввели лишь общие понятия. Специальные понятия, необходимые для иложения конкретного материала, будут даны в соответсвующих главах.

2.1 Задачи поиска путей

Одна из классических задач анализа графов — это задача поиска путей между вершинами с различными ограничениями.

При этом, возможны различные постановки задачи. С одной стороны, по тому, что именно мы хотим получить в качестве результата.

- Наличие пути, удовлетворяющего ограничениям, в графе.
- Наличие пути, удовлетворяющего ограничениям, между некоторыми вершинами: задача достижимости. Достижима ли вершина v_1 из вершины v_2 по пути, удовлетворяющему ограничениям. Такая постановка требует лишь проверить существование, но не обязательно предоставлять путь.
- Поиск одного пути, удовлетворяющего ограничениям. Уже надо предъявлять путь.
- Поиск всех путей. Надо предоставить все пути.

С другой тсороны, задачи различаются ещё и по тому, как фиксируем вершины.

- из одной вершины до всех
- между всеми парами вершин
- межу фиксированной парой вершин
- Между двумя множествами вершин

Итого, можем сгенерировать прямое произведение различных постановок.

Ограничение, имеющее важное прикладное значение, — минимальность длины. Иными словами, важная прикладная задача — поиск кратчайших путей в графе.

Часто добавляют ограничения, что путь должен быть простым и другие.

Список интересных работ по APSP.

2.2 Транзитивне замыкание графа

Заметим, что отношение достижимости (2.4) является тринзитивным. Действительно, если существует путь из v_i в v_j и путь из v_j в v_k, то существует путь из v_i в v_k.

Определение 2.6. *Транзитивным замыканием графа* называется транзитивное замыкание отношения достижимости по всему графу.

Как несложно заметить, транзитивное заамыкание графа — это тоже граф. Более того, если решить задачу поиска кратчайших путей между всеми парами вершин, то мы построим транзитивное замыкание. Потому сразу рассмотрим алгоритм Флойда-Уоршелла, который является общим алгоритмом поиска кратчайших путей (умеет обрабатывать рёбра с отрицательными весами, чего не может, например, алгоритм Дейкстры). При этом, данный алггоритм легко упростить до алгоритма построения транзитивного замыкания.

Listing 1 Алгоритм Флойда-Уоршелла

```
1: function FloydWarshall(𝒢)
2:     M ← матрица смежности 𝒢
3:     n ← |V(𝒢)|
4:     for k = 0; k < n; k++ do
5:         for i = 0; i < n; i++ do
6:             for j = 0; j < n; j++ do
7:                 M[i, j] ← min(M[i, j], M[i, k] + M[k, j])
8:     return M
```

Пример 2.5. Построим транзитивное замыкание

Матрицы.

Про APSP и произведение матриц.

Рассуждения про субкубичность.

Про то, что булево полукольцо.

Интересные ссылки: [8] [26]

2.3 Вопросы и задачи

1. Реализуйте алгоритм построения транзитивного замыкания через матрицы.
2. Реализовать матрицы самим.
3. Взять готовую библиотеку матричных операций: CPU, GPGPU.
4. Реализуйте поиск кратчайших путей через матричные операции.
5. Взять готовую библиотеку матричных операций: CPU, GPGPU.

3 Общие сведения теории формальных языков

В данной главе мы рассмотрим основные понятия из теории формальных языков, которые пригодятся нам в дальнейшем изложении.

Определение 3.1. *Алфавит* — это конечное множество. Элементы этого множества будем называть *символами*.

Пример 3.1. Примеры алфавитов

- Латинский алфавит $\Sigma = \{a, b, c, \ldots, z\}$
- Кириллический алфавит $\Sigma = \{$а, б, в, ..., я$\}$
- Алфавит чисел в шестнадцатеричной записи $\Sigma = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F\}$

Традиционное обозначение для алфавита — Σ. Также мы будем использовать различные прописные буквы латинского алфавита. Для обозначения символов алфавита будем использовать строчные буквы латинского алфавита: $a, b, \ldots, x, y, z$.

Будем считать, что над алфавитом Σ всегда определена операция конкатенации $(\cdot) : \Sigma^* \times \Sigma^* \to \Sigma^*$. При записи выражений символ точки (обозначение операции конкатенации) часто будем опускать: $a \cdot b = ab$.

Определение 3.2. *Слово* над алфавитом Σ — это конечная конкатенация символов алфавита Σ: $\omega = a_0 \cdot a_1 \cdot \ldots \cdot a_m$, где ω — слово, а для любого i $a_i \in \Sigma$.

Определение 3.3. *Слово* над алфавитом Σ — это результат конечной конкатенации символов алфавита Σ: $\omega = a_0 \cdot a_1 \cdot \ldots \cdot a_m$, где ω — слово, а для любого i $a_i \in \Sigma$. Будем называть $m + 1$ длиной слова и обозначать как $|\omega|$.

Определение 3.4. *Язык* над алфавитом Σ — это множество слов над алфавитом Σ.

Пример 3.2. Примеры языков

- Язык целых чисел в двоичной записи $\{0, 1, -1, 10, 11, -10, -11, \ldots\}$
- Язык всех правильных скобочных последовательностей $\{(), (()), ()(), (())(), \ldots\}$

Любой язык над алфавитом Σ является подмножеством Σ^* — множества всех слов над алфавитом Σ

Заметим, что язык не обязан быть конечным множеством, в то время как алфавит всегда конечен и изучаем мы конечные слова.

Способы задания языков

- Перечислить все элементы. Такой способ работает только для конечных языков. Перечислить бесконечное множество не получится.
- Задать генератор — процедуру, которая возвращает очередное слово языка.
- Задаьть распознователь — процедуру, которая по данному слову может определить, принадлежит оно заданному языку ли нет.

3.1 Контекстно-свободные грамматики и языки

Из всего многообразия нас будут интересовать прежде всего контекстно-свободные грамматики.

Определение 3.5. *Контекстно-свободная грамматика* — это четвёрка вида $\langle \Sigma, N, P, S \rangle$, где

- Σ — это терминальный алфавит;
- N — это нетерминальный алфавит;
- P — это множество правил или продукций, таких что каждая продукция имеет вид $N_i \to \alpha$, где $N_i \in N$ и $\alpha \in \{\Sigma \cup N\}^* \cup \varepsilon$;
- S — стартовый нетерминал. Отметим, что $\Sigma \cap N = \varnothing$.

Пример 3.3. Грамматика, задающая язык целых чисел в двоичной записи без лидирующих нулей: $G = \langle \{0, 1, -\}, \{S, N, A\}, P, S \rangle$, где P определено следующим образом:

$$\begin{array}{rcl} S & \to & 0 \mid N \mid -N \\ N & \to & 1A \\ A & \to & 0A \mid 1A \mid \varepsilon \end{array}$$

При спецификации грамматики часто опускают множество терминалов и нетерминалов, оставляя только множество правил. При этом нетерминалы часто обозначаются прописными латинскими буквами, терминалы — строчными, а стартовый нетерминал обозначается буквой S. Мы будем следовать этим обозначениям, если не указано иное.

Определение 3.6. *Отношение непосредственной выводимости.* Мы говорим, что последовательность терминалов и нетерминалов $\gamma\alpha\delta$ *непосредственно выводится из* $\gamma\beta\delta$ *при помощи правила* $\alpha \to \beta$ ($\gamma\alpha\delta \Rightarrow \gamma\beta\delta$), если

- $\alpha \to \beta \in P$
- $\gamma, \delta \in \{\Sigma \cup N\}^* \cup \varepsilon$

Определение 3.7. *Отношение выводимости* является рефлексивно-транзитивным замыканием отношения непосредственной выводимости

- $\alpha \overset{*}{\Rightarrow} \beta$ означает $\exists \gamma_0, \ldots \gamma_k : \ \alpha \Rightarrow \gamma_0 \Rightarrow \gamma_1 \Rightarrow \ldots \Rightarrow \gamma_{k-1} \Rightarrow \gamma_k \Rightarrow \beta$
- Транзитивность: $\forall \alpha, \beta, \gamma \in \{\Sigma \cup N\}^* \cup \varepsilon : \ \alpha \overset{*}{\Rightarrow} \beta, \beta \overset{*}{\Rightarrow} \gamma \Rightarrow \alpha \overset{*}{\Rightarrow} \gamma$
- Рефлексивность: $\forall \alpha \in \{\Sigma \cup N\}^* \cup \varepsilon : \ \alpha \overset{*}{\Rightarrow} \alpha$
- $\alpha \overset{*}{\Rightarrow} \beta$ — α выводится из β
- $\alpha \overset{k}{\Rightarrow} \beta$ — α выводится из β за k шагов
- $\alpha \overset{+}{\Rightarrow} \beta$ — при выводе использовалось хотя бы одно правило грамматики

Пример 3.4. Пример вывода цепочки -1101 в грамматике:

$$\begin{array}{rcl} S & \to & 0 \mid N \mid -N \\ N & \to & 1A \\ A & \to & 0A \mid 1A \mid \varepsilon \end{array}$$

$$S \Rightarrow -N \Rightarrow -1A \Rightarrow -11A \stackrel{*}{\Rightarrow} -1101A \Rightarrow -1101$$

Определение 3.8. *Рефлексивно-транзитивное замыкание отношения* — это наименьшее рефлексивное и транзитивное отношение, содержащее исходное.

Определение 3.9. *Вывод слова в грамматике.* Слово $\omega \in \Sigma^*$ *выводимо в грамматике* $\langle \Sigma, N, P, S \rangle$, если существует некоторый вывод этого слова из начального нетерминала $S \stackrel{*}{\Rightarrow} \omega$.

Определение 3.10. *Левосторонний вывод.* На каждом шаге вывода заменяется самый левый нетерминал.

Пример 3.5. Пример левостороннего вывода цепочки в грамматике

$$\begin{array}{rcl} S & \to & AA \mid s \\ A & \to & AA \mid Bb \mid a \\ B & \to & c \mid d \end{array}$$

$$\boldsymbol{S} \Rightarrow \boldsymbol{A}A \Rightarrow \boldsymbol{B}bA \Rightarrow cb\boldsymbol{A} \Rightarrow cb\boldsymbol{A}A \Rightarrow cba\boldsymbol{A} \Rightarrow cbaa$$

Аналогично можно определить правосторонний вывод.

Определение 3.11. *Язык, задаваемый грамматикой* — множество строк, выводимых в грамматике $L(G) = \{\omega \in \Sigma^* \mid S \stackrel{*}{\Rightarrow} \omega\}$.

Определение 3.12. Грамматики G_1 и G_2 называются *эквивалентными*, если они задают один и тот же язык: $L(G_1) = L(G_2)$

Пример 3.6. Пример эквивалентных грамматик для языка целых чисел в двоичной системе счисления.

$$\begin{array}{rcl} \Sigma & = & \{0, 1, -\} \\ N & = & \{S, N, A\} \\ \\ S & \to & 0 \mid N \mid -N \\ N & \to & 1A \\ A & \to & 0A \mid 1A \mid \varepsilon \end{array}$$

$$\begin{array}{rcl} \Sigma & = & \{0, 1, -\} \\ N & = & \{S, A\} \\ \\ S & \to & 0 \mid 1A \mid -1A \\ A & \to & 0A \mid 1A \mid \varepsilon \end{array}$$

Определение 3.13. *Неоднозначная грамматика* — грамматика, в которой существует 2 и более выводов для одного слова.

Пример 3.7. Неоднозначная грамматика для правильных скобочных последовательностей

$$S \to (S) \mid SS \mid \varepsilon$$

Определение 3.14. *Однозначная грамматика* — грамматика, в которой существует не более одного вывода для каждого слова.

Пример 3.8. Однозначная грамматика для правильных скобочных последовательностей

$$S \to (S)S \mid \varepsilon$$

Определение 3.15. *Существенно неоднозначные языки* — язык, для которого невозможно построить однозначную грамматику

Пример 3.9. Пример существенно неоднозначного языка

$$\{a^n b^n c^m \mid n, m \in \mathbb{Z}\} \cup \{a^n b^m c^m \mid n, m \in \mathbb{Z}\}$$

3.2 Дерево вывода

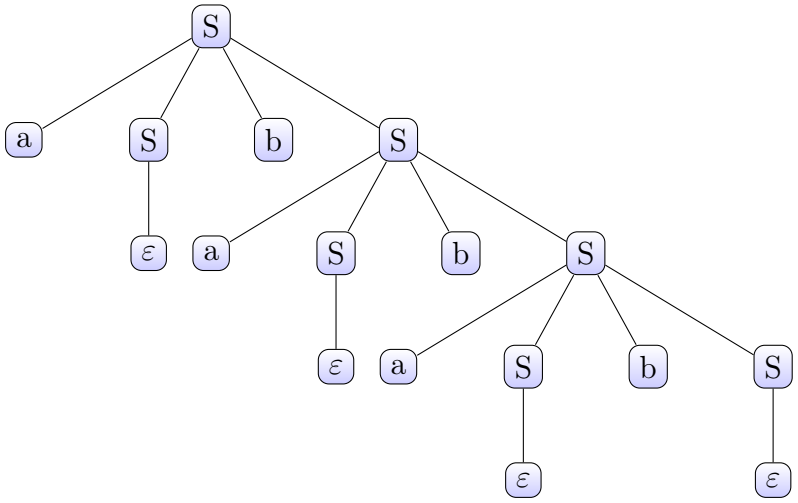
В некоторых случаях не достаточно знать порядок применения правил. Необходимо структурное представление вывода цепочки в грамматике. Таким представлением является *дерево вывода*.

Определение 3.16. Деревом вывода цепочки ω в грамматике $G = \langle \Sigma, N, S, P \rangle$ называется дерево, удовлетворяющее следующим свойствам.

1. Помеченное: метка каждого внутреннего узла — нетерминал, метка каждого листа — терминал или ε.
2. Корневое: корень помечен стартовым нетерминалом.
3. Упорядоченное.
4. В дереве есть узел с меткой N_i и сыновьями $M_j \dots M_k$ тогда и только тогда, когда в грамматике есть правило вида $N_i \to M_j \dots M_k$.
5. Крона образует цепочку.

Пример 3.10. Построим дерево вывода цепочки *ababab* в грамматике

$$G = \langle \{a, b\}, \{S\}, S, \{S \to a\ S\ b\ S, S \to \varepsilon\} \rangle$$



Теорема 3.1. Пусть $G = \langle \Sigma, N, P, S \rangle$ — КС-грамматика. Вывод $S \overset{*}{\Rightarrow} \alpha$, где $\alpha \in (N \cup \Sigma)^*, \alpha \neq \varepsilon$ существует $\Leftrightarrow$ существует дерево вывода в грамматике G с кроной α.

3.3 Пустота КС-языка

Теорема 3.2. Существует алгоритм, определяющий, является ли язык, порождаемый КС грамматикой, пустым

Доказательство. Следующая лемма утверждает, что если в КС языке есть выводимое слово, то существует другое выводимое слово с деревом вывода не глубже количества нетерминалов грамматики. Для доказательства теоремы достаточно привести алгоритм, последовательно строящий все деревья глубины не больше количества нетерминалов грамматики, и проверяющий, являются ли такие деревья деревьями вывода. Если в результате работы алгоритма не удалось построить ни одного дерева, то грамматика порождает пустой язык. □

Лемма 3.3. Если в данной грамматике выводится некоторая цепочка, то существует цепочка, дерево вывода которой не содержит ветвей длиннее m, где m — количество нетерминалов грамматики

Доказательство. Рассмотрим дерево вывода цепочки ω. Если в нем есть 2 узла, соответствующих одному нетерминалу A, обозначим их n_1 и n_2.

Предположим, n_1 расположен ближе к корню дерева, чем n_2.

$S \overset{*}{\Rightarrow} \alpha A_{n_1} \beta \overset{*}{\Rightarrow} \alpha\omega_1\beta; S \overset{*}{\Rightarrow} \alpha\gamma A_{n_2}\delta\beta \overset{*}{\Rightarrow} \alpha\gamma\omega_2\delta\beta$, при этом ω_2 является подцепочкой ω_1.

Заменим в изначальном дереве узел n_1 на n_2. Полученное дерево является деревом вывода $\alpha\omega_2\delta$.

Повторяем процесс замены одинаковых нетерминалов до тех пор, пока в дереве не останутся только уникальные нетерминалы.

В полученном дереве не может быть ветвей длины большей, чем m.

По построению оно является деревом вывода. □

3.4 Нормальная форма Хомского

Определение 3.17. Контекстно-свободная грамматика $\langle \Sigma, N, P, S \rangle$ находится в *Нормальной Форме Хомского*, если она содержит только правила следующего вида:

- $A \to BC$, где $A, B, C \in N^*$
- $A \to a$, где $A \in N, a \in \Sigma$
- $S \to \varepsilon$

Теорема 3.4. Любую КС грамматику можно преобразовать в НФХ.

Доказательство. Алгоритм преобразования в НФХ состоит из следующих шагов:

- Замена неодиночных терминалов
- Удаление длинных правил
- Удаление ε-правил
- Удаление цепных правил

- Удаление бесполезных нетерминалов

То, что каждый из этих шагов преобразует грамматику к эквивалентной, при этом является алгоритмом, доказано в следующих леммах. □

Лемма 3.5. Для любой КС-грамматики можно построить эквивалентную, которая не содержит правила с неодиночными терминалами.

Доказательство. Каждое правило $A \to B_0 B_1 \dots B_k, k \geq 1$ заменить на множество правил:

- $A \to C_0 C_1 \dots C_k$
- $\{C_i \to B_i \mid B_i \in \Sigma, C_i$ — новый нетерминал$\}$

□

Лемма 3.6. Для любой КС-грамматики можно построить эквивалентную, которая не содержит правил длины больше 2.

Доказательство. Каждое правило $A \to B_0 B_1 \dots B_k, k \geq 2$ заменить на множество правил:

- $A \to B_0 C_0$
- $C_0 \to B_1 C_1$
- $\dots$
- $C_{k-3} \to B_{k-2} C_{k-2}$
- $C_{k-2} \to B_{k-1} B_k$

□

Лемма 3.7. Для любой КС-грамматики можно построить эквивалентную, не содержащую ε-правил.

Доказательство. Определим ε-правила:

- $A \to \varepsilon$
- $A \to B_0 \dots B_k, \forall i : \ B_i$ — ε-правило.

Каждое правило $A \to B_0 B_1 \dots B_k$ заменяем на множество правил, где каждое ε-правило удалено во всех возможных комбинациях. □

Лемма 3.8. Можно удалить все цепные правила

Лемма 3.9. Можно удалить все бесполезные нетерминалы

3.5 Вопросы и задачи

1. Предъявить несколько выводов для одной цепочки.
2. Построить выводы
3. Построить деревья вывода !!! Перенести из раздела про SPPF

4 Задача о поиске путей с ограничениями в терминах формальных языков

Что, откуда и зачем.

История вопроса.

4.1 Постановка задачи

Пусть нам дан конечный ориентированный помеченный граф $\mathcal{G} = \langle V, E, L \rangle$. Функция $\omega(\pi) = \omega((v_0, l_0, v_1), (v_1, l_1, v_2), \ldots, (v_{n-1}, l_{n-1}, v_n)) = l_0 \cdot l_1 \cdot \ldots \cdot l_{n-1}$ 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.

Если теперь рассматривать задачу поиска путей, то окажется, что то множество путей, которое мы хотим найти, задаёт множество слов, то есть язык. А значит, критерий поиска мы можем сформулировать следующим образом: нас интересуют такие пути, что слова из меток вдоль них принадлежат заданному языку.

Определение 4.1. *Задача поиска путей с ограничениями в терминах формальных языков* заключается в поиске множества путей $\Pi = \{\pi \mid \omega(\pi) \in \mathcal{L}\}$.

Язык $\mathcal{L}$ может принадлежать разным классам и быть задан разными способами. Например, он может быть регулярным. Или контекстно свободным. Или многокомпонентным контекстно-свободным. Подробно мы рассотрим случай, когда $\mathcal{L}$ — контекстно-свободный язык.

Путь $G = \langle \Sigma, N, P \rangle$ — контекстно-свободная граммтика. Будем считать, что $L \subseteq \Sigma$. Мы не фиксируем стартовый нетерминал в определении граммтики, 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:

В задаче поиска путей мы можем накладывать дополнительные ограничения на путь (например, чтобы он был простым или кратчайшим), но это уже другая история.

4.2 О разрешимости задачи

Сведение к задаче о пересечениии с регулярным.

Замкнутость регулярных.

Проверка пустоты.

Замкнутость контекстно-свободных.

Проверка пустоты.

Про другие классы языков: конъюнктивные, булевы, многокомпонентные.

4.3 Области применения

Где применятеся

Статанализ. Введено Томасом Репсом [20]. Применяется для различных межпроцедурных задач [18, 5, 28].

Графовые БД. Впервые предложил Михалис Яннакакис [27].

Куча ссылок. Примеры?

4.4 Вопросы и задачи

1. Пучть есть граф. Задайте грамматику дл поиска всех путей, таких, что....
2. Существует ли в графе !!! путь из А в Б, такой что!!!
3. Для графа !!! постройте все пути, удовлетворяющие !!!!
4. Задача 1
5. Задача 2

5 Алгоритм на матричных опреациях

5.1 Алгоритм CYK

Алгоритм Cocke-Younger-Kasami — один классических алгоритмов синтаксического анализа. Он требует, чтобы грамматика находилась в Нормальной Форме Хомского. Сложность алгоритма — $O(n^3)$, где n — размер входной строки.

В основе алгоритма лежит динамическое программирование. Используются следующие соображения.

Правило $A \to a$ означает, что

$$A \Rightarrow a \overset{*}{\Rightarrow} \omega \iff \omega = a$$

Правило $A \to BC$ означает, что

$$A \Rightarrow BC \overset{*}{\Rightarrow} \omega \iff \exists \omega_1, \omega_2 : \omega = \omega_1\omega_2, B \overset{*}{\Rightarrow} \omega_1, C \overset{*}{\Rightarrow} \omega_2$$

Или в терминах позиций в строке:

$$A \Rightarrow BC \overset{*}{\Rightarrow} \omega \iff \exists k \in [1 \dots |\omega|] : B \overset{*}{\Rightarrow} \omega[1 \dots k], C \overset{*}{\Rightarrow} \omega[k+1 \dots |\omega|]$$

В процессе работы алгоритма заполняется булева трехмерная матрица размера $N \times n \times n$, где n — размер входной цепочки, N — количество нетерминалов в нормализованной грамматике. $m[A][i][j] = true \iff A \overset{*}{\Rightarrow} \omega[i \dots j]$

Первым шагом инициализируем матрицу, заполнив значения $m[A][i][j]$, где $i = j$:

- $m[A][i][i] = true$, если в грамматике есть правило $A \to \omega[i]$.

- $m[A][i][i] = false$, иначе.

Далее используем динамику: предполагаем, что ячейки матрицы $m[A][i'][j']$ заполнены для всех нетерминалов A и пар $i', j' : j' - i' < m$. Тогда можно заполнить ячейки матрицы $m[A][i][j]$, где $j - i = m$

$$m[A][i][j] = \bigvee_{A \to BC} \bigvee_{k=i}^{j-1} m[B][i][k] \wedge m[C][k][j]$$

По итогу работы алгоритма значение в ячейке $m[S][0][|\omega|]$, где S — стартовый нетерминал грамматики, отвечает на вопрос о выводимости цепочки ω в грамматике.

Пример 5.1. Рассмотрим пример работы алгоритма CYK на грамматике правильных скобочных последовательностей в Нормальной Форме Хомского.

$$\begin{aligned} S_0 &\to LS' \mid \varepsilon \\ S &\to LS' \\ S' &\to RS \mid b \mid SS'' \\ S'' &\to RS \mid b \\ L &\to a \\ R &\to b \end{aligned}$$

Проверим выводимость цепочки $\omega = aabbab$.

Так как трехмерные матрицы рисовать на двумерной бумаге не очень удобно, мы будем иллюстрировать работу алгоритма двумерными матрицами размера $n \times n$, где в ячейках указано множество нетерминалов, выводящих соответствующую подстроку.

Инициализируем матрицу:

	a	a	b	b	a	b
	1	2	3	4	5	6
1	$\{L\}$	∅	∅	∅	∅	∅
2		$\{L\}$	∅	∅	∅	∅
3			$\{R, S', S''\}$	∅	∅	∅
4				$\{R, S', S''\}$	∅	∅
5					$\{L\}$	∅
6						$\{R, S', S''\}$

Заполняем следующую диагональ:

	a	a	b	b	a	b
	1	2	3	4	5	6
1	$\{L\}$	∅	∅	∅	∅	∅
2		$\{L\}$	$\{S\}$	∅	∅	∅
3			$\{R, S', S''\}$	∅	∅	∅
4				$\{R, S', S''\}$	∅	∅
5					$\{L\}$	$\{S\}$
6						$\{R, S', S''\}$

Заполняем следующую диагональ:

	a	a	b	b	a	b
	1	2	3	4	5	6
1	$\{L\}$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$
2		$\{L\}$	$\{S\}$	$\{S'\}$	$\varnothing$	$\varnothing$
3			$\{R, S', S''\}$	$\varnothing$	$\varnothing$	$\varnothing$
4				$\{R, S', S''\}$	$\varnothing$	$\{S', S''\}$
5					$\{L\}$	$\{S\}$
6						$\{R, S', S''\}$

Заполняем следующую диагональ:

	a	a	b	b	a	b
	1	2	3	4	5	6
1	$\{L\}$	$\varnothing$	$\varnothing$	$\{S, S_0\}$	$\varnothing$	$\varnothing$
2		$\{L\}$	$\{S\}$	$\{S'\}$	$\varnothing$	$\varnothing$
3			$\{R, S', S''\}$	$\varnothing$	$\varnothing$	$\varnothing$
4				$\{R, S', S''\}$	$\varnothing$	$\{S', S''\}$
5					$\{L\}$	$\{S\}$
6						$\{R, S', S''\}$

Заполняем следующую диагональ:

	a	a	b	b	a	b
	1	2	3	4	5	6
1	$\{L\}$	$\varnothing$	$\varnothing$	$\{S, S_0\}$	$\varnothing$	$\varnothing$
2		$\{L\}$	$\{S\}$	$\{S'\}$	$\varnothing$	$\{S'\}$
3			$\{R, S', S''\}$	$\varnothing$	$\varnothing$	$\varnothing$
4				$\{R, S', S''\}$	$\varnothing$	$\{S', S''\}$
5					$\{L\}$	$\{S\}$
6						$\{R, S', S''\}$

Заполняем последнюю оставшуюся ячейку:

	a	a	b	b	a	b
	1	2	3	4	5	6
1	$\{L\}$	$\varnothing$	$\varnothing$	$\{S, S_0\}$	$\varnothing$	$\{S, S_0\}$
2		$\{L\}$	$\{S\}$	$\{S'\}$	$\varnothing$	$\{S'\}$
3			$\{R, S', S''\}$	$\varnothing$	$\varnothing$	$\varnothing$
4				$\{R, S', S''\}$	$\varnothing$	$\{S', S''\}$
5					$\{L\}$	$\{S\}$
6						$\{R, S', S''\}$

Стартовый нетерминал находится в верхней правой ячейке, а значит цепочка $aabbab$ выводима в нашей грамматике.

Пример 5.2. Теперь выполним алгоритм на невыводимой цепочке $abaa$.

Инициализируем таблицу:

	a	b	a	a
	1	2	3	4
1	$\{L\}$	$\emptyset$	$\emptyset$	$\emptyset$
2		$\{R, S', S''\}$	$\emptyset$	$\emptyset$
3			$\{L\}$	$\emptyset$
4				$\{L\}$

Заполняем следующую диагональ:

	a	b	a	a
	1	2	3	4
1	$\{L\}$	$\{S, S_0\}$	$\emptyset$	$\emptyset$
2		$\{R, S', S''\}$	$\emptyset$	$\emptyset$
3			$\{L\}$	$\emptyset$
4				$\{L\}$

Больше ни одну ячейку в таблице заполнить нельзя, а значит эта строка не выводится в грамматике правильных скобочных последовательностей.

5.2 Алгоритм для графов на основе CYK

Данный алгоритм накладывает ограничение на форму грамматики: грамматика должна быть в "ослабленной" нормальной форме Хомского 3.4. Не будем требовать отсутсвия пустого и стартового в правой части.

Обобщение. Смотрим на транзитивное замыкание.

5.3 Алгоритм на основе матриц

Ссылка на Валианта [**?**].

Оригинальные матрицы (Рустам) [4]

Кратчайшие.

Одно дерево?

5.4 Конъюнктивные и булевы граммтики

5.4.1 Определения

Охотин [**?**].

5.4.2 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 [**?**]

5.5 Особенности реализации матричного алгоритма

Кое-что про наши реализации [16]

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

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

Распределённые решения. Через GraphBLAS

5.6 Обзор

Китайцы [**?**], Брэдфорд [**?**], Ли [12], Хеллингс [**?**], OpenCypher [11].

Рассуждения про ассимптотику.

Субкубический для частного случая (Брэдфорд) [7].

5.7 Вопросы и задачи

1. !!!

6 Через тензорное произведение

Предыдущий подход позволяет выразить задачу поиска путей с ограничениями в терминах ыормальных языков через набор матричных операций. Это позволяет использовать высокопроизводительные библиотеки и массовопараллельные архитектуры и вообще круто. Однако, такой подход требует, чтобы грамматика находилась в ослабленной нормальной форме Хомского, что приводит к её разростанию. Можно ли как-то избежать этого?

В данном разделе мы попробуем предложить альтернативное сведение задачи поиска путей к матричным опреациям. В результате мы сможем избежать преобразования граммтики в ОНФХ, однако, матрицы, с которыми нам предётся работать, будут существенно бóльшего размера.

В основе подхода лежит использование рекурсивных сетей или рекурсивных автоматов в качестве представления контекстно-свободных грамматик.

6.1 Рекурсивные автоматы и сети

Рекурсивный автомат или сеть — это представление контекстно-свободных грамматик, обобщающее конечные автоматы. В нашей работе мы будем придерживаться термина **рекурсивный автомат**. Классическое определение рекурсивного автомата выглядит следующим образом.

Определение 6.1. Рекурсивный автомат — это кортеж вида $\langle N, \Sigma, S, D \rangle$, где

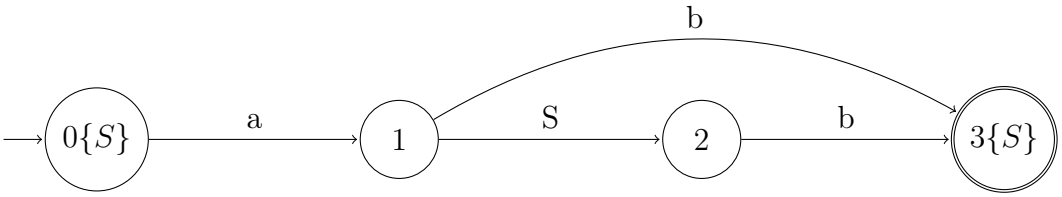
- N — нетерминальный алфавит;
- Σ — терминальный алфавит;

[1] !!!

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

- S — стартовый нетерминал;
- D — конечный автомат над $N \cup \Sigma$ в котором стартовые и финальные состояния помечены подмножествами N.

Построим рекурсивный автомат для грамматики !!!!



Используем стандартные обозначения для стартовых и финальных состояний. Дополнительно в стартовых и финальных состояниях укажем нетерминалы, для которых эти состояния стартовые/финальные.

В некоторых случаях рекурстивный автомат можно рассматривать как конечный автомат над смешанным алфавитом. Именно такой взгляд мы будем использовать при изложении алгоритма.

6.2 Тензорное произведение

Тензорное произведение матриц или произведение Кронекера — это бинарная опирация, обозначаемая $\otimes$ и определяемая следующим образом.

Определение 6.2. Пусть даны две матрицы: A размера $m \times n$ и B размера $p \times q$. Произведение Кронекера или тензорное произведение матриц A и B — это блочная матрица C размера $mp \times nq$, вычисляемая следующим образом:

$$C = A \otimes B = \begin{pmatrix} A_{0,0}B & \cdots & A_{0,n-1}B \\ \vdots & \ddots & \vdots \\ A_{m-1,0}B & \cdots & A_{m-1,n-1}B \end{pmatrix}$$

Пример 6.1.

$$\begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix} \otimes \begin{pmatrix} 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \\ 13 & 14 & 15 & 16 \end{pmatrix} = \begin{pmatrix} 1\begin{pmatrix} 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \\ 13 & 14 & 15 & 16 \end{pmatrix} & 2\begin{pmatrix} 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \\ 13 & 14 & 15 & 16 \end{pmatrix} \\ 3\begin{pmatrix} 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \\ 13 & 14 & 15 & 16 \end{pmatrix} & 4\begin{pmatrix} 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \\ 13 & 14 & 15 & 16 \end{pmatrix} \end{pmatrix} =$$

$$= \left(\begin{array}{cccc|cccc} 5 & 6 & 7 & 8 & 10 & 12 & 14 & 16 \\ 9 & 10 & 11 & 12 & 18 & 20 & 22 & 24 \\ 13 & 14 & 15 & 16 & 26 & 28 & 30 & 32 \\ \hline 15 & 18 & 21 & 24 & 20 & 24 & 28 & 32 \\ 27 & 30 & 33 & 36 & 36 & 40 & 44 & 48 \\ 39 & 42 & 45 & 48 & 52 & 56 & 60 & 64 \end{array}\right) \quad (1)$$

Заметим, что для определения тензорного произведения матриц достаточно определить опреацию умножения на элементах исходных матриц. Также отметим, что произведение Кронекера не является коммутативной. При этом всегда существуют такиедве матрицы перестоновок P и Q, что $A \otimes B = P(B \otimes A)Q$. Это свойство потребуется нам в дальнейшем.

Теперь перейдём к графам. Сперва дадим классическое определение тензорного произведения двух неориентированных графов.

Определение 6.3. 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.

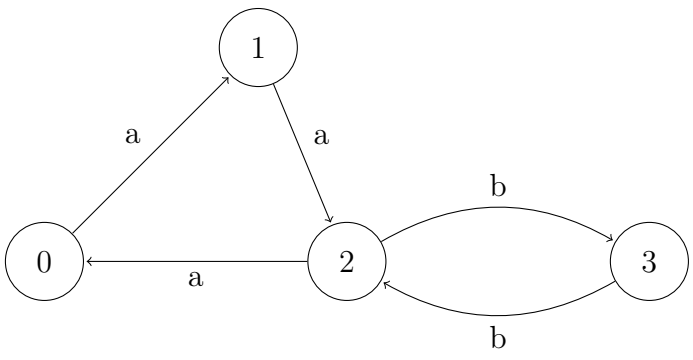
Осталось добавить метки к рёбрам. Это приведёт к логичному усилению требованя к существованию ребра: метки рёбер в исходных графах должны совпадать. Таким образом, мы получаем следующее определение тензорного произведения ориентированных графов с метками на рёбрах.

Определение 6.4. 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 $V_3 = V_1 \times V_2$, $E_3 = \{((v_1, v_2), l, (u_1, u_2)) \mid (v_1, l, u_1) \in E_1 \text{ и } (v_2, l, u_2) \in E_2\}$, $L_3 = L_1 \cap L_2$.

Нетрудно заметить, что матрица смежности графа $\mathcal{G}_3$ равна тенорному произведению матриц смежностей исходных графов $\mathcal{G}_1$ и $\mathcal{G}_2$.

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 6.2. Его матрица смежности имеет следующий вид.

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

Теперь вычислим $M_1 \otimes M_2$.

$$M_3 = M_1 \otimes M_2 = \begin{pmatrix} . & [a] & . & . \\ . & . & [S] & [b] \\ . & . & . & [b] \\ . & . & . & . \end{pmatrix} \otimes \begin{pmatrix} . & [a] & . & . \\ . & . & [a] & . \\ [a] & . & . & [b] \\ . & . & [b] & . \end{pmatrix} =$$

$$= \left(\begin{array}{cccc|cccc|cccc|cccc} . & . & . & . & . & [a] & . & . & . & . & . & . & . & . & . & . \\ . & . & . & . & . & . & [a] & . & . & . & . & . & . & . & . & . \\ . & . & . & . & [a] & . & . & . & . & . & . & . & . & . & . & . \\ . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . \\ \hline . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . \\ . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . \\ . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & [b] \\ . & . & . & . & . & . & . & . & . & . & . & . & . & . & [b] & . \\ \hline . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . \\ . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . \\ . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & [b] \\ . & . & . & . & . & . & . & . & . & . & . & . & . & . & [b] & . \\ \hline . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . \\ . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . \\ . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . \\ . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . \end{array}\right) \quad (2)$$

Побалуемся с некоммутативностью и перестановками.

6.3 Алгоритм

Идея алгоритма основана на обобщении пересечения двух конечных автоматов до пересечения рекурсивного автомата, построенного по грамматике, со входным графом.

Пересечение двух конечных автоматов — тензорное произведение соответствующих графов. Пересечение языкрв коммутативно, тензорное произведение нет, но это не страшно.

Несколько наблюдений Путь из нетерминалов и терминалов. При этом должны быть пути из терминалов. Иначе не задать язык. Будем насыщать граф рёбрами, помеченными нетерминалами.

По грамматике строим автомат. r 13

В цикле: пересекли через тензорное произведение, замкнули через обычное матричное произведение, чтобы найти пути из начальной в конечную в граммтике, поставили в соответствующие ячейки нетерминалы, добавили соответствующие рёбра в исходный граф.

Можно вычислять только разницу. Для этого, правда, потребуется держать ещё одну матрицу. И надо проверять, что вычислительно дешевле: поддерживать разницу и потом

Listing 2 Поиск путей через тензорное произведение

```
1: function contextFreePathQueryingTP(G, 𝒢)
2:     N ← рекурсивный автомат для G
3:     M_1 ← матрица смежности N
4:     M_2 ← матрица смежности 𝒢
5:     T ← the matrix n × n in which each element is ∅
6:     for all (i, x, j) ∈ E do                          ▷ Matrix initialization
7:         T_{i,j} ← T_{i,j} ∪ {A | (A → x) ∈ P}
8:     while matrix T is changing do
9:         M_3 ← M_1 ⊗ M_2                               ▷ Graphs intersection
10:        M_3 ← transitiveClosure(M_3)
11:        for  do
12:            for  do
13:                if  then
14:                    Nt ← M_3[i, j] ∪ {}
15:                    m ← M_3[i, j] ∪ {}
16:                    n ← M_3[i, j] ∪ {}
17:                    M_2[n, m] ← M_2[n, m] ∪ {Nt}
18:    return T
```

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.

6.4 Примеры

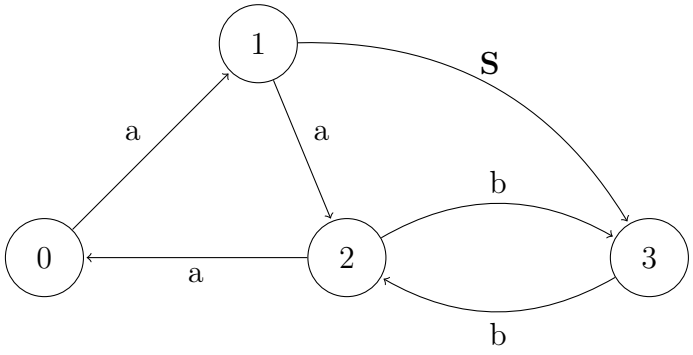
Рассмотрим подробно ряд примеров работы описанного алгоритма. Будем для каждой итерации внешнего цикла выписывать результаты основных операций: тензорного произведения, транзитивного замыкания, обновления матрицы смежности входного грвфа. Новые, по сравнению с пердыдущим состоянием, элементы матриц будем выделять жирным.

Пример 6.2. Теоретический худший случай. Такой же как и для матричного.

Итерация 1 (конец). Начало в разделе!!!, где мы вычислили тензорное произведение матриц смежности. Теепрь нам осталось только вычислить транзитивное замыкание полученной матрицы.

$$tc(M_3) = \left(\begin{array}{cccc|cccc|cccc|cccc} . & . & . & . & . & [a] & . & . & . & . & . & . & . & . & . & . \\ . & . & . & . & . & . & [a] & . & . & . & . & . & . & . & . & \mathbf{[ab]} \\ . & . & . & . & [a] & . & . & . & . & . & . & . & . & . & . & . \\ . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . \\ \hline . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . \\ . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . \\ . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & [b] \\ . & . & . & . & . & . & . & . & . & . & . & . & . & [b] & . & . \\ \hline . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . \\ . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . \\ . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & [b] \\ . & . & . & . & . & . & . & . & . & . & . & . & . & [b] & . & . \\ \hline . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . \\ . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . \\ . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . \\ . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . \end{array}\right) \tag{3}$$

Мы видим, что в результате транзитивного замыкания появилось новое ребро с меткой ab из вершины $(0, 1)$ в вершину $(3, 3)$. Так как вершина 0 является стартовой в рекурсивном автоматие, а 3 является финальной, то слово вдоль пути из вершины 1 в вершину 3 во входном графе выводимо из нетерминала S. Это означает, что в графе должно быть добавлено ребро из 0 в 3 с меткой S, после чего граф будет выглядеть следующим образом:



Матрица смежности обновлённого графа:

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

Итерация закончена. Возвращаемся к началу цикла и вновь вычисляем тензорное произведение.

Итерация 2. Вычисляем тензорное произведение матриц смежности.

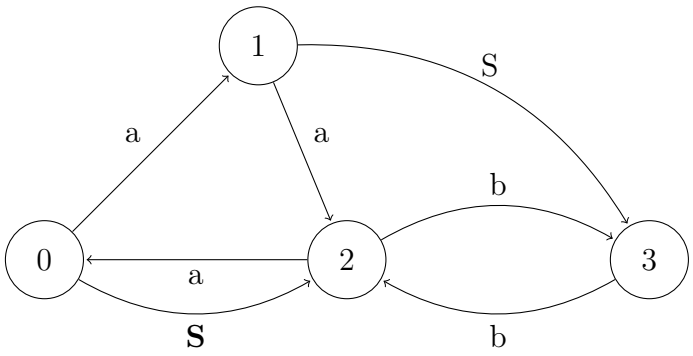
$$M_3 = M_1 \otimes M_2 = \begin{pmatrix} . & [a] & . & . \\ . & . & [S] & [b] \\ . & . & . & [b] \\ . & . & . & . \end{pmatrix} \otimes \begin{pmatrix} . & [a] & . & . \\ . & . & [a] & [S] \\ [a] & . & . & [b] \\ . & . & [b] & . \end{pmatrix} =$$

$$= \left(\begin{array}{cccc|cccc|cccc|cccc} . & . & . & . & . & [a] & . & . & . & . & . & . & . & . & . & . \\ . & . & . & . & . & . & [a] & . & . & . & . & . & . & . & . & . \\ . & . & . & . & [a] & . & . & . & . & . & . & . & . & . & . & . \\ . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . \\ \hline . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . \\ . & . & . & . & . & . & . & . & . & . & . & \mathbf{[S]} & . & . & . & . \\ . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & [b] \\ . & . & . & . & . & . & . & . & . & . & . & . & . & . & [b] & . \\ \hline . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . \\ . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . \\ . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & [b] \\ . & . & . & . & . & . & . & . & . & . & . & . & . & . & [b] & . \\ \hline . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . \\ . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . \\ . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . \\ . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . \end{array}\right) \quad (4)$$

Вычисляем транзитивное замыкание полученной матрицы:

$$tc(M_3) = \left(\begin{array}{cccc|cccc|cccc|cccc} . & . & . & . & . & [a] & . & . & . & . & \mathbf{[aS]} & . & . & . & \mathbf{[aSb]} & . \\ . & . & . & . & . & . & [a] & . & . & . & . & . & . & . & . & [ab] \\ . & . & . & . & [a] & . & . & . & . & . & . & . & . & . & . & . \\ . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . \\ \hline . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . \\ . & . & . & . & . & . & . & . & . & . & [S] & . & . & . & \mathbf{[Sb]} & . \\ . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & [b] \\ . & . & . & . & . & . & . & . & . & . & . & . & . & . & [b] & . \\ \hline . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . \\ . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . \\ . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & [b] \\ . & . & . & . & . & . & . & . & . & . & . & . & . & . & [b] & . \\ \hline . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . \\ . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . \\ . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . \\ . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . \end{array}\right) \quad (5)$$

В транзитивном замыкании появилось три новых ребра, лжнако только одно из них соединяет вершины, соответствующие сартовому и конечному состоянию входного рекурсивного автомата. Только это ребро должно быть соответствующим образом добавлено во входной граф. В итоге, обновлённый граф:



И его матрица смежности:

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

Итерация 3. Снова начинаем с тензорного произведения.

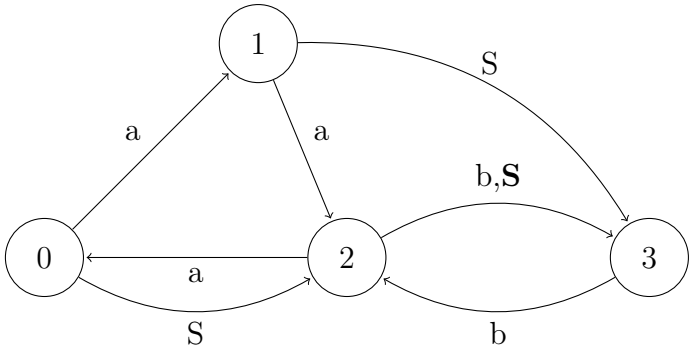
$$M_3 = M_1 \otimes M_2 = \begin{pmatrix} . & [a] & . & . \\ . & . & [S] & [b] \\ . & . & . & [b] \\ . & . & . & . \end{pmatrix} \otimes \begin{pmatrix} . & [a] & [S] & . \\ . & . & [a] & [S] \\ [a] & . & . & [b] \\ . & . & [b] & . \end{pmatrix} =$$

$$= \left(\begin{array}{cccc|cccc|cccc} . & . & . & . & . & [a] & . & . & . & . & . & . \\ . & . & . & . & . & . & [a] & . & . & . & . & . \\ . & . & . & . & [a] & . & . & . & . & . & . & . \\ . & . & . & . & . & . & . & . & . & . & . & . \\ \hline . & . & . & . & . & . & . & . & . & . & . & . \\ . & . & . & . & . & . & . & . & . & . & . & . \\ . & . & . & . & . & . & . & . & . & . & . & [b] \\ . & . & . & . & . & . & . & . & . & . & [b] & . \\ \hline . & . & . & . & . & . & . & . & . & . & . & . \\ . & . & . & . & . & . & . & . & . & . & . & . \\ . & . & . & . & . & . & . & . & . & . & . & [b] \\ . & . & . & . & . & . & . & . & . & . & [b] & . \\ \hline . & . & . & . & . & . & . & . & . & . & . & . \\ . & . & . & . & . & . & . & . & . & . & . & . \\ . & . & . & . & . & . & . & . & . & . & . & . \\ . & . & . & . & . & . & . & . & . & . & . & . \end{array}\right) \quad (6)$$

Затем вычисляем транзитивное замыкание:

$$tc(M_3) = \left(\begin{array}{cccc|cccc|cccc|cccc}
. & . & . & . & . & [a] & . & . & . & . & . & [aS] & . & . & [aSb] & . \\
. & . & . & . & . & . & [a] & . & . & . & . & . & . & . & . & [ab] \\
. & . & . & . & [a] & . & . & . & . & . & \mathbf{[aS]} & . & . & . & . & \mathbf{[aSb]} \\
. & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . \\
\hline
. & . & . & . & . & . & . & . & . & . & [S] & . & . & . & . & \mathbf{[Sb]} \\
. & . & . & . & . & . & . & . & . & . & . & [S] & . & . & [Sb] & . \\
. & . & . & . & . & . & . & . & . & . & . & . & . & . & . & [b] \\
. & . & . & . & . & . & . & . & . & . & . & . & . & . & [b] & . \\
\hline
. & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . \\
. & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . \\
. & . & . & . & . & . & . & . & . & . & . & . & . & . & . & [b] \\
. & . & . & . & . & . & . & . & . & . & . & . & . & . & [b] & . \\
\hline
. & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . \\
. & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . \\
. & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . \\
. & . & . & . & . & . & . & . & . & . & . & . & . & . & . & .
\end{array}\right) \tag{7}$$

И наконец обновляем граф:



Матрица смежности обновлённого графа:

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

Уже можно заметить закономерность: на каждой итерации мы добавляем ровно одно новое ребро во входной граф. То есть находим ровно одну новую пару вешин, между которыми существует интересующий нас путь. Попробуйте спрогонозировать, сколько итераций нам ещё осталось сделать.

Итерауия 4. Продолжаем. Вычисляем тензорное произведение.

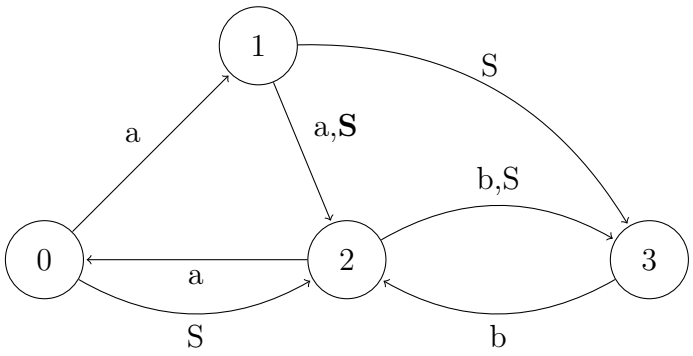
$$M_3 = M_1 \otimes M_2 = \begin{pmatrix} . & [a] & . & . \\ . & . & [S] & [b] \\ . & . & . & [b] \\ . & . & . & . \end{pmatrix} \otimes \begin{pmatrix} . & [a] & [S] & . \\ . & . & [a] & [S] \\ [a] & . & . & [b, S] \\ . & . & [b] & . \end{pmatrix} =$$

$$= \left(\begin{array}{cccc|cccc|cccc|cccc} . & . & . & . & . & [a] & . & . & . & . & . & . & . & . & . & . \\ . & . & . & . & . & . & [a] & . & . & . & . & . & . & . & . & . \\ . & . & . & . & [a] & . & . & . & . & . & . & . & . & . & . & . \\ . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . \\ \hline . & . & . & . & . & . & . & . & . & . & [S] & . & . & . & . & . \\ . & . & . & . & . & . & . & . & . & . & . & [S] & . & . & . & . \\ . & . & . & . & . & . & . & . & . & . & . & \mathbf{[S]} & . & . & . & [b] \\ . & . & . & . & . & . & . & . & . & . & . & . & . & . & [b] & . \\ \hline . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . \\ . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . \\ . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & [b] \\ . & . & . & . & . & . & . & . & . & . & . & . & . & . & [b] & . \\ \hline . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . \\ . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . \\ . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . \\ . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . \end{array}\right) \tag{8}$$

Затем транзитивное замыкание:

$$tc(M_3) = \left(\begin{array}{cccc|cccc|cccc|cccc} . & . & . & . & . & [a] & . & . & . & . & [aS] & . & . & [aSb] & . & . \\ . & . & . & . & . & . & [a] & . & . & . & . & \mathbf{[aS]} & . & \mathbf{[aSb]} & [ab] & . \\ . & . & . & . & [a] & . & . & . & . & . & [aS] & . & . & . & [aSb] & . \\ . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . \\ \hline . & . & . & . & . & . & . & . & . & . & [S] & . & . & . & [Sb] & . \\ . & . & . & . & . & . & . & . & . & . & . & [S] & . & [Sb] & . & . \\ . & . & . & . & . & . & . & . & . & . & . & [S] & . & \mathbf{[Sb]} & [b] & . \\ . & . & . & . & . & . & . & . & . & . & . & . & . & [b] & . & . \\ \hline . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . \\ . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . \\ . & . & . & . & . & . & . & . & . & . & . & . & . & . & [b] & . \\ . & . & . & . & . & . & . & . & . & . & . & . & . & [b] & . & . \\ \hline . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . \\ . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . \\ . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . \\ . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . \end{array}\right) \tag{9}$$

И снова обновляем граф, так как в транзитивном замыкании оявился один (и снова ровно один) новый элемент, соответствующий принимающему пути в автомате.



Матрица смежности обновлённого графа:

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

Итерация 5. Приступаем к выполнению следующей итерации основного цикла. Вычисляем тензорное произведение.

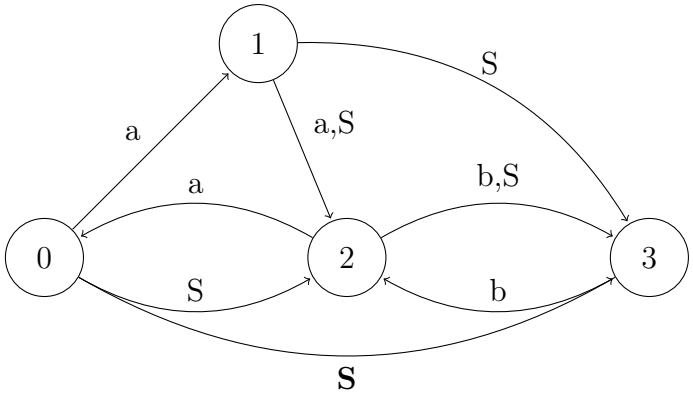
$$M_3 = M_1 \otimes M_2 = \begin{pmatrix} . & [a] & . & . \\ . & . & [S] & [b] \\ . & . & . & [b] \\ . & . & . & . \end{pmatrix} \otimes \begin{pmatrix} . & [a] & [S] & . \\ . & . & [a, S] & [S] \\ [a] & . & . & [b, S] \\ . & . & [b] & . \end{pmatrix} =$$

$$= \left(\begin{array}{cccc|cccc|cccc|cccc} . & . & . & . & . & [a] & . & . & . & . & . & . & . & . & . & . \\ . & . & . & . & . & . & [a] & . & . & . & . & . & . & . & . & . \\ . & . & . & . & [a] & . & . & . & . & . & . & . & . & . & . & . \\ . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . \\ \hline . & . & . & . & . & . & . & . & . & . & [S] & . & . & . & . & . \\ . & . & . & . & . & . & . & . & . & . & \mathbf{[S]} & [S] & . & . & . & . \\ . & . & . & . & . & . & . & . & . & . & . & [S] & . & . & . & [b] \\ . & . & . & . & . & . & . & . & . & . & . & . & . & . & [b] & . \\ \hline . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . \\ . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . \\ . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & [b] \\ . & . & . & . & . & . & . & . & . & . & . & . & . & . & [b] & . \\ \hline . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . \\ . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . \\ . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . \\ . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . \end{array}\right) \tag{10}$$

Затем вычисляем транзитивное замыкание:

$$tc(M_3) = \left(\begin{array}{cccc|cccc|cccc|cccc}
. & . & . & . & . & [a] & . & . & . & . & \mathbf{[aS]} & [aS] & . & . & [aSb] & \mathbf{[aSb]} \\
. & . & . & . & . & . & [a] & . & . & . & . & [aS] & . & . & [aSb] & [ab] \\
. & . & . & . & [a] & . & . & . & . & . & [aS] & . & . & . & . & [aSb] \\
. & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . \\
\hline
. & . & . & . & . & . & . & . & . & . & [S] & . & . & . & . & [Sb] \\
. & . & . & . & . & . & . & . & . & . & [S] & [S] & . & . & [Sb] & \mathbf{[Sb]} \\
. & . & . & . & . & . & . & . & . & . & . & [S] & . & . & [Sb] & [b] \\
. & . & . & . & . & . & . & . & . & . & . & . & . & . & [b] & . \\
\hline
. & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . \\
. & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . \\
. & . & . & . & . & . & . & . & . & . & . & . & . & . & . & [b] \\
. & . & . & . & . & . & . & . & . & . & . & . & . & . & [b] & . \\
\hline
. & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . \\
. & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . \\
. & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . \\
. & . & . & . & . & . & . & . & . & . & . & . & . & . & . & .
\end{array}\right) \tag{11}$$

Обновляем граф:



Матрица смежности обновлённого графа:

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

Итерация 6. И наконец последняя содержательная итерация основного цикла.

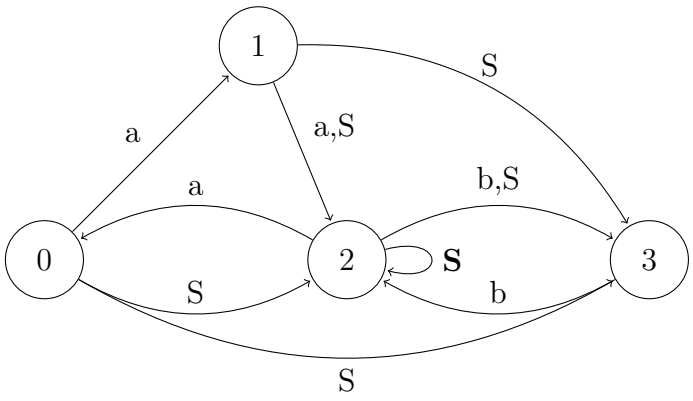
$$M_3 = M_1 \otimes M_2 = \begin{pmatrix} . & [a] & . & . \\ . & . & [S] & [b] \\ . & . & . & [b] \\ . & . & . & . \end{pmatrix} \otimes \begin{pmatrix} . & [a] & [S] & [S] \\ . & . & [a,S] & [S] \\ [a] & . & . & [b,S] \\ . & . & [b] & . \end{pmatrix} =$$

$$= \left(\begin{array}{cccc|cccc|cccc|cccc} . & . & . & . & . & [a] & . & . & . & . & . & . & . & . & . & . \\ . & . & . & . & . & . & [a] & . & . & . & . & . & . & . & . & . \\ . & . & . & . & [a] & . & . & . & . & . & . & . & . & . & . & . \\ . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . \\ \hline . & . & . & . & . & . & . & . & . & . & [S] & \mathbf{[S]} & . & . & . & . \\ . & . & . & . & . & . & . & . & . & . & [S] & [S] & . & . & . & . \\ . & . & . & . & . & . & . & . & . & . & . & [S] & . & . & . & [b] \\ . & . & . & . & . & . & . & . & . & . & . & . & . & . & [b] & . \\ \hline . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . \\ . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . \\ . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & [b] \\ . & . & . & . & . & . & . & . & . & . & . & . & . & . & [b] & . \\ \hline . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . \\ . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . \\ . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . \\ . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . \end{array}\right) \tag{12}$$

Транзитивное замыкание:

$$tc(M_3) = \left(\begin{array}{cccc|cccc|cccc|cccc} . & . & . & . & . & [a] & . & . & . & . & [aS] & [aS] & . & . & [aSb] & [aSb] \\ . & . & . & . & . & . & [a] & . & . & . & . & [aS] & . & . & [aSb] & [ab] \\ . & . & . & . & [a] & . & . & . & . & . & [aS] & \mathbf{[aS]} & . & . & \mathbf{[aSb]} & [aSb] \\ . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . \\ \hline . & . & . & . & . & . & . & . & . & . & [S] & \texttt{[S]} & . & . & \mathbf{[Sb]} & [Sb] \\ . & . & . & . & . & . & . & . & . & . & [S] & [S] & . & . & [Sb] & [Sb] \\ . & . & . & . & . & . & . & . & . & . & . & [S] & . & . & [Sb] & [b] \\ . & . & . & . & . & . & . & . & . & . & . & . & . & . & [b] & . \\ \hline . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . \\ . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . \\ . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & [b] \\ . & . & . & . & . & . & . & . & . & . & . & . & . & . & [b] & . \\ \hline . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . \\ . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . \\ . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . \\ . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . \end{array}\right) \tag{13}$$

Обновлённый граф:



И матрица смежности:

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

Следующая итерация не приведёт к изменению графа. Читатель может убедиться в этом самостоятельно. Соответственно алгоритм можно завершать. Нам потребовалось семь итераций (шесть содержательных и одна проверочная).

Матрица смежности получилась такая же, как и раньше, рответ правильный. Мы видим, что количество итераций внешнего цикла такое же как и у алгоритма !!!. И ещё что-то видим и можем понять.

Пример 6.3. В данном примере мы увидим, как структура грамматики и, соответственно, рекурсивного автомата, влияет на процесс вычислений.

Интуитивно понятно, что чем меньше состояний в рекурсивной сети, тем лучше. То есть желательно получить как можно более компактное описание контекстно-свободного языка.

Для примера возьмём в качестве КС языка язык Дика на одном типе скобок и опишем его двумя различными грамматиками. Первая граммтика классическая:

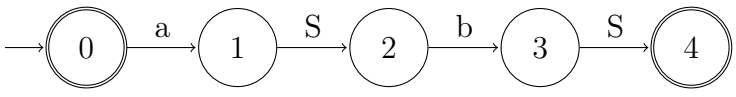
$$G_1 = \langle \{a,\ b\}, \{S\}, \{S \to a\ S\ b\ S \mid \varepsilon\} \rangle$$

Во второй грамматике мы будем использовать конструкции регулярных выражений в правой части правил. То есть вторая грамматика находитмся в EBNF [**?**].

$$G_2 = \langle \{a,\ b\}, \{S\}, \{S \to (a\ S\ b)^*\} \rangle$$

Построим рекурсивные автоматы N_1 и N_2 и их матрицы смежности для этих грамматик.

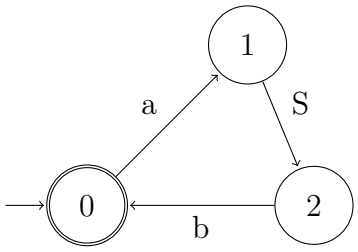
Рекурсивный автомат N_1 для грамматики G_1:



Матрица смежности N_1:

$$M_1^1 = \begin{pmatrix} . & [a] & . & . & . \\ . & . & [S] & . & . \\ . & . & . & [b] & . \\ . & . & . & . & [S] \\ . & . & . & . & . \end{pmatrix}$$

Рекурсивный автомат N_2 для грамматики G_2:



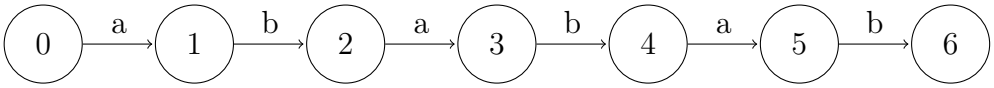
Матрица смежности N_2:

$$M_1^2 = \begin{pmatrix} . & [a] & . \\ . & . & [S] \\ [b] & . & . \end{pmatrix}$$

Первое, очевидное, наблюдение — количество состояний в N_2 меньше, чем в N_1. Это значит, что матрицы будут меньше, считать быстрее.

Для того, чтобы проще было сделать второе, сперва выполним пошагово алгоритм для двух заданных рекурсивных сетей.

Вход возьмём линейный:



Сразу дополним матрицу смежности нетерминалами, выводящими пустую строку, и получим следующую матрицу:

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

Сперва запустим алгоритм на входе и N_2. Первый шаг первой итерации — вычисление тензорного произведения $M_1^2 \otimes M_2$.

$$M_3 = M_1^2 \otimes M_2 = \begin{pmatrix} . & [a] & . \\ . & . & [S] \\ [b] & . & . \end{pmatrix} \otimes \begin{pmatrix} [S] & [a] & . & . & . & . & . \\ . & [S] & [b] & . & . & . & . \\ . & . & [S] & [a] & . & . & . \\ . & . & . & [S] & [b] & . & . \\ . & . & . & . & [S] & [a] & . \\ . & . & . & . & . & [S] & [b] \\ . & . & . & . & . & . & [S] \end{pmatrix} =$$

$$= \left(\begin{array}{ccccccc|ccccccc|ccccccc} . & . & . & . & . & . & . & . & [a] & . & . & . & . & . & . & . & . & . & . & . & . \\ . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . \\ . & . & . & . & . & . & . & . & . & . & [a] & . & . & . & . & . & . & . & . & . & . \\ . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . \\ . & . & . & . & . & . & . & . & . & . & . & . & [a] & . & . & . & . & . & . & . & . \\ . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . \\ . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . \\ \hline . & . & . & . & . & . & . & . & . & . & . & . & . & . & [S] & . & . & . & . & . & . \\ . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & [S] & . & . & . & . & . \\ . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & [S] & . & . & . & . \\ . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & [S] & . & . & . \\ . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & [S] & . & . \\ . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & [S] & . \\ . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & [S] \\ \hline . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . \\ . & . & [b] & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . \\ . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . \\ . & . & . & . & [b] & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . \\ . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . \\ . & . & . & . & . & . & [b] & . & . & . & . & . & . & . & . & . & . & . & . & . & . \\ . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . \end{array}\right) \quad (14)$$

Опустим промежуточные шаги вычисления транзитивного замыкания M_3 и сразу пред-

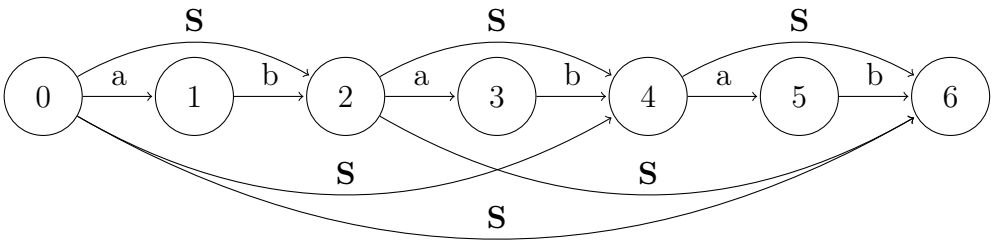
ставим конечный результат:

$$tc(M_3) =$$

$$\left(\begin{array}{cccccc|cccccc|cccccc}
. & . & \text{[aSb]} & . & \text{[aSbaSb]} & . & \text{[aSbaSbaSb]} & . & [a] & . & \text{[aSba]} & . & \text{[aSbaSba]} & . & . & \text{[aS]} & . & \text{[aSbaS]} & . & \text{[aSbaSbaS]} & . \\
. & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . \\
. & . & . & . & \text{[aSb]} & . & \text{[aSbaSb]} & . & . & . & [a] & . & \text{[aSba]} & . & . & . & . & \text{[aS]} & . & \text{[aSbaS]} & . \\
. & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . \\
. & . & . & . & . & . & \text{[aSb]} & . & . & . & . & . & [a] & . & . & . & . & . & . & \text{[aS]} & . \\
. & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . \\
. & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . \\
\hline
. & . & . & . & . & . & . & . & . & . & . & . & . & . & [S] & . & . & . & . & . & . \\
. & . & . & . & . & . & . & . & . & . & . & . & . & . & . & [S] & . & . & . & . & . \\
. & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & [S] & . & . & . & . \\
. & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & [S] & . & . & . \\
. & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & [S] & . & . \\
. & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & [S] & . \\
. & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & [S] \\
\hline
. & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . \\
. & . & [b] & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . \\
. & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . \\
. & . & . & . & [b] & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . \\
. & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . \\
. & . & . & . & . & . & [b] & . & . & . & . & . & . & . & . & . & . & . & . & . & . \\
. & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & .
\end{array}\right) \tag{15}$$

В результате вычисления транзитивного замыкания появилось большое количество новых рёбер, однако нас будут интересновать только те, информация о которых храниться в левом верхнем блоке. Остальные рёбра не соответствуют принимающим путям в рекурсивном автомате (убедитесь в этом самостоятельно).

После добавления соответствующих рёбер, мы получим следующий граф:



Его матрица смежности:

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

Таким образом видно, что для выбранных входных данных алгоритму достаточно двух итераций основного цикла: первая содержательная, вторая, как обычно, проверочная. Читателю предлагается самостоятельно выяснить, сколько умножений матриц потребуется, стобы вычислить транзитивное замыкание на первой итерации.

Теперь запустим алгоритм на второй грамматике и том же входе.

$$
M_3 = M_1^1 \otimes M_2 = \begin{pmatrix} . & [a] & . & . & . \\ . & . & [S] & . & . \\ . & . & . & [b] & . \\ . & . & . & . & [S] \\ . & . & . & . & . \end{pmatrix} \otimes \begin{pmatrix} [S] & [a] & . & . & . & . & . \\ . & [S] & [b] & . & . & . & . \\ . & . & [S] & [a] & . & . & . \\ . & . & . & [S] & [b] & . & . \\ . & . & . & . & [S] & [a] & . \\ . & . & . & . & . & [S] & [b] \\ . & . & . & . & . & . & [S] \end{pmatrix} =
$$

$$
= \left(\begin{array}{ccccccc|ccccccc|ccccccc|ccccccc|ccccccc}
. & . & . & . & . & . & . & . & [a] & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . \\
. & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . \\
. & . & . & . & . & . & . & . & . & . & [a] & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . \\
. & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . \\
. & . & . & . & . & . & . & . & . & . & . & . & [a] & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . \\
. & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . \\
. & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . \\
\hline
. & . & . & . & . & . & . & . & . & . & . & . & . & . & [S] & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . \\
. & . & . & . & . & . & . & . & . & . & . & . & . & . & . & [S] & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . \\
. & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & [S] & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . \\
. & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & [S] & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . \\
. & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & [S] & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . \\
. & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & [S] & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . \\
. & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & [S] & . & . & . & . & . & . & . & . & . & . & . & . & . & . \\
\hline
. & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . \\
. & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & [b] & . & . & . & . & . & . & . & . & . & . & . \\
. & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . \\
. & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & [b] & . & . & . & . & . & . & . & . & . \\
. & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . \\
. & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & [b] & . & . & . & . & . & . & . \\
. & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . \\
\hline
. & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & [S] & . & . & . & . & . & . \\
. & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & [S] & . & . & . & . & . \\
. & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & [S] & . & . & . & . \\
. & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & [S] & . & . & . \\
. & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & [S] & . & . \\
. & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & [S] & . \\
. & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & [S] \\
\hline
. & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . \\
. & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . \\
. & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . \\
. & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . \\
. & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . \\
. & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . \\
. & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & .
\end{array}\right) \tag{16}
$$

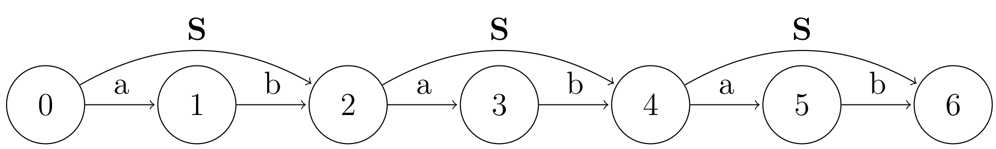
Уже сейчас можно заметить, что размер матриц, с которыми нам придётся работать, существенно увеличился, по сравнению с предыдущим вариантом. Это, конечно, закономерно, ведь в рекурсивном автомате для предыдущего варианта меньше состояний, а значит и матрица смежности имеет меньший размер.

Транзитивное замыкание.

$$
tc(M_3) =
$$

$$
= \left(\begin{array}{ccccccc|ccccccc|ccccccc|ccccccc|ccccccc}
. & . & . & . & . & . & . & . & [a] & . & . & . & . & . & . & [aS] & . & . & . & . & . & . & . & [aSb] & . & . & . & . & . & . & [aSbS] & . & . & . & . \\
. & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . \\
. & . & . & . & . & . & . & . & . & . & [a] & . & . & . & . & . & . & [aS] & . & . & . & . & . & . & . & [aSb] & . & . & . & . & . & . & [aSbS] & . & . \\
. & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . \\
. & . & . & . & . & . & . & . & . & . & . & . & [a] & . & . & . & . & . & . & [aS] & . & . & . & . & . & . & . & [aSb] & . & . & . & . & . & . & [aSbS] \\
. & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . \\
. & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . \\
\hline
. & . & . & . & . & . & . & . & . & . & . & . & . & . & [S] & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . \\
. & . & . & . & . & . & . & . & . & . & . & . & . & . & . & [S] & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . \\
. & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & [S] & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . \\
. & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & [S] & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . \\
. & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & [S] & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . \\
. & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & [S] & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . \\
. & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & [S] & . & . & . & . & . & . & . & . & . & . & . & . & . & . \\
\hline
. & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . \\
. & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & [b] & . & . & . & . & . & . & . & . & . & . & . \\
. & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . \\
. & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & [b] & . & . & . & . & . & . & . & . & . \\
. & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . \\
. & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & [b] & . & . & . & . & . & . & . \\
. & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . \\
\hline
. & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & [S] & . & . & . & . & . & . \\
. & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & [S] & . & . & . & . & . \\
. & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & [S] & . & . & . & . \\
. & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & [S] & . & . & . \\
. & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & [S] & . & . \\
. & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & [S] & . \\
. & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & [S] \\
\hline
. & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . \\
. & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . \\
. & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . \\
. & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . \\
. & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . \\
. & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . \\
. & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & .
\end{array}\right) \tag{17}
$$

Обновлённый граф



Его матрица смежности:

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

Потребуется ещё одна итерация.

$$M_3 = M_1^1 \otimes M_2 = \begin{pmatrix} . & [a] & . & . & . \\ . & . & [S] & . & . \\ . & . & . & [b] & . \\ . & . & . & . & [S] \\ . & . & . & . & . \end{pmatrix} \otimes \begin{pmatrix} [S] & [a] & [S] & . & . & . & . \\ . & [S] & [b] & . & . & . & . \\ . & . & [S] & [a] & [S] & . & . \\ . & . & . & [S] & [b] & . & . \\ . & . & . & . & [S] & [a] & [S] \\ . & . & . & . & . & [S] & [b] \\ . & . & . & . & . & . & [S] \end{pmatrix} =$$

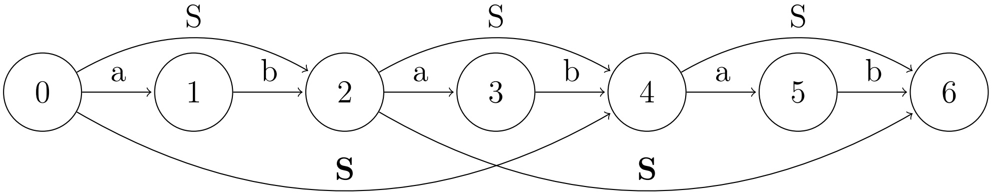
$$= \left(\begin{array}{ccccccc|ccccccc|ccccccc|ccccccc|ccccccc}
. & . & . & . & . & . & . & . & [a] & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . \\
. & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . \\
. & . & . & . & . & . & . & . & . & . & [a] & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . \\
. & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . \\
. & . & . & . & . & . & . & . & . & . & . & . & [a] & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . \\
. & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . \\
. & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . \\
\hline
. & . & . & . & . & . & . & . & . & . & . & . & . & . & [S] & . & \mathbf{[S]} & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . \\
. & . & . & . & . & . & . & . & . & . & . & . & . & . & . & [S] & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . \\
. & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & [S] & . & \mathbf{[S]} & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . \\
. & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & [S] & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . \\
. & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & [S] & . & \mathbf{[S]} & . & . & . & . & . & . & . & . & . & . & . & . & . & . \\
. & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & [S] & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . \\
. & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & [S] & . & . & . & . & . & . & . & . & . & . & . & . & . & . \\
\hline
. & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . \\
. & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & [b] & . & . & . & . & . & . & . & . & . & . & . \\
. & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . \\
. & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & [b] & . & . & . & . & . & . & . & . & . \\
. & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . \\
. & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & [b] & . & . & . & . & . & . & . \\
. & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . \\
\hline
. & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & [S] & . & \mathbf{[S]} & . & . & . & . \\
. & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & [S] & . & . & . & . & . \\
. & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & [S] & . & \mathbf{[S]} & . & . \\
. & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & [S] & . & . & . \\
. & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & [S] & . & \mathbf{[S]} \\
. & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & [S] & . \\
. & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & [S] \\
\hline
. & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . \\
. & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . \\
. & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . \\
. & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . \\
. & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . \\
. & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . \\
. & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & .
\end{array}\right) \tag{18}$$

Транзитивное замыкание.

$tc(M_3) =$

$$
= \left(\begin{array}{ccccccc|ccccccc|ccccccc|ccccccc|ccccccc}
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & [a] & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & [\mathrm{aS}] & \cdot & [\mathbf{aS}] & \cdot & \cdot & \cdot & \cdot & \cdot & [\mathrm{aSb}] & \cdot & [\mathbf{aSb}] & \cdot & \cdot & \cdot & \cdot & [\mathrm{aSbS}] & \cdot & [\mathbf{aSbS}] & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & [a] & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & [\mathrm{aS}] & \cdot & [\mathbf{aS}] & \cdot & \cdot & \cdot & \cdot & \cdot & [\mathrm{aSb}] & \cdot & [\mathbf{aSb}] & \cdot & \cdot & \cdot & \cdot & [\mathrm{aSbS}] & \cdot & [\mathbf{aSbS}] \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & [a] & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & [\mathrm{aS}] & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & [\mathrm{aSb}] & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & [\mathrm{aSbS}] \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
\hline
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & [S] & \cdot & [S] & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & [S] & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & [S] & \cdot & [S] & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & [S] & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & [S] & \cdot & [S] & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & [S] & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & [S] & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
\hline
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & [b] & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & [b] & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & [b] & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
\hline
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & [S] & \cdot & [S] & \cdot & \cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & [S] & \cdot & \cdot & \cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & [S] & \cdot & [S] & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & [S] & \cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & [S] & \cdot & [S] \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & [S] & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & [S] \\
\hline
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot
\end{array}\right) \tag{19}
$$

Обновлённый граф. На этом шаге мы смогли “склеить” из подстрок, вoводимых из S, более длинные пути. Однако, согласно правилам грамматики, мы согли “скленить” только две подстроки в единое целое.



Матрица смежности обновлённого графа:

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

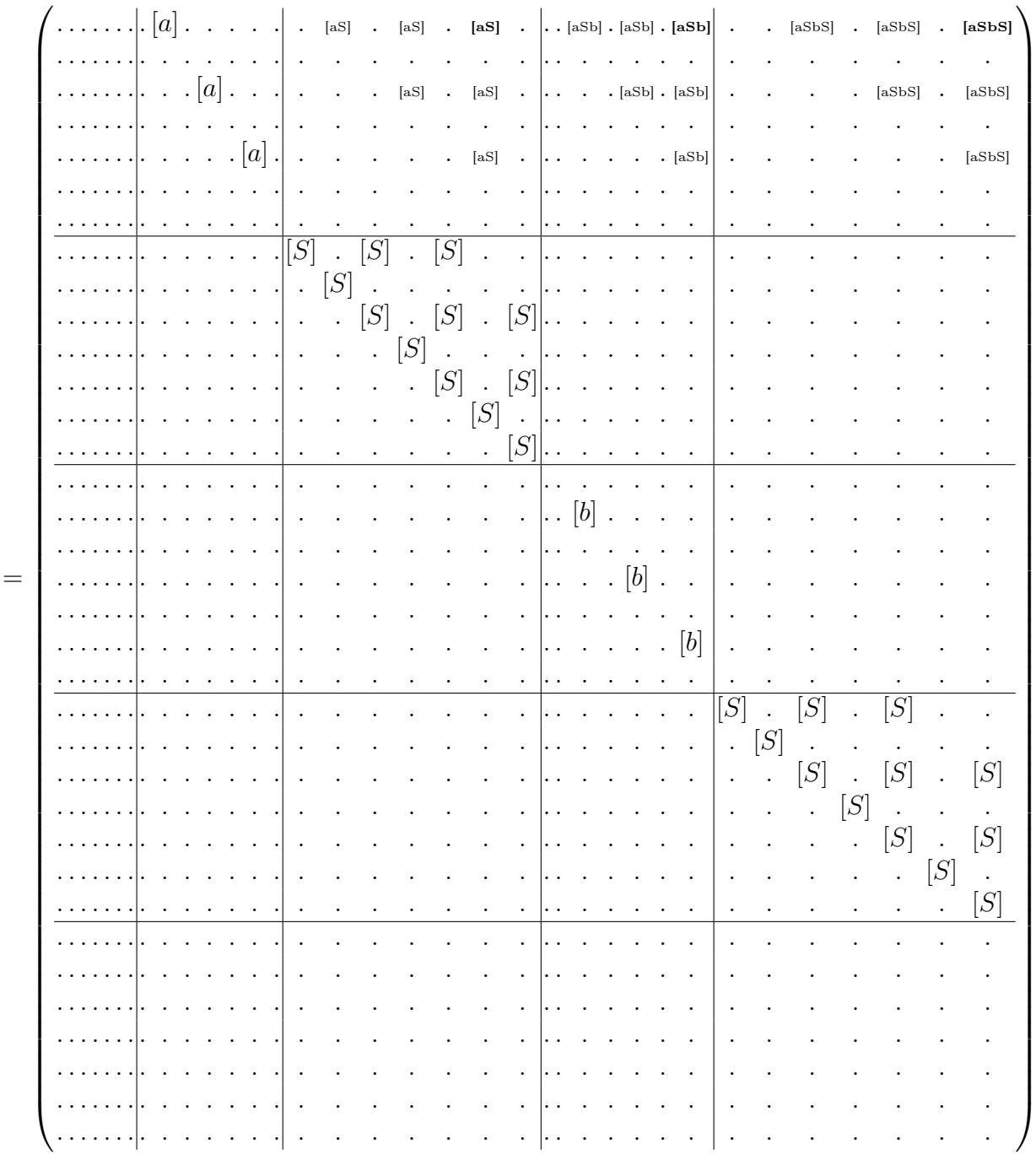
И, наконец, последняя содержательная итерация.

$$
M_3 = M_1^1 \otimes M_2 = \begin{pmatrix} . & [a] & . & . & . \\ . & . & [S] & . & . \\ . & . & . & [b] & . \\ . & . & . & . & [S] \\ . & . & . & . & . \end{pmatrix} \otimes \begin{pmatrix} [S] & [a] & [S] & . & [S] & . & . \\ . & [S] & [b] & . & . & . & . \\ . & . & [S] & [a] & [S] & . & [S] \\ . & . & . & [S] & [b] & . & . \\ . & . & . & . & [S] & [a] & [S] \\ . & . & . & . & . & [S] & [b] \\ . & . & . & . & . & . & [S] \end{pmatrix} =
$$

$$
= \left(\begin{array}{ccccccc|ccccccc|ccccccc|ccccccc|ccccccc}
. & . & . & . & . & . & . & . & [a] & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . \\
. & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . \\
. & . & . & . & . & . & . & . & . & . & [a] & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . \\
. & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . \\
. & . & . & . & . & . & . & . & . & . & . & . & [a] & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . \\
. & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . \\
. & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . \\
\hline
. & . & . & . & . & . & . & . & . & . & . & . & . & . & [S] & . & [S] & . & \mathbf{[S]} & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . \\
. & . & . & . & . & . & . & . & . & . & . & . & . & . & . & [S] & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . \\
. & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & [S] & . & [S] & . & \mathbf{[S]} & . & . & . & . & . & . & . & . & . & . & . & . & . & . \\
. & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & [S] & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . \\
. & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & [S] & . & [S] & . & . & . & . & . & . & . & . & . & . & . & . & . & . \\
. & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & [S] & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . \\
. & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & [S] & . & . & . & . & . & . & . & . & . & . & . & . & . & . \\
\hline
. & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . \\
. & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & [b] & . & . & . & . & . & . & . & . & . & . & . \\
. & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . \\
. & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & [b] & . & . & . & . & . & . & . & . & . \\
. & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . \\
. & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & [b] & . & . & . & . & . & . & . \\
. & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . \\
\hline
. & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & [S] & . & [S] & . & \mathbf{[S]} & . & . \\
. & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & [S] & . & . & . & . & . \\
. & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & [S] & . & [S] & . & \mathbf{[S]} \\
. & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & [S] & . & . & . \\
. & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & [S] & . & [S] \\
. & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & [S] & . \\
. & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & [S] \\
\hline
. & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . \\
. & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . \\
. & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . \\
. & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . \\
. & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . \\
. & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . \\
. & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & .
\end{array}\right) \tag{20}
$$

Транзитивное замыкание.

$tc(M_3) =$



(21)

В конечном итоге мы получаем такой же результат как и при первом запуске. Однако нам потребовалось выполнить существенно больше итераций внешнего цикла, а именно четыре (три содержательных и одна проверочная). При этом, в ходе работы нам пришлось оперировать сущесвенно бóльшими матрицами: 35×35 против 21×21

Таким образом — минимизация рекурсивного автомата как конечного автомата над смешанным алфавитом может быть полезна.

6.5 Замечания о реализации

Блочная структура матриц даёт хорошую основу для распределённого умнодения при построении транзитивного замыкания.

Кодирование данных в ячейках.

Транзитивное замыкание.

6.6 Вопросы и задачи

1. Оценить пространсвенную сложность алгоритма.
2. Оценить временную сложность алгоритма.
3. Найти библиотеку для тензорного произведения. Реализовать алгоритм. Можно предполагать, что запросы содержат ограниченное число терминалов и нетерминалов. Провести замеры. Сравнить с матричным.
4. Реализовать распределённое решение. См. блочную структуру

7 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 искомое 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.

Прежде, чем двинуться дальше, рекомендуется вспомнить всё, что касается деревьев вывода 3.2.

7.1 Лес разбора как представление контекстно-свободной грамматики

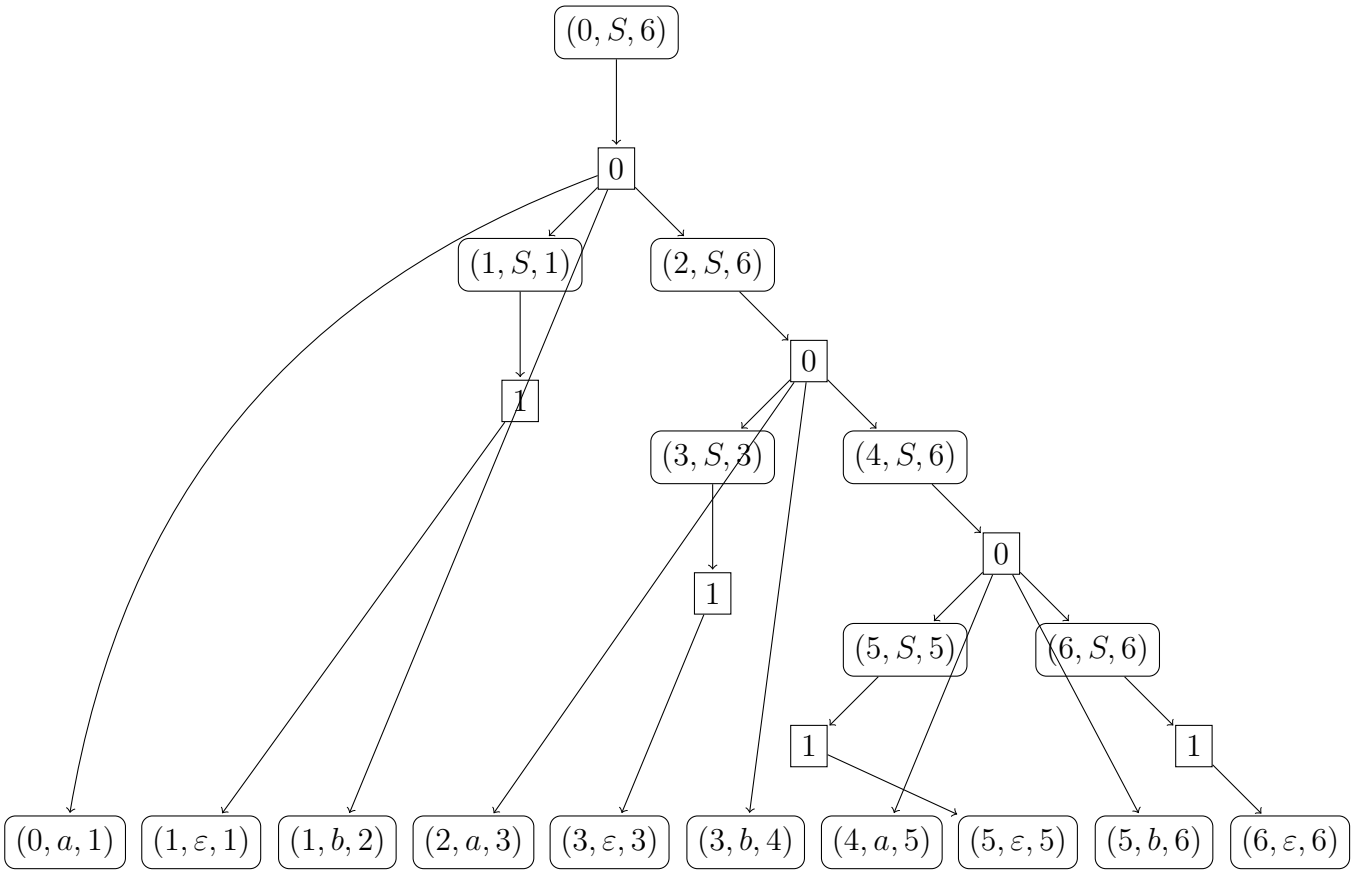
Для начала нам потребуется внести некоторые изменения в конструкцию дерева вывода.

Во-первых, заметим, что в дереве вывода каждый узел соответсвует выводу какой-то подстроки с известными позициями начала и конца. Давайте будем сохранять эту информацию в узлах дерева. Таким образом, метка любого узла это тройка вида (i, q, j), где i — координата начала подстроки, соответствующей этому узлу, j — координата конца, $q \in \Sigma \cup N$ — метка как в исходном определении.

Во-вторых, заметим, что внутренний узел со своими сыновьями связаны с продукцией в грамматике: узел появляется благодаря применению конкретной продукции в процессе вывода. Давайте занумеруем все продукции в грамматике и добавим в дерево вывода ещё один тип узлов (дополнительные узлы), в которых будем хранить номер применённой продукции. Получим следующую конструкцию: непосредственный предок дополнительного узла — это левая часть продукции, а непосредственные сыновья дополнительного узла — это правая часть продукции.

Пример 7.1. Построим модифицированное дерево вывода цепочки ${}_0a_1b_2a_3b_4a_5b_6$ в грамматике

$$
\begin{aligned}
G = \langle & \quad \{a, b\}, \{S\}, S, \\
& \{ \\
& \quad (0) S \to a\ S\ b\ S, \\
& \quad (1) S \to \varepsilon \\
& \} \\
& \rangle
\end{aligned}
$$

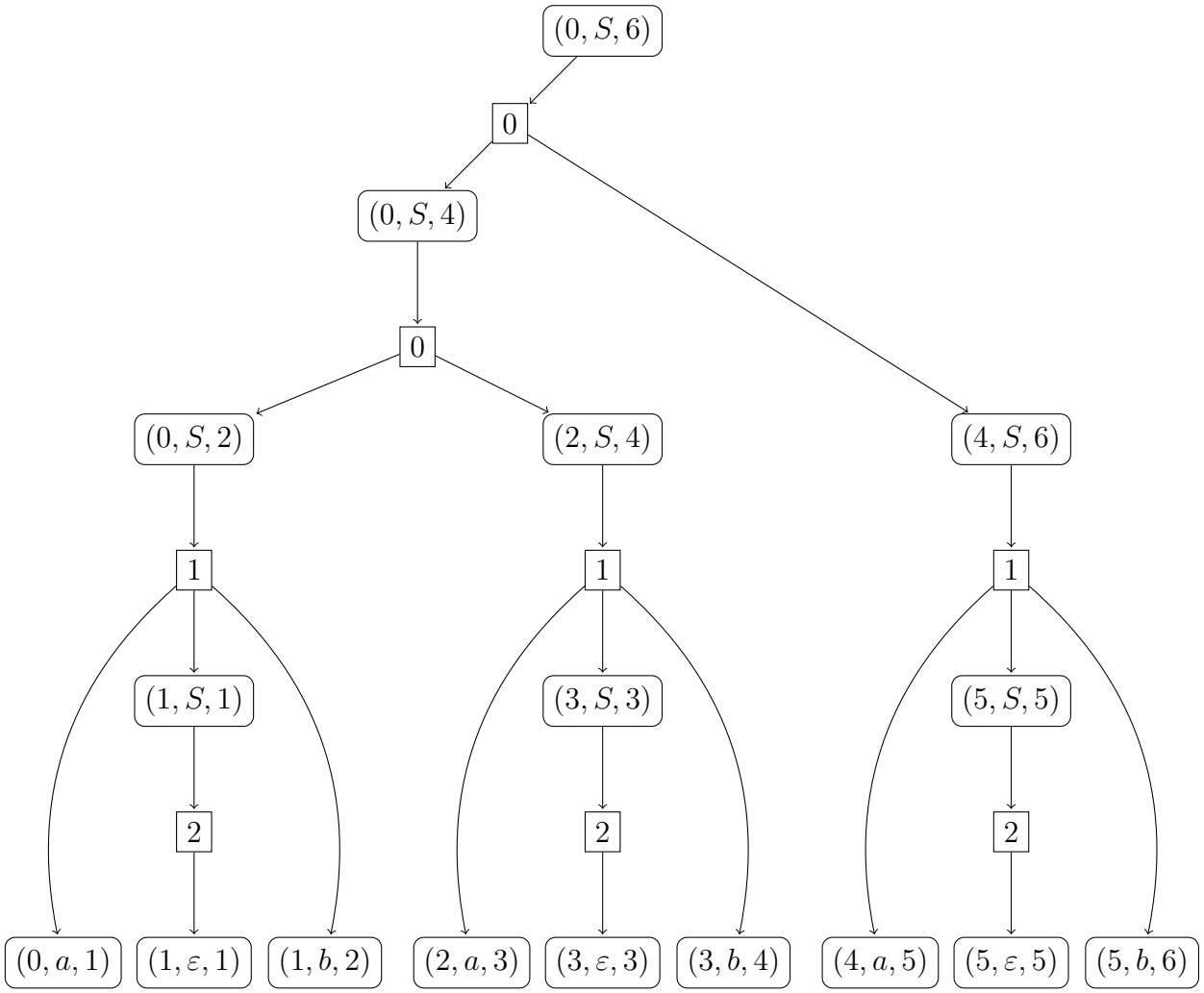


Сохраняемая нами дополнительная информация позволит переиспользовать узлы в том случае, если деревьев вывода оказалось несколько (в случае неоднозначной грамматики). При этом мы можем не бояться, что переиспользование узлов может привести к появлению ранее несуществовавших деревьев вывода, так как дополнительная информация позволяет делать только "безопасные" склейки и затем восстанавливать только корректные деревья.

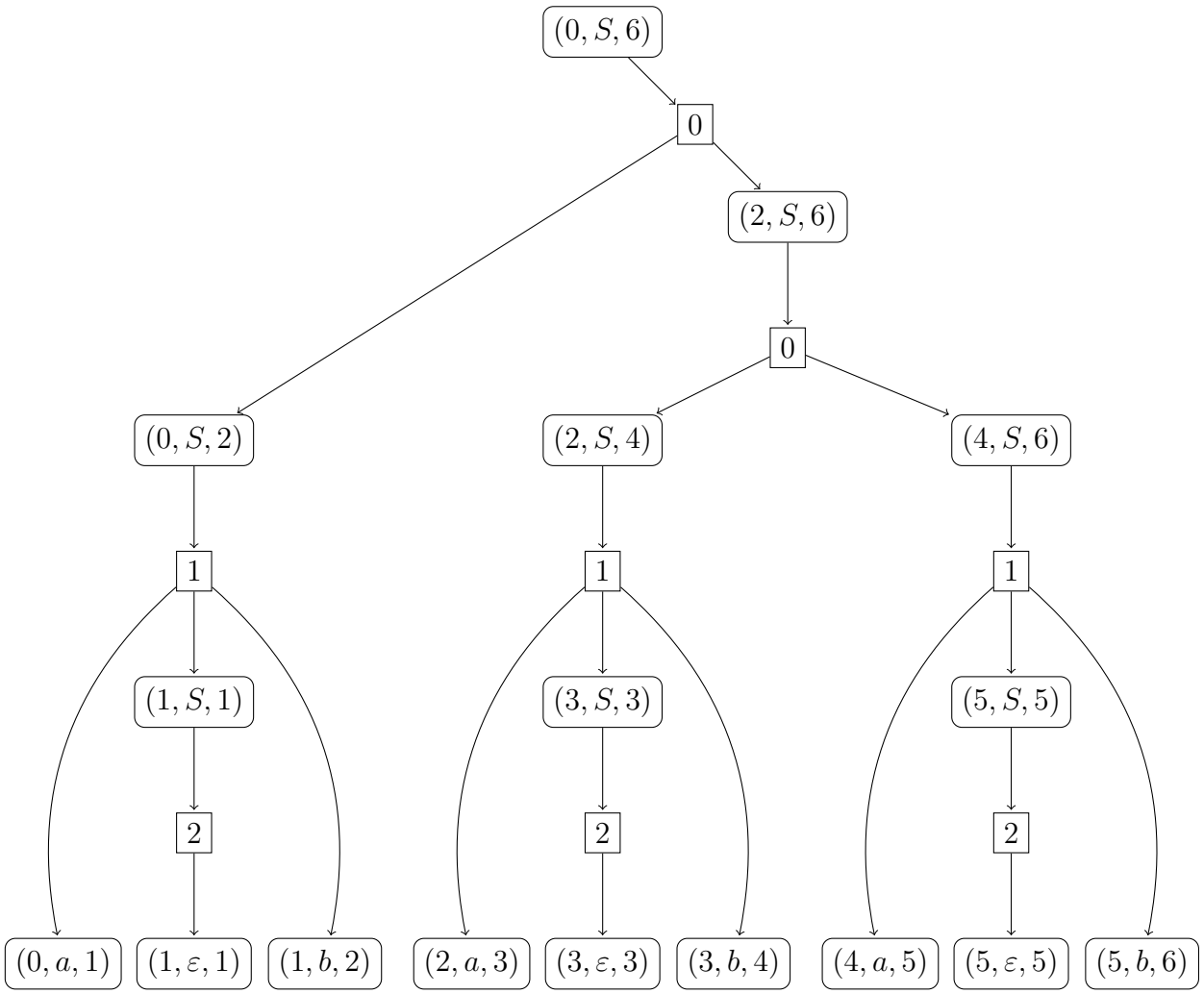
Пример 7.2. Сжатие леса вывода. Построим несколько деревьев вывода цепочки ${}_0a_1b_2a_3b_4a_5b_6$ в грамматике

$$
\begin{aligned}
G_1 = \langle & \qquad\qquad \{a, b\}, \{S\}, S, \\
& \{ \\
& \qquad\qquad (0) S \to SS, \\
& \qquad\qquad (1) S \to a\ S\ b, \\
& \qquad\qquad (2) S \to \varepsilon \\
& \} \\
& \qquad\qquad \rangle
\end{aligned}
$$

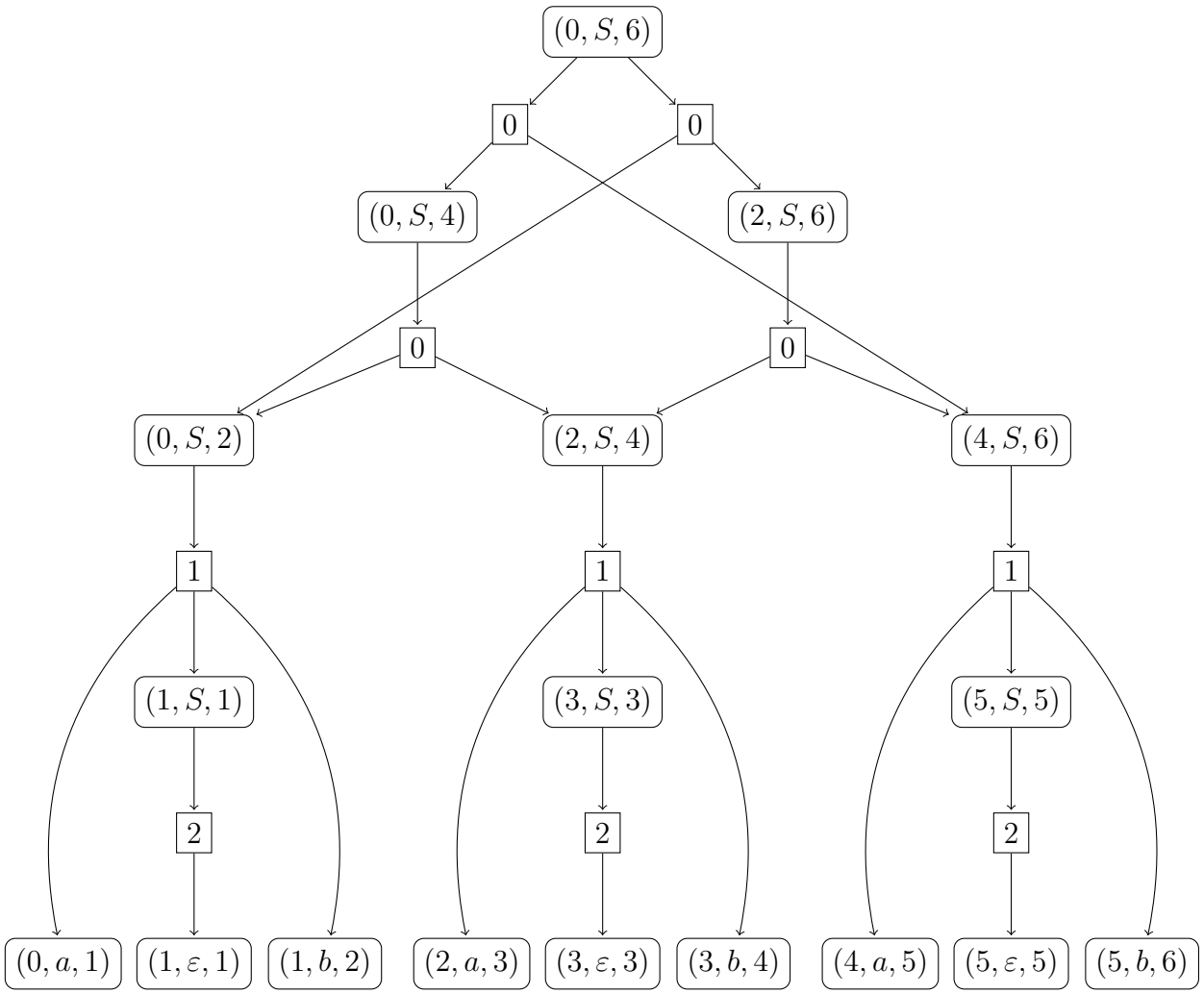
Пердположим, что мы строим левосторонний вывод. Тогда после первого применеия продукции 0 у нас есть два варианта переписывания первого нетерминала: либо с применением продукции 0, либо с применением продукции 1. В первом случае мы примени переписываение по подукции 0. Дальнейшие шаги деретерминированы и в результате мы получим следующее дерево разбора:



Второй вариант — применить продукцию 1. Остальные шаги также детерминированы. Тогда мы получим следующее дерево вывода:



В двух построенных деревьях большое количество одинаковых узлов. Построим структуру, которая содержит оба дерева и при этом никакие нетерминальные и терминальные узлы не встречаются дважды. В результате мы молучим следующий граф:



Мы получили очень простой вариант сжатого представления леса разбора (Shared Packed Parse Forest, SPPF). Впервые подобная идея была предложена Джоаном Рекерсом в его кандидатской диссертации [19]. В дальнейшем она нашла широкое применеие в обобщённом (generalized) синтаксическом анализе и получила серьёзное развитие. В частности, наш вариант, хоть и позволяет избежать экспоненциального разростания леса разбора, всё же не является оптимальным. Оптимальное асимптотическое поведение достигается при использовании бинаризованного SPPF [6] — в этом случае объём леса составляет $O(n^3)$, где n — это длина входной строки.

SPPF применяется в таких алгоритмах синтаксического анализа, как RNGLR [21], бинаризованная верся SPPF в BRNGLR [23] и GLL [22, 2].

Вообще говоря, никто не мешает иметь в этом лесу циклы. При линейном случае тоже могут быть — эпсилон-циклы.

А у нас будут произвольные.

Давайте пока методом пристольного взгляда.

Заметим, что это грамматика..

Ну а дальше будем смотреть на алгоритмы, которые что-то таое умеют строить.

7.2 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.

8 Алгоритм на основе восходящего анализа

Наша реализация [24]

8.1 Восходящий синтаксический анализ

Основы LR-анализа.

Таблицы, конфликты.

Пример автомата и таблиц.

Немного про рекурсивно-восходящй анализ.

8.2 Кс запросы

с [21]

8.3 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.
5. Реализовать рекурствно-восходящий анализ
6. Реализовать !!!!

9 Алгоритм на основе нисходящего анализа

GLL [9]

Другие реализации [13]

9.1 Нисходящий синтаксический анализ

Рекурсивный спуск, LL, таблицы, неоднозначности, левая рекурсия.

9.2 GLL для КС запросов

9.3 Вопросы и задачи

1. Задача 1
2. Задача 2

10 Комбинаторы для КС апросов

10.1 Парсер комбинаторы

Что это, с чем едят, плюсы, минусы. Про семантику, безопасность, левую рекурсию и т.д. Набор примитивных парсеров и функций, которые умеют из существующих арсеров строить более сложные (собственно, комбинаторы парсеров).

Разобрать символ, разобрать последовательность, разобрать альтернативу. впринципе, этого достаточно, но это не очень удобно.

Проблемы с левой рекурсией. Существуют решения. Одно из них — Meerkat. Подробно про него?

10.2 Комбинаторы для КС запросов

Вообще говоря, идея использовать комбинаторы для навигации по графам достаточно очевидно и не нова. немного про Trails [10].

Комбинаторы для запросов к графам на основе Meerkat [25]

Обобщённые запросы, типобезопасность и всё такое. Примеры запросов.

10.3 Вопросы и задачи

1. Реализовать библиотеку парсер комбинаторов.
2. Что-нибудь полезное с ними сделать.

11 Производные для КС запросов

11.1 Производные

Общая теория.

Определения.

11.2 Парсинг на производных

Статьи [14, 1, 15, 3] Реализации. На Scala [3], на Racket [4].

11.3 Адоптация для КС запросов

Для регулярных запросов над графами [17]. Хорошо работают в распределённых системах, в которых реализовван параллелизм уровня вершин. Например Google Pregel.

11.4 Вопросы и задачи

1. Предъявить несколько выводов для одной цепочки.
2. Построить выводы
3. Построить деревья вывода !!! Перенести из раздела про SPPF

12 От CFPQ к вычислению Datalog-запросов

12.1 Datalog

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

12.2 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.

[3] `https://github.com/djspiewak/parseback`

[4] `https://bitbucket.org/ucombinator/derp-3/src/86bca8a720231e010a3ad6aefd1aa1c0f35cbf6b/src/derp.rkt?at=master&fileviewer=file-view-default`

12.3 Обобщение GLL для вычисления Datalog-запросов

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

12.4 Вопросы и задачи

1. Написать синтаксический анализатор раз.
2. Написать синтаксический анализатор два.
3. Побаловаться с неоднозначными грамматиками
4. Побаловаться с конъюнктивными грамматиками.
5. Графы?

Список литературы

[1] M. D. Adams, C. Hollenbeck, and M. Might. On the complexity and performance of parsing with derivatives. In *Proceedings of the 37th ACM SIGPLAN Conference on Programming Language Design and Implementation*, PLDI '16, pages 224–236, New York, NY, USA, 2016. ACM.

[2] A. Afroozeh and A. Izmaylova. Faster, practical gll parsing. In B. Franke, editor, *Compiler Construction*, pages 89–108, Berlin, Heidelberg, 2015. Springer Berlin Heidelberg.

[3] L. Andersen. Parsing with derivatives.

[4] 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.

[5] O. Bastani, S. Anand, and A. Aiken. Specification inference using context-free language reachability. In *ACM SIGPLAN Notices*, volume 50, pages 553–566. ACM, 2015.

[6] S. Billot and B. Lang. The structure of shared forests in ambiguous parsing. In *Proceedings of the 27th Annual Meeting on Association for Computational Linguistics*, ACL '89, pages 143–151, Stroudsburg, PA, USA, 1989. Association for Computational Linguistics.

[7] 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.

[8] T. M. Chan. All-pairs shortest paths with real weights in $o(n^3/log\,n)$ time. *Algorithmica*, 50(2):236–243, Feb 2008.

[9] 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.

[10] D. Kröni and R. Schweizer. Parsing graphs: Applying parser combinators to graph traversals. In *Proceedings of the 4th Workshop on Scala*, SCALA '13, pages 7:1–7:4, New York, NY, USA, 2013. ACM.

[11] 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.

[12] L. Lee. Fast context-free grammar parsing requires fast boolean matrix multiplication. *J. ACM*, 49(1):1–15, Jan. 2002.

[13] 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.

[14] M. Might and D. Darais. Yacc is dead. *CoRR*, abs/1010.5023, 2010.

[15] M. Might, D. Darais, and D. Spiewak. Parsing with derivatives: A functional pearl. *SIGPLAN Not.*, 46(9):189–195, Sept. 2011.

[16] 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.

[17] M. Nolé and C. Sartiani. Regular path queries on massive graphs. In *Proceedings of the 28th International Conference on Scientific and Statistical Database Management*, SSDBM '16, pages 13:1–13:12, New York, NY, USA, 2016. ACM.

[18] P. Pratikakis, J. S. Foster, and M. Hicks. Existential label flow inference via cfl reachability. In *SAS*, volume 6, pages 88–106. Springer, 2006.

[19] J. G. Rekers. *Parser generation for interactive environments.* PhD thesis, Universiteit van Amsterdam, 1992.

[20] T. Reps. Program analysis via graph reachability. In *Proceedings of the 1997 International Symposium on Logic Programming*, ILPS '97, pages 5–19, Cambridge, MA, USA, 1997. MIT Press.

[21] E. Scott and A. Johnstone. Right nulled glr parsers. *ACM Trans. Program. Lang. Syst.*, 28(4):577–618, July 2006.

[22] E. Scott and A. Johnstone. Gll parsing. *Electron. Notes Theor. Comput. Sci.*, 253(7):177–189, Sept. 2010.

[23] E. Scott, A. Johnstone, and R. Economopoulos. Brnglr: A cubic tomita-style glr parsing algorithm. *Acta Inf.*, 44(6):427–461, Sept. 2007.

[24] E. Verbitskaia, S. Grigorev, and D. Avdyukhin. Relaxed parsing of regular approximations of string-embedded languages. In M. Mazzara and A. Voronkov, editors, *Perspectives of System Informatics*, pages 291–302, Cham, 2016. Springer International Publishing.

[25] 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.

[26] V. V. Williams and R. Williams. Subcubic equivalences between path, matrix and triangle problems. In *Proceedings of the 2010 IEEE 51st Annual Symposium on Foundations of Computer Science*, FOCS '10, pages 645–654, Washington, DC, USA, 2010. IEEE Computer Society.

[27] M. Yannakakis. Graph-theoretic methods in database theory. In *Proceedings of the ninth ACM SIGACT-SIGMOD-SIGART symposium on Principles of database systems*, pages 230–242. ACM, 1990.

[28] X. Zheng and R. Rugina. Demand-driven alias analysis for c. In *Proceedings of the 35th Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, POPL '08, pages 197–208, New York, NY, USA, 2008. ACM.