Формальные грамматики. HW#2

Тураев Тимур, SPbAU, SE, 604 group

1. *Постоить обыкновенную грамматику в нормальном виде Хомского для языка Дика* $D = \{\varepsilon, ab, aabb, abab, aaabbb, \ldots\}$ *над алфавитом* $\{a, b\}$. *Для этой грамматики и для входной строки* $w = abaabba \notin D$, *построить таблицу разбора* $T_{i,j}$, *как в алгоритме Кокка–Касами–Янгера.*

Обыкновенная грамматика не в нормальной форме:

$$S \to aSb \mid SS \mid \varepsilon$$

Приведем эту грамматику к нормальной форме Хомского:

- Удалим длинное правило $S \to aSb$

$$\begin{aligned} &S \to Tb \mid SS \mid \varepsilon \\ &T \to aS \end{aligned}$$

- Удалим ε-правила:

$$\begin{aligned} &S \to C \mid \varepsilon \\ &C \to Tb \mid CC \\ &T \to a \mid aC \end{aligned}$$

- Удалим цепные правила:

$$\begin{aligned} &S \to Tb \mid CC \mid \varepsilon \\ &C \to Tb \mid CC \\ &T \to a \mid aC \end{aligned}$$

- Последний шаг: заменим терминалы на нетерминалы (бесполезных символов в грамматике нет):

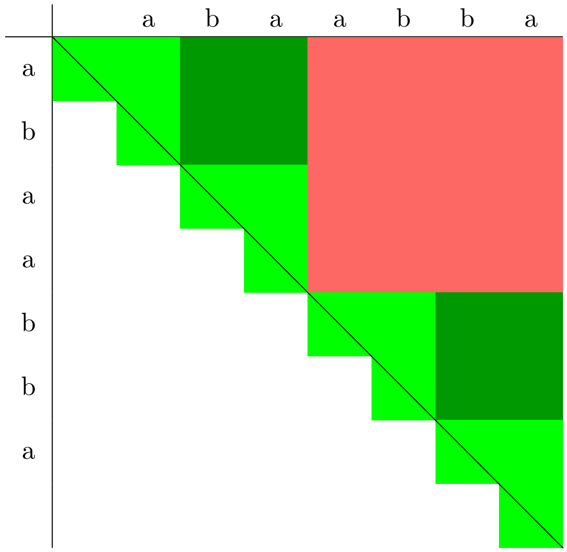
$$\begin{aligned} &S \to TB \mid CC \mid \varepsilon \\ &C \to TB \mid CC \\ &T \to a \mid AC \\ &A \to a \\ &B \to b \end{aligned}$$

Таблица разбора в алгоритме Кокка–Касами–Янгера:

	a	b	a	a	b	b	a
a	{A, T}	{S, C}	∅	∅	∅	{S, C}	∅
b		{B}	∅	∅	∅	∅	∅
a			{A, T}	∅	{T}	{S, C}	∅
a				{A, T}	{S, C}	∅	∅
b					{B}	∅	∅
b						{B}	∅
a							{A, T}

По значению отсутствию нетерминала S в $T_{0,7}$ видно что да, данная строка не принадлежит языку.

2. *Рассмотреть работу алгоритма Валианта для грамматики, построенной в прошлом упражнении. Среди всех действий, производимых алгоритмом, найти то произведение булевых матриц, после вычисления которого станет верным условие $S \in f(P_{0,6})$, где S — начальный символ грамматики. Описать, когда и как именно вычисляется это произведение — то есть, какая процедура, вызванная с какими значениями, и какой оператор в ней умножает какие две булевы матрицы какого размера, каков результат умножения, и какие элементы $P_{i,j}$ будут этим затронуты?*



Рассмотрим работу алгоритма Валианта. Единственной функцией, вызванной из $main()$, будет $compute(0,8)$. Она вызовет $compute(0,4)$ и $compute(4,8)$ на двух больших зеленых треугольниках. Эти две функции, в свою очередь, запустят $compute(0,2)$, $compute(2,4)$, $compute(4,6)$, $compute(6,8)$, который обещают посчитать маленькие светло-зеленые треугольники; и успешно их обсчитают: каждая такая функция $compute(k, k{+}2)$ запустит единственную $complete(k, k{+}1, k{+}1, k{+}2)$, которая, выполнив 19-я строку алгоритма, заполнит все $T_{i,j}$, стоящие на светло-зеленых клетках. Получится следующее:

		a	b	a	a	b	b	a
a	–	{A, T}						
b		–						
a			–	{A, T}				
a				–				
b					–	{B}		
b						–		
a							–	{A, T}
								–

Кроме обсчета маленьких светло-зеленых треугольников, функии $compute(0,4)$ и $compute(4,8)$ обсчитают и темно-зеленые квадраты:

		a	b	a	a	b	b	a
a	–	{A, T}	{S, C}	∅				
b		–	{B}	∅				
a			–	{A, T}				
a				–				
b					–	{B}	∅	∅
b						–	{B}	∅
a							–	{A, T}
								–

Таким образом, функции $compute(0,4)$ и $compute(4,8)$ (внутри вызова $compute(0,8)$) закончат свою работу. Далее вызовется функция **complete(0, 4, 4, 8)**. Она поделит большой красный квадрат на 4 квадрата D, E, C, D' и запустит $complete(C)$, то есть $complete(2,4,4,6)$. Эта функция его обсчитает аналогично тому как обсчитывались темно-зеленые квадраты. 9-я строка алгоритма внутри **complete(0, 4, 4, 8)** выполнилась.

		a	b	a	a	b	b	a
a	–	{A, T}	{S, C}	∅	D		E	
b		–	{B}	∅				
a			–	{A, T}	∅	{T}	D’	
a				–	{A, T}	{S, C}		
b					–	{B}	∅	∅
b						–	{B}	∅
a							–	{A, T}
								–

Далее выполнятся строки 10-11 (обсчет матрицы D) и стркои 12-13 (обсчет матрицы D') внутри **complete(0, 4, 4, 8)**.

		a	b	a	a	b	b	a
a	–	{A, T}	{S, C}	∅	∅	∅	E	
b		–	{B}	∅	∅	∅		
a			–	{A, T}	∅	{T}	{S, C}	∅
a				–	{A, T}	{S, C}	∅	∅
b					–	{B}	∅	∅
b						–	{B}	∅
a							–	{A, T}
								–

Наконец алгоритм перейдет к строке **14**: $P_E = P_E \cup (T_B \times T_{D'})$. Подматрица P_E это матрица $\left(\begin{smallmatrix} P_{0,6} & P_{0,7} \\ P_{1,6} & P_{1,7} \end{smallmatrix}\right)$. Подматрица T_B на рисунке выше – это первая темно-зеленая, то есть $\left(\begin{smallmatrix} P_{0,2} & P_{0,3} \\ P_{1,2} & P_{1,3} \end{smallmatrix}\right) = \left(\begin{smallmatrix} \{S,C\} & \varnothing \\ \{B\} & \varnothing \end{smallmatrix}\right)$, а матрица T_D' это темно-красная матрица $\left(\begin{smallmatrix} P_{2,6} & P_{2,7} \\ P_{3,6} & P_{3,7} \end{smallmatrix}\right) = \left(\begin{smallmatrix} \{S,C\} & \varnothing \\ \varnothing & \varnothing \end{smallmatrix}\right)$. Именно их алгоритм собираться перемножать, используя произведение булевых матриц.

Алгоритм работает так: по каждой паре нетерминалов $\{A, B\}$ (их у нас $|N|^2 = 25$) строются две булевы матрицы: матрица X^A, элемент которой (булев бит) означает есть ли нетерминал A в соответствующем элементе первой матрицы; и матрица Y^B, элемент которой (булев бит) означает есть ли нетерминал B в соответствующем элементе второй матрицы. Получившаяся в результате умножения булева матрица, которую можно назвать Z^{AB}, означает «где в итоговой матрице $(T_B \times T_{D'})$ будет элемент-пара нетерминалов $\{A, B\}$».

В нашем случае, когда дело дойдет до рассмотрения пары $\{\mathbf{C}, \mathbf{C}\}$, матрица X^C будет иметь вид $\left(\begin{smallmatrix} 1 & 0 \\ 0 & 0 \end{smallmatrix}\right)$, а матрица Y^C $\left(\begin{smallmatrix} 1 & 0 \\ 0 & 0 \end{smallmatrix}\right)$. Их произведение: $\left(\begin{smallmatrix} 1 & 0 \\ 0 & 0 \end{smallmatrix}\right)$, что означает, пара $\{C, C\}$ гарантированно будет стоять на первой строке первого столбца матрицы P_E. А это в свою очередь означает, что когда алгоритм дойдет до 16 строки ($complete(E)$), затем вызовется ($complete(D) = complete(0, 1, 6, 7)$), в матрице T на месте $(0, 6)$ появится нетерминал S, потому что существует правило $S \to CC$ (вместе с ним, в этом же множестве нетерминалов, из-за пары $\{C, C\}$ появится еще и C, так как существует правило $C \to CC$).

Итак, собираем все вместе и отвечаем на вопросы в задании:

- какая процедура с какими параметрами – $\mathbf{complete(0, 4, 4, 8)}$ (вызванная из $compute(0, 8)$, которая в свою очередь вызвана из $main()$)
- какой оператор в ней – 14-я строка алгоритма, $(\mathbf{T_B} \times \mathbf{T_{D'}})$
- какие две булевы матрицы какого размера – умножаются 2 матрицы 2×2: $\mathbf{X^C} = \left(\begin{smallmatrix} 1 & 0 \\ 0 & 0 \end{smallmatrix}\right)$ и $\mathbf{Y^C} = \left(\begin{smallmatrix} 1 & 0 \\ 0 & 0 \end{smallmatrix}\right)$
- каков результат умножения этих булевых матриц – $\mathbf{Z^{CC}} = \left(\begin{smallmatrix} 1 & 0 \\ 0 & 0 \end{smallmatrix}\right)$
- общий результат умножения $(T_B \times T_{D'})$ – $\left(\begin{smallmatrix} \{\{S,S\},\{C,C\},\{S,C\},\{C,S\}\} & \varnothing \\ \{\{B,S\},\{B,C\} & \varnothing \end{smallmatrix}\right)$
- какие элементы $P_{i,j}$ будут этим затронуты – при этом умножении затронуты будут элементы подматрицы $P_E = \left(\begin{smallmatrix} P_{0,6} & P_{0,7} \\ P_{1,6} & P_{1,7} \end{smallmatrix}\right)$

3. *Замкнут ли класс LL языков относительно пересечения с регулярными языками? Если замкнут, привести построение, а если незамкнут, привести пример LL грамматики и регулярного языка с доказательством несуществования LL грамматики для их пересечения*

Рассмотрим язык L_1, который задает все строки четной длины, первая половина которых состоит только из букв a, а вторая – из любых сочетаний букв b и c:

$$L_1 = \{a^n(b|c)^n \mid n \geqslant 0\}$$

Это $LL(1)$-язык, для него можно построить такую грамматику:

$$S \to aSB \mid \varepsilon$$
$$B \to b \mid c$$

и следующую таблицу разбора:

	a	b	c	ε
S	S → aSB	S → ε	S → ε	S → ε
B	–	B → b	B → c	B → ε

Наряду с L_1 рассмотрим язык L_2, задающий все слова, в котором сначала идет какое-то (возможно нулевое) число букв a, а затем какое-то (возможно нулевое) число букв b:

$$L_2 = \{a^n b^m + a^n c^m \mid n, m \geqslant 0\}$$

Это тоже $LL(1)$-язык, для него можно построить такую грамматику:

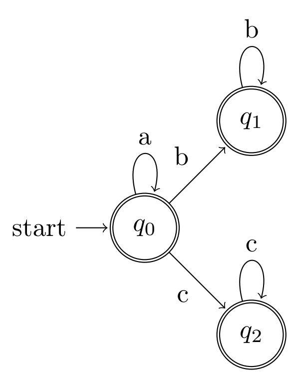
$$S \to AE$$
$$A \to aA \mid \varepsilon$$
$$E \to bB \mid cC \mid \varepsilon$$
$$B \to bB \mid \varepsilon$$
$$C \to cC \mid \varepsilon$$

и следующую таблицу разбора:

	a	b	c	ε
S	S → AE	S → AE	S → AE	S → AE
E	–	E → bB	E → cC	E → ε
B	–	B → bB	–	B → ε
C	–	–	C → cC	C → ε

Кроме того, этот язык, очевидно, является регулярным:

$$L_2 = a^*(b^* \mid c^*)$$



Пересечением этих языков является язык L_3:

$$L_1 \cap L_2 = L_3 = \{a^n b^n + a^n c^n \mid n \geqslant 0\}$$

который, как мы знаем, не является $LL(k)$ ни для какого k (см пример 8.4 конспекта 11 лекции).
Отсюда делаем вывод, что класс LL языков не замкнут относительно пересечения с регулярными языками.