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 Й.Хеллингс в работе [30], также она реализована в работе [77].

Псевдокод алгоритма Хеллингса представлен в листинге 3.

Algorithm 3 Алгоритм Хеллингса

function CONTEXTFREEPATHQUERYING($G = \langle\Sigma, N, P, S\rangle$, $\mathcal{G} = \langle V, E, L\rangle$)
    $r \leftarrow \{(N_i, v, v) \mid v \in V \wedge N_i \to \varepsilon \in P\} \cup \{(N_i, v, u) \mid (v, t, u) \in E \wedge N_i \to t \in P\}$
    $m \leftarrow r$
    while $m \neq \varnothing$ **do**
        $(N_i, v, u) \leftarrow$ m.pick()
        for $(N_j, v', v) \in r$ **do**
            for $N_k \to N_j N_i \in P$ 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 $N_k \to N_i N_j \in P$ 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 5.2.2. 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 $S \to A\ B$. 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
$$m = \{(S_1, 2, 2)\},$$
$$r = \{(A, 0, 1), (A, 1, 2), (A, 2, 0), (B, 2, 3), (B, 3, 2), (S, 1, 3), (S_1, 1, 2), (S, 0, 2), (S_1, 0, 3), (S, 2, 3), (S_1, 2, 2)\}.$$
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

$$m = \{(S, 1, 2)\},$$

$$r = \{(A, 0, 1), (A, 1, 2), (A, 2, 0), (B, 2, 3), (B, 3, 2), (S, 1, 3), (S_1, 1, 2), (S, 0, 2), (S_1, 0, 3), (S, 2, 3), (S_1, 2, 2), (S, 1, 2)\}.$$

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

$$m = \{(S_1, 1, 3)\},$$

$$r = \{(A, 0, 1), (A, 1, 2), (A, 2, 0), (B, 2, 3), (B, 3, 2), (S, 1, 3), (S_1, 1, 2), (S, 0, 2), (S_1, 0, 3), (S, 2, 3), (S_1, 2, 2), (S, 1, 2), (S_1, 1, 3)\}.$$

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

$$m = \{(S, 0, 3)\},$$

$$r = \{(A, 0, 1), (A, 1, 2), (A, 2, 0), (B, 2, 3), (B, 3, 2), (S, 1, 3), (S_1, 1, 2), (S, 0, 2), (S_1, 0, 3), (S, 2, 3), (S_1, 2, 2), (S, 1, 2), (S_1, 1, 3), (S, 0, 3)\}.$$

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

$$m = \{(S_1, 0, 2)\},$$

$$r = \{(A, 0, 1), (A, 1, 2), (A, 2, 0), (B, 2, 3), (B, 3, 2), (S, 1, 3), (S_1, 1, 2), (S, 0, 2), (S_1, 0, 3), (S, 2, 3), (S_1, 2, 2), (S, 1, 2), (S_1, 1, 3), (S, 0, 3), (S_1, 0, 2)\}.$$

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

$$m = \{(S, 2, 2)\},$$

$$r = \{(A, 0, 1), (A, 1, 2), (A, 2, 0), (B, 2, 3), (B, 3, 2), (S, 1, 3), (S_1, 1, 2), (S, 0, 2), (S_1, 0, 3), (S, 2, 3), (S_1, 2, 2), (S, 1, 2), (S_1, 1, 3), (S, 0, 3), (S_1, 0, 2), (S, 2, 2)\}.$$

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

$$m = \{(S_1, 2, 3)\},$$

$$r = \{(A, 0, 1), (A, 1, 2), (A, 2, 0), (B, 2, 3), (B, 3, 2), (S, 1, 3), (S_1, 1, 2), (S, 0, 2), (S_1, 0, 3), (S, 2, 3), (S_1, 2, 2), (S, 1, 2), (S_1, 1, 3), (S, 0, 3), (S_1, 0, 2), (S, 2, 2), (S_1, 2, 3)\}.$$

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.

Глава 6

КС и конъюнктивная достижимость через произведение матриц

В данном разделе мы рассмотрим алгоритм решения задачи контекстно-свободной и конъюнктивной достижимости, основанный на произведении матриц. Будет показано, что при использовании конъюнктивных граммтик, представленный алгоритм находит переапроксимацию истинного решения задачи.

6.1 КС достижимость через произведение матриц

В главе 5.2 был изложен алгоритм для решения задачи КС достижимости на основе CYK. Заметим, что обход матрицы напоминает умножение матриц в ячейках которых множества нетерминалов:

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

, где сумма — это объединение множеств:

$$\sum_{k=1}^{n} = \bigcup_{k=1}^{n}$$

, а поэлементное умножение определено следующим образом:

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

O(n^3/log)

Таким образом, алгоритм решения задачи КС достижимости может быть выражена в терминах перемножения матриц над полукольцом с соответствующими операциями.

Для частного случая этой задачи, синтаксического анализа линейного входа, существует алгоритм Валианта [68], использующий эту идею. Однако он не обобщается на графы из-за того, что существенно использует возможность упорядочить обход матрицы (см. разницу в CYK для линейного случая и для графа). Поэтому, хотя для линейного случая алгоритм Валианта является алгоритмом синтаксического анализа для произвольных КС граммтик за субкубическое время, его обобщение до задачи КС достижимости в произвольных графах с сохранением асимптотики является нетривиальной задачей [75]. В настоящее время алгоритм с субкубической сложностью получен только для частного случая — языка Дика с одним типом скобок — Филипом Брэдфорloм [15].

В случае с линейным входом, отдельного внимания заслуживает работа Лиллиан Ли (Lillian Lee) [42], где она показывает, что задача перемножения матриц сводима к синтаксическому анализу линейного входа. Аналогичных результатов для графов на текущий момент не известно.

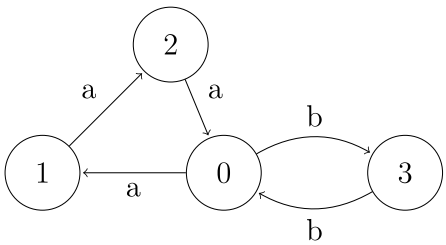
Поэтому рассмотрим более простую идею, изложенную в статье Рустама Азимова [5]: будем строить транзитивное замыкание графа через наивное (не через возведение в квадрат) умножение матриц.

Пусть $\mathcal{G} = (V, E)$ — входной граф и $G = (N, \Sigma, P)$ — входная грамматика. Тогда алгоритм может быть сформулирован как представлено в листинге 4.

Algorithm 4 Context-free recognizer for graphs

1: **function** CONTEXTFREEPATHQUERYING($\mathcal{G}$, G)
2: $\quad n \leftarrow$ количество узлов в $\mathcal{G}$
3: $\quad E \leftarrow$ направленные ребра в $\mathcal{G}$
4: $\quad P \leftarrow$ набор продукций из G
5: $\quad T \leftarrow$ матрица $n \times n$, в которой каждый элемент $\varnothing$
6: $\quad$ **for all** $(i, x, j) \in E$ **do** $\quad\triangleright$ Инициализация матрицы
7: $\qquad T_{i,j} \leftarrow T_{i,j} \cup \{A \mid (A \to x) \in P\}$
8: $\quad$ **for all** $i \in 0 \dots n-1$ **do** $\quad\triangleright$ Добавление петель для нетерминалов, порождающих пустую строку
9: $\qquad T_{i,i} \leftarrow T_{i,i} \cup \{A \in N \mid A \to \varepsilon\}$
10: $\quad$ **while** матрица T меняется **do**
11: $\qquad T \leftarrow T \cup (T \times T)$ $\quad\triangleright$ Вычисление транзитивного замыкания
12: $\quad$ **return** T

Пример 6.1.1 (Пример работы)**.** Пусть есть граф $\mathcal{G}$:



и грамматика G:

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

Пусть T_i — матрица, полученная из T после применения цикла, описанного в строках **8-9** алгоритма 4, i раз. Тогда T_0, полученная напрямую из графа, выглядит следующим образом:

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

Далее показано получение матрицы T_1.

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

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

После первой итерации цикла нетерминал в ячейку $T[2,3]$ добавился нетерминал S. Это означает, что существует такой путь π из вершины 2 в вершину 3 в графе $\mathcal{G}$, что $S \xrightarrow{*} \omega(\pi)$. В данном примере путь состоит из двух ребер $2 \xrightarrow{a} 0$ и $0 \xrightarrow{b} 3$, так что $S \xrightarrow{*} ab$.

Вычисление транзитивного замыкания заканчивается через k итераций, когда достигается неподвижная точка процесса: $T_{k-1} = T_k$. Для данного примера $k = 13$, так как $T_{13} = T_{12}$. Весь процесс рабты алгоритма (все матрицы T_i) 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 $T_{13} = T_{12}$. Заметим, что для данного алгоритма приведённый пример также является худшим случаем: на каждой итерации в матрицу добавляется ровно один нетерминал, при том, что необходимо заполнить порядка $O(n^2)$ ячеек.

6.1.1 Расширение алгоритма для конъюнктивных грамматик

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 6.1.1. Пусть M_1 и M_2 матрицы размера n. Определим операцию $\circ$ сдедующим образом:

$$M_1 \circ M_2 = M_3,$$