Приложения теории формальных языков и синтаксического анализа

Семён Григорьев

12 ноября 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 таком 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.

Подобный класс задач часто возникает в областях, связанных с анализом данных и активно исследуется [8, 4, 31, 68, 7, 8]. Исследуются как классы языков, так и различные постановки задачи.

Применения в статическом анализе кода.

Пример 1.1. Примеры

Применения в графовых базах данных.

Пример 1.2. Примеры

Ещё где-нибудь.

Структура данной работы такова.

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

Далее рассмотрим различные варианты постановки задачи, обсудим базовые свойства задач, их разрешимость и т.д..

И различные алгоритмы решения. Обычно в связке: классический синтаксический анализ — адоптация для обработки графов.

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 последующей работе.

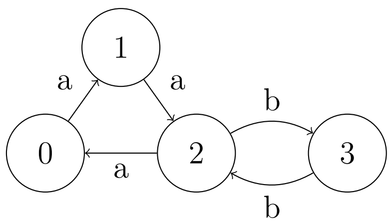
В дальнейшем нам будут нужны конечные ориентированные помеченные графы. Мы будем использовать термин *граф* подразумевая именно конечный ориентированный помеченный граф, если только не оговорено противное.

2.1 Основные определения

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

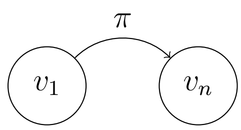
Пример 2.1 (Пример графа и его графического представления)**.** Пусть граф $\mathcal{G}_1 = \langle \{0, 1, 2, 3\}, \{(0, a, 1), (1, a, 2), (2, a, 0), (2, b, 3), (3, b, 2)\}, \{a, b\} \rangle$. Графическое представление графа $\mathcal{G}_1$:



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 обозначать 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: $(v_i, v_j) \in P \iff \exists v_i \pi v_j$.

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.

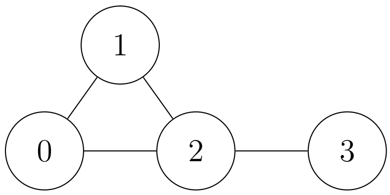
Один из способов задать граф — это задать его *матрицу смежности*.

Определение 2.5. *Матрица смежности* графа $\mathcal{G} = \langle V, E, L \rangle$ — это квадратная матрица M размера $n \times n$, где $|V| = n$ и 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.

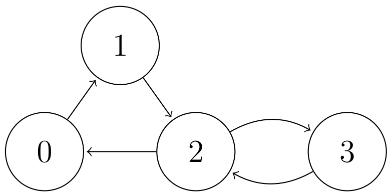
Пример 2.4. Неориентированный граф:



И его матрица смежности:

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

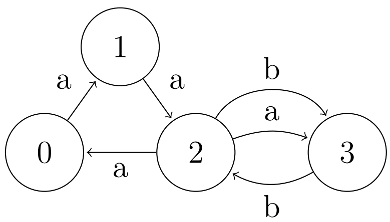
Пример 2.5. Ориентированный граф:



И его матрица смежности:

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

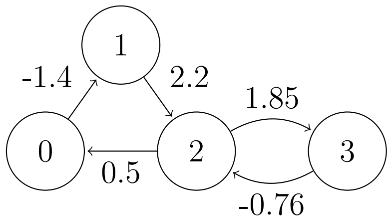
Пример 2.6. Помеченный граф:



И его матрица смежности:

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

Пример 2.7. 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 изложения конкретного материала, будут даны в соответствующих главах.

2.2 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 стороны, 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 (англ. APSP — all-pairs shortest paths)*

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

2.3 APSP и транзитивное замыкание графа

Заметим, что отношение достижимости (2.4) является транзитивным. 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.

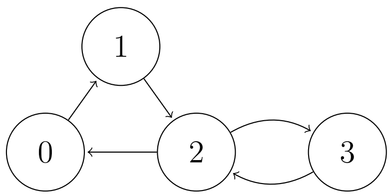
Определение 2.6. *Транзитивным замыканием графа* называется транзитивное замыкание отношения достижимости по всему графу.

Как несложно заметить, транзитивное замыкание графа — это тоже граф. Более того, если решить задачу поиска кратчайших путей между всеми парами вершин, то мы построим транзитивное замыкание. Поэтому сразу рассмотрим алгоритм Флойда-Уоршелла [23, 53, 70], который является общим алгоритмом поиска кратчайших путей (умеет обрабатывать рёбра с отрицательными весами, чего не может, например, алгоритм Дейкстры). Его сложность: $O(n^3)$, где n — количество вершин в графе. При этом, данный алгоритм легко упростить до алгоритма построения транзитивного замыкания.

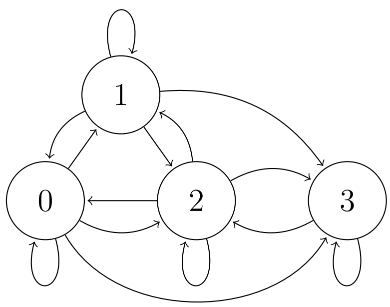
Listing 1 Алгоритм Флойда-Уоршелла

function FloydWarshall($\mathcal{G}$)
    $M \leftarrow$ матрица смежности $\mathcal{G}$
    $n \leftarrow |V(\mathcal{G})|$
    for k = 0; k < n; k++ **do**
        for i = 0; i < n; i++ **do**
            for j = 0; j < n; j++ **do**
                $M[i,j] \leftarrow \min(M[i,j], M[i,k] + M[k,j])$
    return M

Пример 2.8. Пусть дан следующий граф:



Построим его транзитивное замыкание:



Remark. На самом деле, петли у вершины 3 может и не быть, т.к. это зависит от определения.

Remark. Приведем список интересных работ по APSP для ориентированных графов с вещественными весами (здесь n — количество вершин в графе):

- M.L. Fredman (1976) — $O(n^3(\log\log n/\log n)^{\frac{1}{3}})$ — использование дерева решений [24]
- W. Dobosiewicz (1990) — $O(n^3/\sqrt{\log n})$ — использование операций на Word Random Access Machine [20]
- T. Takaoka (1992) — $O(n^3\sqrt{\log\log n/\log n})$ — использование таблицы поиска [61]
- Y. Han (2004) — $O(n^3(\log\log n/\log n)^{\frac{5}{7}})$ [26]
- T. Takoaka (2004) — $O(n^3(\log\log n)^2/\log n)$ [62]
- T. Takoaka (2005) — $O(n^3\log\log n/\log n)$ [63]
- U. Zwick (2004) — $O(n^3\sqrt{\log\log n}/\log n)$ [75]
- T.M. Chan (2006) — $O(n^3/\log n)$ — многомерный принцип "разделяй и властвуй" [16]
- и др.

2.4 APSP и произведение матриц

Алгоритм Флойда-Уоршелла можно переформулировать в терминах перемножения матриц. Для этого введём обозначение.

Определение 2.7. Пусть даны матрицы A и B размера $n \times n$. Определим операцию $\star$, которую называют *Min-plus matrix multiplication*:

$A \star B = C$ — матрица размера $n \times n$, т.ч. $C[i,j] = \min_{k=1\ldots n}\{A[i,k] + B[k,j]\}$

Также, обозначим за $D[i,j](k)$ минимальную длину пути из вершины i в j, содержащий максимум k рёбер. Таким образом, $D(1) = M$, где M — это матрица смежности, а решением APSP является $D(n-1)$, т.к. чтобы соединить n вершин требуется не больше $n-1$ рёбер.

$$\begin{gathered}
D(1) = M \\
D(2) = D(1) \star M = M^2 \\
D(3) = D(2) \star M = M^3 \\
\cdots \\
D(n-1) = D(n-2) \star M = M^{n-1}
\end{gathered}$$

Итак, мы можем решить APSP за $O(nK(n))$, где $K(n)$ — сложность алгоритма умножения матриц. Сразу заметим, что, например, $D(3)$ считать не обязательно, т.к. можем посчитать $D(4)$ как $D(2) \star D(2)$. Поэтому:

$$\begin{gathered}
D(1) = M \\
D(2) = M^2 = M \star M \\
D(4) = M^4 = M^2 \star M^2 \\
D(8) = M^8 = M^4 \star M^4 \\
\cdots \\
D(2^{\log(n-1)}) = M^{2^{\log(n-1)}} = M^{2^{\log(n-1)-1}} \star M^{2^{\log(n-1)-1}} \\
D(n-1) = D(2^{\log(n-1)})
\end{gathered}$$

Теперь вместо n итераций нам нужно $\log n$. В итоге, сложность — $O(\log nK(n))$. Данный алгоритм называется *repeated squaring*[1].

[1]Пример решения APSP с помощью repeated squaring: `http://users.cecs.anu.edu.au/~Alistair.Rendell/Teaching/apac_comp3600/module4/all_pairs_shortest_paths.xhtml`

2.5 Умножение матриц с субкубической сложностью

В предыдущем подразделе мы свели проблему APSP к проблеме min-plus matrix multiplication, поэтому взглянем на эффективные алгоритмы матричного умножения.

Сложность наивного произведения двух матриц составляет $O(n^3)$, это приемлемо только для малых матриц, для больших же лучше использовать алгоритмы с субкубической сложностью. Отметим, что мы называем сложность субкубической, если она равна $O(n^{3-\varepsilon})$, где $\varepsilon > 0$, иначе говоря, меньшей, чем $O(n^3)$.

Первый субкубический алгоритм опубликовал Ф. Штрассен в 1969 году, его сложность — $O(n^{\log_2 7}) \approx O(n^{2.81})$ [60]. Основная идея — рекурсивное разбиение на блоки 2×2 и вычисление их произведения с помощью только 7 умножений, а не 8. Впоследствии алгоритмы усовершенствовались до $\approx O(n^{2.5})$ [49, 11, 54, 18]. В настоящее время наиболее асимптотически быстрым является алгоритм Копперсмита — Винограда со сложностью $O(n^{2.376})$ [19].

Несмотря на то, что у приведенных алгоритмов неплохая алгоритмическая сложность, мы всё же не можем использовать их для вычисления min-plus matrix multiplication, так как в субкубических алгоритмах требуется, чтобы элементы образовывали кольцо. Покажем, что $\mathbb{R} \cup \{+\infty\}$ с операциями min и $+$ являются полукольцом, а не кольцом:

1. $min(a, b) = min(b, a)$
2. $min(min(a, b)), c) = min(a, min(b, c)))$
3. $min(a, +\infty) = min(+\infty, a) = a$, т.е. $+\infty$ — нейтральный элемент относительно операции min
4. $(a + b) + c = a + (b + c)$
5. $min(a, b) + c = min(a + c, b + c)$
6. $a + (B \vee c) = (a + b) \vee (a + c)$
7. $a + \infty = \infty + a = \infty$
8. Но для произвольного элемента a: $\nexists d$, т.ч. $min(a, d) = min(d, a) = +\infty$, т.е. для любого элемента нет обратного относительно операции min

Таким образом, вопрос о субкубических алгоритмах решения APSP всё ещё открыт [17]. Кроме того, более простая задача APSP с булевыми матрицами также не решена за субкубическую сложность. Приведем обоснование этого факта.

Определение 2.8. Матрица называется *булевой*, если она состоит из 0 и 1.

Булевы матрицы с поэлементными операциями дизъюнкции и конъюнкции являются полукольцом. Покажем это: пусть A, B и C — булевы матрицы, тогда:

1. $A \vee B = B \vee A$
2. $(A \vee B) \vee C = A \vee (B \vee C)$
3. $A \vee N = N \vee A = A$, где N — матрица, полностью состоящая из 0
4. $(A \wedge B) \wedge C = A \wedge (B \wedge C)$

5. $(A \vee B) \wedge C = (A \wedge C) \vee (B \wedge C)$

6. $A \wedge (B \vee C) = (A \wedge B) \vee (A \wedge C)$

7. $A \wedge N = N \wedge A = N$

Булевы матрицы тоже не являются кольцом, т.к. не имеют обратный элемент относительно операции дизъюнкции (т.е. для произвольной булевой матрицы A: $\nexists D$, т.ч. D — булева матрица и $A \vee D = D \vee A = N$). Следовательно, субкубические алгоритмы не подходят для перемножения булевых матриц, т.к. в них используются обратные элементы (например, для разности). В частности, они не применимы к классической матрице смежности, которая ведёт себя как булева матрица.

2.6 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 $\Sigma = \{a, b, c, \dots, z\}$
- Кириллический алфавит $\Sigma = \{$а, б, в, $\dots$, 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: $a, b, \dots, x, y, z$.

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 $\exists \gamma_0, \dots \gamma_k : \ \alpha \Rightarrow \gamma_0 \Rightarrow \gamma_1 \Rightarrow \ldots \Rightarrow \gamma_{k-1} \Rightarrow \gamma_k \Rightarrow \beta$
- 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:

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

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

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 $S \overset{*}{\Rightarrow} \omega$.

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 $L(G) = \{\omega \in \Sigma^* \mid S \overset{*}{\Rightarrow} \omega\}$.

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.

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

Определение 3.13. *Неоднозначная грамматика* — грамматика, в которой существует 2 и более левосторонних (правосторонних) 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 левостороннего (правостороннего) 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

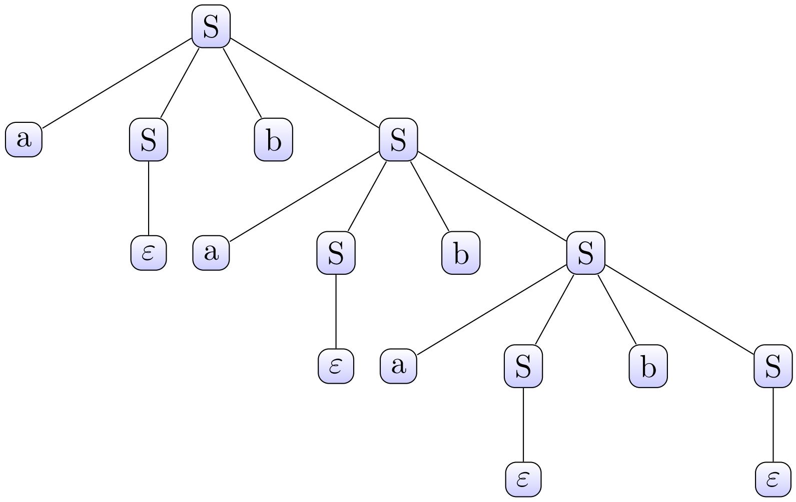
В некоторых случаях не достаточно знать порядок применения правил. Необходимо структурное представление вывода цепочки в грамматике. Таким представлением является *дерево вывода*.

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 $A, B, C \in N$, S не содержится в правой части правила
- $A \to a$, 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

Доказательство. *Цепное правило* — правило вида $A \to B$, где $A, B \in N$
Цепная пара — упорядоченная пара (A, B), в которой $A \overset{*}{\Rightarrow} B$, используя только цепные правила.

Алгоритм:

1. Найти все цепные пары в грамматике G. Найти все цепные пары можно по индукции: Базис: (A, A) — цепная пара для любого нетерминала, так как $A \overset{*}{\Rightarrow} A$ за ноль шагов. Индукция: Если пара (A, B_0) — цепная, и есть правило $B_0 \to B_1$, то (A, B_1) — цепная пара.
2. Для каждой цепной пары (A, B) добавить в грамматику G' все правила вида $A \to a$, где $B \to a$ — нецепное правило из G.
3. Удалить все цепные правила

Пусть G — контекстно-свободная грамматика. G' — грамматика, полученная в результате применения алгоритма к G. Тогда $L(G) = L(G')$. □

Определение 3.18. Нетерминал A называется *порождающим*, если из него может быть выведена конечная терминальная цепочка. Иначе он называется *непорождающим*.

Лемма 3.9. Можно удалить все бесполезные (непорождающие) нетерминалы

Доказательство. После удаления из грамматики правил, содержащих непорождающие нетерминалы, язык не изменится, так как непорождающие нетерминалы по определению не могли участвовать в выводе какого-либо слова.

Алгоритм нахождения порождающих нетерминалов:

1. Множество порождающих нетерминалов пустое.
2. Найти правила, не содержащие нетерминалов в правых частях и добавить нетерминалы, встречающихся в левых частях таких правил, в множество.
3. Если найдено такое правило, что все нетерминалы, стоящие в его правой части, уже входят в множество, то добавить в множество нетерминалы, стоящие в его левой части.
4. Повторить предыдущий шаг, если множество порождающих нетерминалов изменилось.

В результате получаем множество всех порождающих нетерминалов грамматики, а все нетерминалы, не попавшие в него, являются непорождающими. Их можно удалить. □

Пример 3.11. Приведем в Нормальную Форму Хомского однозначную грамматику правильных скобочных последовательностей: $S \to aSbS \mid \varepsilon$

Первым шагом добавим новый нетерминал и сделаем его стартовым:

$$\begin{aligned} S_0 &\to S \\ S &\to aSbS \mid \varepsilon \end{aligned}$$

Заменим все терминалы на новые нетерминалы:

$$\begin{aligned} S_0 &\to S \\ S &\to LSRS \mid \varepsilon \\ L &\to a \\ R &\to b \end{aligned}$$

Избавимся от длинных правил:

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

Избавимся от ε-продукций:

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

Избавимся от цепных правил:

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

Определение 3.19. Контекстно-свободная грамматика $\langle\Sigma, N, P, S\rangle$ находится в *ослабленной Нормальной Форме Хомского*, если она содержит только правила следующего вида:

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

То есть ослабленная НФХ отличается от НФХ тем, что:

1. ε может выводиться из любого нетерминала
2. S может появляться в правых частях правил

3.5 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 $G = \langle\{a, b\}, \{S\}, \{S \to \varepsilon \mid a\ S\ b\ |S\ S\}, S\rangle$.
3. Постройте все правосторонние выводы цепочки $w = ababab$ в грамматике $G = \langle\{a, b\}, \{S\}, \{S \to \varepsilon \mid a\ S\ b\ |S\ S\}, S\rangle$.
4. Постройте все деревья вывода цепочки $w = ababab$ в грамматике $G = \langle\{a, b\}, \{S\}, \{S \to \varepsilon \mid a\ S\ b\ |S\ S\}, S\rangle$, соответствующие левосторонним выводам.
5. Постройте все деревья вывода цепочки $w = ababab$ в грамматике $G = \langle\{a, b\}, \{S\}, \{S \to \varepsilon \mid a\ S\ b\ |S\ S\}, S\rangle$, соответствующие правосторонним выводам.

4 Задача о поиске путей с ограничениями в терминах формальных языков

В данной главе сформулируем постановку задачи о поиске путей в графе с ограничениями. А также приведём несколько примеров областей, в которых применяются алгоритмы решения этой задачи.

4.1 Постановка задачи

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

В задаче поиска путей мы можем накладывать дополнительные ограничения на путь (например, чтобы он был простым, кратчайшим или Эйлеровым [36]), но это уже другая история.

Другим вариантом постановки задачи является задача достижимости.

Определение 4.2. *Задача достижимости* заключается в поиске множества пар вершин, для которых найдется путь с началом и концом в этих вершинах, что слово, составленное из меток рёбер пути, будет принадлежать заданному языку. $\Pi' = \{(v_i, v_j) \mid \exists v_i \pi v_j, \omega(\pi) \in \mathcal{L}\}$.

При этом, множество Π может являться бесконечным, тогда как Π' конечно, по причине конечности графа $\mathcal{G}$.

Язык $\mathcal{L}$ может принадлежать разным классам и быть задан разными способами. Например, он может быть регулярным, или контекстно-свободным, или многокомпонентным контекстно-свободным.

Если $\mathcal{L}$ — регулярный, $\mathcal{G}$ можно рассматривать как недетерминированный конечный автомат (НКА), в котором все вершины и стартовые, и конечные. Тогда задача поиска путей, в которой $\mathcal{L}$ — регулярный, сводится к пересечению двух регулярных языков.

Более подробно мы рассмотрим случай, когда $\mathcal{L}$ — контекстно-свободный язык.

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

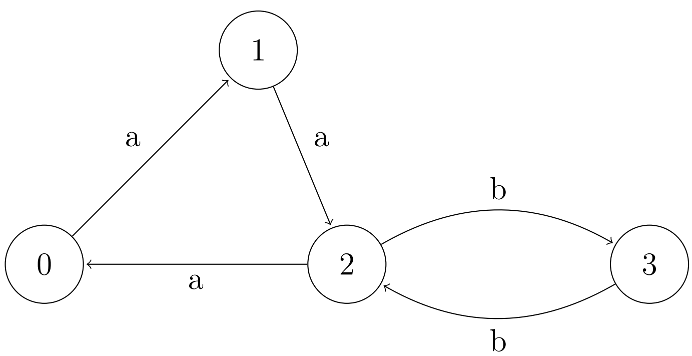
Пример 4.1. Пример задачи поиска путей.

Дана грамматика G :

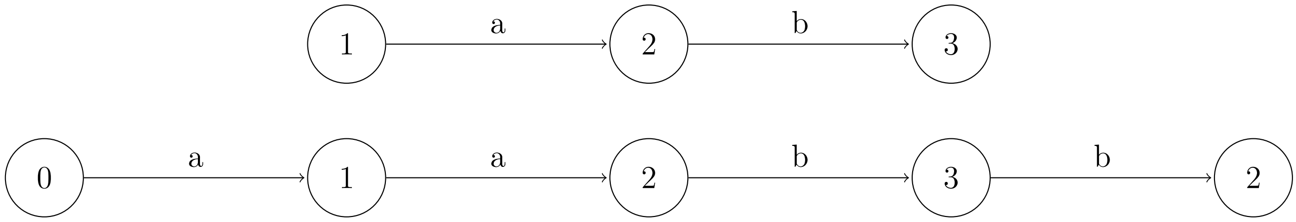
$$S \to ab$$
$$S \to aSb$$

Эта грамматика задаёт язык $\mathcal{L} = a^n b^n$.

И дан граф $\mathcal{G}$:



Тогда примерами путей, принадлежащих множеству $\Pi = \{\pi \mid \omega(\pi) \in \mathcal{L}\}$, являются:



4.2 О разрешимости задачи

Задачи из определения 4.1 и 4.2 сводятся к построению пересечения языка $\mathcal{L}$ и языка, задаваемого путями графа, R. А мы для обсуждения разрешимости задачи рассмотрим более слабую постановку задачи:

Определение 4.3. Необходимо проверить, что существует хотя бы один такой путь π для данного графа, для данного языка $\mathcal{L}$, что $\omega(\pi) \in \mathcal{L}$.

Эта задача сводится к проверке пустоты пересечения языка $\mathcal{L}$ с R — регулярным языком, задаваемым графом. От класса языка $\mathcal{L}$ зависит её разрешимость:

- Если $\mathcal{L}$ регулярный, то получаем задачу пересечения двух регулярных языков:

  $\mathcal{L} \cap R = R'$. R' — также регулярный язык. Проверка регулярного языка на пустоту — разрешимая проблема.

- Если $\mathcal{L}$ контекстно-свободный, то получаем задачу

  $\mathcal{L} \cap R = CF$ — контекстно-свободный. Проверка контекстно-свободного языка на пустоту — разрешимая проблема.

- Помимо иерархии Хомского существуют и другие классификации языков. Так например, класс конъюнктивных (Conj) языков [43] является строгим расширением контекстно-свободных языков и все так же позволяет полиномиальный синтаксический анализ.

  Пусть $\mathcal{L}$ — конъюнктивный. При пересечении конъюнктивного и регулярного языков получается конъюнктивный ($\mathcal{L} \cap R = Conj$), а проблема проверки Conj на пустоту не разрешима [45].

- Ещё один класс языков из альтернативной иерархии, не сравнимой с Иерархией Хомского, — MCFG (multiple context-free grammars) [58]. Как его частный случай — TAG (tree adjoining grammar) [33].

  Если $\mathcal{L}$ принадлежит классу MCFG, то $\mathcal{L} \cap R$ также принадлежит MCFG. Проблема проверки пустоты MCFG разрешима [58].

Существует ещё много других классификаций языков, но поиск универсальной иерархии до сих пор продолжается.

Далее, для изучения алгоритмов решения, нас будет интересовать задача $R \cap CF$.

4.3 Области применения

- Statанализ. Введено Томасом Репсом [52].
- Социальные сети [28].
- RDF обработка [73].
- Биоинформатика [59].
- Применяется для различных межпроцедурных задач [50, 9, 74].
- Графовые БД Впервые предложил Михалис Яннакакис [72].

4.4 Вопросы и задачи

1. Пусть есть граф. Задайте грамматику для 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 CYK для вычисления КС запросов

В данной главе мы рассмотрим алгоритм CYK, позволяющий установить принадлежность слова грамматике и предоставить его вывод, если таковой имеется.

Наш главный интерес заключается в возможности применения данного алгоритма для решения описанной в предыдущей главе задачи — поиска путей с ограничениями в терминах формальных языков. Как уже было указано выше, будем рассматривать случай контекстно-свободных языков.

5.1 Алгоритм CYK

Алгоритм CYK (Cocke-Younger-Kasami) — один из классических алгоритмов синтаксического анализа. Его асимптотическая сложность в худшем случае — $O(n^3 * |G|)$ (n — размер входной строки, G — входная грамматика), что выгодно выделяет его среди других алгоритмов парсинга. [32]

Для его применения необходимо, чтобы подаваемая на вход грамматика находилась в Нормальной Форме Хомского (НФХ) 3.4.

В основе алгоритма лежит принцип динамического программирования. Используются два соображения (здесь ω — слово, A, B, C — нетерминалы грамматики, a — терминал грамматики):

1. Для правила вида $A \to a$ очевидно, что из A выводится ω (с применением этого правила) тогда и только тогда, когда $a = \omega$:

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

2. Для правила вида $A \to BC$ понятно, что из A выводится ω (с применением этого правила) тогда и только тогда, когда существуют две цепочки ω_1 и ω_2 такие, что ω_1 выводима из B, ω_2 выводима из C и при этом $\omega = \omega_1\omega_2$:

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

Или в терминах позиций в строке:

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

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

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

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

Далее используем динамику: на шаге $m > 1$ предполагаем, что ячейки матрицы $M[i', j', A]$ заполнены для всех нетерминалов A и пар $i', j' : j' - i' < m$. Тогда можно заполнить ячейки матрицы $M[i, j, A]$, где $j - i = m$

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

По итогу работы алгоритма значение в ячейке $M[0, |\omega|, S]$, 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.

$$\begin{aligned}
S &\to AS_2 \mid \varepsilon \\
S_1 &\to AS_2 \\
S_2 &\to b \mid BS_1 \mid SS_3 \\
S_3 &\to b \mid BS_1 \\
A &\to a \\
B &\to b
\end{aligned}$$

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.

Шаг 1. Инициализируем матрицу элементами на главной диагонали:

$$\begin{pmatrix} \{A\} & \varnothing & \varnothing & \varnothing & \varnothing & \varnothing \\ \varnothing & \{A\} & \varnothing & \varnothing & \varnothing & \varnothing \\ \varnothing & \varnothing & \{B, S_2, S_3\} & \varnothing & \varnothing & \varnothing \\ \varnothing & \varnothing & \varnothing & \{B, S_2, S_3\} & \varnothing & \varnothing \\ \varnothing & \varnothing & \varnothing & \varnothing & \{A\} & \varnothing \\ \varnothing & \varnothing & \varnothing & \varnothing & \varnothing & \{B, S_2, S_3\} \end{pmatrix}$$

Шаг 2. Заполняем диагональ, находящуюся над главной:

$$\begin{pmatrix} \{A\} & \varnothing & \varnothing & \varnothing & \varnothing & \varnothing \\ \varnothing & \{A\} & \{S_1\} & \varnothing & \varnothing & \varnothing \\ \varnothing & \varnothing & \{B, S_2, S_3\} & \varnothing & \varnothing & \varnothing \\ \varnothing & \varnothing & \varnothing & \{B, S_2, S_3\} & \varnothing & \varnothing \\ \varnothing & \varnothing & \varnothing & \varnothing & \{A\} & \{S_1\} \\ \varnothing & \varnothing & \varnothing & \varnothing & \varnothing & \{B, S_2, S_3\} \end{pmatrix}$$

Шаг 3. Заполняем следующую диагональ:

$$\begin{pmatrix} \{A\} & \varnothing & \varnothing & \varnothing & \varnothing & \varnothing \\ \varnothing & \{A\} & \{S_1\} & \{S_2\} & \varnothing & \varnothing \\ \varnothing & \varnothing & \{B, S_2, S_3\} & \varnothing & \varnothing & \varnothing \\ \varnothing & \varnothing & \varnothing & \{B, S_2, S_3\} & \varnothing & \{S_2, S_3\} \\ \varnothing & \varnothing & \varnothing & \varnothing & \{A\} & \{S_1\} \\ \varnothing & \varnothing & \varnothing & \varnothing & \varnothing & \{B, S_2, S_3\} \end{pmatrix}$$

Шаг 4. И следующую за ней:

$$\begin{pmatrix} \{A\} & \varnothing & \varnothing & \{S_1, S\} & \varnothing & \varnothing \\ \varnothing & \{A\} & \{S_1\} & \{S_2\} & \varnothing & \varnothing \\ \varnothing & \varnothing & \{B, S_2, S_3\} & \varnothing & \varnothing & \varnothing \\ \varnothing & \varnothing & \varnothing & \{B, S_2, S_3\} & \varnothing & \{S_2, S_3\} \\ \varnothing & \varnothing & \varnothing & \varnothing & \{A\} & \{S_1\} \\ \varnothing & \varnothing & \varnothing & \varnothing & \varnothing & \{B, S_2, S_3\} \end{pmatrix}$$

Шаг 5 Заполняем предпоследнюю диагональ:

$$\begin{pmatrix} \{A\} & \varnothing & \varnothing & \{S_1, S\} & \varnothing & \varnothing \\ \varnothing & \{A\} & \{S_1\} & \{S_2\} & \varnothing & \{S_2\} \\ \varnothing & \varnothing & \{B, S_2, S_3\} & \varnothing & \varnothing & \varnothing \\ \varnothing & \varnothing & \varnothing & \{B, S_2, S_3\} & \varnothing & \{S_2, S_3\} \\ \varnothing & \varnothing & \varnothing & \varnothing & \{A\} & \{S_1\} \\ \varnothing & \varnothing & \varnothing & \varnothing & \varnothing & \{B, S_2, S_3\} \end{pmatrix}$$

Шаг 6. Завершаем выполнение алгоритма:

$$\begin{pmatrix} \{A\} & \varnothing & \varnothing & \{S_1, S\} & \varnothing & \{S_1, S\} \\ \varnothing & \{A\} & \{S_1\} & \{S_2\} & \varnothing & \{S_2\} \\ \varnothing & \varnothing & \{B, S_2, S_3\} & \varnothing & \varnothing & \varnothing \\ \varnothing & \varnothing & \varnothing & \{B, S_2, S_3\} & \varnothing & \{S_2, S_3\} \\ \varnothing & \varnothing & \varnothing & \varnothing & \{A\} & \{S_1\} \\ \varnothing & \varnothing & \varnothing & \varnothing & \varnothing & \{B, S_2, S_3\} \end{pmatrix}$$

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

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

Шаг 1. Инициализируем таблицу:

$$\begin{pmatrix} \{A\} & \varnothing & \varnothing & \varnothing \\ \varnothing & \{B, S_2, S_3\} & \varnothing & \varnothing \\ \varnothing & \varnothing & \{A\} & \varnothing \\ \varnothing & \varnothing & \varnothing & \{A\} \end{pmatrix}$$

Шаг 2. Заполняем следующую диагональ:

$$\begin{pmatrix} \{A\} & \{S_1, S\} & \varnothing & \varnothing \\ \varnothing & \{B, S_2, S_3\} & \varnothing & \varnothing \\ \varnothing & \varnothing & \{A\} & \varnothing \\ \varnothing & \varnothing & \varnothing & \{A\} \end{pmatrix}$$

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

Теперь изменим вид входного слова и немного модифицируем алгоритм. Прежде мы сопоставляли каждому символу слова номер его позиции в цепочке, поэтому при инициализации заполняли главную диагональ матрицы. Вместо этого обозначим числами позиции между буквами таким образом (в качестве примера рассмотрим слово $aabbab$ из предыдущего пункта):

$$0 \xrightarrow{a} 1 \xrightarrow{a} 2 \xrightarrow{b} 3 \xrightarrow{b} 4 \xrightarrow{a} 5 \xrightarrow{b} 6$$

Что нужно изменить в описании алгоритма, чтобы он продолжал работать при подобной нумерации? Каждая буква теперь идентифицируется не одним числом, а парой – номера слева и справа от нее. При этом чисел стало на одно больше, чем при прежнем способе нумерации.

Возьмем матрицу $|N| \times (n+1) \times (n+1)$ и при инициализации будем заполнять не главную диагональ, а диагональ прямо над ней. Таким образом, мы начинаем наш алгоритм с определения значений $M[i, j, A]$, где $j = i+1$. При этом наши дальнейшие действия в рамках алгоритма не изменятся.

На шаге инициализации матрица выглядит следующим образом:

$$\begin{pmatrix} \varnothing & \{A\} & \varnothing & \varnothing & \varnothing & \varnothing & \varnothing \\ \varnothing & \varnothing & \{A\} & \varnothing & \varnothing & \varnothing & \varnothing \\ \varnothing & \varnothing & \varnothing & \{B, S_2, S_3\} & \varnothing & \varnothing & \varnothing \\ \varnothing & \varnothing & \varnothing & \varnothing & \{B, S_2, S_3\} & \varnothing & \varnothing \\ \varnothing & \varnothing & \varnothing & \varnothing & \varnothing & \{A\} & \varnothing \\ \varnothing & \varnothing & \varnothing & \varnothing & \varnothing & \varnothing & \{B, S_2, S_3\} \\ \varnothing & \varnothing & \varnothing & \varnothing & \varnothing & \varnothing & \varnothing \end{pmatrix}$$

А в результате работы алгоритма имеем:

$$
\begin{pmatrix}
\varnothing & \{A\} & \varnothing & \varnothing & \{S_1, S\} & \varnothing & \{S_1, S\} \\
\varnothing & \varnothing & \{A\} & \{S_1\} & \{S_2\} & \varnothing & \{S_2\} \\
\varnothing & \varnothing & \varnothing & \{B, S_2, S_3\} & \varnothing & \varnothing & \varnothing \\
\varnothing & \varnothing & \varnothing & \varnothing & \{B, S_2, S_3\} & \varnothing & \{S_2, S_3\} \\
\varnothing & \varnothing & \varnothing & \varnothing & \varnothing & \{A\} & \{S_1\} \\
\varnothing & \varnothing & \varnothing & \varnothing & \varnothing & \varnothing & \{B, S_2, S_3\} \\
\varnothing & \varnothing & \varnothing & \varnothing & \varnothing & \varnothing & \varnothing
\end{pmatrix}
$$

Мы представили входную строку в виде линейного графа, а на шаге инициализации получили его матрицу смежности. Шаги алгоритма очень напоминают построение транзитивного замыкания графа. Различие заключается в том, что мы “добавляем ребра“ только для тех пар нетерминалов, для которых существует правило в грамматике, их выводящее.

Алгоритм можно обобщить и на произвольные графы с метками, рассматриваемые в этом курсе. При этом накладывается ограничение на форму входной грамматики: она должна находиться в ослабленной Нормальной Форме Хомского 3.4. То есть будем требовать выполнение только следующих правил:

- $A \to BC$, где $A, B, C \in N$
- $A \to a$, где $A \in N, a \in \Sigma$

Шаг инициализации в алгоритме заключается в том, что мы с помощью продукций вида

$$A \to a, \text{ где } A \in N, a \in \Sigma$$

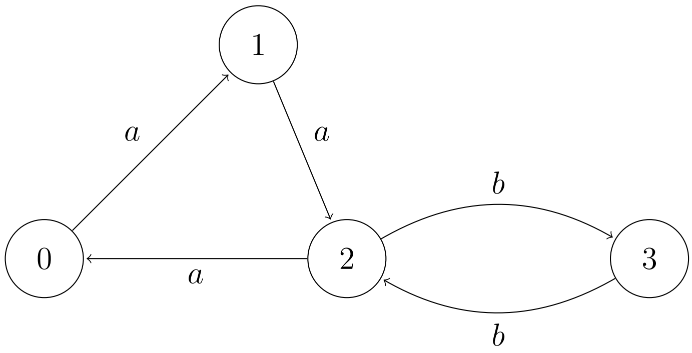
заменяем терминалы на ребрах входного графа на нетерминалы, из которых они выводятся. Затем используем матрицу смежности получившегося графа (обозначим ее M) в качестве начального значения. Дальнейший ход алгоритма можно описать псевдокодом:

Algorithm 2 Context-free recognizer for graphs

function ContextFreePathQuerying(G, $\mathcal{G}$)
    $n \leftarrow$ the number of nodes in $\mathcal{G}$
    $M \leftarrow$ the modified adjacency matrix of $\mathcal{G}$
    $P \leftarrow$ the set of production rules in G
    while M is changing **do**
        for $k \in 0..n$ **do**
            for $i \in 0..n$ **do**
                for $j \in 0..n$ **do**
                    for all $N_1 \in M[i,k]$, $N_2 \in M[k,j]$ **do**
                        if $N_3 \to N_1 N_2 \in P$ **then**
                            $M[i,j] \mathrel{+}= \{N_3\}$
    return M

Если в некоторой ячейке результируюшей матрицы с номером (i, j) находятся стартовый нетерминал, то это означает, что существует путь из вершины i в вершину j, удовлетворяющий данной грамматике.

Пример 5.3. Рассмотрим работу алгоритма на графе

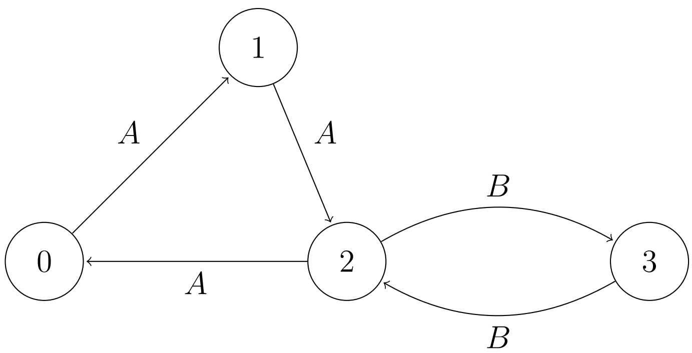


и грамматике:

$$
\begin{aligned}
S &\to AB\\
S &\to AS_1\\
S_1 &\to SB\\
A &\to a\\
B &\to b
\end{aligned}
$$

Данный пример является классическим и еще не раз будет использоваться в рамках данного курса.

Инициализация. Заменяем терминалы на ребрах графа на нетерминалы, из которых они выводятся, и строим матрицу смежности получившегося графа:



$$
\begin{pmatrix}
\varnothing & \{A\} & \varnothing & \varnothing\\
\varnothing & \varnothing & \{A\} & \varnothing\\
\{A\} & \varnothing & \varnothing & \{B\}\\
\varnothing & \varnothing & \{B\} & \varnothing
\end{pmatrix}
$$

Итерация 1. Итерируемся по k, i и j, пытаясь найти пары нетерминалов, для которых существуют правила вывода, их выводящие. Нам интересны следующие случаи:

- $k = 2, i = 1, j = 3 : A \in M[1,2], B \in M[2,3]$, так как в грамматике присутствует правило $S \to AB$, добавляем нетерминал S в ячейку $M[1,3]$.

- $k = 3, i = 1, j = 2 : S \in M[1,3], B \in M[3,2]$, поскольку в грамматике есть правило $S_1 \to SB$, добавляем нетерминал S_1 в ячейку $M[1,2]$.

В остальных случаях либо какая-то из клеток пуста, либо не существует продукции в грамматике, выводящей данные два нетерминала.

Матрица после данной итерации:

$$\begin{pmatrix} \varnothing & \{A\} & \varnothing & \varnothing \\ \varnothing & \varnothing & \{A, S_1\} & \{S\} \\ \{A\} & \varnothing & \varnothing & \{B\} \\ \varnothing & \varnothing & \{B\} & \varnothing \end{pmatrix}$$

Итерация 2. Снова итерируемся по k, i, j. Рассмотрим случаи:

- $k = 1, i = 0, j = 2 : A \in M[0,1], S_1 \in M[1,2]$, так как в грамматике присутствует правило $S \to AS_1$, добавляем нетерминал S в ячейку $M[0,2]$.

- $k = 2, i = 0, j = 3 : S \in M[0,2], B \in M[2,3]$, поскольку в грамматике есть правило $S_1 \to SB$, добавляем нетерминал S_1 в ячейку $M[0,3]$.

Матрица на данном шаге:

$$\begin{pmatrix} \varnothing & \{A\} & \{S\} & \{S_1\} \\ \varnothing & \varnothing & \{A, S_1\} & \{S\} \\ \{A\} & \varnothing & \varnothing & \{B\} \\ \varnothing & \varnothing & \{B\} & \varnothing \end{pmatrix}$$

Итерация 3. Рассматриваемые на данном шаге случаи:

- $k = 0, i = 2, j = 3 : A \in M[2,0], S_1 \in M[0,3]$, так как в грамматике присутствует правило $S \to AS_1$, добавляем нетерминал S в ячейку $M[2,3]$.

- $k = 3, i = 2, j = 2 : S \in M[2,3], B \in M[3,2]$, поскольку в грамматике есть правило $S_1 \to SB$, добавляем нетерминал S_1 в ячейку $M[2,2]$.

Матрица после этой итерации:

$$\begin{pmatrix} \varnothing & \{A\} & \{S\} & \{S_1\} \\ \varnothing & \varnothing & \{A, S_1\} & \{S\} \\ \{A\} & \varnothing & \{S_1\} & \{B, S\} \\ \varnothing & \varnothing & \{B\} & \varnothing \end{pmatrix}$$

Итерация 4. Рассмариваемые случаи:

- $k = 2, i = 1, j = 2 : A \in M[1,2], S_1 \in M[2,2]$, так как в грамматике присутствует правило $S \to AS_1$, добавляем нетерминал S в ячейку $M[1,2]$.

- $k = 2, i = 1, j = 3 : S \in M[1,2], B \in M[2,3]$, поскольку в грамматике есть правило $S_1 \to SB$, добавляем нетерминал S_1 в ячейку $M[1,3]$.

Матрица:

$$\begin{pmatrix} \varnothing & \{A\} & \{S\} & \{S_1\} \\ \varnothing & \varnothing & \{A, S, S_1\} & \{S, S_1\} \\ \{A\} & \varnothing & \{S_1\} & \{B, S\} \\ \varnothing & \varnothing & \{B\} & \varnothing \end{pmatrix}$$

Итерация 5. Рассмотрим на это шаге:

- $k = 1, i = 0, j = 3 : A \in M[0,1], S_1 \in M[1,3]$, поскольку в грамматике есть правило $S \to AS_1$, добавляем нетерминал S в ячейку $M[0,3]$.

- $k = 3, i = 0, j = 2 : S \in M[0,3], B \in M[3,2]$, поскольку в грамматике есть правило $S_1 \to SB$, добавляем нетерминал S_1 в ячейку $M[0,2]$.

Матрица на этой итерации:

$$\begin{pmatrix} \varnothing & \{A\} & \{S, S_1\} & \{S, S_1\} \\ \varnothing & \varnothing & \{A, S, S_1\} & \{S, S_1\} \\ \{A\} & \varnothing & \{S_1\} & \{B, S\} \\ \varnothing & \varnothing & \{B\} & \varnothing \end{pmatrix}$$

Итерация 6. Интересующие нас на этом шаге случаи:

- $k = 0, i = 2, j = 2 : A \in M[2,0], S_1 \in M[0,2]$, поскольку в грамматике есть правило $S \to AS_1$, добавляем нетерминал S в ячейку $M[2,2]$.

- $k = 2, i = 2, j = 3 : S \in M[2,2], B \in M[2,3]$, поскольку в грамматике есть правило $S_1 \to SB$, добавляем нетерминал S_1 в ячейку $M[2,3]$.

Матрица после данного шага:

$$\begin{pmatrix} \varnothing & \{A\} & \{S, S_1\} & \{S, S_1\} \\ \varnothing & \varnothing & \{A, S, S_1\} & \{S, S_1\} \\ \{A\} & \varnothing & \{S, S_1\} & \{B, S, S_1\} \\ \varnothing & \varnothing & \{B\} & \varnothing \end{pmatrix}$$

На следующей итерации матрица не изменяется, поэтому заканчиваем работу алгоритма. В результате, если ячейка $M[i,j]$ содержит стартовый нетерминал S, то существует путь из i в j, удовлетворяющий ограничениям, заданным грамматикой.

Можно заметить, что мы делаем много лишних итераций. Можно переписать алгоритм так, чтобы он не просматривал заведомо пустые ячейки. Данную модификацию предложил Хеллингс [27], также она реализована в работе [73].

Псевдокод алгоритма Хеллингса.

Algorithm 3 Context-free recognizer for graphs. Hellings.

function CONTEXTFREEPATHQUERYING($G = \langle \Sigma, N, P, S \rangle$, $\mathcal{G} = \langle V, E, L \rangle$)
    $r \leftarrow \{(N_i, v, v) \mid v \in V \wedge N_i \rightarrow \varepsilon \in P\} \cup \{(N_i, v, u) \mid (v, t, u) \in E \wedge N_i \rightarrow t \in P\}$
    $m \leftarrow r$
    while $m \neq \varnothing$ **do**
        $(N_i, v, u) \leftarrow$ m.pick()
        for $(N_j, v', v) \in r$ **do**
            for $N_k \rightarrow N_j N_i \in P$ таких что $((N_k, v', u) \notin r)$ **do**
                $m \leftarrow m \cup \{(N_k, v', u)\}$
                $r \leftarrow r \cup \{(N_k, v', u)\}$
        for $(N_j, u, v') \in r$ **do**
            for $N_k \rightarrow N_i N_j \in P$ таких что $((N_k, v, v') \notin r)$ **do**
                $m \leftarrow m \cup \{(N_k, v, v')\}$
                $r \leftarrow r \cup \{(N_k, v, v')\}$
    return r

Пример 5.4. Запустим алгоритм Хеллингса на нашем примере.

Инициализация

$$m = r = \{(A, 0, 1), (A, 1, 2), (A, 2, 0), (B, 2, 3), (B, 3, 2)\}$$

Итерации внешнего цикла. Будем считеть, что r и m — упорядоченные списки и *pick* возврпщает его голову, оставляя хвост. Новые элементы добавляются в конец.

1. Обрабатываем $(A, 0, 1)$. Ни один из вложенных циклов не найдёт новых путей, так как для рассматриваемого ребра есть только две возможности достроить путь: $2 \xrightarrow{A} 0 \xrightarrow{A} 1$ и $0 \xrightarrow{A} 1 \xrightarrow{A} 2$ и ни одна из соответствующих строк не выводтся в заданной граммтике.

2. Перед началом итерации
$$m = \{(A, 1, 2), (A, 2, 0), (B, 2, 3), (B, 3, 2)\},$$
r не изменилось. Обрабатываем $(A, 1, 2)$. В данной ситуации второй цикл найдёт тройку $(B, 2, 3)$ и соответсвующее правило $S \rightarrow A\ B$. Это значит, что и в m и в r добавится тройка $(S, 1, 3)$.

3. Перед началом итерации
$$m = \{(A, 2, 0), (B, 2, 3), (B, 3, 2), (S, 1, 3)\},$$
$$r = \{(A, 0, 1), (A, 1, 2), (A, 2, 0), (B, 2, 3), (B, 3, 2), (S, 1, 3)\}.$$
Обрабатываем $(A, 2, 0)$. Внутринние циклы ничего не найдут, новых путей н появится.

4. Перед началом итерации
$$m = \{(B, 2, 3), (B, 3, 2), (S, 1, 3)\},$$
$$r = \{(A, 0, 1), (A, 1, 2), (A, 2, 0), (B, 2, 3), (B, 3, 2), (S, 1, 3)\}.$$
Обрабатываем $(B, 2, 3)$. Первый цикл мог бы найти $(A, 1, 2)$, однако при проверке во вложенном цикле выяснится, что $(S, 1, 3)$ уже найдена. В итоге, на данной итерации новых путей н появится.

5. Перед началом итерации
$$m = \{(B,3,2),(S,1,3)\},$$
$$r = \{(A,0,1),(A,1,2),(A,2,0),(B,2,3),(B,3,2),(S,1,3)\}.$$
Обрабатываем $(B,3,2)$. Первый цикл найдёт $(S,1,3)$ и соответствующее правило $S_1 \to S\ B$. Это значит, что и в m и в r добавится тройка $(S_1,1,2)$.

6. Перед началом итерации
$$m = \{(S,1,3),(S_1,1,2)\},$$
$$r = \{(A,0,1),(A,1,2),(A,2,0),(B,2,3),(B,3,2),(S,1,3),(S_1,1,2)\}.$$
Обрабатываем $(S,1,3)$. Второй цикл мог бы найти $(B,3,2)$, однако при проверке во вложенном цикле выяснится, что $(S_1,1,2)$ уже найдена. В итоге, на данной итерации новых путей н появится.

7. Перед началом итерации
$$m = \{(S_1,1,2)\},$$
$$r = \{(A,0,1),(A,1,2),(A,2,0),(B,2,3),(B,3,2),(S,1,3),(S_1,1,2)\}.$$
Обрабатываем $(S_1,1,2)$. Первый цикл найдёт $(A,0,1)$ и соответствующее правило $S \to A\ S_1$. Это значит, что и в m и в r добавится тройка $(S,0,2)$.

8. Перед началом итерации
$$m = \{(S,0,2)\},$$
$$r = \{(A,0,1),(A,1,2),(A,2,0),(B,2,3),(B,3,2),(S,1,3),(S_1,1,2),(S,0,2)\}.$$
Обрабатываем $(S,0,2)$. Найдено: $(B,2,3)$ и соответствующее правило $S_1 \to S\ B$. В m и в r добавится тройка $(S_1,0,3)$.

9. Перед началом итерации
$$m = \{(S_1,0,3)\},$$
$$r = \{(A,0,1),(A,1,2),(A,2,0),(B,2,3),(B,3,2),(S,1,3),(S_1,1,2),(S,0,2),(S_1,0,3)\}.$$
Обрабатываем $(S_1,0,3)$. Найдено: $(A,2,0)$ и соответствующее правило $S \to A\ S_1$. В m и в r добавится тройка $(S,2,3)$.

10. Перед началом итерации
$$m = \{(S,2,3)\},$$
$$r = \{(A,0,1),(A,1,2),(A,2,0),(B,2,3),(B,3,2),(S,1,3),(S_1,1,2),(S,0,2),(S_1,0,3),(S,2,3)\}.$$
Обрабатываем $(S,2,3)$. Найдено: $(B,3,2)$ и соответствующее правило $S_1 \to S\ B$. В m и в r добавится тройка $(S_1,2,2)$.

11. Перед началом итерации
$$m = \{(S_1,2,2)\},$$
$$r = \{(A,0,1),(A,1,2),(A,2,0),(B,2,3),(B,3,2),(S,1,3),(S_1,1,2),(S,0,2),(S_1,0,3),(S,2,3),(S_1,2$$
Обрабатываем $(S_1,2,2)$. Найдено: $(A,1,2)$ и соответствующее правило $S \to A\ S_1$. В m и в r добавится тройка $(S,1,2)$.

12. Перед началом итерации
$$m = \{(S, 1, 2)\},$$
$$r = \{(A, 0, 1), (A, 1, 2), (A, 2, 0), (B, 2, 3), (B, 3, 2), (S, 1, 3), (S_1, 1, 2), (S, 0, 2), (S_1, 0, 3), (S, 2, 3), (S_1, 2$$
Обрабатываем $(S, 1, 2)$. Найдено: $(B, 2, 3)$ и соответствующее правило $S_1 \to S\ B$. В m и в r добавится тройка $(S_1, 1, 3)$.

13. Перед началом итерации
$$m = \{(S_1, 1, 3)\},$$
$$r = \{(A, 0, 1), (A, 1, 2), (A, 2, 0), (B, 2, 3), (B, 3, 2), (S, 1, 3), (S_1, 1, 2), (S, 0, 2), (S_1, 0, 3), (S, 2, 3), (S_1, 2$$
Обрабатываем $(S_1, 1, 3)$. Найдено: $(A, 0, 1)$ и соответствующее правило $S \to A\ S_1$. В m и в r добавится тройка $(S, 0, 3)$.

14. Перед началом итерации
$$m = \{(S, 0, 3)\},$$
$$r = \{(A, 0, 1), (A, 1, 2), (A, 2, 0), (B, 2, 3), (B, 3, 2), (S, 1, 3), (S_1, 1, 2), (S, 0, 2), (S_1, 0, 3), (S, 2, 3), (S_1, 2$$
Обрабатываем $(S, 0, 3)$. Найдено: $(B, 3, 2)$ и соответствующее правило $S_1 \to S\ B$. В m и в r добавится тройка $(S_1, 0, 2)$.

15. Перед началом итерации
$$m = \{(S_1, 0, 2)\},$$
$$r = \{(A, 0, 1), (A, 1, 2), (A, 2, 0), (B, 2, 3), (B, 3, 2), (S, 1, 3), (S_1, 1, 2), (S, 0, 2), (S_1, 0, 3), (S, 2, 3), (S_1, 2$$
Обрабатываем $(S_1, 0, 2)$. Найдено: $(A, 2, 0)$ и соответствующее правило $S \to A\ S_1$. В m и в r добавится тройка $(S, 2, 2)$.

16. Перед началом итерации
$$m = \{(S, 2, 2)\},$$
$$r = \{(A, 0, 1), (A, 1, 2), (A, 2, 0), (B, 2, 3), (B, 3, 2), (S, 1, 3), (S_1, 1, 2), (S, 0, 2), (S_1, 0, 3), (S, 2, 3), (S_1, 2$$
Обрабатываем $(S, 2, 2)$. Найдено: $(B, 2, 3)$ и соответствующее правило $S_1 \to S\ B$. В m и в r добавится тройка $(S_1, 2, 3)$.

17. Перед началом итерации
$$m = \{(S_1, 2, 3)\},$$
$$r = \{(A, 0, 1), (A, 1, 2), (A, 2, 0), (B, 2, 3), (B, 3, 2), (S, 1, 3), (S_1, 1, 2), (S, 0, 2), (S_1, 0, 3), (S, 2, 3), (S_1, 2$$
Обрабатываем $(S_1, 2, 3)$. Могло бы быть найдено: $(A, 1, 2)$ и соответствующее правило $S \to A\ S_1$, однако тройка $(S, 1, 3)$ уже есть в r. А значит никаких новых троек найдено не будет и m становится пустым. Это была последняя итерация внешнего цикла, в r на текущий момент уже содержится всё ршение.

Как можно заметить, количество итераций внешнего цикла также получилось достаточно большим. Проверьте, зависит ли оно от порядка обработки элементов из m. При этом внутренние циклы в нашем случае достаточно короткие, так как просматриваются только “существенные” элементы и избегается дублирование.

5.3 Вопросы и задачи

1. Проверить работу алгоритма CYK для цепочек на грамматике

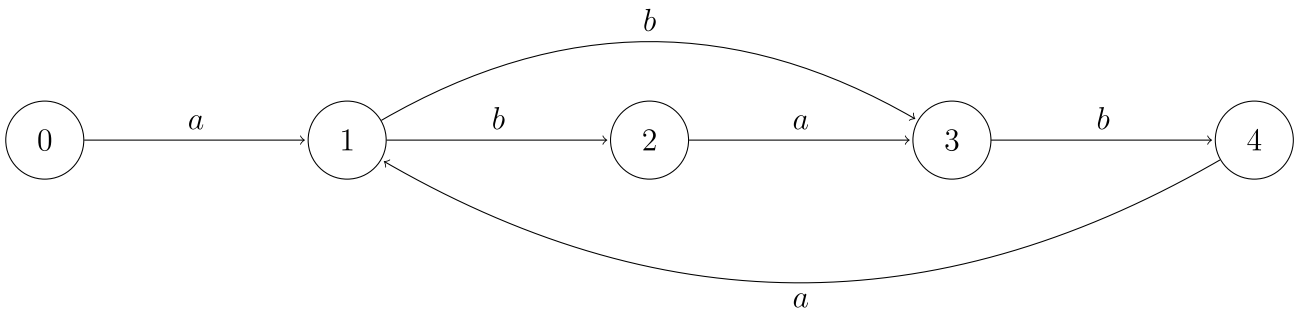
   $E \to E + E$
   $E \to E * E$
   $E \to (E)$
   $E \to n$

   и словах (алфавит $\Sigma = \{n, +, *, (,)\}$)

   $(n + n) * n$
   $n + n * n$
   $n + n + n + n$
   $n + (n * n) + n$

2. Посчитать вычислительную сложность алгоритма CYK для матриц в зависимости от размера входного графа (размер грамматики считать фиксированным)

3. Проверить работу алгоритма CYK для графов на графе



   И грамматике

   $S \to SS$
   $S \to AB$
   $A \to a$
   $B \to b$

4. Оцените временную сложность алгоритма Хеллингса и сравните её с оценкой для наивного обобщения CYK.

6 Алгоритм на матричных операциях

6.1 Описание

В главе 5.2 был изложен алгоритм для графов на основе CYK. Заметим, что обход матрицы напоминает матричное умножение:

$$M_1 \times M_2 = M_3$$
$$M_3[i,j] = \sum_{k=1}^{n} M[i,k] * M[k,j]$$

, где

$$\sum_{k=1}^{n} = \bigcup_{k=1}^{n}$$
$$S_1 * S_2 = \{N_1^0 ... N_1^m\} * \{N_2^0 ... N_2^l\} = \{N_3 \mid (N_3 \to N_1^i N_2^j) \in P\}$$

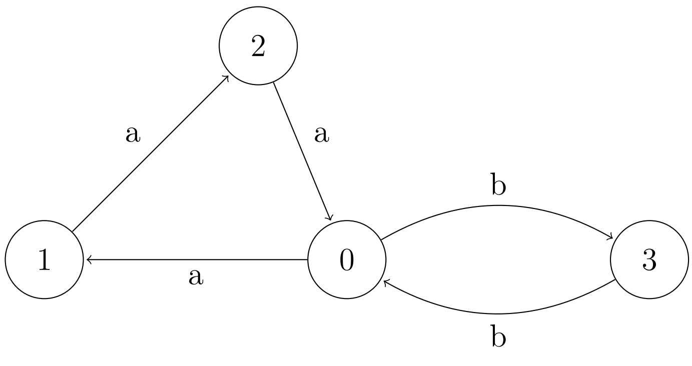
Для линейного входа существует алгоритм Валианта [65], но он не обобщается на графы с сохранением асимптотики [72], поэтому рассмотрим идею, изложенную в статье Рустама Азимова [5].

Пусть $\mathcal{G} = (V, E)$ — входной граф и $G = (N, \Sigma, P)$ — входная грамматика.

Algorithm 4 Context—free recognizer for graphs

```
function contextFreePathQuerying(𝒢, G)
    n ← количество узлов в 𝒢
    E ← направленные ребра в 𝒢
    P ← набор продукций из G
    T ← матрица n × n, в которой каждый элемент ∅
    for all (i, x, j) ∈ E do                          ▷ Инициализация матрицы
        T_{i,j} ← T_{i,j} ∪ {A | (A → x) ∈ P}
    while матрица T меняется do
        T ← T ∪ (T × T)                               ▷ Вычисление транзитивного замыкания
    return T
```

Пример 6.1 (Пример работы)**.** Пусть есть граф $\mathcal{G}$:



и грамматика G:

$$S \to AB$$
$$S \to AS_1$$
$$S_1 \to SB$$
$$A \to a$$
$$B \to b$$

Тогда T_0, полученная напрямую из графа, выглядит так:

$$T_0 = \begin{pmatrix} \varnothing & \{A\} & \varnothing & \{B\} \\ \varnothing & \varnothing & \{A\} & \varnothing \\ \{A\} & \varnothing & \varnothing & \varnothing \\ \{B\} & \varnothing & \varnothing & \varnothing \end{pmatrix}$$

Пусть T_i — матрица, полученная из T после применения цикла на строках **8-9** алгоритма 4 i раз. Далее показано получение матрицы T_1.

$$T_0 \times T_0 = \begin{pmatrix} \varnothing & \varnothing & \varnothing & \varnothing \\ \varnothing & \varnothing & \varnothing & \varnothing \\ \varnothing & \varnothing & \varnothing & \{S\} \\ \varnothing & \varnothing & \varnothing & \varnothing \end{pmatrix}$$

$$T_1 = T_0 \cup (T_0 \times T_0) = \begin{pmatrix} \varnothing & \{A\} & \varnothing & \{B\} \\ \varnothing & \varnothing & \{A\} & \varnothing \\ \{A\} & \varnothing & \varnothing & \{\boldsymbol{S}\} \\ \{B\} & \varnothing & \varnothing & \varnothing \end{pmatrix}$$

Когда алгоритм находит новые пути в графе $\mathcal{G}$, он добавляет соответствующие нетерминалы в матрицу T. Например, после первого цикла нетерминал S добавляется к матрице T. Этот нетерминал добавляется в ячейку $[2, 3]$. Это означает, что существует такой путь π из вершины 2 в вершину 3, что $S \xrightarrow{*} \omega(\pi)$. В данном примере путь состоит из двух ребер с именами a и b, так что $S \xrightarrow{*} ab$.

Вычисление транзитивного замыкания заканчивается через k итераций, когда достигается фиксированная точка: $T_{k-1} = T_k$. Для данного примера $k = 13$, так как $T_{13} = T_{12}$. Процесс вычисления транзитивного замыкания показан ниже (на каждой итерации новые элементы выделены жирным).

$$T_2 = \begin{pmatrix} \varnothing & \{A\} & \varnothing & \{B\} \\ \varnothing & \varnothing & \{A\} & \varnothing \\ \{A, \boldsymbol{S_1}\} & \varnothing & \varnothing & \{S\} \\ \{B\} & \varnothing & \varnothing & \varnothing \end{pmatrix} \quad T_3 = \begin{pmatrix} \varnothing & \{A\} & \varnothing & \{B\} \\ \{\boldsymbol{S}\} & \varnothing & \{A\} & \varnothing \\ \{A, S_1\} & \varnothing & \varnothing & \{S\} \\ \{B\} & \varnothing & \varnothing & \varnothing \end{pmatrix}$$

$$T_4 = \begin{pmatrix} \varnothing & \{A\} & \varnothing & \{B\} \\ \{S\} & \varnothing & \{A\} & \{\boldsymbol{S_1}\} \\ \{A, S_1\} & \varnothing & \varnothing & \{S\} \\ \{B\} & \varnothing & \varnothing & \varnothing \end{pmatrix} \quad T_5 = \begin{pmatrix} \varnothing & \{A\} & \varnothing & \{B, \boldsymbol{S}\} \\ \{S\} & \varnothing & \{A\} & \{S_1\} \\ \{A, S_1\} & \varnothing & \varnothing & \{S\} \\ \{B\} & \varnothing & \varnothing & \varnothing \end{pmatrix}$$

$$T_6 = \begin{pmatrix} \{\boldsymbol{S_1}\} & \{A\} & \varnothing & \{B, S\} \\ \{S\} & \varnothing & \{A\} & \{S_1\} \\ \{A, S_1\} & \varnothing & \varnothing & \{S\} \\ \{B\} & \varnothing & \varnothing & \varnothing \end{pmatrix} \quad T_7 = \begin{pmatrix} \{S_1\} & \{A\} & \varnothing & \{B, S\} \\ \{S\} & \varnothing & \{A\} & \{S_1\} \\ \{A, S_1, \boldsymbol{S}\} & \varnothing & \varnothing & \{S\} \\ \{B\} & \varnothing & \varnothing & \varnothing \end{pmatrix}$$

$$T_8 = \begin{pmatrix} \{S_1\} & \{A\} & \varnothing & \{B, S\} \\ \{S\} & \varnothing & \{A\} & \{S_1\} \\ \{A, S_1, S\} & \varnothing & \varnothing & \{S, \boldsymbol{S_1}\} \\ \{B\} & \varnothing & \varnothing & \varnothing \end{pmatrix} \quad T_9 = \begin{pmatrix} \{S_1\} & \{A\} & \varnothing & \{B, S\} \\ \{S\} & \varnothing & \{A\} & \{S_1, \boldsymbol{S}\} \\ \{A, S_1, S\} & \varnothing & \varnothing & \{S, S_1\} \\ \{B\} & \varnothing & \varnothing & \varnothing \end{pmatrix}$$

$$T_{10} = \begin{pmatrix} \{S_1\} & \{A\} & \varnothing & \{B, S\} \\ \{S, \boldsymbol{S_1}\} & \varnothing & \{A\} & \{S_1, S\} \\ \{A, S_1, S\} & \varnothing & \varnothing & \{S, S_1\} \\ \{B\} & \varnothing & \varnothing & \varnothing \end{pmatrix} \quad T_{11} = \begin{pmatrix} \{S_1, \boldsymbol{S}\} & \{A\} & \varnothing & \{B, S\} \\ \{S, S_1\} & \varnothing & \{A\} & \{S_1, S\} \\ \{A, S_1, S\} & \varnothing & \varnothing & \{S, S_1\} \\ \{B\} & \varnothing & \varnothing & \varnothing \end{pmatrix}$$

$$T_{12} = \begin{pmatrix} \{S_1, S\} & \{A\} & \varnothing & \{B, S, \boldsymbol{S_1}\} \\ \{S, S_1\} & \varnothing & \{A\} & \{S_1, S\} \\ \{A, S_1, S\} & \varnothing & \varnothing & \{S, S_1\} \\ \{B\} & \varnothing & \varnothing & \varnothing \end{pmatrix} \quad T_{13} = \begin{pmatrix} \{S_1, S\} & \{A\} & \varnothing & \{B, S, S_1\} \\ \{S, S_1\} & \varnothing & \{A\} & \{S_1, S\} \\ \{A, S_1, S\} & \varnothing & \varnothing & \{S, S_1\} \\ \{B\} & \varnothing & \varnothing & \varnothing \end{pmatrix}$$

Таким образом, результат алгоритма 4 для нашего примера — это матрица $T_{13} = T_{12}$.

6.2 Конъюнктивные и булевы грамматики

6.2.1 Определения

Впервые конъюнктивные и булевы грамматики были предложены Александром Охотиным [43, 44]. Дадим определение конъюнктивной грамматики.

Определение 6.1. *Конъюнктивной грамматикой* называется $G = (\Sigma, N, P, S)$, где:

- Σ и N — дизъюнктивные конечные непустые множества терминалов и нетерминалов.
- P — конечное множество продукций, каждая вида

$$A \to \alpha_1 \& ... \& \alpha_n$$

  ,где $A \in N, n \geq 1$ и $\alpha_1, ..., \alpha_n \in (\Sigma \cup N)^*$.

- $S \in N$ — стартовый нетерминал.

Конъюнктивная грамматика генерирует строки, выводя их из начального символа, так же, как это происходит в контекстно-свободных грамматиках в параграфе 3.1. Промежуточные строки, используемые в процессе вывода, являются формулами следующего вида:

Определение 6.2. Пусть $G = (\Sigma, N, P, S)$ — конъюнктивная грамматика. Множество конъюнктивных формул $\mathcal{F}$ определяется индуктивно:

- Пустая строка ε — конъюнктивная формула.
- Любой символ из $(\Sigma \cup N)$ — формула.
- Если $\mathcal{A}$ и $\mathcal{B}$ непустые формулы, тогда $\mathcal{AB}$ — формула.
- Если $\mathcal{A}_1, \ldots, \mathcal{A}_n$ $(n \geqslant 1)$ — формула, тогда $(\mathcal{A}_1 \& \ldots \& \mathcal{A}_n)$ — формула.

Определение 6.3. Пусть $G = (\Sigma, N, P, S)$ — конъюнктивная грамматика. Аналогично определению отношения непосредственной выводимости в контекстно-свободной грамматике 3.6 определим $\underset{G}{\Rightarrow}$ как отношение непосредственной выводимости на множестве конъюнктивных формул.

- Любой нетерминал в любой формуле может быть перезаписан телом любого правила для этого терминала заключенным в скобки. То есть для любых $s^{'}, s^{''} \in (\Sigma \cup N \cup \{(, \&,)\})^*$ и $A \in N$, таких что $s^{'}As^{''}$ — формула, и для всех правил вида $A \to \alpha_1 \& \ldots \& \alpha_n \in P$, имеем $s^{'}As^{''} \underset{G}{\Rightarrow} s^{'}(\alpha_1 \& \ldots \& \alpha_n)s^{''}$.
- Если формула содержит подформулу в виде конъюнкции одной или нескольких одинаковых терминальных строк, заключенных в скобки, тогда подформула может быть перезаписана терминальной строкой без скобок. То есть для любых $s^{'}, s^{''} \in (\Sigma \cup N \cup \{(, \&,)\})^*$, $(n \geqslant 1)$ и $w \in \Sigma^*$, таких что $s^{'}(w \& \ldots \& w)s^{''}$ — формула, имеем $s^{'}(w \& \ldots \& w)s^{''} \underset{G}{\Rightarrow} s^{'}ws^{''}$.

Как и в случае контекстно-свободной грамматики обозначим $\underset{G}{\Rightarrow}^*$ рефлексивное транзитивное замыкание отношения $\underset{G}{\Rightarrow}$.

Определение 6.4. Пусть $G = (\Sigma, N, P, S)$ — конъюнктивная грамматика. Язык, порождаемый формулой, это множество всех терминальных строк выводимых из этой формулы: $L_G(\mathcal{A}) = \{w \in \Sigma^* \mid \mathcal{A} \underset{G}{\Rightarrow}^* w\}$. Очевидно, что язык порождаемый грамматикой, это язык порождаемый стартовым нетерминалом $S : L(G) = L_G(S) = L(S)$.

Теорема 6.1. Пусть $G = (\Sigma, N, P, S)$ — конъюнктивная грамматика. Пусть $\mathcal{A}_1, \ldots, \mathcal{A}_n, \mathcal{B}$ — формулы, $A \in N$, $a \in \Sigma$. Тогда,

1. $L(\varepsilon) = \{\varepsilon\}$.
2. $L(a) = \{a\}$.
3. $L(A) = \bigcup_{A \to \alpha_1 \& \ldots \& \alpha_n \in P} L((\alpha_1 \& \ldots \& \alpha_m))$.
4. $L(\mathcal{AB}) = L(\mathcal{A}) * L(\mathcal{B})$
5. $L((\mathcal{A}_1 \& \ldots \& \mathcal{A}_n)) = \bigcap_{i=1}^{n} L(\mathcal{A}_i)$.

Теорема 6.1 уже подразумевает интерпретацию грамматики как системы уравнений. Используем математический подход, чтобы лучше охарактеризовать конъюнктивные языки с помощью систем уравнений.

Определение 6.5 (Выражение)**.** Пусть Σ конечный непустой алфавит. Пусть $X = \{X_1, \ldots, X_N\}$ вектор переменных. Выражение над алфавитом Σ, зависящее от переменных X, определяется индуктивно:

- ε — выражение.
- Любой символ $a \in \Sigma$ — выражение.
- Любая переменная $X_i \in X$ — выражение.
- Если ϕ_1 и ϕ_2 выражения, то $\phi_1\phi_2, (\phi_1 \mid \phi_2), (\phi_1 \& \phi_2)$ также выражения.

Заметим, что любая формула, в терминах определения 6.2, является выражением, где нетерминалы формулы это переменные выражения. С другой стороны, любое выражение, не содержащее дизъюнкции, формула.

Предположим, что переменные X_i приняли в качестве значений слова из языка над алфавитом Σ. Определим значение всего выражения.

Определение 6.6 (Значение выражения)**.** Пусть $L = (L_1, \ldots, L_n)(L_i \subseteq \Sigma^*)$ вектор из n языков над Σ, где $n \geqslant 1$. Пусть ϕ выражение над Σ, зависящее от переменных $X_1, \ldots, X_n$. Значение выражения ϕ на векторе L — это язык над тем же алфавитом Σ. Обозначим его $\phi(L)$ и определим индуктивно на структуре выражения:

- $\varepsilon(L) = \{\varepsilon\}$.
- $a(L) = \{a\}$ для любого $a \in \Sigma$.
- $X_i(L) = L_i$ для любого $X_i \in X$.
- $\phi_1\phi_2 = \phi_1(L) * \phi_2(L), (\phi_1 \mid \phi_2)(L) = \phi_1(L) \cup \phi_2(L), (\phi_1 \& \phi_2)(L) = \phi_1(L) \cap \phi_2(L)$ для любых выражений ϕ_1 и ϕ_2.

Обобщим определение 6.6 на случай вектора выражений.

Определение 6.7 (Значение вектора выражений)**.** Пусть $L = (L_1, \ldots, L_n)(L_i \subseteq \Sigma^*)$ вектор из n языков над Σ, где $n \geqslant 1$. Пусть $\phi_1, \ldots, \phi_m$ выражения над Σ, зависящее от переменных $X_1, \ldots, X_n$. Значение вектора выражений $P = (\phi_1, \ldots, \phi_m)$ на векторе L — это вектор языков $P(L) = (\phi_1(L), \ldots, \phi_m(L))$ над тем же алфавитом Σ.

Зададим частичный порядок относительно включения “ $\preccurlyeq$ ” на множестве языков и расширим его на вектора языков длины n: $(L_1^{'}, \ldots, L_n^{'}) \preccurlyeq (L_1^{''}, \ldots, L_n^{''})$ если и только если $L_i^{'} \subseteq L_i^{''}$ для любого $1 \leqslant i \leqslant n$

Определение 6.8. $X = P(X)$ система уравнений над алфавитом Σ и $X = \{X_1, \ldots, X_n\}$, где $P = (\phi_1, \ldots, \phi_n)$ вектор выражений над алфавитом Σ, зависящий от X.

Вектор языков $L = (L_1, \ldots, L_n)$ является решением системы уравнений если $L = P(L)$.

Наименьшее решение L это вектор языков, такой что для любого другого сравнимого вектора языков $L^{'}$ выполняется $L \preccurlyeq L^{'}$.

Заметим, что оператор P на множестве $2^{\Sigma} \times \ldots \times 2^{\Sigma}$, что решение системы 6.8 это неподвижная точка P и что наименьшее решение системы это наименьшая неподвижная точка оператора P.

Теорема 6.2. Для любой системы из определения 6.8 с переменными $X_1, \dots, X_n$, оператор $P = (\phi_1, \dots, \phi_n)$ имеет наименьшую неподвижную точку $L = (L_1, \dots, L_n) = \lim_{i\to\infty} P^i \underbrace{(\varnothing, \dots, \varnothing)}_{n}$.

Приведем пример конъюнктивной грамматики.

Пример 6.2 (Пример конъюнктивной грамматики)**.** Следующая конъюнктивная грамматика G порождает язык $\{a^n b^n c^n \mid n \geq 0\}$:

1. $S \to AB\&DC$
2. $A \to aA \mid \varepsilon$
3. $B \to bBc \mid \varepsilon$
4. $C \to cC \mid \varepsilon$
5. $D \to aDb \mid \varepsilon$

Легко видеть, что $L(AB) = \{a^i b^j c^k \mid j = k\}$ и $L(DC) = \{a^i b^j c^k \mid i = j\}$. Тогда $L(S) = L(AB) \cap L(DC) = \{a^n b^n c^n \mid n \geq 0\}$.

В этой грамматике строка abc может быть получена следующим образом. Для начала представим грамматику в виде системы уравнений:

$$
\begin{aligned}
S &= AB \cap DC \\
A &= \{a\}A \cup \varepsilon \\
B &= \{b\}B\{c\} \cup \varepsilon \\
C &= \{c\}C \cup \varepsilon \\
D &= \{a\}D\{b\} \cup \varepsilon
\end{aligned}
$$

Используя теорему 6.2, будем итеративно вычислять $P^i \underbrace{(\varnothing, \dots, \varnothing)}_{5}$. На каждом шаге будем подставлять все терминальные строки из языков, порожденных нетерминалами на предыдущем шаге, в соответствующие нетерминалы правой части каждого уравнения и записывать получившиеся терминальные строки в языки нетерминалов текущего шага. Продолжаем до тех пор пока язык, порождаемый нетерминалом S, не будет содержать терминальную строку "abc".

1. На начальном этапе имеем $P^0(\varnothing, \dots, \varnothing) = (S : \varnothing, A : \varnothing, B : \varnothing, C : \varnothing, D : \varnothing)$

2. Подставляем в первое уравнение терминальные строки из шага 1 в соответствующие нетерминалы, т.е.

$$
\begin{aligned}
S : \varnothing &= \varnothing\varnothing \cap \varnothing\varnothing \\
A : \{\varepsilon\} &= \{a\}\varnothing \cup \{\varepsilon\} \\
B : \{\varepsilon\} &= \{b\}\varnothing\{c\} \cup \{\varepsilon\} \\
C : \{\varepsilon\} &= \{c\}\varnothing \cup \{\varepsilon\} \\
D : \{\varepsilon\} &= \{a\}\varnothing\{b\} \cup \{\varepsilon\}
\end{aligned}
$$

   В конце итерации получаем $P^1(\varnothing, \dots, \varnothing) = (S : \varnothing, A : \{\varepsilon\}, B : \{\varepsilon\}, C : \{\varepsilon\}, D : \{\varepsilon\})$

3. Делаем еще одну итерацию,

$$\begin{aligned} S:\{\varepsilon\} &= \{\varepsilon\}\{\varepsilon\} \cap \{\varepsilon\}\{\varepsilon\} \\ A:\{a,\varepsilon\} &= \{a\}\{\varepsilon\} \cup \{\varepsilon\} \\ B:\{bc,\varepsilon\} &= \{b\}\{\varepsilon\}\{c\} \cup \{\varepsilon\} \\ C:\{c,\varepsilon\} &= \{c\}\{\varepsilon\} \cup \{\varepsilon\} \\ D:\{ab,\varepsilon\} &= \{a\}\{\varepsilon\}\{b\} \cup \{\varepsilon\} \end{aligned}$$

   В конце итерации получаем $P^2(\varnothing,\ldots,\varnothing) = (S:\{\varepsilon\}, A:\{a,\varepsilon\}, B:\{bc,\varepsilon\}, C:\{c,\varepsilon\}, D:\{ab,\varepsilon\})$

4. Еще одна итерация,

$$\begin{aligned} S:\{\boxed{\text{abc}},\varepsilon\} &= \{a,\varepsilon\}\{bc,\varepsilon\} \cap \{ab,\varepsilon\}\{c,\varepsilon\} \\ A:\{a,aa,\varepsilon\} &= \{a\}\{a,\varepsilon\} \cup \{\varepsilon\} \\ B:\{bc,bbcc,\varepsilon\} &= \{b\}\{bc,\varepsilon\}\{c\} \cup \{\varepsilon\} \\ C:\{c,cc,\varepsilon\} &= \{c\}\{c,\varepsilon\} \cup \{\varepsilon\} \\ D:\{ab,aabb,\varepsilon\} &= \{a\}\{ab,\varepsilon\}\{b\} \cup \{\varepsilon\} \end{aligned}$$

   В конце итерации получили $P^3(\varnothing,\ldots,\varnothing) = (S:\{\boxed{\text{abc}},\varepsilon\}, A:\{a,aa,\varepsilon\}, B:\{bc,bbcc,\varepsilon\}, C:\{c,cc,\varepsilon\}, D:\{ab,aabb,\varepsilon\})$. Заметим, что терминальная строка “abc” появилась в языке, который порождает стартовый нетерминал S. Т.е. терминальная строка “abc” выводима в грамматике G, что и требовалось показать.

Заметим, что строку “abc” также можно получить применением правил вывода. здесь цифра над стрелкой соответствует номеру примененного правила.

$$\begin{aligned} S &\overset{1}{\Rightarrow} (AB\&DC) \\ &\overset{2}{\Rightarrow} (aAB\&DC) \overset{2}{\Rightarrow} (a\varepsilon B\&DC) \\ &\overset{3}{\Rightarrow} (abBc\&DC) \overset{3}{\Rightarrow} (ab\varepsilon c\&DC) \\ &\overset{5}{\Rightarrow} (abc\&aDbC) \overset{5}{\Rightarrow} (abc\&a\varepsilon bC) \\ &\overset{4}{\Rightarrow} (abc\&abcC) \overset{4}{\Rightarrow} (abc\&abc\varepsilon) \\ &\Rightarrow (abc\&abc) \Rightarrow abc \end{aligned}$$

Пример 6.3. Конъюнктивная грамматика G для языка $L = \{wcw \mid w \in \{a,b\}^*\}$:

$$\begin{aligned} &S \to C\&D \\ &C \to aCa \mid aCb \mid bCa \mid bCb \mid c \\ &D \to aA\&aD \mid bB\&bD \mid cE \\ &A \to aAa \mid aAb \mid bAa \mid bAb \mid cEa \\ &B \to aBa \mid aBb \mid bBa \mid bBb \mid cEb \\ &E \to aE \mid bE \mid \varepsilon \end{aligned}$$

Подробнее о конъюнктивных грамматиках можно прочитать в статьях [43, 48, 45, 46].

Дадим определение булевой грамматики.

Определение 6.9. *Булевой грамматикой* называется $G = (\Sigma, N, P, S)$, где:

- Σ и N — дизъюнктивные конечные непустые множества терминалов и нетерминалов.
- P — конечное множество продукций, каждая вида

$$A \to \alpha_1 \& ... \& \alpha_m \& \neg\beta_1 \& ... \& \neg\beta_n$$

,где $A \in N, m, n >= 0, m + n \geq 1$ и $\alpha_i, \beta_j \in (\Sigma \cup N)^*$.

- $S \in N$ — стартовый нетерминал.

Приведем пример булевой грамматики.

Пример 6.4. Следующая булева грамматика порождает язык $\{a^m b^n c^n \mid m, n \geq 0, m \neq n\}$:

$$\begin{aligned}
&S \to AB \& \neg DC\\
&A \to aA \mid \varepsilon\\
&B \to bBc \mid \varepsilon\\
&C \to cC \mid \varepsilon\\
&D \to aDb \mid \varepsilon
\end{aligned}$$

Очевидно, что $L(AB) = \{a^m b^n c^n \mid m, n \in \mathbb{N}\}$ и $L(DC) = \{a^n b^n c^m \mid m, n \in \mathbb{N}\}$. Тогда $L(AB) \cap \overline{L(DC)} = \{a^m b^n c^n \mid m, n \geq 0, m \neq n\}$.

Подробнее о булевых грамматиках можно прочитать в статьях [44, 47].

6.2.2 Матричный алгоритм для конъюнктивных грамматик

Определим бинарную нормальную форму конъюнктивной грамматики.

Определение 6.10 (Бинарная нормальная форма)**.** Конъюнктивная грамматика $G = (\Sigma, N, P, S)$ находится в бинарной нормальной форме, если каждое правило из P имеет вид,

- $A \to B_1C_1 \& \dots \& B_mC_m$, где $m \geqslant 1; A, B_i, C_i \in N$.
- $A \to a$, где $A \in N, a \in \Sigma$.
- $S \to \varepsilon$, если только S не содержится в правой части всех правил.

Теорема 6.3. Для каждой конъюнктивной грамматики G можно построить конъюнктивную грамматику в бинарной нормальной форме G', такую что $L(G) = L(G')$.

Доказательство теоремы 6.3 описано в статье [43].

Матричный алгоритм для конъюнктивных грамматик отличается от алгоритма 4 для контекстно-свободных грамматик только операцией умножения матриц, в остальном алгоритм остается без изменений. Определим операцию умножения матриц.

Определение 6.11. Пусть M_1 и M_2 матрицы размера n. Определим операцию $\circ$. $M_1 \circ M_2 = M_3$, где $M_3[i,j] = \{A \mid \exists (A \to B_1C_1 \& \dots \& B_mC_m) \in P, (B_k, C_k) \in d[i,j] \forall k = 1, \dots, m\}$, где $d[i,j] = \bigcup_{k=1}^{n} M_1[i,k] \times M_2[k,j]$.

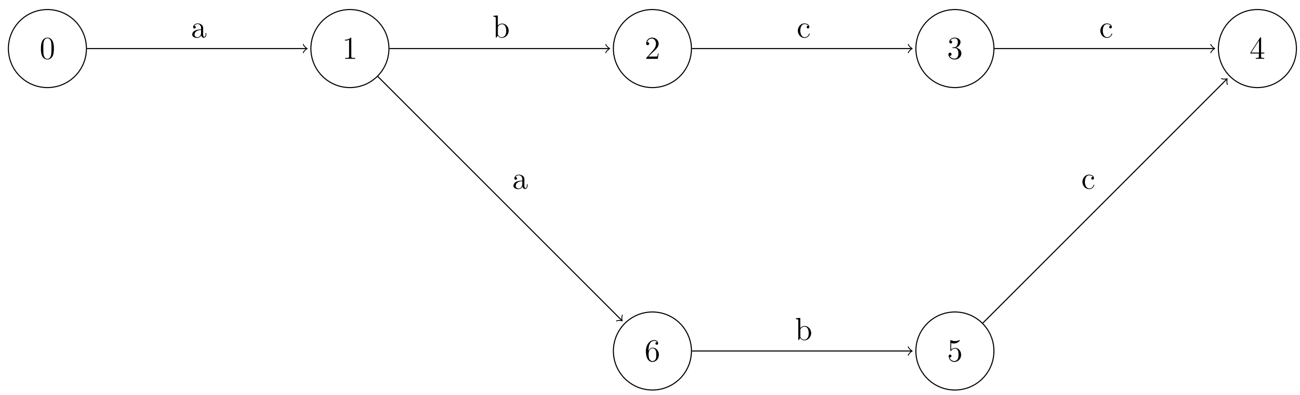
Важно заметить, что алгоритм для конъюнктивных грамматик, в отличие от алгоритма для контекстно-свободных грамматик, дает лишь верхнюю оценку. То есть все нетерминалы, которые должны быть в ячейках матрицы результата, содержатся там, но вместе с ними содержатся и лишние нетерминалы. Рассмотрим пример, иллюстрирующий появление лишних нетерминалов.

Пример 6.5. Грамматика G:

$$
\begin{aligned}
&S \to AB\&DC \\
&A \to a \\
&B \to BC \mid b \\
&C \to c \\
&D \to DC \mid b
\end{aligned}
$$

Очевидно, что грамматика G задает язык из одного слова $L(G) = \{abc\} = \{abc^*\} \cap \{a^*bc\}$.

Пусть есть граф $\mathcal{G}$:



Применяя алгоритм, получим, что существует путь из вершины 0 в вершину 4 из нетерминала S. Очевидно, что такого пути нет. Это происходит из-за того, что существует путь "abcc", соответствующий $L(AB) = \{abc^*\}$ и путь "aabc", соответствующий $L(DC) = \{a^*bc\}$, но они различны и проверить их в общем случае нельзя. Поэтому для классической семантики результат работы алгоритма является оценкой сверху.

Существует альтернативная семантика, когда мы трактуем конъюнкцию как в Datalog. Подробнее о Datalog в параграфе 13. Т.е если есть правило $S \to AB\&DC$, то должен быть путь принадлежащий языку $L(AB)$ и путь принадлежащий языку $L(DC)$. В альтернативной семантике алгоритм дает точный ответ.

Подробнее алгоритм описан в статье Рустама Азимова и Семена Григорьева [6].

В общем случае для булевых грамматик подобного алгоритма не существует.

6.3 Особенности реализации матричного алгоритма

Матричные алгоритмы удобны тем, что их удобно вычислять на GPU и для них можно создать модификацию, которая будет работать параллельно на нескольких вычислителях [41].

Так как множество нетерминалов и правил конечно, то мы можем свести алгоритм к булевым матрицам: для каждого нетерминала заводим матрицу, такую что в ячейке стоит 1 тогда и только тогда, когда в исходной матрице в соответствующей ячейке был этот нетерминал. Тогда перемножение пары таких матриц — это применение правила.

Пример 6.6. Представим в таком виде следующую матрицу:

$$T_0 = \begin{pmatrix} \varnothing & \{A\} & \varnothing & \{B\} \\ \varnothing & \varnothing & \{A\} & \varnothing \\ \{A\} & \varnothing & \varnothing & \varnothing \\ \{B\} & \varnothing & \varnothing & \varnothing \end{pmatrix}$$

Тогда:

$$T_{0\_A} = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \quad T_{0\_B} = \begin{pmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix}$$

Тогда при наличии правила $S \to AB$ получим $T_{1\_S} = T_{0\_A} \times T_{0\_B}$:

$$T_{1\_S} = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

С другой стороны, для небольших запросов практически может быть выгодно представлять множества нетерминалов в виде битового вектора. Нумеруем все нетерминалы с нуля, в векторе стоит 1 на i позиции, если в множестве есть нетерминал с номером i. Таким образом, в каждой ячейке хранится битовый вектор длины $|N|$. Тогда операцию умножения надо определить следующим образом:

$$v_1 \times v_2 = \{v \mid \exists (v, v_3) \in P, append(v_1, v_2) \& v_3 = v_3\},$$

где & — побитовое ``и''. Чтобы это было возможно, правила надо кодировать соответственно: продукция это пара, где первый элемент — битовый вектор длины N с единственной единицей в позиции, соответствующей нетерминалу в правой части, а второй элемент — вектор длины $2|N|$, с двумя единицами кодирующими первый и второй нетерминалы, соответственно.

Пример 6.7. Пусть $N = \{S, A, B\}$ и есть продукция $S \to AB$. Тогда занумеруем нетерминалы $S \to 0, A \to 1, B \to 2$ и продукция примет вид $[1, 0, 0] \to [0, 1, 0, 0, 0, 1]$. Матрица T_0 примет вид(в нашей нотации . означает, что в ячейке стоит $[0, 0, 0]$):

$$T_0 = \begin{pmatrix} . & [0,1,0] & . & [0,0,1] \\ . & . & [0,1,0] & . \\ [0,1,0] & . & . & . \\ [0,0,1] & . & . & . \end{pmatrix}$$

После выполнения умножения получим $T_1 = T_0 \times T_0$:

$$T_1 = \begin{pmatrix} . & [0,1,0] & . & [0,0,1] \\ . & . & [0,1,0] & . \\ [0,1,0] & . & . & [1,0,0] \\ [0,0,1] & . & . & . \end{pmatrix}$$

На практике в роли векторов могут выступать беззнаковые целые. Например, 32 бита под ячейки в матрице и 64 бита под правила (или 8 и 16, если запросы совсем маленькие, или 16 и 32). Тогда умножение выражается через битовые операции и сравнение.

Это может оказаться быстрее — в данном случае скорость зависит от деталей реализации.

При таких представлениях данные часто оказываются разреженны — возникает вопрос, как представлять матрицу. Среди способов — разреженные матрицы, GraphBLAS[2], GPGPU, CUTLASS[3]. Quad Tree [22].

Для вычислений лучше всего, когда все нужные для вычисления матрицы помещаются на одну карту. Хуже — если только одна пара матриц. Хуже всего, когда не помещается даже пара матриц. Поэтому хороши распределенные решения, например через GraphBLAS.

6.4 Обзор

- Lee. О конвертации парсеров КС-грамматик в перемножение булевых матриц [37]
- OpenCypher [35]
- J.Hellings. CFPQ [27, 29, 28]
- Zhang. CFPQ on rdf graphs [73]
- Bradford [12, 69, 15, 13]

Асимптотически приведенные алгоритмы имеют большую сложность, например $O(n^2|N|^2|P|\ (MM(n))$, где $MM(n)$ — сложность перемножения матриц $n \times n$, $|P|$ — мощность множества продукций, $|N|$ — мощность множества нетерминалов. Однако такая большая сложность компенсируется возможностью их распараллеливания, в результате чего они работают быстрее однопоточных алгоритов с лучшей сложностью.

Брэдфорт получил субкубическую сложность для частного случая — языка Дика с одним типом скобок [14].

6.5 Вопросы и задачи

1. Находить кратчайшие пути в графах, используя идеи из секции 6.1.
2. Превратить граф, использующийся для CFPQ, в дерево.
3. Реализовать предложенные идеи на различных архитектурах.
4. Замерить производительность и расходы памяти по сравнению с существующими реализациями.

7 Через тензорное произведение

Предыдущий подход позволяет выразить задачу поиска путей с ограничениями в терминах формальных 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 разрастанию. Можно ли как-то избежать этого?

[2]GraphBLAS — открытый стандарт для графовых алгоритмов — https://github.com/gunrock/graphblast

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

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

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

7.1 Рекурсивные автоматы и сети

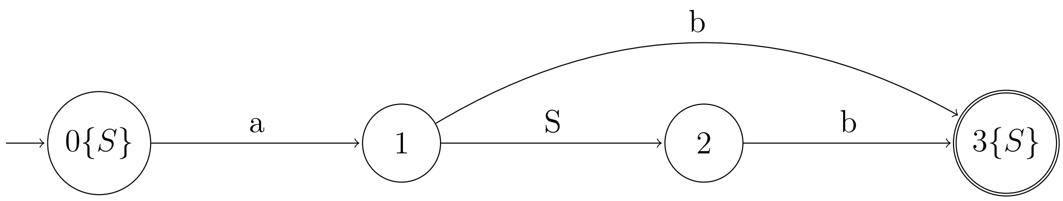
Рекурсивный автомат или сеть — это представление контекстно-свободных грамматик, обобщающее конечные автоматы. В нашей работе мы будем придерживаться термина **рекурсивный автомат**. Классическое определение рекурсивного автомата выглядит следующим образом.

Определение 7.1. Рекурсивный автомат — это кортеж вида $\langle N, \Sigma, S, D \rangle$, где

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

Построим рекурсивный автомат для грамматики G:

$$
\begin{aligned}
S &\to ASB \\
S &\to AB \\
A &\to a \\
B &\to b
\end{aligned}
$$



(1)

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 рекурсивный автомат можно рассматривать как конечный автомат над смешанным алфавитом. Именно такой взгляд мы будем использовать при изложении алгоритма.

7.2 Тензорное произведение

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

Определение 7.2. 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 7.1.

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

$$= \left(\begin{array}{cccc|cccc} 5 & 6 & 7 & 8 & 10 & 12 & 14 & 16 \\ 9 & 10 & 11 & 12 & 18 & 20 & 22 & 24 \\ 13 & 14 & 15 & 16 & 26 & 28 & 30 & 32 \\ \hline 15 & 18 & 21 & 24 & 20 & 24 & 28 & 32 \\ 27 & 30 & 33 & 36 & 36 & 40 & 44 & 48 \\ 39 & 42 & 45 & 48 & 52 & 56 & 60 & 64 \end{array}\right) \tag{2}$$

Заметим, что для определения тензорного произведения матриц достаточно определить операцию умножения на элементах исходных матриц. Также отметим, что произведение Кронекера не является коммутативным. При этом всегда существуют две матрицы перестоновок P и Q такие, 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 7.3. 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 $V_3 = V_1 \times V_2$, $E_3 = \{((v_1, v_2), (u_1, u_2)) \mid (v_1, u_1) \in E_1$ и $(v_2, u_2) \in E_2\}$.

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

1. множество вершин — это прямое произведение множеств 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 ребра 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.

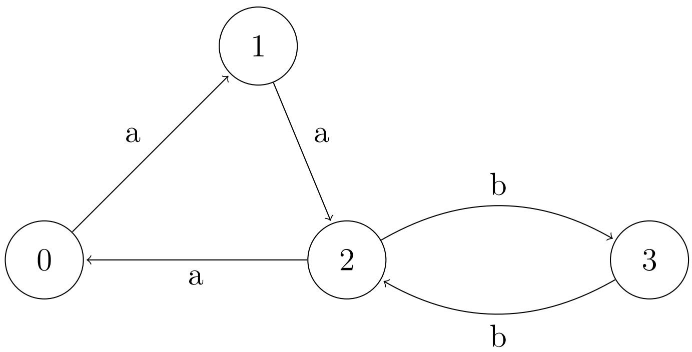
Осталось добавить метки к рёбрам. Это приведёт к логичному усилению требованя к существованию ребра: метки рёбер в исходных графах должны совпадать. Таким образом, мы получаем следующее определение тензорного произведения ориентированных графов с метками на рёбрах.

Определение 7.4. 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 $V_3 = V_1 \times V_2$, $E_3 = \{((v_1, v_2), l, (u_1, u_2)) \mid (v_1, l, u_1) \in E_1$ 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 тензорному произведению матриц смежностей исходных графов $\mathcal{G}_1$ и $\mathcal{G}_2$.

Пример 7.2. Рассмотрим пример. В качестве одного из графов возьмём рекурсивный автомат, построенный ранее (изображение 1). Его матрица смежности выглядит следующим образом.

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



(3)

Второй граф представлен на изображении 3. Его матрица смежности имеет следующий вид.

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

Теперь вычислим $M_1 \otimes M_2$.

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

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

Побалуемся с некоммутативностью и перестановками.

7.3 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 языков коммутативно, тензорное произведение нет, но это не страшно.

В итоговом автомате для любого нетерминала должен существовать хотя бы один путь из правильного стартового состояния в финальное, содержащий только терминалы. Если это не так, то данная грамматика непорождающая, то есть она не может породить ни одной цепочки, содержайщей только терминалы. Это делает задачу бессодержательной.

Алгоритм исполняется до тех пор, пока матрица смежности M_2 изменяется. На каждой итерации цикла алгоритм последовательно проделывает следующие команды: пересечение двух автоматов через тензорное произведение, транзитивное замыкание результата тензорного произведения и итерация по всем ячейкам получившейся после транзитивного замыкания матрицы. Во время итерации по ячейкам матрицы транзитивного замыкания алгоритм сначала проверяет наличие ребра в данной ячейке, а затем — принадлежность стартовой и конечной вершин ребра к стартовому и конечному состоянию входного рекурсивного автомата. При удовлетворении этих условий алгоритм добавляет нетерминал/нетерминалы, соответствующие стартовой и конечной вершинам ребра, в ячейку матрицы M_2, полученной с благодаря функции $getCoordinates(i, j)$.

Вместо того, чтобы перезаписывать каждый раз матрицу смежности M_2 можно вычислять только разницу. Для этого, правда, потребуется хранить в памяти ещё одну матрицу. Также для данной реализации надо проверять, что вычислительно дешевле: поддерживать

Listing 5 Поиск путей через тензорное произведение

```
function contextFreePathQueryingTP(G, 𝒢)
    R ← рекурсивный автомат для G
    N ← нетерминальный алфавит для R
    S ← стартовые состояния для R
    F ← конечные состояния для R
    M_1 ← матрица смежности R
    M_2 ← матрица смежности 𝒢
    while матрица M_2 изменяется do
        M_3 ← M_1 ⊗ M_2                                        ▷ Пересечение графов
        tC_3 ← transitiveClosure(M_3)
        n ← количество строк и столбцов матрицы M_3      ▷ размер матрицы M_3 = n × n
        for i ∈ 0..n do
            for j ∈ 0..n do
                if tC_3[i, j] then
                    s ← стартовая вершина ребра tC_3[i, j]
                    f ← конечная вершина ребра tC_3[i, j]
                    if s ∈ S and f ∈ F then
                        x, y ← getCoordinates(i, j)
                        M_2[x, y] ← M_2[x, y] ∪ {getNonterminals(s, f)}
    return M_2
```

разницу и потом каждый раз поэлементно складывать две матрицы или каждый раз вычислять полностью произведение.

Отметим, что для реальных графов и запросов результат тензорного произведения будет очень разрежен. Таким образом на готовых либах вычисление должно быть быстро.

7.4 Примеры

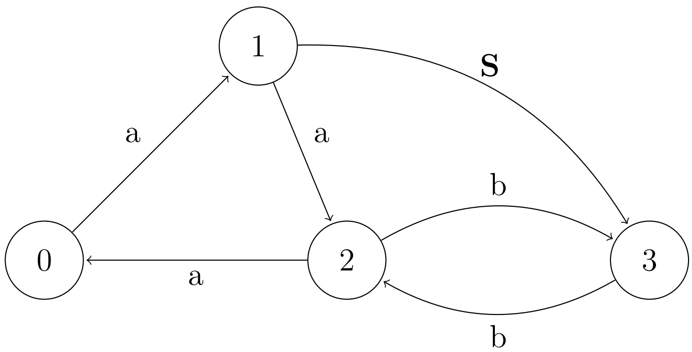
Рассмотрим подробно ряд примеров работы описанного алгоритма. Будем для каждой итерации внешнего цикла выписывать результаты основных операций: тензорного произведения, транзитивного замыкания, обновления матрицы смежности входного графа. Новые, по сравнению с предыдущим состоянием, элементы матриц будем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

Пример 7.3. Теоретически худший случай. Такой же как и для матричного.

Итерация 1 (конец). Начало в разделе 7.2, где мы вычислили тензорное произведение матриц смежности. Теперь нам осталось только вычислить транзитивное замыкание полученной матрицы.

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

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 автомате, 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.

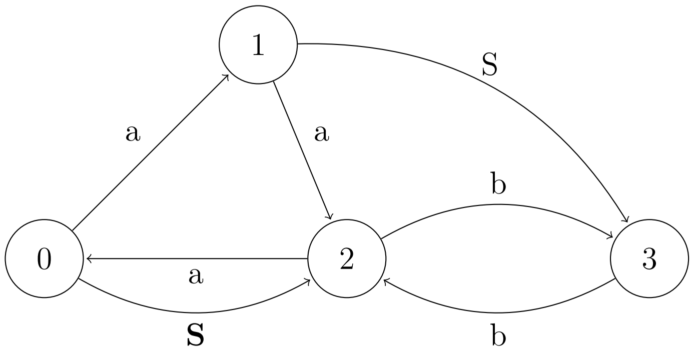
$$
M_3 = M_1 \otimes M_2 = \begin{pmatrix} \cdot & [a] & \cdot & \cdot \\ \cdot & \cdot & [S] & [b] \\ \cdot & \cdot & \cdot & [b] \\ \cdot & \cdot & \cdot & \cdot \end{pmatrix} \otimes \begin{pmatrix} \cdot & [a] & \cdot & \cdot \\ \cdot & \cdot & [a] & [S] \\ [a] & \cdot & \cdot & [b] \\ \cdot & \cdot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

$$
= \left(\begin{array}{cccc|cccc|cccc|cccc}
\cdot & \cdot & \cdot & \cdot & \cdot & [a] & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & [a] & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & [a] & \cdot & \cdot & \cdot & \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 & \mathbf{[S]} & \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 & [b] & \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 & [b] \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & [b] & \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 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot
\end{array}\right) \tag{6}
$$

Вычисляем транзитивное замыкание полученной матрицы:

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

В транзитивном замыкании появилось три новых ребра, однако только одно из них соединяет вершины, соответствующие стартовому и конечному состоянию входного рекурсивного автомата. Таким образом только это ребро должно быть 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.

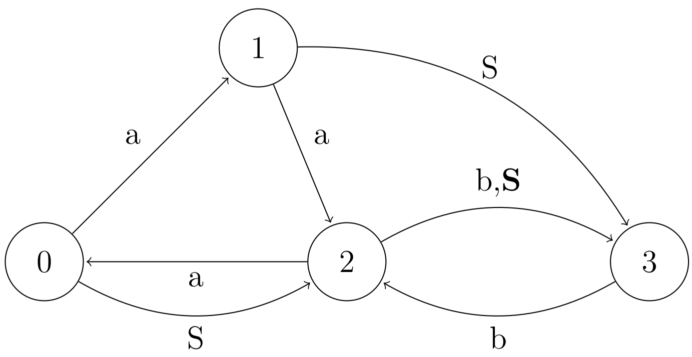
$$M_3 = M_1 \otimes M_2 = \begin{pmatrix} . & [a] & . & . \\ . & . & [S] & [b] \\ . & . & . & [b] \\ . & . & . & . \end{pmatrix} \otimes \begin{pmatrix} . & [a] & [S] & . \\ . & . & [a] & [S] \\ [a] & . & . & [b] \\ . & . & [b] & . \end{pmatrix} =$$

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

Затем вычисляем транзитивное замыкание:

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

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.

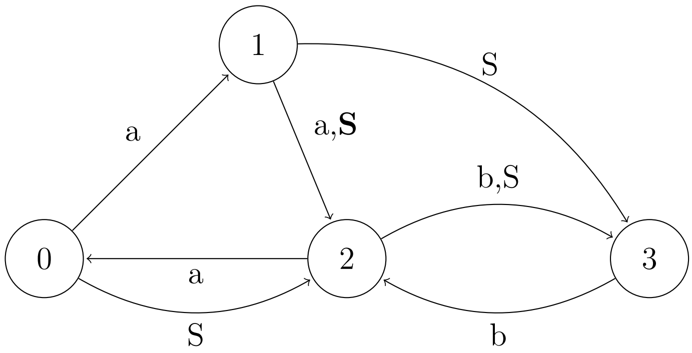
$$M_3 = M_1 \otimes M_2 = \begin{pmatrix} \cdot & [a] & \cdot & \cdot \\ \cdot & \cdot & [S] & [b] \\ \cdot & \cdot & \cdot & [b] \\ \cdot & \cdot & \cdot & \cdot \end{pmatrix} \otimes \begin{pmatrix} \cdot & [a] & [S] & \cdot \\ \cdot & \cdot & [a] & [S] \\ [a] & \cdot & \cdot & [b, S] \\ \cdot & \cdot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

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

Затем транзитивное замыкание:

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

И снова обновляем граф, так как в транзитивном замыкании появился 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.

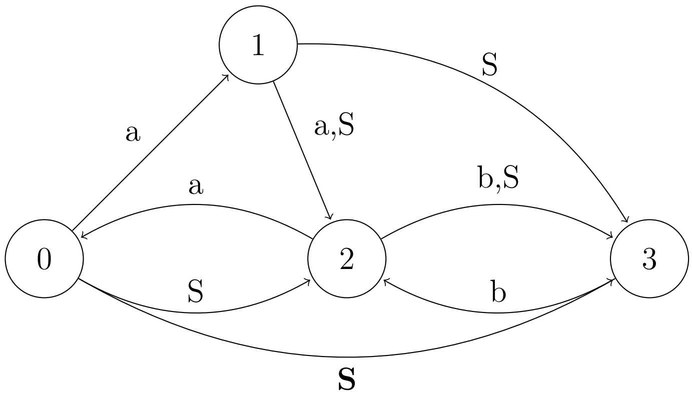
$$M_3 = M_1 \otimes M_2 = \begin{pmatrix} . & [a] & . & . \\ . & . & [S] & [b] \\ . & . & . & [b] \\ . & . & . & . \end{pmatrix} \otimes \begin{pmatrix} . & [a] & [S] & . \\ . & . & [a, S] & [S] \\ [a] & . & . & [b, S] \\ . & . & [b] & . \end{pmatrix} =$$

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

Затем вычисляем транзитивное замыкание:

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

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.

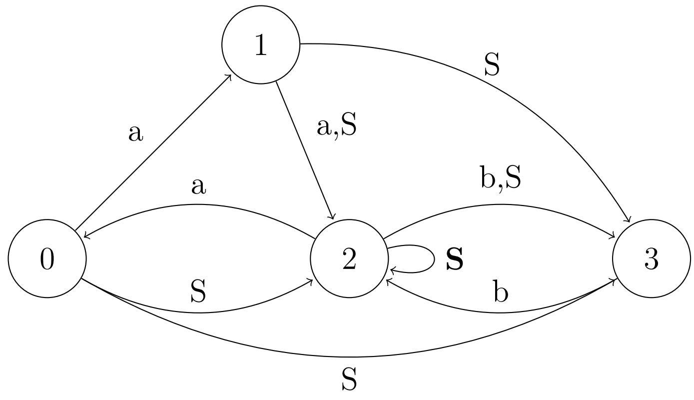
$$M_3 = M_1 \otimes M_2 = \begin{pmatrix} \cdot & [a] & \cdot & \cdot \\ \cdot & \cdot & [S] & [b] \\ \cdot & \cdot & \cdot & [b] \\ \cdot & \cdot & \cdot & \cdot \end{pmatrix} \otimes \begin{pmatrix} \cdot & [a] & [S] & [S] \\ \cdot & \cdot & [a,S] & [S] \\ [a] & \cdot & \cdot & [b,S] \\ \cdot & \cdot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

$$= \left(\begin{array}{cccc|cccc|cccc|cccc}
\cdot & \cdot & \cdot & \cdot & \cdot & [a] & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & [a] & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & [a] & \cdot & \cdot & \cdot & \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 & [S] & \mathbf{[S]} & \cdot & \cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & [S] & [S] & \cdot & \cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & [S] & \cdot & \cdot & \cdot & [b] \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & [b] & \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 & [b] \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & [b] & \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 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot
\end{array}\right) \tag{14}$$

Транзитивное замыкание:

$$tc(M_3) = \left(\begin{array}{cccc|cccc|cccc|cccc}
\cdot & \cdot & \cdot & \cdot & \cdot & [a] & \cdot & \cdot & \cdot & \cdot & [aS] & [aS] & \cdot & \cdot & [aSb] & [aSb] \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & [a] & \cdot & \cdot & \cdot & \cdot & [aS] & \cdot & \cdot & [aSb] & [ab] \\
\cdot & \cdot & \cdot & \cdot & [a] & \cdot & \cdot & \cdot & \cdot & \cdot & [aS] & \mathbf{[aS]} & \cdot & \cdot & \mathbf{[aSb]} & [aSb] \\
\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 & [S] & \mathtt{[S]} & \cdot & \cdot & \mathbf{[Sb]} & [Sb] \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & [S] & [S] & \cdot & \cdot & [Sb] & [Sb] \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & [S] & \cdot & \cdot & [Sb] & [b] \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & [b] & \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 & [b] \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & [b] & \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 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot
\end{array}\right) \tag{15}$$

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, ответ правильный. Мы видим, что количество итераций внешнего цикла такое же как и у алгоритма CYK (пример 5.3). И ещё что-то видим и можем понять.

Пример 7.4. 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:

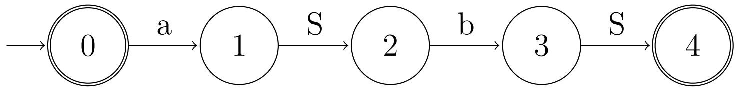
$$G_1 = \langle \{a,\ b\}, \{S\}, \{S \to a\ S\ b\ S \mid \varepsilon\} \rangle$$

Во второй грамматике мы будем использовать конструкции регулярных выражений в правой части правил. То есть вторая грамматика находитмся в EBNF (Расширенная форма Бэкуса-Наура [30, 71]).

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

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

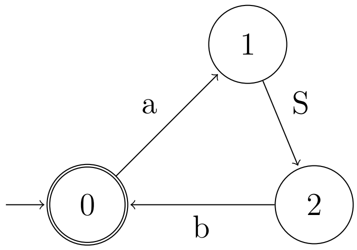
Рекурсивный автомат N_1 для грамматики G_1:



Матрица смежности N_1:

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

Рекурсивный автомат N_2 для грамматики G_2:



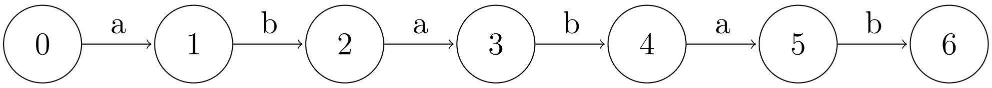
Матрица смежности N_2:

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

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 $M_1^2 \otimes M_2$.

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

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

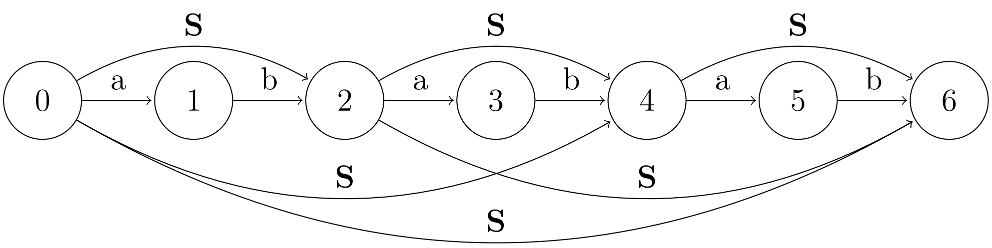
Опустим промежуточные шаги вычисления транзитивного замыкания M_3 и сразу представим конечный результат:

$$tc(M_3) =$$

$$\left(\begin{array}{ccccccc|ccccccc|ccccccc}
. & . & \mathbf{[aSb]} & . & \mathbf{[aSbaSb]} & . & \mathbf{[aSbaSbaSb]} & . & [a] & . & \mathbf{[aSba]} & . & \mathbf{[aSbaSba]} & . & . & \mathbf{[aS]} & . & \mathbf{[aSbaS]} & . & \mathbf{[aSbaSbaS]} & . \\
. & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . \\
. & . & . & . & \mathbf{[aSb]} & . & \mathbf{[aSbaSb]} & . & . & . & [a] & . & \mathbf{[aSba]} & . & . & . & . & \mathbf{[aS]} & . & \mathbf{[aSbaS]} & . \\
. & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . \\
. & . & . & . & . & . & \mathbf{[aSb]} & . & . & . & . & . & [a] & . & . & . & . & . & . & \mathbf{[aS]} & . \\
. & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . \\
. & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . \\
\hline
. & . & . & . & . & . & . & . & . & . & . & . & . & . & [S] & . & . & . & . & . & . \\
. & . & . & . & . & . & . & . & . & . & . & . & . & . & . & [S] & . & . & . & . & . \\
. & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & [S] & . & . & . & . \\
. & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & [S] & . & . & . \\
. & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & [S] & . & . \\
. & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & [S] & . \\
. & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & [S] \\
\hline
. & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . \\
. & . & [b] & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . \\
. & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . \\
. & . & . & . & [b] & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . \\
. & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . \\
. & . & . & . & . & . & [b] & . & . & . & . & . & . & . & . & . & . & . & . & . & . \\
. & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & .
\end{array}\right) \tag{17}$$

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, чтобы вычислить транзитивное замыкание на первой итерации.

Теперь запустим алгоритм на второй грамматике и том же входе.

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

$$= \left(\begin{array}{ccccccc|ccccccc|ccccccc|ccccccc|ccccccc}
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & [a] & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \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 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \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 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \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 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \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 & \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 & \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 & \cdot & \cdot & \cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \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 & \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 & \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{18}$$

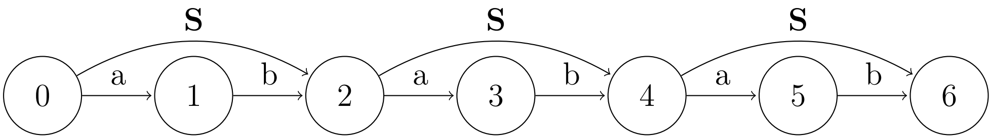
Уже сейчас можно заметить, что размер матриц, с которыми нам придётся работать, существенно увеличился, по сравнению с предыдущим вариантом. Это, конечно, закономерно, ведь в рекурсивном автомате для предыдущего варианта меньше состояний, а значит и матрица смежности имеет меньший размер.

Транзитивное замыкание:

$$tc(M_3) =$$

$$= \left(\begin{array}{ccccccc|ccccccc|ccccccc|ccccccc|ccccccc}
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & [a] & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \mathbf{[aS]} & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \mathbf{[aSb]} & \cdot & \cdot & \cdot & \cdot & \cdot & \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 & \cdot & \cdot & [a] & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \mathbf{[aS]} & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \mathbf{[aSb]} & \cdot & \cdot & \cdot & \cdot & \cdot & \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 & \cdot & \cdot & [a] & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \mathbf{[aS]} & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \mathbf{[aSb]} & \cdot & \cdot & \cdot & \cdot & \cdot & \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 & \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 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \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 & \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 & \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 & \cdot & \cdot & \cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \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 & \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 & \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:



Его матрица смежности:

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

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{20}
$$

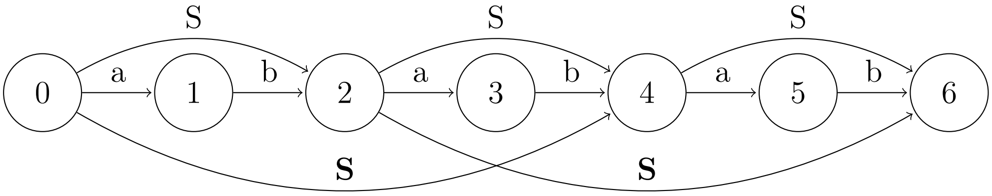
Транзитивное замыкание:

$tc(M_3) =$

$$
= \left(\begin{array}{ccccccc|ccccccc|ccccccc|ccccccc|ccccccc}
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & [a] & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \text{[aS]} & \cdot & \textbf{[aS]} & \cdot & \cdot & \cdot & \cdot & \cdot & \text{[aSb]} & \cdot & \textbf{[aSb]} & \cdot & \cdot & \cdot & \cdot & \text{[aSbS]} & \cdot & \textbf{[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 & \text{[aS]} & \cdot & \textbf{[aS]} & \cdot & \cdot & \cdot & \cdot & \cdot & \text{[aSb]} & \cdot & \textbf{[aSb]} & \cdot & \cdot & \cdot & \cdot & \text{[aSbS]} & \cdot & \textbf{[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 & \text{[aS]} & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \text{[aSb]} & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \text{[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{21}
$$

Обновлённый граф:

На этом шаге мы смогли “склеить” из подстрок, выводимых из S, более длинные пути. Однако, согласно правилам грамматики, мы смогли “склеить” 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{22}
$$

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 & \text{[aS]} & \cdot & \text{[aS]} & \cdot & \textbf{[aS]} & \cdot & \cdot & \cdot & \text{[aSb]} & \cdot & \text{[aSb]} & \cdot & \textbf{[aSb]} & \cdot & \cdot & \text{[aSbS]} & \cdot & \text{[aSbS]} & \cdot & \textbf{[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 & [a] & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \text{[aS]} & \cdot & \text{[aS]} & \cdot & \cdot & \cdot & \cdot & \cdot & \text{[aSb]} & \cdot & \text{[aSb]} & \cdot & \cdot & \cdot & \cdot & \text{[aSbS]} & \cdot & \text{[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 & \text{[aS]} & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \text{[aSb]} & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \text{[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 & [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 & [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 & [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 & [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 & [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 & [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 & [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 & [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{23}
$$

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 существенно бóльшими матрицами: 35×35 против 21×21.

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

7.5 Замечания о реализации

Блочная структура матриц даёт хорошую основу для распределённого умножения при построении транзитивного замыкания. В наших матрицах куча пустых ячеек, для которых можно даже не выделять память. Аналогично и для пустых блоков. Поэтому каждую матрицу можно представить в виде блочно-разреженной.

Транзитивное замыкание.

7.6 Вопросы и задачи

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

8 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.

8.1 Лес разбора как представление контекстно-свободной грамматики

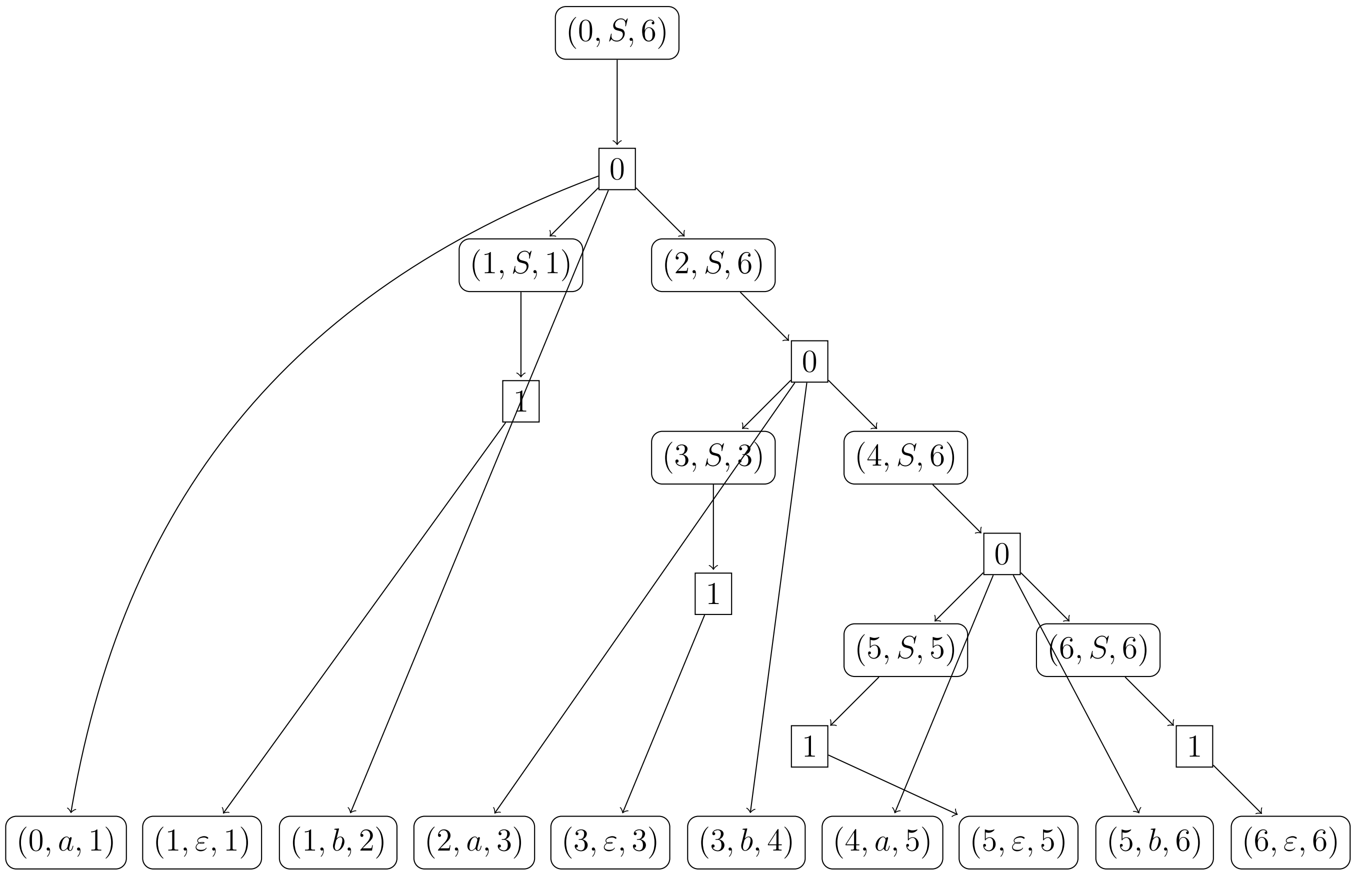
Для начала нам потребуется внести некоторые изменения в конструкцию дерева вывода.

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 граммтике: узел появляется благодаря применению конкретной продукции в процессе вывода. Давайте занумеруем все продукции в граммтике 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 8.1. Построим модифицированное дерево вывода цепочки ${}_0a_1b_2a_3b_4a_5b_6$ в грамматике

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

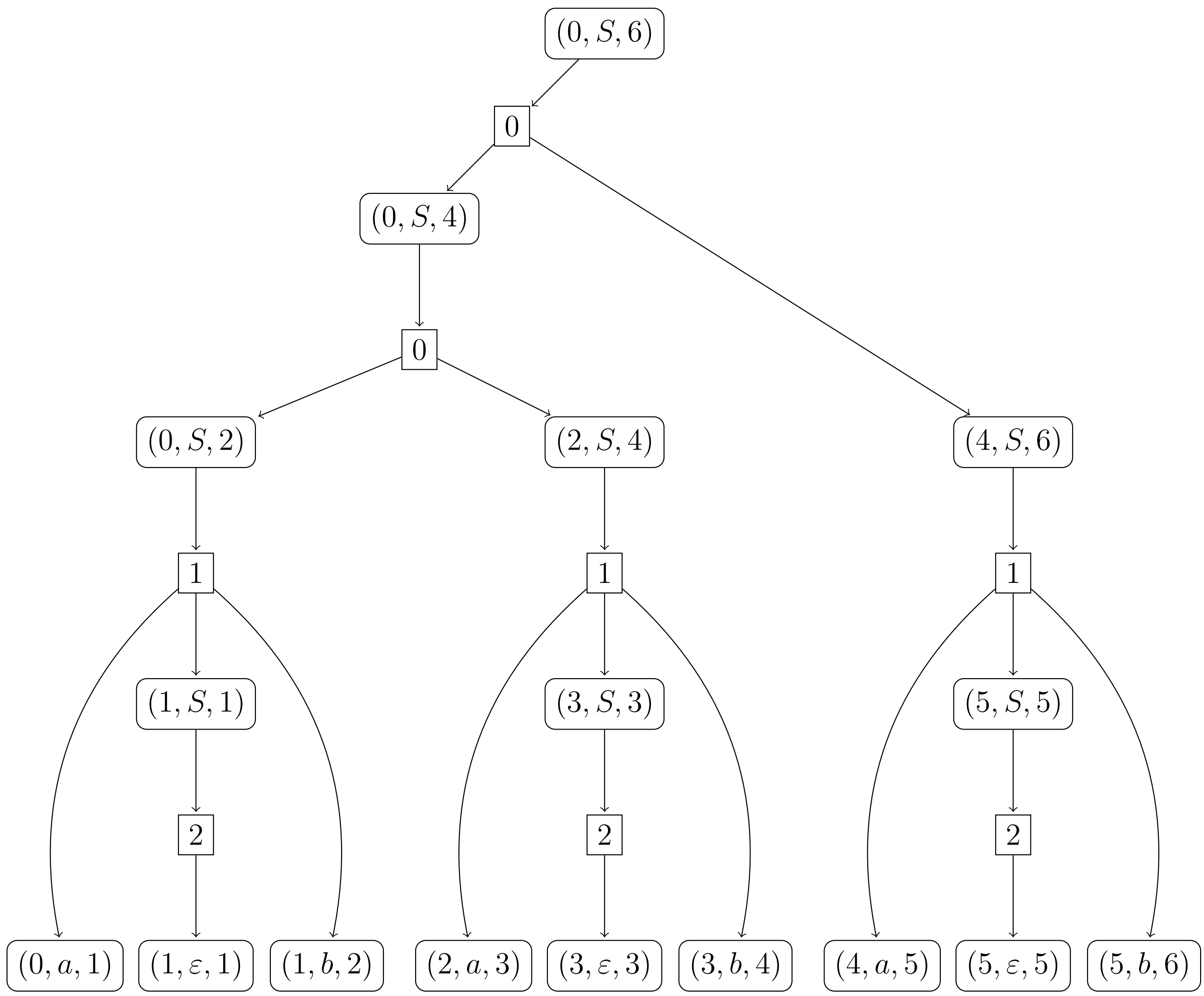


Сохраняемая нами дополнительная информация позволит переиспользовать узлы в том случае, если деревьев вывода оказалось несколько (в случае неоднозначной грамматики). При этом мы можем не бояться, что переиспользование узлов может привести к появлению ранее несуществовавших деревьев вывода, так как дополнительная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

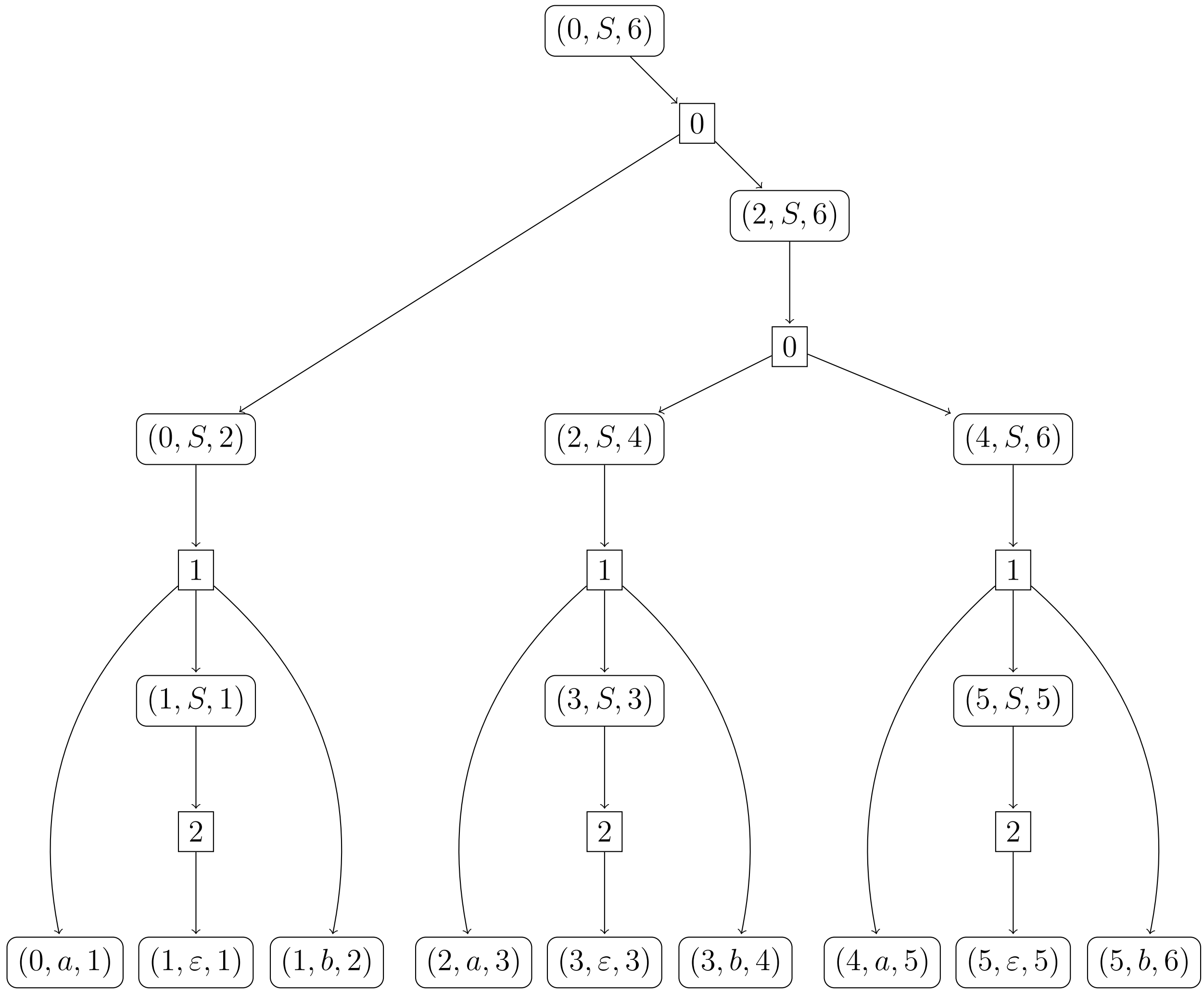
Пример 8.2. Сжатие леса вывода. Построим несколько деревьев вывода цепочки ${}_0a_1b_2a_3b_4a_5b_6$ в грамматике

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

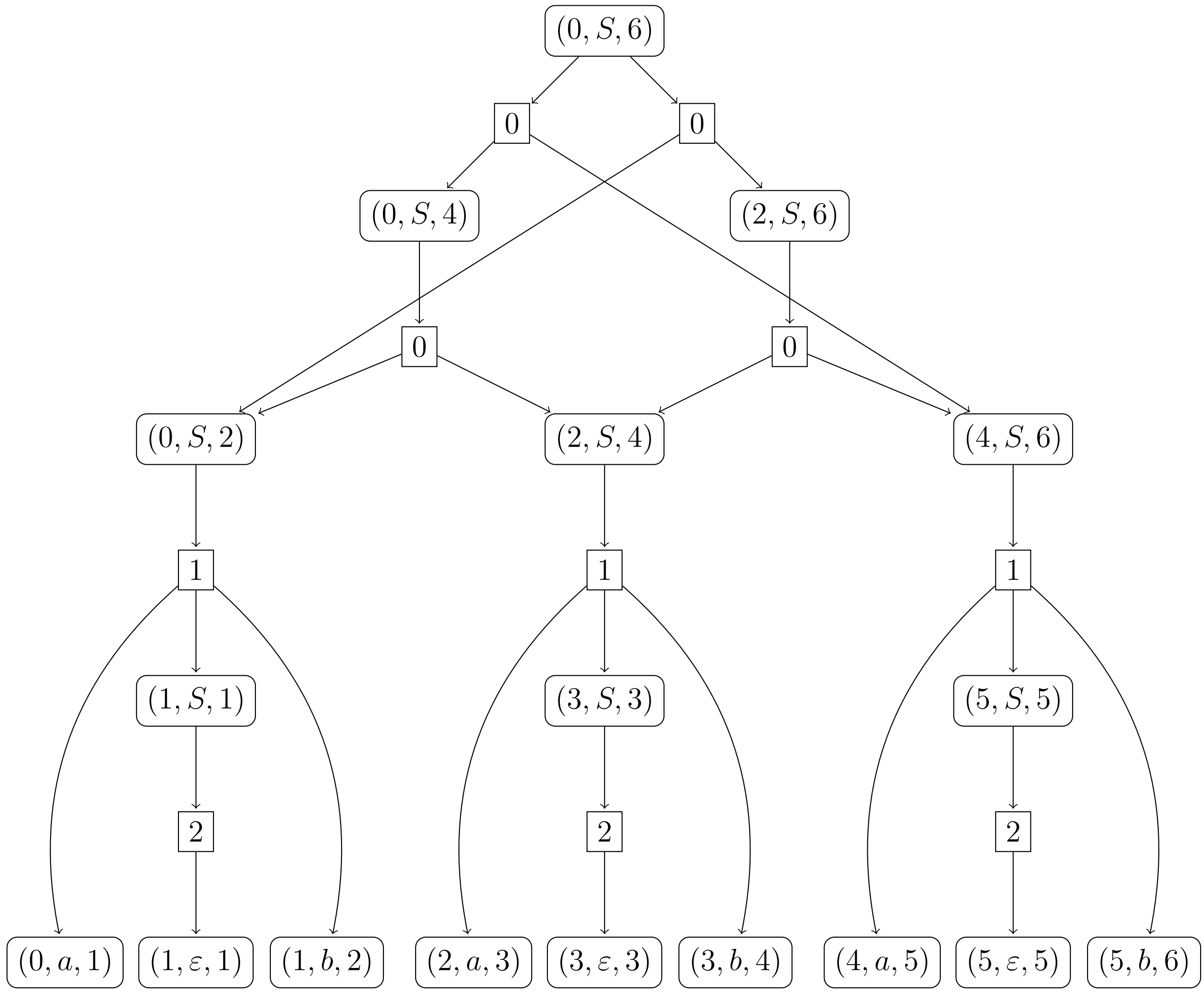
Пердположим, что мы строим левосторонний вывод. Тогда после первого применеия продукции 0 у нас есть два варианта переписывания первого нетерминала: либо с применением продукции 0, либо с применением продукции 1. В первом случае мы примени переписываение по подукции 0. Дальнейшие шаги деретерминированы 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 [51]. 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 [10] — в этом случае объём леса составляет $O(n^3)$, где n — это длина входной строки.

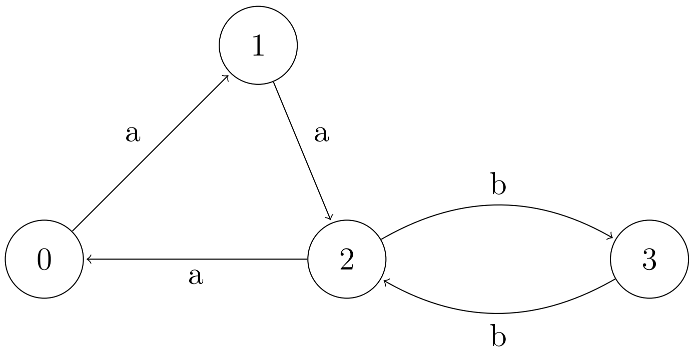
SPPF применяется в таких алгоритмах синтаксического анализа, как RNGLR [55], бинаризованная верся SPPF в BRNGLR [57] и GLL [56, 2][4].

В действительности SPPF может содержать в себе циклы. Для линейного входа можно получить граф, когда есть возможность выводить по грамматике бесконечные эпсилон-цепочки. Циклы будут вырожденными, но они будут. SPPF также можно построить и для графов.

В графе может существовать множество способов получить путь из одной вершины в другую. И точно так же при построении деревьев вывода путей может появиться несколько одинаковых нетерминалов, получаемых в разных деревьях по-разному. При объединении в SPPF может оказаться, что какой-то путь из вершины a в вершину b является подпутем другого пути из вершины a в вершину b, просто более длинного. То есть появятся циклические зависимости.

[4]Ещё немного полезной информации про SPPF: `http://www.bramvandersanden.com/post/2014/06/shared-packed-parse-forest/`.

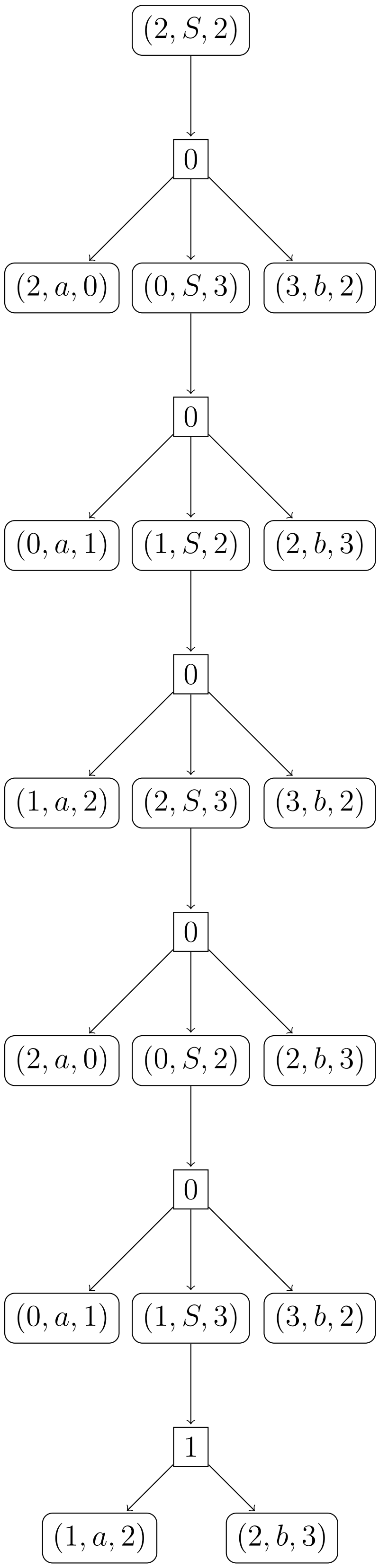
Пример 8.3. Дан граф $\mathcal{G}$:



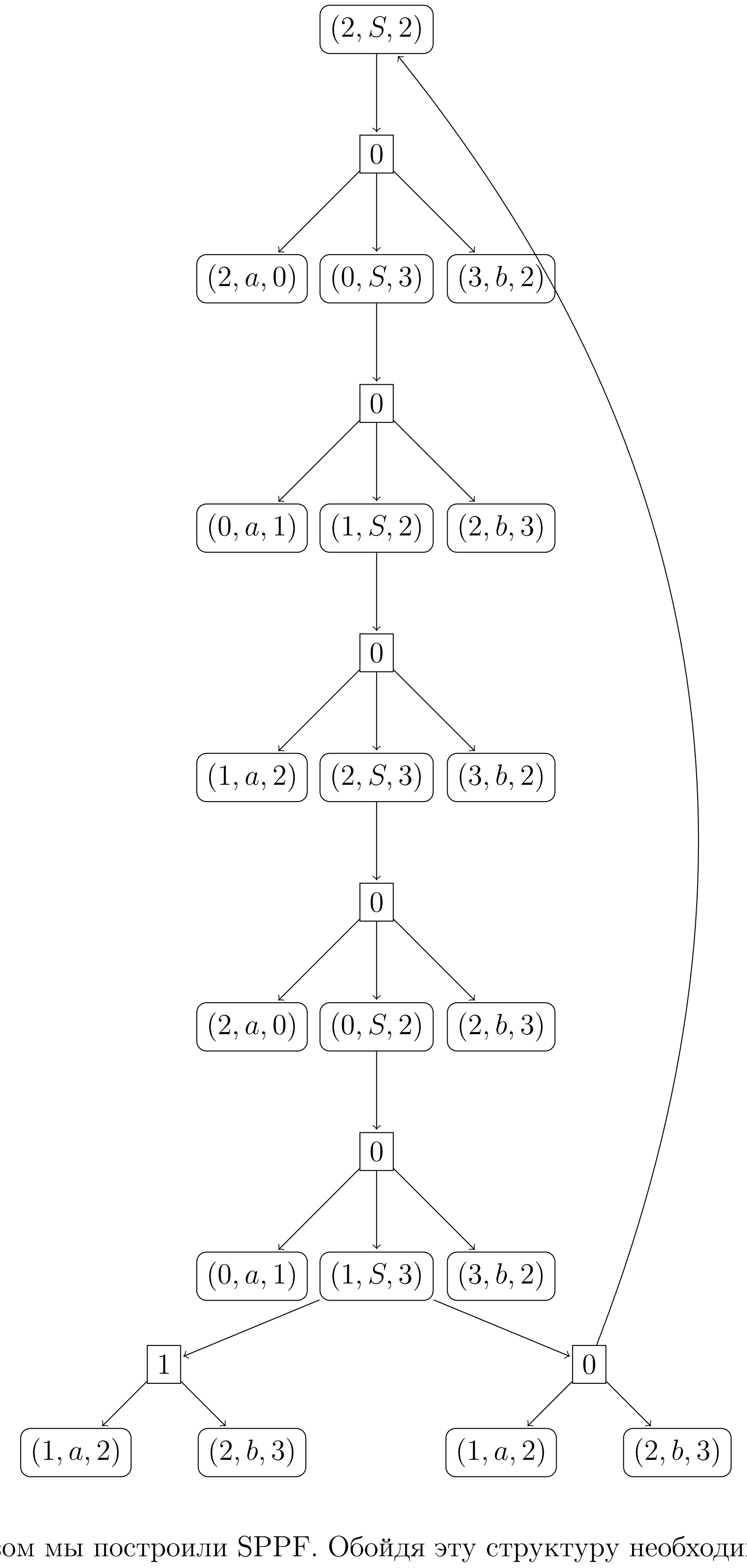
Дана грамматика

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

Попробуем найти все пути из вершины 2 в вершину 2, выводимые из нетерминала S. Методом пристального взгляда найдем один из них. Пусть это будет ${}_2a_0a_1a_2a_0a_1a_2b_3b_2b_3b_2b_3b_2$. Построим дерево его вывода.



Так как это только один путь, а таких путей гораздо больше, то к нетерминалу ${}_1S_3$ применим также применим продукцию 0. Тогда мы получим нетерминал ${}_2S_2$, который и создаст цикл.



Таким образом мы построили SPPF. Обойдя эту структуру необходимое количество раз, мы можем получить любой путь, удовлетворяющий условию. Более того, в полученном графе можно получать любые другие пути по соответствующим нетерминалам, содержащимся в узлах леса.

По полученному графу построим грамматику:

$$
\begin{array}{l}
(0)\ {}_2S_2 \to\ {}_2a_0\ {}_0S_3\ {}_3b_2 \\
(1)\ {}_0S_3 \to\ {}_0a_1\ {}_1S_2\ {}_2b_3 \\
(2)\ {}_1S_2 \to\ {}_1a_2\ {}_2S_3\ {}_3b_2 \\
(3)\ {}_2S_3 \to\ {}_2a_0\ {}_0S_2\ {}_2b_3 \\
(4)\ {}_0S_2 \to\ {}_0a_1\ {}_1S_3\ {}_3b_2 \\
(5)\ {}_1S_3 \to\ {}_1a_2\ {}_2S_2\ {}_2b_3 \\
(6)\ {}_1S_3 \to\ {}_1a_2\ {}_2b_3
\end{array}
$$

Видим, что для одного единственного нетерминала ${}_1S_3$ существует 2 правила, одно из которых рекурсивное. Попробуем получить левосторонний вывод какой-нибудь цепочки в этой грамматике:

$$
\begin{array}{l}
\boldsymbol{{}_2S_2} \xRightarrow{(0)} \\
{}_2a_0\ \boldsymbol{{}_0S_3}\ {}_3b_2 \xRightarrow{(1)} \\
{}_2a_0\ {}_0a_1\ \boldsymbol{{}_1S_2}\ {}_2b_3\ {}_3b_2 \xRightarrow{(2)} \\
{}_2a_0\ {}_0a_1\ {}_1a_2\ \boldsymbol{{}_2S_3}\ {}_3b_2\ {}_2b_3\ {}_3b_2 \xRightarrow{(3)} \\
{}_2a_0\ {}_0a_1\ {}_1a_2\ {}_2a_0\ \boldsymbol{{}_0S_2}\ {}_2b_3\ {}_3b_2\ {}_2b_3\ {}_3b_2 \xRightarrow{(4)} \\
{}_2a_0\ {}_0a_1\ {}_1a_2\ {}_2a_0\ {}_0a_1\ \boldsymbol{{}_1S_3}\ {}_3b_2\ {}_2b_3\ {}_3b_2\ {}_2b_3\ {}_3b_2 \xRightarrow{(5)} \\
{}_2a_0\ {}_0a_1\ {}_1a_2\ {}_2a_0\ {}_0a_1\ {}_1a_2\ \boldsymbol{{}_2S_2}\ {}_2b_3\ {}_3b_2\ {}_2b_3\ {}_3b_2\ {}_2b_3\ {}_3b_2 \xRightarrow{(0)} \\
{}_2a_0\ {}_0a_1\ {}_1a_2\ {}_2a_0\ {}_0a_1\ {}_1a_2\ {}_2a_0\ \boldsymbol{{}_0S_3}\ {}_3b_2\ {}_2b_3\ {}_3b_2\ {}_2b_3\ {}_3b_2\ {}_2b_3\ {}_3b_2 \xRightarrow{(1)} \\
{}_2a_0\ {}_0a_1\ {}_1a_2\ {}_2a_0\ {}_0a_1\ {}_1a_2\ {}_2a_0\ {}_0a_1\ \boldsymbol{{}_1S_2}\ {}_2b_3\ {}_3b_2\ {}_2b_3\ {}_3b_2\ {}_2b_3\ {}_3b_2\ {}_2b_3\ {}_3b_2 \xRightarrow{(2)} \\
{}_2a_0\ {}_0a_1\ {}_1a_2\ {}_2a_0\ {}_0a_1\ {}_1a_2\ {}_2a_0\ {}_0a_1\ {}_1a_2\ \boldsymbol{{}_2S_3}\ {}_3b_2\ {}_2b_3\ {}_3b_2\ {}_2b_3\ {}_3b_2\ {}_2b_3\ {}_3b_2\ {}_2b_3\ {}_3b_2 \xRightarrow{(3)} \\
{}_2a_0\ {}_0a_1\ {}_1a_2\ {}_2a_0\ {}_0a_1\ {}_1a_2\ {}_2a_0\ {}_0a_1\ {}_1a_2\ {}_2a_0\ \boldsymbol{{}_0S_2}\ {}_2b_3\ {}_3b_2\ {}_2b_3\ {}_3b_2\ {}_2b_3\ {}_3b_2\ {}_2b_3\ {}_3b_2\ {}_2b_3\ {}_3b_2 \xRightarrow{(4)} \\
{}_2a_0\ {}_0a_1\ {}_1a_2\ {}_2a_0\ {}_0a_1\ {}_1a_2\ {}_2a_0\ {}_0a_1\ {}_1a_2\ {}_2a_0\ {}_0a_1\ \boldsymbol{{}_1S_3}\ {}_3b_2\ {}_2b_3\ {}_3b_2\ {}_2b_3\ {}_3b_2\ {}_2b_3\ {}_3b_2\ {}_2b_3\ {}_3b_2\ {}_2b_3\ {}_3b_2 \xRightarrow{(6)} \\
{}_2a_0\ {}_0a_1\ {}_1a_2\ {}_2a_0\ {}_0a_1\ {}_1a_2\ {}_2a_0\ {}_0a_1\ {}_1a_2\ {}_2a_0\ {}_0a_1\ {}_1a_2\ {}_2b_3\ {}_3b_2\ {}_2b_3\ {}_3b_2\ {}_2b_3\ {}_3b_2\ {}_2b_3\ {}_3b_2\ {}_2b_3\ {}_3b_2\ {}_2b_3\ {}_3b_2
\end{array}
$$

Мы получили цепочку, которая действительно является путем из вершины 2 в вершину 2 в заданном графе. Таким образом выводятся и любые другие соответствующие пути.

8.2 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 $G = \langle\{a, b\}, \{S\}, \{S \to \varepsilon \mid a\ S\ b\ |S\ S\}, S\rangle$.

3. Постройте все правосторонние выводы цепочки $w = ababab$ в грамматике $G = \langle\{a, b\}, \{S\}, \{S \to \varepsilon \mid a\ S\ b\ |S\ S\}, S\rangle$.

4. Постройте все деревья вывода цепочки $w = ababab$ в грамматике $G = \langle\{a,b\},\{S\},\{S \to \varepsilon \mid a\ S\ b\ |S\ S\}, S\rangle$, соответствующие левосторонним выводам.

5. Постройте все деревья вывода цепочки $w = ababab$ в грамматике $G = \langle\{a,b\},\{S\},\{S \to \varepsilon \mid a\ S\ b\ |S\ S\}, S\rangle$, 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.

9 Алгоритм на основе восходящего анализа

Традиционно, алгоритмы, применияемые для анализа языков программирвания как раз умеют строить дерево разбора — то, что нам надо. Только нам бы лес. Вот и посмотрим, как это можно сделать.

Сперва погововрим про классический синтаксический анализ, потом про его адоптацию к анализу графов.

9.1 Восходящий синтаксический анализ

Основы LR-анализа.

LR-автомат. Ситуации (item). Ядро, замыкание, предпросмотр.

Сперва будет про LR(0)

Таблицы, конфликты.

Пример 9.1. Пример автомата и таблиц.

Пример 9.2. Ну и пример разбора.

На практике конфликты стараются решать ещё и на этапе генерации. Да, реальные тулы могут сгенерировать парсер по неоднозначной грамматике: из переноса или свёртки выбирать перенос, из нескольких свёрток — первую в каком-то порядке (обычно в порядке появления соответствующих продукций в граммтике).

Про модификации: SLR, LALR — меньше конфликтов. иногда больше таблицы.

Немного про рекурсивно-восходящй анализ.

Но вообще, бывают обобщённые анализаторы, которые умеют обрабатывать все конфликты.

Томита — вообще говоря не все граммтики, RNGLR [55] — все грамматики, но вообще говоря не за куб, а за произвольный полином, более продвинутый BRNGLR [**?**] — все граммтики за куб.

Куча интересностей и подробностей [21].

Общая идея — объединение состояний.

Пример 9.3. Ну и пример с конфликтами и объединением состояний.

9.2 Кс запросы

Наша реализация [66]

Конфликты типа перенос-перенос — ветвления в графах.

Слияние состояний в циклах.

Проходящие редукции.

А почему терминируется?

9.3 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 !!!!

10 Алгоритм на основе нисходящего анализа

GLL [25]

Другие реализации [38]

10.1 Нисходящий синтаксический анализ

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

10.1.1 Рекурсивный спуск

Пример 10.1. Постороим функцию рекурсивного спуска для продукции $S \to aSbS$.

Если возвращаемое значение этой функции пусто, то разбор завершился успехом.

Listing 6 Функция рекурсивного спуска

```
function S(hd′, tl′)
    res = {}
    if hd′ = a then
        res = S(tl′)
        else error
    if head(res) = b then
        res = S(tail(res))
        else error
    return res
```

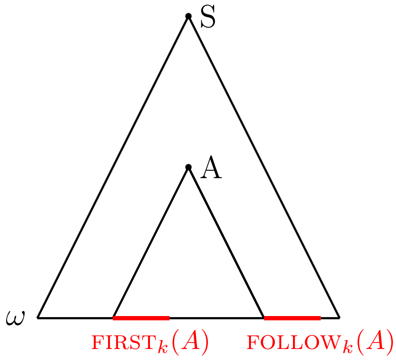
10.1.2 LL(k)-алгоритм синтаксического анализа

LL(k) — алгоритм синтаксического анализа — нисходящий анализ без отката, но с предпросмотром. Решение о том, какую продукцию применять, принимается на основании k следующих за текущим символом. Временная сложность алгоритма $O(n)$, где n — длина слова.

Алгоритм использует входной буфер, стек для хранения промежуточных данных и таблицу анализатора, которая управляет процессом разбора. В ячейке таблицы указано правило, которое нужно применять, если рассматривается нетерминал A, а следующие m символов строки — $t_1 \dots t_m$, где $m \leq k$. Также в таблице выделена отдельная колонка для \$ — маркера конца строки.

	$\dots$	$t_1 \dots t_m$	$\dots$	\$
$\dots$	$\dots$	$\dots$	$\dots$	$\dots$
A	$\dots$	$A \to \alpha$	$\dots$	$\dots$
$\dots$	$\dots$	$\dots$	$\dots$	$\dots$

Для построения таблицы вычисляются множества FIRST_k и FOLLOW_k. Идейно их можно понимать, как первые или последующие k символов в результирующем выводе, при использовании нетерминала A. Данную мысль хорошо иллюстрирует рисунок:



Определим их формально:

Определение 10.1. Пусть $G = \langle N, \Sigma, P, S \rangle$ — КС-грамматика. Множество FIRST_k определено для сентециальной формы α следующим образом:

$$\text{FIRST}_k(\alpha) = \{\omega \in \Sigma^* \mid \alpha \overset{*}{\Rightarrow} \omega \text{ и } |\omega| < k \text{ либо } \exists \beta : \alpha \overset{*}{\Rightarrow} \omega\beta \text{ и } |\omega| = k\}, \text{ где } \alpha, \beta \in (N \cup \Sigma)^*$$

Определение 10.2. Пусть $G = \langle N, \Sigma, P, S \rangle$ — КС-грамматика. Множество $\textsc{follow}_k$ определено для сентециальной формы β следующим образом:

$$\textsc{follow}_k(\beta) = \{\omega \in \Sigma^* \mid \exists \gamma, \alpha : S \overset{*}{\Rightarrow} \gamma\beta\alpha \text{ и } \omega \in \textsc{first}_k(\alpha)\}$$

В частном случае для $k = 1$:

$$\textsc{first}_1(\alpha) = \{a \in \Sigma \mid \exists \gamma \in (N \cup \Sigma)^* : \alpha \overset{*}{\Rightarrow} a\gamma\}, \text{ где } \alpha \in (N \cup \Sigma)^*$$

$$\textsc{follow}_1(\beta) = \{a \in \Sigma \mid \exists \gamma, \alpha \in (N \cup \Sigma)^* : S \overset{*}{\Rightarrow} \gamma\beta a\alpha\}, \text{ где } \beta \in (N \cup \Sigma)^*$$

Множество $\textsc{first}_1$ можно вычислить, пользуясь следующими соотношениями:

- $\textsc{first}_1(a\alpha) = \{a\}, a \in \Sigma, \alpha \in (N \cup \Sigma)^*$
- $\textsc{first}_1(\varepsilon) = \{\varepsilon\}$
- $\textsc{first}_1(\alpha\beta) = \textsc{first}_1(\alpha) \cup (\textsc{first}_1(\beta), \text{если } \varepsilon \in \textsc{first}_1(\alpha))$
- $\textsc{first}_1(A) = \textsc{first}_1(\alpha) \cup \textsc{first}_1(\beta)$, если в грамматике есть правило $A \to \alpha \mid \beta$

Алгоритм для вычисления множества $\textsc{follow}_1$:

- Положим $\textsc{follow}_1(X) = \varnothing, \forall X \in N$
- $\textsc{follow}_1(S) = \textsc{follow}_1(S) \cup \{\$\}$, где S— стартовый нетерминал
- Для всех правил вида $A \to \alpha X \beta : \textsc{follow}_1(X) = \textsc{follow}_1(X) \cup (\textsc{first}_1(\beta) \setminus \{\varepsilon\})$
- Для всех правил вида $A \to \alpha X$ и $A \to \alpha X \beta$, где $\varepsilon \in \textsc{first}_1(\beta)$: $\textsc{follow}_1(X) = \textsc{follow}_1(X) \cup \textsc{follow}_1(A)$
- Последние два пункта применяются пока есть что добавлять в строящиеся множества.

Пример множеств $\textsc{first}_1$ для нетерминалов следующей грамматики:

$$\begin{array}{ll}
S \to aS' & \textsc{first}_1(S) = \{a\} \\
S' \to AbBS' \mid \varepsilon & \textsc{first}_1(A) = \{a, \varepsilon\} \\
A \to aA' \mid \varepsilon & \textsc{first}_1(A') = \{a, b\} \\
A' \to b \mid a & \textsc{first}_1(B) = \{c, \varepsilon\} \\
B \to c \mid \varepsilon & \textsc{first}_1(S') = \{a, b, \varepsilon\}
\end{array}$$

Пример множеств $\textsc{follow}_1$ для нетерминалов следующей грамматики:

$$\begin{array}{llr}
S \to aS' & \textsc{follow}_1(S) = \{\$\} & \\
S' \to AbBS' \mid \varepsilon & \textsc{follow}_1(S') = \{\$\} & (S \to aS') \\
A \to aA' \mid \varepsilon & \textsc{follow}_1(A) = \{b\} & (S' \to AbBS') \\
A' \to b \mid a & \textsc{follow}_1(A') = \{b\} & (A \to aA') \\
B \to c \mid \varepsilon & \textsc{follow}_1(B) = \{a, b, \$\} & (S' \to AbBS', \varepsilon \in \textsc{first}_1(S'))
\end{array}$$

Таблица заполняется следующим образом: продукции $A \to \alpha, \alpha \neq \varepsilon$ помещаются в ячейки (A, a), где $a \in \text{FIRST}_1(A)$, продукции $A \to \varepsilon$ — в ячейки (A, a), где $a \in \text{FOLLOW}_1(A)$

Пример 10.2. Пример таблицы для грамматики $S \to aSbS \mid \varepsilon$

N	FIRST$_1$	FOLLOW$_1$	a	b	\$
S	$\{a, \varepsilon\}$	$\{b, \$\}$	$S \to aSbS$	$S \to \varepsilon$	$S \to \varepsilon$

Однако, не для всех грамматик по множествам FIRST$_k$ и FOLLOW$_k$ возможно выбрать применяемую продукцию, а значит, нельзя однозначно построить таблицу, необходимую для работы алгоритма, поэтому данный алгоритм применим только для грамматик особого класса — LL(k).

Определение 10.3. LL(k) грамматика — грамматика, для которой на основании множеств FIRST$_k$ и FOLLOW$_k$ можно однозначно определить, какую продукцию применять.

Важно заметить, что при больших k строимая нами таблица сильно разрастается, поэтому на практике данный алгоритм применим для небольших значений k.

Ход работы: Интерпретатор автомата принимает входную строку и построенную управляющую таблицу и работает следующим образом. В каждый момент времени конфигурация автомата это позиция во входной строке и стек. В начальный момент времени стэк пуст, а позиция во входной строке соответствует её началу. На певом шаге в стек добавляются последовательно сперва симаол концы строки, затем стартовый нетерминал. На каждом шаге анализируется существующая конфигурация и совершается одно из действий.

- Если текущая позиция — конец строки и вершина стека — символ конца строки, то успешно завершаем разбор.
- Если текушая вершина стека — терминал, то проверяем, что позиция в строке соответствует этому терминалу. Если да, то снимаем элемент со стека, сдвигаем позицию на единицу и продолжаем разбор. Иначе завершаем разбор с ошибкой.
- Если текущая врешина стека — нетерминал N_i и текущий входной символ t_j, то ищем в управляющей таблице ячейку с координатами (N_i, t_j) и записываем на стек содержимое этой ячейки.

Пример 10.3. Пример работы LL анализатора. Рассмотрим грамматику $S \to aSbS \mid \varepsilon$ и выводимое слово $\omega = abab$.

Расмотрим пошагово работу алгоритма, будем использовать таблицу, построенную в предыдущем примере:

1. Начало работы.

   Стек:

\$

   входное слово:

a	b	a	b	\$

   Финальный символ лежит на стеке, а указатель указывает на первый символ слова.

2. кладем стартовый символ на стек

   Стек:

S
\$

   входное слово:

a	b	a	b	\$

3. Ищем ячейку с координатами (S, a), применяем продукцию из ячейки.

Стек:

a
S
b
S
$

входное слово:

a	b	a	b	$

4. Снимаем терминал a со стека и двигаем указатель.

Стек:

S
b
S
$

входное слово:

a	b	a	b	$

5. Ищем ячейку с координатами (S, b), применяем продукцию из ячейки.

Стек:

b
S
$

входное слово:

a	b	a	b	$

6. Снимаем терминал b со стека и двигаем указатель.

Стек:

S
$

входное слово:

a	b	a	b	$

7. Ищем ячейку с координатами (S, a), применяем продукцию из ячейки.

Стек:

a
S
b
S
$

входное слово:

a	b	a	b	$

8. Снимаем терминал a со стека и двигаем указатель.

Стек:

S
b
S
$

входное слово:

a	b	a	b	$

9. Ищем ячейку с координатами (S, b), применяем продукцию из ячейки.

Стек:

b
S
$

входное слово:

a	b	a	b	$

10. Снимаем терминал b со стека и двигаем указатель.

Стек:

S
$

входное слово:

a	b	a	b	$

11. Ищем ячейку с координатами (S, \$), применяем продукцию из ячейки.

    Стек:

$

    входное слово:

a	b	a	b	$

12. Оказались в конце строки и на вершине стека символ конца — завершаем разбор.

Еще одна существенная проблема данного алгоритма — грамматики, содержащие леворекурсивные правила, т.е правила вида $Q \to Q\omega$. Действительно, встретив на вершине стека нетерминал Q и применив данное правило, на вершине стека снова окажется Q и мы будем вынуждены вновь применять это правило, таким образом алгоритм зациклится. Та же проблема встречаестя и в грамматиках со скрытой левой рекурсией.

А ещё надо бы деревья научиться строить.

Пример 10.4. Пример работы LL анализатора с деревом. Таблца вида (стек, указатель во входе, дерево, комментарии)

Итого. По некоторым грамматикам можно построить LL(k) анализатор (назовём их LL(k) граммтиками), но не по всем. С левой рекурсией, конечно, можно бороться, существуют алгоритмы устранения левой и скрытой левой рекурсии, а вот с неоднозначностями ничего не поделаешь.

10.2 GLL и его применение для КС запросов

Можно построить анализатор, работающий с произвольными КС-грамматиками. Generalized LL (GLL) [56, 2]

Принцип работы остается абсолютно таким же как и для табличного LL:

- Сначала по грамматике строится *управляющая* таблица
- Затем построенная таблица команд и непосредственно анализируемое слово поступают на вход абстрактному интерпретатору.
- Для своей работы интерпретатор поддерживает некоторую вспомогательную структуру данных (стек для LL).
- Один шаг разбора состоит в том, чтобы рассмотреть текущую позицию в слове, применить соответствующее ей правило из таблицы и при возможности сдвинуть позицию разбора вправо.

Где в этой схеме возникают ограничения на вид обрабатываемой грамматики для алгоритма LL? На самом первом шаге — при построении таблицы может возникнуть ситуация, когда одному нетерминалу N_j и последовательности $first_k(N_j)$ соответствует несколько продукций грамматики. В этом случае грамматика признавалась не соответствующей классу LL(k) и отвергалась анализатором.

Теперь же мы разрешим такую ситуацию и в этом случае в ячейку таблицы будем записывать все продукции грамматики, соответствующие этой ячейке. Однако сразу же возникает вопрос — а что делать интерпретатору, когда при разборе ему необходимо применить правило, состоящее из нескольких продукций? Общий ответ такой — необходим некоторый вид недетерминизма, при котором интерпретатор мог бы “параллельно” обрабатывать несколько возможных вариантов синтаксического разбора.

Эти два свойства (модифицированная управляющая таблица и недетерминизм) суть главные принципиальные отличия GLL(k) от LL(k). Далее мы перейдем к рассмотрению непосредственно технической реализации описанного алгоритма.

Нам необходимо научиться задавать различные ветви (пути) синтаксического разбора и переключаться между ними. Заметим, что состояние любой ветви в любой момент времени суть следующее: необходимо распознать символ $N_j \in N \cup \Sigma$ из продукции X, начиная с элемента слова под индексом i. Т.е. имеем позицию в слове и позицию символа в продукции. Последнее принято называть *слотом грамматики*.

Определение 10.4. Пусть $G = \langle N, \Sigma, P, S \rangle$ — КС-грамматика. *Слотом грамматики* G (позицией грамматики G) назовем пару из продукции $X \in P$ и позиции $0 \leq q \leq length(body(X))$ тела продукции X. При этом введем следующее обозначение $X ::= \alpha \cdot \beta, \quad \alpha, \beta \in (N \cup \Sigma)^*$, где $\cdot$ указывает на позицию в продукции.

Описанная пара позиций уже однозначно задает состояние синтаксического разбора. Имеем множество состояний и переходов между ними — возникает естественное желание воспользоваться терминами графов для представления этой структуры. Такую конструкцию называют *граф-структурированный стек* или *GSS* (Graph Structured Stack), который впервые был предложен Масару Томитой [64] в контексте восходящего анализа. GSS будет являться рабочей структурой нашего нового интерпретатора вместо стека для LL. Состояние разбора вместе с узлом GSS мы будет называть *дескриптором*.

Определение 10.5. Пусть $G = \langle N, \Sigma, P, S \rangle$ — КС-грамматика, X слот грамматики G, i позиция в слове w над алфавитом Σ, а u узел GSS. *Дескриптором* назовём тройку (X, u, i).

Есть несколько способов задания GSS для алгоритма GLL. Вариант, предложенный самими авторами алгоритма, оперирует непосредственно парами из позиции слова и слота грамматики в качестве состояний (и узлов графа) — такой метод является довольно простым и наглядным, но, как описано в работе [2], не самым эффективным. Предложим сразу чуть более оптимальное представление: заметим, что шаги разбора, соответствующие одному и тому же нетерминалу и позиции слова, должны выдавать один и тот же результат независимо от конкретной продукции грамматики, в которой стоит этот нетерминал. Поэтому заводить по узлу на каждый слот грамматики довольно избыточно — вместо этого в качестве состояния будет использовать пары из нетерминала и позиции слова, а позиции грамматики будем записывать на рёбрах.

Итак, мы научились задавать состояния с помощью дескрипторов, а также определились со вспомогательной структурой GSS. Теперь можно перейти к рассмотрению непосредственно самого алгоритма, суть которого довольно проста и напоминает BFS по неявному графу.

Дескриптор задает состояние, которое необходимо обработать. При этом мы без какой-либо дополнительной информации можем продолжить анализ входа из состояния, задаваемого этим дескриптором. В процессе обработки мы можем получить несколько новых состояний. Поэтому будем поддерживать множество R дескрипторов на обработку — на каждом шаге извлекаем один из множества, проводим анализ и кладем в множество новые полученные.

При каких условиях этот процесс будет конечен? Ну, например, если мы каждое состояние будем обрабатывать не более одного раза. И действительно, поскольку наш интерпретатор является "чистым" в том смысле, что для оного и того же состояния каждый раз будут получены одинаковые результаты, проводить анализ дважды не имеет смысла. Поэтому будем также поддерживать множество U всех полученных в ходе разбора дескрипторов, и добавлять в R только те, которых еще нет в U.

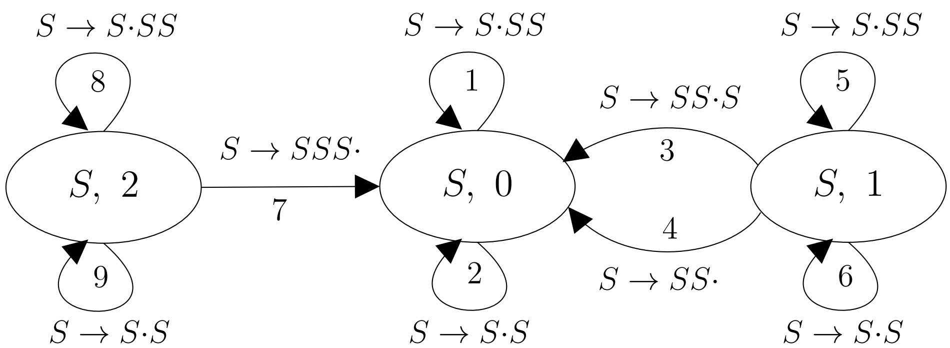
И наконец, заключительная и самая главная часть — как происходит обработка дескриптора? Пусть дескриптор имеет вид (X, u, i), а входное слово обозначим W. Есть три возможных варианта, в зависимости от вида позиции грамматики X — разберем каждый из них по отдельности;

- $X ::= \alpha \cdot t\beta$, т.е. указатель смотрит на терминал — в этом случае новых дескрипторов добавлено не будет. Если $W[i] = t$, то мы сдвигаем указатель слота, переходя к рассмотрению $X ::= \alpha t \cdot \beta$, и инкрементируем позицию i в слове. В противном же случае сразу переходим к следующему дескриптору, т.о. терминируя текущую ветвь разбора.
- $X ::= \alpha \cdot A\beta$, т.е. указатель смотрит на нетерминал. Нам нужен GSS узел v вида (A, i) и ребро $(u, X ::= \alpha A \cdot \beta, v)$ (ребро из u в v с пометкой $X ::= \alpha A \cdot \beta$). Если такой узел и ребро уже существуют в нашем GSS, берем их, иначе — создаём. Далее в R добавляем по дескриптору для узла v и каждого правила грамматики из ячейки управляющей таблицы для нетерминала A (конечно, если их еще не было в U). На этом обработка текущего дескриптора завершается.
- $X ::= \alpha\cdot$, т.е. указатель находится в конце продукции. Продукция разобрана, а значит, интерпретатору необходимо вернуться из разбора X к вызывающему правилу и продолжить разбор там (это, в некотором смысле, соответствует возврату из функции разбора нетерминала в методе рекурсивного спуска). По каждому исходящему ребру (u, Y, v) добавляем (если уже не существует) дескриптор (Y, v, i).

Результатом синтаксического разбора является успех тогда и только тогда, когда был достигнут дескриптор вида $(S ::= \alpha\cdot, s, n)$, где слот грамматики представляет собой любое правило для аксиомы S, узел GSS s состоит из аксиомы S и 0, а позиция входного слова равна его длине n. Если же после разбора всех полученных дескрипторов указанный найден не был, результатом будет являться провал.

Давайте посмотрим, как такой алгоритм справится с неоднозачной грамматикой с леворекурсивным правилом.

Пример 10.5. Пусть грамматика G имеет вид $S \to SSS \mid SS \mid a$, а разбираемое слово $w = aaa$. Тогда GSS, соответствующий разбору $S \Rightarrow SSS \Rightarrow aSS \Rightarrow aaS \Rightarrow aaa$, будет выглядеть следующим образом (для удобства каждое ребро дополнительно пронумеровано):



Далее мы пошагово рассмотрим процесс его построения, а пока отметим несколько особенностей:

- Это *неполный* GSS. Для задачи синтаксического анализа такого достаточно, поскольку если в какой-то момент был достигнут финальный дескриптор, то обрабатывать все последующие уже не нужно. Однако, для задачи построения SPPF, как мы отметим далее, это уже не так, поскольку она требует агрегирования всех возможных путей разбора.

- Обратите особое внимание на наличие петель. Они как раз-таки и обеспечивают эффективную работу с леворекурсивными правилами, поскольку переиспользуются уже существующие узлы. При этом кратных петель, понятно, не создается, т.к. мы запоминаем все достигнутые дескрипторы в множестве U и дублирующих дескрипторов в рабочее множество R не добавляем.
- В GSS не создаются узлы, соответствующие разбору терминалов (например, $a, 0$). В действительности так можно было бы сделать. Но тогда при обработке слота, указывающего на терминал, сначала бы создался узел GSS, затем интерпретатор сверил бы терминал и символ в слове, после чего, если они совпали, произошел бы возврат из узла, а если нет, узел был бы отброшен и интерпретатор перешел бы к другому дескриптору. Таким образом, при любом случае сначала создается узел, затем выполняется проверка, после чего узел сразу отбрасывается. Для того, чтобы не создавать такие "одноразовые" узлы, проверка терминалов выполняется in-place.

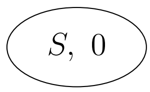
Пронумеруем продукции и выпишем управляющую таблицу:

$$\begin{array}{ll} S \to SSS & (0) \\ S \to SS & (1) \\ S \to a & (2) \end{array}$$

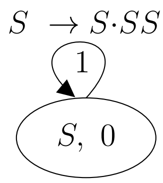
N	FIRST_1	a	\$
S	$\{a\}$	0,1,2	

Разумеется, что конкретный порядок исполнения алгоритма будет зависеть, например, от используемой в качестве рабочего множества R структуры данных и от порядка обработки правил из ячейки управляющей таблицы. Рассмотрим лишь один из возможных вариантов:

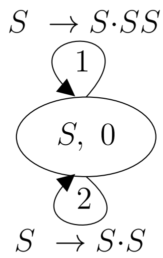
1. Для начала мы создаем узел GSS $s_0 = (S, 0)$ и дескрипторы для правил из ячейки таблицы S, a: $(S \to \cdot SSS, s_0, 0), (S \to \cdot SS, s_0, 0), (S \to \cdot a, s_0, 0)$.



2. При обработке $(S \to \cdot SSS, s_0, 0)$ образовываются петля 1 и дескрипторы $(S \to \cdot SSS, s_0, 0), (S \to \cdot SS, s_0, 0), (S \to \cdot a, s_0, 0)$, которые уже содержатся в множестве U после шага 1 и поэтому не добавляются повторно.

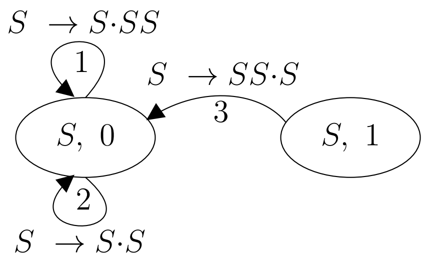


3. При обработке $(S \to \cdot SS, s_0, 0)$ образовывается петля 2, а в остальном аналогично шагу 2.

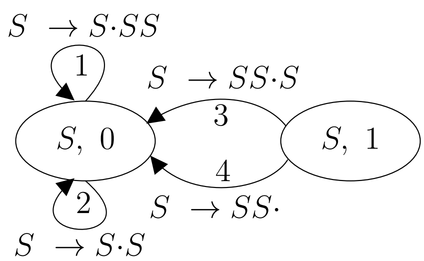


4. При обработке $(S \to \cdot a, s_0, 0)$ мы распознаем терминал a на позиции 0 и, возвращаясь по петлям 1 и 2, добавляем дескрипторы $(S \to S \cdot SS, s_0, 1), (S \to S \cdot S, s_0, 1)$.

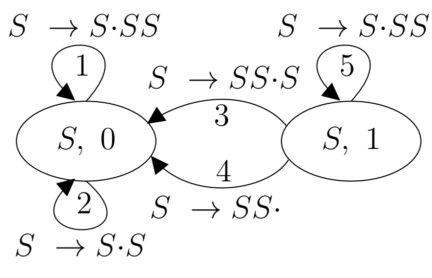
5. При обработке $(S \to S \cdot SS, s_0, 1)$ образовываем узел $s_1 = (S, 1)$ с исходящим ребром 3 и добавляем дескрипторы $(S \to \cdot SSS, s_1, 1), (S \to \cdot SS, s_1, 1), (S \to \cdot a, s_1, 1)$.



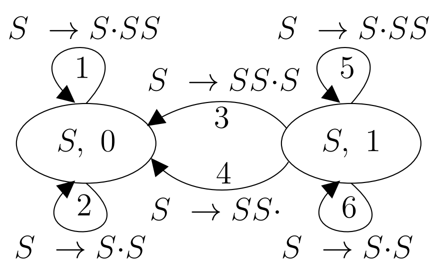
6. При обработке $(S \to S \cdot S, s_0, 1)$ образовываем ребро 4, новых дескрипторов не добавляется.



7. Обработка дескриптора $(S \to \cdot SSS, s_1, 1)$ аналогична шагу 2 с добавлением петли 5.

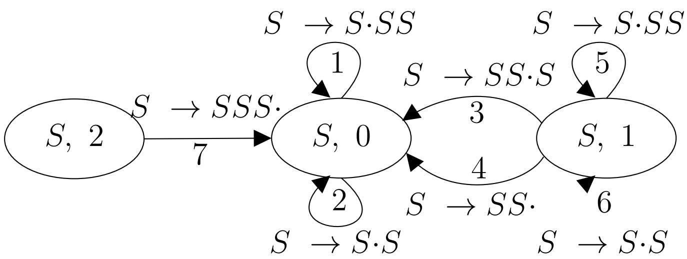


8. Обработка дескриптора $(S \to \cdot SS, s_1, 1)$ аналогична шагу 3 с добавлением петли 6.

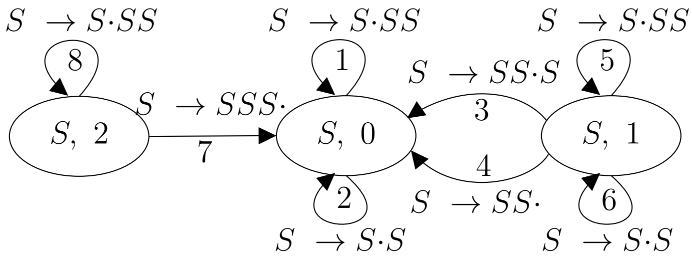


9. При обработке $(S \to \cdot a, s_1, 1)$ мы распознаем терминал a на позиции 1 и, возвращаясь по ребрам 3 и 4, добавляем дескрипторы $(S \to SS \cdot S, s_0, 2), (S \to SS\cdot, s_0, 2)$, а также, возвращаясь по петлям 5 и 6, добавляем дескрипторы $(S \to S{\cdot}SS, s_1, 2), (S \to S{\cdot}S, s_1, 2)$.

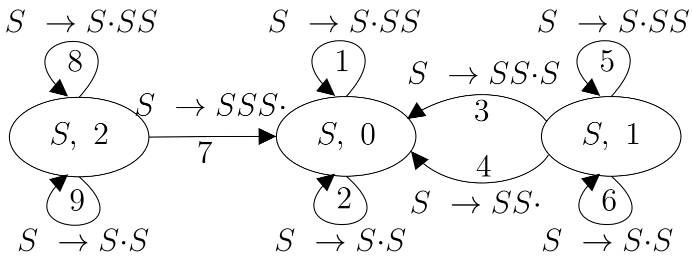
10. При обработке $(S \to SS \cdot S, s_0, 2)$ образовываем узел $s_2 = (S, 2)$ с исходящим ребром 7 и добавляем дескрипторы $(S \to \cdot SSS, s_2, 2), (S \to \cdot SS, s_2, 2), (S \to \cdot a, s_2, 2)$.



11. Обработка дескриптора $(S \to \cdot SSS, s_2, 2)$ аналогична шагу 2 с добавлением петли 8.



12. Обработка дескриптора $(S \to \cdot SS, s_2, 2)$ аналогична шагу 3 с добавлением петли 9.



13. При обработке $(S \to \cdot a, s_2, 2)$ мы распознаем терминал a на позиции 2 и, возвращаясь по ребру 7, добавляем дескриптор $(S \to SSS\cdot, s_0, 3)$, а также, возвращаясь по петлям 8 и 9, добавляем дескрипторы $(S \to S \cdot SS, s_2, 3), (S \to S \cdot S, s_2, 3)$.

14. Мы достигли финального дескриптора $(S \to SSS\cdot, s_0, 3)$, синтаксический разбор успешен.

Внимательный читатель мог заметить, что если бы в этом примере шаг 4 был выполнен перед шагом 2, разбор довольно быстро бы завершился неудачей. Отсюда вытекает следующее наблюдение: если в какой-то момент из существующего узла появилось новое ребро, необходимо пересчитать все входящие в него пути.

Для построения SPPF требуется внести лишь несколько небольших добавлений:

1. В дескриптор необходимо добавить узел SPPF w, который будет представлять уже разобранный префикс.

2. Необходимо поддерживать множество P из элементов вида (u, z), где u это узел GSS, а z соответствующий ему узел SPPF, для того, чтобы переиспользовать результаты разбора, ассоциированные с узлами GSS.

3. При обработке терминала t на позиции i ищется узел вида $(t, i, i+1)$, либо создается, если такого еще нет.

4. При обработке нетерминала с помощью P ищется или при необходимости создается промежуточный узел вида (X, l, r), где X соответствующий слот грамматики, а l и r узлы SPPF, отвечающие за разбор левой и правой частей слота соответственно.

Конкретные шаги построения SPPF будут зависеть от выбранного для него формата. Описание эффективного бинаризованного SPPF и детали его построения при выполнении GLL представлены в работе [2].

GLL довольно естественно обобщается на граф [25]: позициями входа теперь будем считать не индексы линейного слова, а вершины графа. В самом же алгоритме требуется внести лишь два небольших дополнения:

1. Теперь при обработке терминала "следующих" символов может быть несколько — рассматриваем каждый из них отдельно, сдвигаясь по соответствующему ребру.

2. При обработке нетерминала аналогично правила управляющей таблицы применяются для каждого из "следующих" символов в графе. Соответственно новых дескрипторов будет сгенерировано больше, но все они по-прежнему независимы и просто добавляются в рабочее множество R.

Подробное описание алгоритма и псевдокод представлены в работе [25].

Напоследок сделаем небольшое замечание об эффективной реализации: в качестве рабочего множества R можно использовать несколько различных структур данных и, как правило, выбирают очередь. Однако иногда (в особенности для графов) лучше использовать стек дескрипторов, так как в этом случае выше локальность данных — мы кладём пачку дескрипторов, соответствующих исходящим рёбрам. И если граф представлен списком смежности, то исходящие будут храниться рядом и их лучше обработать сразу.

10.3 Вопросы и задачи

1. Проведите алгоритм GLL для грамматики $S \to aSbS \mid \varepsilon$. Правда ли, что эта грамматика принадлежит классу $LL(1)$? Пронаблюдайте, как использование GSS вырождается в работу с обычным стеком.

2. Доразберите все не рассмотренные в примере 10.5 дескрипторы, постройте полный GSS.

11 Комбинаторы для КС апросов

11.1 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?

11.2 Комбинаторы для КС запросов

Вообще говоря, идея использовать комбинаторы для навигации по графам достаточно очевидно и не нова. немного про Trails [34].

Комбинаторы для запросов к графам на основе Meerkat [67]

Обобщённые запросы, типобезопасность и всё такое. Примеры запросов.

11.3 Вопросы и задачи

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

12 Производные для КС запросов

12.1 Производные

Общая теория.

Определения.

12.2 Парсинг на производных

Статьи [39, 1, 40, 3] Реализации. На Scala [5], на Racket [6].

12.3 Адоптация для КС запросов

Для регулярных запросов над графами [42]. Хорошо работают в распределённых системах, в которых реализовван параллелизм уровня вершин. Например Google Pregel.

12.4 Вопросы и задачи

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

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

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

13 От CFPQ к вычислению Datalog-запросов

13.1 Datalog

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

13.2 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 граммтика 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.

13.3 Обобщение GLL для вычисления Datalog-запросов

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

13.4 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. Axelsson and M. Lange. Formal language constrained reachability and model checking propositional dynamic logics. In *International Workshop on Reachability Problems*, pages 45–57. Springer, 2011.

[5] 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.

[6] R. Azimov and S. Grigorev. Path querying using conjunctive grammars. *Proceedings of the Institute for System Programming of the RAS*, 30:149–166, 01 2018.

[7] C. Barrett, K. Bisset, M. Holzer, G. Konjevod, M. Marathe, and D. Wagner. Label constrained shortest path algorithms: An experimental evaluation using transportation networks. *March*, 9:2007, 2007.

[8] C. Barrett, R. Jacob, and M. Marathe. Formal-language-constrained path problems. *SIAM Journal on Computing*, 30(3):809–837, 2000.

[9] 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.

[10] 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.

[11] D. A. Bini, M. Capovani, F. Romani, and G. Lotti. $o(n^{2.7799})$ complexity for approximate matrix multiplication. *Information Processing Letters*, 8(5):234–235, 1979.

[12] P. G. Bradford. Quickest path distances on context-free labeled graphs. In *Appear in 6-th WSEAS Conference on Computational Intelligence, Man-Machine Systems and Cybernetics*. Citeseer, 2007.

[13] P. G. Bradford. Language constrained graph problems: A microcosm of engineering research and development. In *Proceedings of the 2Nd WSEAS International Conference on Computer Engineering and Applications*, CEA'08, pages 71–76, Stevens Point, Wisconsin, USA, 2008. World Scientific and Engineering Academy and Society (WSEAS).

[14] 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.

[15] P. G. Bradford and V. Choppella. Fast point-to-point dyck constrained shortest paths on a dag. In *2016 IEEE 7th Annual Ubiquitous Computing, Electronics & Mobile Communication Conference (UEMCON)*, pages 1–7. IEEE, 2016.

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

[17] T. M. Chan. More algorithms for all-pairs shortest paths in weighted graphs. *SIAM Journal on Computing*, 39(5):2075–2089, 2010.

[18] D. Coppersmith and S. Winograd. On the asymptotic complexity of matrix multiplication. *SIAM Journal on Computing*, 11(3):472–492, 1982.

[19] D. Coppersmith and S. Winograd. Matrix multiplication via arithmetic progressions. *Journal of symbolic computation*, 9(3):251–280, 1990.

[20] W. Dobosiewicz. A more efficient algorithm for the min-plus multiplication. *International journal of computer mathematics*, 32(1-2):49–60, 1990.

[21] G. R. Economopoulos. *Generalised LR parsing algorithms.* PhD thesis, Royal Holloway, University of London, UK, 2006.

[22] N. El abbadi. An efficient storage format for large sparse matrices based on quadtree. *International Journal of Computer Applications*, 105:25–30, 11 2014.

[23] R. W. Floyd. Algorithm 97: shortest path. *Communications of the ACM*, 5(6):345, 1962.

[24] M. L. Fredman. New bounds on the complexity of the shortest path problem. *SIAM Journal on Computing*, 5(1):83–89, 1976.

[25] 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.

[26] Y. Han. Improved algorithm for all pairs shortest paths. *Information Processing Letters*, 91(5):245–250, 2004.

[27] J. Hellings. Conjunctive context-free path queries. In *Proceedings of ICDT'14*, pages 119–130, 2014.

[28] J. Hellings. Path results for context-free grammar queries on graphs. *ArXiv*, abs/1502.02242, 2015.

[29] J. Hellings. Querying for paths in graphs using context-free path queries, 2015.

[30] K. Hemerik. Towards a taxonomy for ecfg and rrpg parsing. *Proceedings of the 3rd International Conference on Language and Automata Theory and Applications*, pages 410–421, 2009.

[31] M. Holzer, M. Kutrib, and U. Leiter. Nodes connected by path languages. In G. Mauri and A. Leporati, editors, *Developments in Language Theory*, pages 276–287, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg.

[32] J. E. Hopcroft and J. D. Ullman. *Introduction to Automata Theory, Languages, and Computation.* Addison-Wesley Publishing Company, 1979.

[33] A. K. Joshi and Y. Schabes. *Tree-Adjoining Grammars*, pages 69–123. Springer Berlin Heidelberg, Berlin, Heidelberg, 1997.

[34] 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.

[35] 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.

[36] O. Kupferman and G. Vardi. Eulerian paths with regular constraints. In *41st International Symposium on Mathematical Foundations of Computer Science (MFCS 2016)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2016.

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

[38] 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.

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

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

[41] 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.

[42] 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.

[43] A. Okhotin. Conjunctive grammars. *Journal of Automata, Languages and Combinatorics*, 6(4):519–535, 2001.

[44] A. Okhotin. Boolean grammars. In *Proceedings of the 7th International Conference on Developments in Language Theory*, DLT'03, pages 398–410, Berlin, Heidelberg, 2003. Springer-Verlag.

[45] A. Okhotin. On the closure properties of linear conjunctive languages. *Theor. Comput. Sci.*, 299(1-3):663–685, 2003.

[46] A. Okhotin. Conjunctive and boolean grammars: The true general case of the context-free grammars. *Computer Science Review*, 9:27–59, 8 2013.

[47] A. Okhotin. Parsing by matrix multiplication generalized to boolean grammars. *Theor. Comput. Sci.*, 516:101–120, Jan. 2014.

[48] A. S. Okhotin. Conjunctive grammars and systems of language equations. *Programming and Computer Software*, 28(5):243–249, Sep 2002.

[49] V. Y. Pan. Strassen's algorithm is not optimal trilinear technique of aggregating, uniting and canceling for constructing fast algorithms for matrix operations. In *19th Annual Symposium on Foundations of Computer Science (sfcs 1978)*, pages 166–176. IEEE, 1978.

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

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

[52] 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.

[53] B. Roy. Transitivité et connexité. *Comptes Rendus Hebdomadaires Des Seances De L Academie Des Sciences*, 249(2):216–218, 1959.

[54] A. Schönhage. Partial and total matrix multiplication. *SIAM Journal on Computing*, 10(3):434–455, 1981.

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

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

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

[58] H. Seki, T. Matsumura, M. Fujii, and T. Kasami. On multiple context-free grammars. *Theoretical Computer Science*, 88(2):191 – 229, 1991.

[59] P. Sevon and L. Eronen. Subgraph queries by context-free grammars. *Journal of Integrative Bioinformatics*, 5, 06 2008.

[60] V. Strassen. Gaussian elimination is not optimal. *Numerische mathematik*, 13(4):354–356, 1969.

[61] T. Takaoka. A new upper bound on the complexity of the all pairs shortest path problem. *Information Processing Letters*, 43(4):195–199, 1992.

[62] T. Takaoka. A faster algorithm for the all-pairs shortest path problem and its application. In *Computing and Combinatorics*, volume 3106, pages 278–289, 2004.

[63] T. Takaoka. An $o(n^3 \log\log n/\log n)$ time algorithm for the all-pairs shortest path problem. *Information Processing Letters*, 96:155–161, 2005.

[64] M. Tomita. Graph-structured stack and natural language parsing. In *26th Annual Meeting of the Association for Computational Linguistics*, pages 249–257, Buffalo, New York, USA, June 1988. Association for Computational Linguistics.

[65] L. G. Valiant. General context-free recognition in less than cubic time. *J. Comput. Syst. Sci.*, 10(2):308–315, Apr. 1975.

[66] 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.

[67] 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.

[68] C. B. Ward and N. M. Wiegand. Complexity results on labeled shortest path problems from wireless routing metrics. *Comput. Netw.*, 54(2):208–217, Feb. 2010.

[69] C. B. Ward, N. M. Wiegand, and P. G. Bradford. A distributed context-free language constrained shortest path algorithm. In *2008 37th International Conference on Parallel Processing*, pages 373–380. IEEE, 2008.

[70] S. Warshall. A theorem on boolean matrices. 1962.

[71] N. Wirth. What can we do about the unnecessary diversity of notation for syntactic definitions? *Communications of the ACM*, 20(11):822–823, 1977.

[72] 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.

[73] X. Zhang, Z. Feng, X. Wang, G. Rao, and W. Wu. Context-free path queries on rdf graphs. In P. Groth, E. Simperl, A. Gray, M. Sabou, M. Krötzsch, F. Lecue, F. Flöck, and Y. Gil, editors, *The Semantic Web – ISWC 2016*, pages 632–648, Cham, 2016. Springer International Publishing.

[74] 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.

[75] U. Zwick. A slightly improved sub-cubic algorithm for the all pairs shortest paths problem with real edge lengths. In *International Symposium on Algorithms and Computation*, pages 921–932. Springer, 2004.