

Теория формальных языков. Рубежный контроль №2

Вариант №23

Киселев Кирилл

Теоретическая информатика и компьютерные
технологии
МГТУ им. Н.Э. Баумана
декабрь 2023

Содержание

1	Задача 1	2
1.1	Решение	2
2	Задача 2	5
2.1	Решение	5
3	Задача 3	6
3.1	Решение	6

1 Задача 1

Язык SRS $a \rightarrow bab, a^3 \rightarrow a^2, ba \rightarrow ac$ над множеством базисных слов $b^n a^n$

1.1 Решение

Замечания:

1. Любое не пустое слово содержит хотя бы одну a
2. Буквы a могут только уменьшаться
3. $|w|_a \leq |w|_b + |w|_c$
4. Можно бесконечно двигать влево самую первую букву a
5. Слова могут начинаться только с $b^i a^k$, где $k > 0$
6. Применение правила 3 ограничивает сдвиг тех a , которые находились правее буквы a , к которой было применено правило.

Начальное количество букв a и b равно, далее количество исходных b может либо остаться таким же, либо с помощью правила 3 некоторые b , могут перейти в c , но суммарное количество b и c останется равным n . Буквы a могут быть переписаны с помощью правила 2, и их количество станет меньше n . Каждой буквой a с помощью первого и третьего правил могут порождаться некоторые блоки, в которых количество букв b и c сбалансированно, т.к. правило 1 порождает блоки вида $b^k a b^k$, в которых количество букв b сбалансированно. Поэтому при построении PDA стоит ввести два стековых символа, первый будет использовать для подсчета исходных b и букв c , полученных из этих букв b . Вторым символом необходим для проверки баланса букв b и c , которые были порождены с помощью повторения правил 2, 3. Также этот блок может содержать один внутренний блок такой же структуры, из-за наличия правила 3, тогда исходный блок должен заканчиваться на c^r , после чтения которых в стеке должны оставаться лишь символы первого вида.

Докажем, что данный язык не принадлежит классу DCFL. Рассмотрим следующие слова:

- $w_1 = b^{3n} a b^n b^{2n} c$
- $w_2 = b^{3n} a b^n b^n a b^{2n} c^{2n} a^{n-2}$

Параметр n - это длина накачки, считаем, что n больше нуля, иначе сделаем замену $n = n + 1$. Пусть $x = b^{3n} a b^n$, $y = b^{2n} c$, $z = b^n a b^{2n} c^{2n} a^{n-2}$. Рассмотрим два случая:

1. Накачка общего префикса x . Пусть $x = x_0 x_1 x_2 x_3 x_4$. Рассмотрим слово w_2 . Если $x_1 x_2 x_3 \in b^{3n}$, то отрицательная накачка выводит слово из языка, теряется баланс с буквами c . Если фрагмент $x_1 x_2 x_3 \in b^n$ и располагается справа от первой буквы a , то любая ($i \neq 1$) накачка выводит слово из языка, т.к. теряется баланс между буквами b , которые

были поражены с помощью первого правила. Если $x_1 \in b^{3n}$, $x_3 \in b^n$, то отрицательная накачка выводит слово из языка, т.к. нарушаются обе указанных выше причины. Пусть $x_1 = b^{k_1}ab^{k_2}$, $x_2 = b^{k_3}$ и $i = 2$. Получим слово: $b^n b^{2n} a [b^{k_2} b^{k_1} a b^{2n+k_3} a b^{2n} c^{2n}] a^{n-2}$. Подслово выделенное квадратными нарушает баланс между b и c , т.к. $k_1 + k_2 < n < 2n$. Аналогично для случая $x_2 = b^{k_1}ab^{k_2}$, $x_1 = b^{k_3}$

2. Синхронная накачка. Рассмотрим слово w_1 . Пусть $x = x_0 x_1 x_2$, $y = y_0 y_1 y_2$. Т.к по условию леммы $|x_1 x_2| \leq n$, то $x_1 = b^{k_1}$ и $x_2 = b^{k_2}$, $k_1 + k_2 \leq n$, $k_1 > 0$. Если $y_1 = b^{k_3}$, $k_3 \geq 0$, то накачка при $i > 1$ выводит слово из языка, т.к. теряется баланс между буквами b , порожденными первым правилом. Если $y_1 = c$, то накачка при $i > 2$ выводит слово из языка, т.к максимальная длина суффикса состоящего из букв c в данном случае может быть только единицей. Пусть $y_1 = b^{k_3}c$, $k_3 > 0$, тогда при $i = 0$ получим слово $w_3 = b^{3n} a b^{3n-k_1-k_3}$, которое не принадлежит языку, потому что оно могло быть получено только из базового слова ba , но т.к. $k_1 + k_3 \geq 2$, то слово из которого получено w_3 должно быть $b^{k_4}a$, $k_4 \geq 2$, а это слово не является базовым.

Таким образом, данный язык не принадлежит классу DCFL.



Рис. 1: Автомат для языка L

2 Задача 2

Язык $\{w \mid |w|_{ab} = |w|_{baa} \ \& \ w = w^R\}$. Алфавит $\{a, b\}$

2.1 Решение

Пусть $L_1 = \{w \mid |w|_{ab} = |w|_{baa}\}$, $L_2 = \{w \mid w = w^R\}$. Язык L_1 регулярный, а язык L_2 контекстно-свободный. Значит исходный язык L является КС, как пересечение КС и регулярного языков.

Докажем недетерминированность L . Пусть n - длина накачки, положим $k = n + 1$. Тогда возьмем следующие слова:

$$w_1 = a^{2k}b^{2k}a^{2k}, w_2 = a^{2k}b^{2k}a^{3k}b^{2k}a^{3k}b^{2k}a^{2k}$$

Пусть $x = a^{2k}b^{2k}a^{2k-1}$, $y = a$, $z = a^{k+1}b^{2k}a^{3k}b^{2k}a^{2k}$. Необходимо рассмотреть 2 случая:

1. Рассмотрим общий префикс x . Пусть $x = x_0x_1x_2x_3x_4$. В случаях: $x_1 = a^k$ и $x_3 = a^p$, $x_1 = a^k$ и $x_3 = b^p$, $x_1 = b^k$ и $x_3 = b^p$; отрицательная накачка в w_2 выводит слово из языка, т.к. полученное слово уже не будет являться палиндромом. Если $x_1 = a^{k_1}b^{k_2}$, либо $x_2 = a^{k_1}b^{k_2}$, то отрицательная накачка выводит оба слова из языка
2. Пусть $x = x_0x_1x_2$, $y = y_0y_1y_2$, $z = z_0z_1z_2$. Т.к по условию леммы $|x_1x_2| \leq n$, то $x_1 = a^{k_1}$ и $x_2 = a^{k_2}$, $k_1 + k_2 \leq n$, $k_1 > 0$. Также y_1 равно либо пустому слову, либо a , тогда слово $x_0x_1^ix_2y_0y_1^iy_2$ при любом $i \neq 1$ не принадлежит L , т.к. не является палиндромом.

Следовательно, данный язык не является детерминированным КС языком.

3 Задача 3

Язык атрибутивной грамматики для регулярных:

$$\begin{aligned}
[S] &\rightarrow [Regex] && ; \\
[Regex] &\rightarrow [Regex][Regex] && ; \quad Regex_1.val \neq \varepsilon, Regex_2.val \neq \varepsilon \\
&&& \quad Regex_0.val := Regex_1.val + +Regex_2.val \\
[Regex] &\rightarrow ([Regex]|[Regex]) && ; \quad Regex_1.val \neq \varepsilon \vee Regex_2.val \neq \varepsilon, \\
&&& \quad Regex_1.val \neq |, Regex_0.val := | \\
[Regex] &\rightarrow ([Regex])^* && ; \quad Regex_1.val \neq \varepsilon \\
&&& \quad Regex_1.val \neq *, Regex_0.val := * \\
[Regex] &\rightarrow \varepsilon && ; \quad Regex.val := \varepsilon \\
[Regex] &\rightarrow a && ; \quad Regex.val := a \\
[Regex] &\rightarrow b && ; \quad Regex.val := b
\end{aligned}$$

3.1 Решение

Рассмотрим подвыражения, которые запрещены согласно ограничениям налагаемым условиями на атрибут:

1. $(\varepsilon|\varepsilon)$
2. $((\cdot | \cdot) | \cdot)$
3. $(\varepsilon)^*$
4. $((\cdot)^*)^*$

Для исключения подслов вида 1, 2 введем три новых нетерминала:

$$\begin{aligned}
[Regex'] &\rightarrow (\varepsilon|[Regex_{rhs}]) && [Regex_{lhs}] &\rightarrow [Regex_{lhs}][Regex_{lhs}] \\
[Regex'] &\rightarrow ([Regex_{lhs}]|\varepsilon) && [Regex_{lhs}] &\rightarrow ([Regex_{iter}])^* \\
[Regex'] &\rightarrow ([Regex_{lhs}][Regex_{rhs}]) && [Regex_{lhs}] &\rightarrow a \\
[Regex_{rhs}] &\rightarrow [Regex'] && [Regex_{lhs}] &\rightarrow b \\
[Regex_{rhs}] &\rightarrow ([Regex_{iter}])^* \\
[Regex_{rhs}] &\rightarrow [Regex_{rhs}][Regex_{rhs}] \\
[Regex_{rhs}] &\rightarrow a \\
[Regex_{rhs}] &\rightarrow b
\end{aligned}$$

Для того, чтобы исключить выражения вида 3, 4 введем новый нетерминал $Regex_{iter}$

$$\begin{aligned}
[Regex_{iter}] &\rightarrow [Regex'] \\
[Regex_{iter}] &\rightarrow [Regex_{iter}][Regex_{iter}] \\
[Regex_{iter}] &\rightarrow a \\
[Regex_{iter}] &\rightarrow b
\end{aligned}$$

В итоге получим следующую грамматику описывающий язык данной атрибутивной грамматики:

$[S]$	\rightarrow	$[Regexp]$	$[Regexp_{iter}]$	\rightarrow	$[Regexp']$
$[Regexp]$	\rightarrow	$[Regexp][Regexp]$	$[Regexp_{iter}]$	\rightarrow	$[Regexp_{iter}][Regexp_{iter}]$
$[Regexp]$	\rightarrow	$[Regexp']$	$[Regexp_{iter}]$	\rightarrow	a
$[Regexp]$	\rightarrow	$([Regexp_{iter}])^*$	$[Regexp_{iter}]$	\rightarrow	b
$[Regexp']$	\rightarrow	$(\varepsilon [Regexp_{rhs}])$	$[Regexp_{lhs}]$	\rightarrow	$[Regexp_{lhs}][Regexp_{lhs}]$
$[Regexp']$	\rightarrow	$([Regexp_{lhs}] \varepsilon)$	$[Regexp_{lhs}]$	\rightarrow	$([Regexp_{iter}])^*$
$[Regexp']$	\rightarrow	$([Regexp_{lhs}][Regexp_{rhs}])$	$[Regexp_{lhs}]$	\rightarrow	a
$[Regexp_{rhs}]$	\rightarrow	$[Regexp']$	$[Regexp_{lhs}]$	\rightarrow	b
$[Regexp_{rhs}]$	\rightarrow	$([Regexp_{iter}])^*$			
$[Regexp_{rhs}]$	\rightarrow	$[Regexp_{rhs}][Regexp_{rhs}]$			
$[Regexp_{rhs}]$	\rightarrow	a			
$[Regexp_{rhs}]$	\rightarrow	b			
$[Regexp]$	\rightarrow	ε			
$[Regexp]$	\rightarrow	a			
$[Regexp]$	\rightarrow	b			

Язык L описываемый изначальной КС грамматикой для регулярных выражений является детерминированным. Пример DPDA распознающего регулярные выражения изображен на рисунке 2

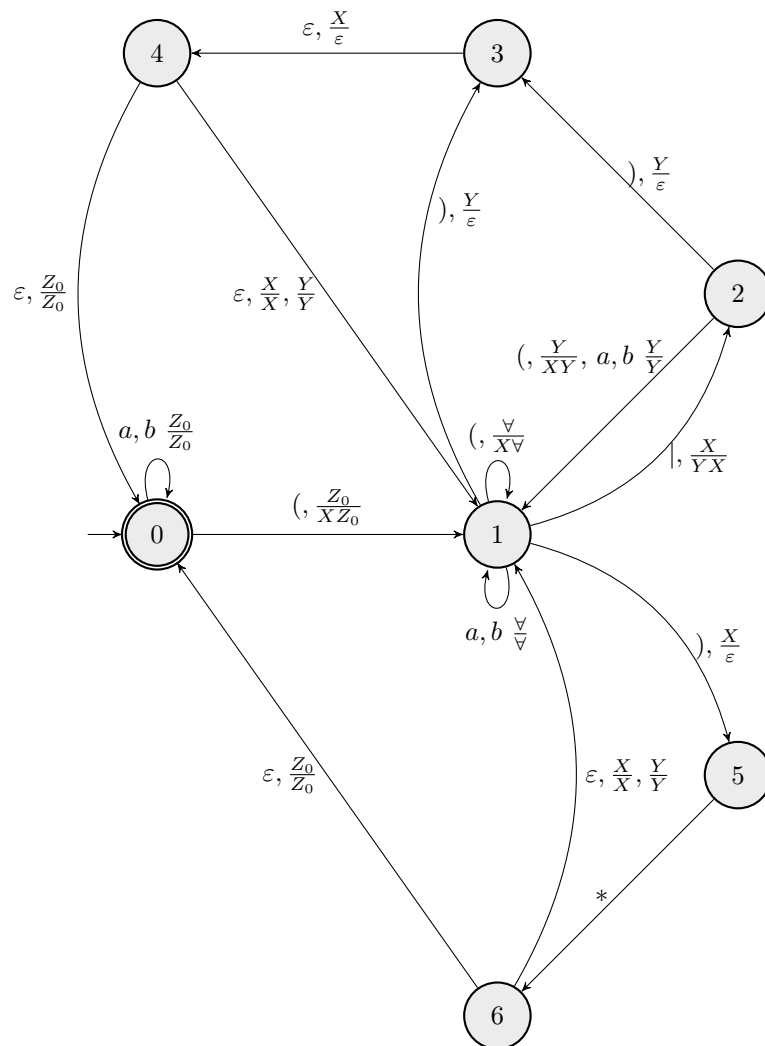


Рис. 2: Автомат для языка регулярных выражений

На основе DPDA для регулярных выражений построим новый DPDA, который не будет принимать слова, содержащие под слова вида 2. Для экономии места, переходы с одинаковыми символами, но разными стековыми символами будут изображаться одним переходом с перечислением возможных конфигураций стека.

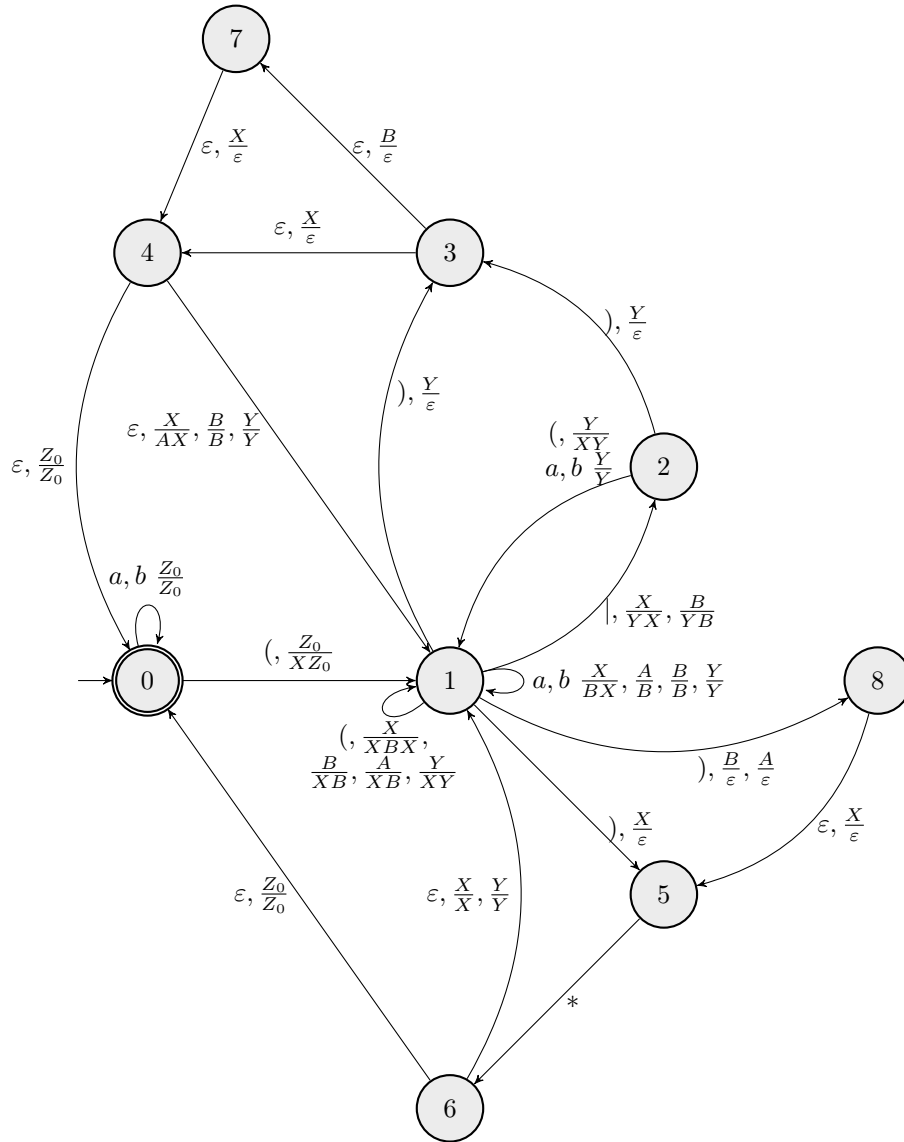


Рис. 3: Автомат для языка регулярных выражений с ограничением 2

Пусть $A = \{a, b, |, *, (,)\}$, L - язык порождаемый автоматом с рисунка 3. Далее рассмотрим регулярный язык P , который описывается регулярным выражением $((a | b | \varepsilon | (|) | | *))^*$. Данный язык содержит в себе L . Пусть далее r обозначает регулярное выражение $((a | b | \varepsilon | (|) | | *))^*$. Пусть язык P' описывается регулярным выражением $r()^*r$, тогда он содержит все слова в алфавите A , которые содержат $()^*$ в качестве подслова. Пусть язык P'' описывается регулярным выражением $r((r)^*)^*r$, тогда он содержит все слова в алфавите A , которые содержат $((r)^*)^*$ в качестве подслова. Пусть язык L^{IV} описывается регулярным выражением $r(|)^*r$, тогда он содержит все слова в алфавите A , которые содержат $(|)^*$ в качестве подслова. Рассмотрим язык $L' = (((P \setminus P') \setminus P'') \setminus P''')$. Полученный язык является регулярным

языком, т.к. множество регулярных языков замкнуто относительно разности. Пересечем языки L и L' , получим новый детерминированный язык эквивалентный языку, который описывает данная атрибутная грамматика.

Полученный язык не является регулярным. Пусть n - это длина накачки, считаем $n > 0$

$$w = \underbrace{(|(|(|(| \dots (| a)) \dots))}_{2n} \underbrace{) \dots)}_{n}$$

Тогда отрицательная накачка в суффиксе w при любом разбиении выводит слово из языка, т.к. теряется скобочный баланс. Если накачиваемый фрагмент состоит только из символа $|$, то при отрицательной накачке, получим выражение в скобках, которое не имеет вид $(w')^*$ или $(w'|w'')$.