Формальные грамматики 2014: домашнее задание 1

А. Охотин

8 октября 2014 г.

Задача 1. *Построить обыкновенную грамматику для языка всех палиндромов:* $L_1 = \{w \in \{a,b\}^* \mid w = w^R\}$. *Показать, как строка ababa выводится с помощью перезаписи строк. Показать, что эта же строка принадлежит наименьшему решению системы языковых уравнений, построив несколько шагов последовательности* $\varphi^k(\bot)$.

Грамматика:

$$S \to aSa \mid bSb \mid a \mid b \mid \varepsilon$$

Перезапись:

$$S \Longrightarrow aSa \Longrightarrow abSba \Longrightarrow ababa$$

Уравнение:

$$S = (\{a\} \cdot S \cdot \{a\}) \cup (\{b\} \cdot S \cdot \{b\}) \cup \{a, b, \varepsilon\}$$

Последовательность:

$$\begin{aligned}
&\varphi^0(\bot) = \varnothing \\
&\varphi^1(\bot) = \{\varepsilon, a, b\} \\
&\varphi^2(\bot) = \{\varepsilon, a, b, aa, bb, aaa, aba, bab, bbb\} \\
&\varphi^3(\bot) = \{\varepsilon, a, b, aa, bb, aaa, aba, bab, bbb, aaaa, abba, baab, bbbb, \\
&\qquad\qquad aaaaa, aabaa, ababa, abbba, baaab, babab, bbabb, bbbbb\}
\end{aligned}$$

Задача 2. *Доказать, что не существует обыкновенной грамматики для языка* $L_2 = \{a^{k_1}b \ldots a^{k_n}b \mid n \geqslant 1, 0 \leqslant k_1 \leqslant \ldots \leqslant k_n\}$

Первый способ. Если существует обыкновенная грамматика для этого языка, то существует и грамматика для его пересечения с регулярным языком $a^*ba^*ba^*b$. В пересечении получается язык

$$L_2' = \{a^{k_1}ba^{k_2}ba^{k_3}b \mid 0 \leqslant k_1 \leqslant k_2 \leqslant k_3\}.$$

К этому языку проще применять лемму накачки, поскольку ни один из трёх символов b теперь не может попасть в область накачки. Лемма накачки даёт константу $p \geqslant 0$. Расмотрим строку $w = a^pba^pba^pb$. Тогда, согласно лемме, существует разбиение $w = xuyvz$, где $|u|+|v| \geqslant 1$ и $|uyv| \leqslant p$, и для всякого $i \geqslant 0$ строка xu^iyv^iz принадлежит языку. Тогда $u, v \in a^*$, так как в противном случае накачка с $i = 2$ даст строку, содержащую более трёх символов b. Стало быть, каждая из подстрок u, v полностью лежит в одном из блоков.

- Если u или v попали в первый блок, то хотя бы в один из двух оставшихся блоков не попадёт ни u, ни v. Тогда накачка при $i = 2$ даёт строку xu^2yv^2z, в которой в первом блоке будет больше символов a, чем в одном из последующих блоков.

- Если ни u ни v не попали в первый блок, то накачка при $i = 0$ даёт строку xyz, в которой в первом блоке останется p символов a, а в одном из последующих блоков их станет меньше, чем p.

В обоих случаях выходит противорение.

Вариант: использование леммы Огдена с первыми p помеченными позициями позволяет устранить второй случай.

Второй способ. Применить лемму накачки непосредственно к L_2. Лемма даёт константу p, по ней строится всё та же строка $w = a^pba^pba^pb$, лемма даёт её разбиение $w = xuyvz$. Случаев будет больше, но их тоже можно разобрать.

- Пусть u содержит символ b, то есть, $u = a^kba^\ell$, где $\ell + k < p$. Тогда, накачивая два раза, получаем строку xu^2yv^2z, содержащую подстроку $ba^\ell a^kb$, составленную из двух соседних экземпляров u. Эта подстрока образует блок размера $\ell + k < p$, причём это не первый блок в строке, в то время как первый имеет длину ровно p. Потому строка не принадлежит языку.

- Если v содержит b, доказательство совершенно такое же, только первый блок будет иметь длину не менее чем p (не обязательно ровно p).

Отсюда выходит, что $u, v \in a^*$, и можно продолжить доказательство по прошлому способу.

Задача 3. *Построить конъюнктивную грамматику для языка L_2.*

Проверка слева направо. Первый конъюнкт пропускает первый блок и сравнивает все оставшиеся блоки с помощью S, второй конъюнкт сравнивает первые два блока с помощью символа C и пропускает остаток (E).

$$
\begin{aligned}
&S \to AbS\&CbE \mid Ab\\
&A \to aA \mid \varepsilon\\
&C \to aCa \mid bA\\
&E \to AbE \mid \varepsilon
\end{aligned}
$$

Вариант: можно обойтись без символа E и дважды ссылаться на S. Тогда потребуется отдельно рассмотреть случай двух блоков ($n = 2$).

$$
\begin{aligned}
&S \to AbS\&CbS \mid Cb \mid Ab\\
&A \to aA \mid \varepsilon\\
&C \to aCa \mid bA
\end{aligned}
$$

- Распространённая ошибка: дважды ссылаться на S, но не рассматривать случай двух блоков ($n = 2$).

Проверка справа налево. Первый конъюнкт пропускает последний блок и сравнивает остальные с помощью S, второй конъюнкт сравнивает последнюю пару с помощью C и пропускает всё, что было раньше.

$$\begin{aligned}
&S \to SAb\&ECb \mid Ab\\
&A \to aA \mid \varepsilon\\
&C \to aCa \mid bA\\
&E \to AbE \mid \varepsilon
\end{aligned}$$

Как и при проверке слева направо, возможен вариант без E.

Третий способ. Язык можно задать в виде пересечения двух обыкновенных языков, использовав конъюнкцию лишь однажды, без рекурсивных определений через конъюнкцию. Грамматика будет чуть побольше.

Задача 4. *Построить однозначную обыкновенную грамматику для языка* $L_3 = \{c^m a^{\ell_0} b \dots a^{\ell_{m-1}} b a^{\ell_m} b \dots a^{\ell_z} b d^n \mid m, n, \ell_i \geqslant 0, z \geqslant 1, \ell_m = n\}$

В описании языка закралась неточность: следует разрешить числу z быть равным нулю, иначе в грамматике приходится рассматривать неприятный особый случай. Поэтому зададим грамматику для языка

$$L_3' = \{c^m a^{\ell_0} b \dots a^{\ell_{m-1}} b a^{\ell_m} b \dots a^{\ell_z} b d^n \mid m, n, \ell_i \geqslant 0, z \geqslant 0, \ell_m = n\}$$

Вот эта грамматика; она не просто неоднозначна, а даже LL(1).

$$\begin{aligned}
&S \to CD\\
&C \to cCAb \mid \varepsilon\\
&A \to aA \mid \varepsilon\\
&D \to aDd \mid bE\\
&E \to AbE \mid \varepsilon
\end{aligned}$$

- Авторы нескольких работ обратили внимание на неточность в описании языка и рассмотрели особый случай; большинство решивших задачу построили грамматику для языка L_3'.

Задача 5. *Является ли язык L_3 линейным конъюнктивным?*

Нет, не является, поскольку функция $f_{L_3}(k)$ из леммы Терье не ограничена никакой экспоненциальной функцией. Чтобы убедиться в этом, для всякого k и для всякого набора целых чисел $\ell_0, \dots, \ell_{k-1} \geqslant 0$, построим строку

$$w_{\ell_0,\dots,\ell_{k-1}} = c^{k-1} a^{\ell_0} b \dots a^{\ell_{k-1}} b d^{k-1}.$$

Отрезая $i \in \{0, \dots, k-1\}$ символов от начала этой строки и $j \in \{0, \dots, k-1\}$ символов от её конца, получаем строку

$$c^{k-1-i} a^{\ell_0} b \dots a^{\ell_{k-1}} b d^{k-1-j},$$

принадлежащую языку L_3 тогда и только тогда, когда $\ell_{k-1-i} = k-1-j$. Следовательно, соответствующее исходной строке $w_{\ell_0,\dots,\ell_{k-1}}$ множество $S_{L_3,k,w_{\ell_0,\dots,\ell_{k-1}}}$ состоит из всех

таких пар (i, j), что $\ell_{k-1-i} = k - 1 - j$ и $i + j < k$. Для каждого первого компонента $i \in \{0, \ldots, k-1\}$, множество $S_{L_3,k,w_{\ell_0,\ldots,\ell_{k-1}}}$ содержит либо единственную пару $(i, k-1-\ell_{k-1-i})$, если $\ell_{k-1-i} \in \{i, i+1, \ldots, k-1\}$, либо ни одной пары вида (i, j), если $\ell_{k-1-i} < i$ или $\ell_{k-1-i} \geqslant k$.

Это позволяет построить попарно различные множества S для каждого набора значений $\ell_{k-1-i} \in \{i, i+1, \ldots, k-1, k\}$, для всех $i \in \{0, \ldots, k-1\}$. Всего таких множеств будет $\prod_{i=0}^{k-1}(k-i+1) = (k+1)!$, и, соответственно, $f_{L_3}(k) \geqslant (k+1)! = 2^{\Theta(n \log n)}$.

Задача 6. *Пусть $D = \{\varepsilon, ab, aabb, abab, aaabbb, \ldots\}$ — язык Дика над алфавитом $\{a, b\}$. Существует ли грамматика обёртывания пар для языка $L_4 = \{wc^{|w|} \mid w \in D\} = \{\varepsilon, abcc, aabbcccc, ababcccc, aaabbbcccccc, \ldots\}$?*

Запишем обыкновенную грамматику для языка Дика.

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

Грамматика обёртывания пар для языка L_4 будет порождать все пары вида $w : c^{|w|}$, где w — строка из языка Дика. На первых компонентах пар она будет работать так же, как обыкновенная грамматика для языка Дика, а на вторых компонентах будет приписывать по символу c всякий раз, когда в первом компоненте добавляются a или b.

$$S \to (a : c)S(b : c) \mid SS \mid \varepsilon : \varepsilon$$

Задача 7. *Построить грамматику 1-го порядка для языка $\{ww \mid w \in \{a, b\}^*\}$.*

Один из способов. Определить предикат $A(x, y, x', y')$, сравнивающий подстроку от x до y с подстрокой от x' до y'. Затем угадать середину с помощью квантора существования.

$$\begin{aligned}
\sigma &= S(\underline{\text{begin}}, \underline{\text{end}}) \\
S(x, y) &= (\exists z)\, A(x, z, z, y) \\
A(x, y, x', y') &= \big(\big[(a(x+1) \wedge a(y+1)) \vee (b(x'+1) \wedge b(x'+1))\big] \wedge \\
&\qquad \wedge (x = y \wedge \vee A(x+1, y, x'+1, y'))\big) \vee (x = y \wedge x' = y')
\end{aligned}$$

Так как четвёртый аргумент A всегда равен $y' = \underline{\text{end}}$, без него можно обойтись.

Есть и много других способов: например, можно определить трёхместный предикат $A(x, y, z)$, проверяющий условие $x + y = z$, и затем выразить условие принадлежности строки языку целиком в формуле σ.