Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное
образовательное учреждение высшего образования
«Московский физико-технический институт
(национальный исследовательский университет)»

Кафедра математических основ управления

С. А. Шестаков

Регулярные языки: структурные свойства

Учебно-методическое пособие

МОСКВА
МФТИ
2025

УДК 519.713(075)
ББК 22.18я73
Ш51

Рецензент
Доцент кафедры дискретной математики МФТИ,
кандидат физико-математических наук *Д. В. Мусатов*

Шестаков, Сергей Алексеевич,

Ш51 Регулярные языки: структурные свойства : учеб.-метод. пособие / С. А. Шестаков ; М-во науки и высшего образования Рос. Федерации, Моск. физ.-техн. ин-т (нац. исслед. ун-т). – Москва : МФТИ, Физтех, 2025. – 44 с.

Содержит сведения о структурных свойствах регулярных языков. Большая часть представленного материала входит в состав курса теории алгоритмов и моделей вычислений для второкурсников физтех-школы прикладной математики и информатики (ФПМИ) МФТИ. Рассмотрены условия и критерии регулярности языков, теорема Майхилла–Нероуда. Представлены свойства морфизмов регулярных языков, в том числе распознавание регулярных языков конечными моноидами. Теоремы, леммы и свойства снабжены примерами.

Предназначено для студентов, изучающих курсы по формальным языкам и конечным автоматам и обучающихся по направлению подготовки «Прикладные математика и физика».

Учебное издание

Шестаков Сергей Алексеевич

**Регулярные языки:
структурные свойства**

Редактор *Н. Е. Кобзева*
Компьютерная верстка *Н. Е. Кобзевой*
Подписано в печать 00.08.2025. Формат 60×84 $^1/_{16}$.
Усл. печ. л. 2,75. Уч.-изд. л. 2,4. Тираж 20 экз. Заказ № 000.
Федеральное государственное автономное образовательное учреждение
высшего образования «Московский физико-технический институт
(национальный исследовательский университет)»
141700, Московская обл., г. Долгопрудный, Институтский пер., 9
Тел. (495) 408-58-22, e-mail: rio@mipt.ru

Отдел оперативной полиграфии «Физтех-полиграф»
141700, Московская обл., г. Долгопрудный, Институтский пер., 9
E-mail: polygraph@mipt.ru

Печатается по решению Редакционно-издательского совета Московского физико-технического института (национального исследовательского университета)

Содержание

Введение

В настоящем пособии рассматриваются структурные свойства регулярных языков, условия и критерии регулярности и нерегулярности языка. Обсуждаются некоторые дополнительные модели вычислений, приводящие к альтернативному описанию регулярного языка. Предполагается знакомство с базовыми свойствами регулярных языков. В частности, требуется знание о различных эквивалентных способах представления регулярных языков (регулярные выражения и деревья и конечные автоматы).

1. Замкнутость класса регулярных языков

Конструкция произведения ДКА позволяет доказать, что регулярные языки замкнуты относительно произвольных теоретико-множественных операций. Рассмотрим несколько более сложных операций, которые сохраняют регулярность языка.

1.1. Деление

Пусть L_1 и L_2 – два языка (над алфавитом Σ). (**Правым**) **частным** от деления L_1 на L_2 называется язык

$$L_1/L_2 = \{x \in \Sigma^* \mid \exists y \in L_2 \; xy \in L_1\}.$$

То есть из L_1 выбираются те слова, суффиксы которых лежат в L_2, а затем эти суффиксы отбрасываются (всеми способами, если таковых несколько). Операция получения (правого) частного, естественно, называется (правым) **делением**. Деление языков интересует нас, поскольку регулярные языки замкнуты относительно деления в следующем смысле.

Теорема. *Пусть L_1 и L_2 – языки над Σ. Пусть L_1 – регулярный язык. Тогда L_1/L_2 – регулярный язык.*

Доказательство. Так как язык L_1 регулярный, то существует ДКА $A = (\Sigma, Q, s, F, \delta)$, распознающий его. Модифицируем множество принимающих состояний, построим ДКА $A' = (\Sigma, Q, s, F', \delta)$, где $F' \subseteq Q$ определяется так:

$$F' = \{q \in Q \mid \exists y \in L_2 \;\; \delta^*(q, y) \in F\}.$$

Для слова x над Σ теперь выполняется следующая цепочка эквивалентностей:

x принимается A',
$\iff \delta^*(s,x) \in F'$,
$\iff \delta^*(s,x) = q$ и существует $y \in L_2$ такой, что $\delta^*(q,y) \in F$,
$\iff$ существует $y \in L_2$ такой, что $\delta^*(s,xy) \in F$,
$\iff$ существует $y \in L_2$ такой, что $xy \in L_1$,
$\iff x \in L_1/L_2$.

Значит, A' принимает язык L_1/L_2, и тогда последний регулярен.

Заметим, что на язык L_2 не накладывалось никаких ограничений, в частности от него не требуется регулярности.

Примеры

Для любого языка L верно, что $L/\varnothing = \varnothing/L = \varnothing$. Если L непуст, то $\Sigma^*/L = \Sigma^*$, поскольку для любого вообще слова x можно взять некоторое слово $y \in L$ (которое существует по предположению) и тогда $xy \in \Sigma^*$.

Пусть теперь L – регулярный язык над алфавитом Σ. Из теоремы выше следует, что L/Σ^* регулярен. Языку L/Σ^* принадлежат такие слова x, что существует какое-то слово y (без ограничений на это слово, поскольку $y \in \Sigma^*$), что xy принадлежит L. Иначе говоря, для каждого слова $w \in L$ и каждого способа разбиения $w = xy$ слово x лежит в языке L/Σ^*. Следовательно, $L/\Sigma^* = \mathsf{pref}(L)$ – язык всех префиксов всех слов из L. Получаем, что язык всех префиксов регулярного языка регулярен.

1.2. Морфизмы

Пусть дано два алфавита Δ и Σ. **Морфизмом** (или гомоморфизмом) назовём всюду определённое отображение

$$h : \Delta^* \to \Sigma^*,$$

для которого на любых $x, y \in \Delta^*$ выполнено $h(xy) = h(x)h(y)$. Из сохранения конкатенации сразу следует, что $h(\varepsilon) = \varepsilon$. Более того, если задать h на буквах алфавита Δ, то на произвольные слова h продолжается (по конкатенации) единственным образом, так что можно считать, что h переводит буквы Δ в слова над Σ.

Морфизм естественным образом распространяется на целый язык:

$$h(L) = \bigcup_{w \in L} \{h(w)\}.$$

Теперь "поднимемся на ступеньку выше", будем вместо букв подставлять не строки, а целые языки. **Подстановка языка** – это всюду определённое отображение

$$s : \Delta^* \to 2^{\Sigma^*},$$

для которого на любых $x, y \in \Delta^*$ выполнены условия $s(xy) = s(x)s(y)$ и $s(\varepsilon) = \{\varepsilon\}$. Последнее свойство нужно для корректности и однозначности операций (подошло бы ещё $s(x) = \varnothing$ на каждом x, но это не очень полезно). В результате подстановки языка каждая буква из Δ переводится в некоторый язык над алфавитом Σ.

Приведём пример: пусть $\Delta = \{a, b\}$, $\Sigma = \{0, 1\}$, пусть $s(a) = \{0, 00\}$, $s(b) = \mathsf{L}((01)^*10)$ (напомним, это язык, определяемый регулярным выражением «$(01)^*10$»). Тогда

$$s(aba) = \mathsf{L}((0 + 00)(01)^*10(0 + 00)).$$

В этом примере вместо букв подставляются регулярные языки – это основной интересующий нас случай, но регулярность подстановки не требуется в определении подстановки языка.

Морфизм h можно считать частным случаем подстановки языка s, если строку w отождествить с языком $\{w\}$.

Подстановка языка также естественным образом распространяется на целый язык:

$$s(L) = \bigcup_{w \in L} s(w).$$

Теорема. *Класс $\mathcal{REG}$ замкнут относительно подстановок регулярных языков, а именно: пусть L – регулярный язык над алфавитом Δ, пусть $s : \Delta^* \to 2^{\Sigma^*}$ – подстановка языка такая, что для любого символа $a \in \Delta$ язык $s(a)$ регулярен. Тогда $s(L)$ – регулярный язык.*

Доказательство.

Шаг 1.

Докажем сначала, что для любой подстановки языка s и любых языков L_1, L_2 выполняются три условия:

1) $s(L_1 \cup L_2) = s(L_1) \cup s(L_2)$,

2) $s(L_1 L_2) = s(L_1)s(L_2)$,

3) $s(L^*) = s(L)^*$.

Рассмотрим 1):

$$s(L_1 \cup L_2) = \bigcup_{x \in L_1 \cup L_2} s(x) = \bigcup_{i \in \{1,2\}} \bigcup_{x \in L_i} s(x) = \bigcup_{i \in \{1,2\}} s(L_i) = s(L_1) \cup s(L_2).$$

Заметим также, что такое же равенство выполнено для любого объединения, не только объединения двух языков: вместо $\bigcup_{i\in\{1,2\}}$ можно написать $\bigcup_{i\in I}$ для любого индексного множества I.

Рассмотрим 2):

$$s(L_1L_2) = \bigcup_{x\in L_1, y\in L_2} s(xy) = \bigcup_{x\in L_1, y\in L_2} s(x)s(y) =$$
$$= \left(\bigcup_{x\in L_1} s(x)\right)\left(\bigcup_{y\in L_2} s(y)\right) = s(L_1)s(L_2).$$

Для 3) докажем по индукции, что $s(L^n) = s(L)^n$. База при $n = 0$ выполнена, так как обе части равны $\{\varepsilon\}$. В предположении, что утверждение выполняется для всех $j < n$, получим

$$s\left(L^n\right) = s\left(LL^{n-1}\right) = s(L)s\left(L^{n-1}\right) = s(L)s(L)^{n-1} = s(L)^n.$$

Здесь второе равенство выполнено по пункту 2), а третье – по предположению индукции.

Тогда

$$s\left(L^*\right) = s\left(\bigcup_{i=0}^{\infty} L^i\right) = \bigcup_{i=0}^{\infty}\left(L^i\right) = \bigcup_{i=0}^{\infty} s(L)^i = s(L)^*.$$

Второе равенство выполнено по замечанию к пункту 1).

Шаг 2.

Пусть L регулярен, рассмотрим его дерево разбора.

Если $L = \varnothing$, то $s(L) = \varnothing$ – регулярный язык.

Если $L = \{a\}$ для некоторой буквы $a \in \Delta$, то $s(L)$ регулярен по условию теоремы.

Иначе (индуктивно по числу вершин в дереве регулярного выражения) L – это $L_1 + L_2$, L_1L_2 или L_1^* для некоторых регулярных L_1, L_2. Тогда, соответственно, части 1), 2) или 3) из предыдущего шага доказывают регулярность: в случае 1) $s(L_1 \cup L_2) = s(L_1) \cup s(L_2)$, оба $s(L_1)$ и $s(L_2)$ регулярны по индукции, их объединение регулярно. Случаи 2) и 3) аналогичны.

В частности, теорема выше доказывает, что регулярные языки замкнуты относительно морфизмов (потому что морфизм – частный случай подстановки языка).

1.3. Обратный морфизм

Пусть Δ и Σ – два алфавита, пусть $h : \Delta^* \to \Sigma^*$ – морфизм, пусть $L \subseteq \Sigma^*$. Определим **обратный морфизм** языка:

$$h^{-1}(L) = \{x \in \Delta^* \mid h(x) \in L\}.$$

Обратный морфизм определён только для языков – он не переводит буквы в строки, и в том числе не является морфизмом. Обратный морфизм языка является языком (над Δ).

Теорема. *Пусть Δ и Σ – два алфавита, пусть $h : \Delta^* \to \Sigma^*$ – морфизм, пусть $L \subseteq \Sigma^*$ – регулярный язык. Тогда $h^{-1}(L)$ – регулярный язык.*

Доказательство. L – регулярный язык, пусть $A = (\Sigma, Q, s, F, \delta)$ – ДКА, распознающий L. Рассмотрим ДКА $A' = (\Delta, Q, s, F, \delta')$, где δ' определена следующим образом:

$$\delta'(q, a) = \delta^*(q, h(a)).$$

Докажем по индукции (по длине слова), что $\delta'^*(q, x) = \delta^*(q, h(x))$.

База: $x = \varepsilon$, тогда $\delta'^*(q, x) = \delta^*(q, h(x)) = q$ по определению. Предположим, что результат верен для всех строк короче x. Пусть $x = ay$, где $a \in \Delta$, тогда

$$\delta^*(q, h(ay)) = \delta^*(q, h(a)h(y)) \overset{1}{=} \delta^*(\delta^*(q, h(a)), h(y)) \overset{2}{=}$$
$$\overset{2}{=} \delta^*\left(\delta'(q, a), h(y)\right) \overset{3}{=} \delta'^*\left(\delta'(q, a), y\right) \overset{4}{=} \delta'^*(q, ay).$$

Равенство (1) выполняется по свойству функции δ^*, равенство (2) – по построению δ', равенство (3) – по предположению индукции, а равенство (4) – по свойству функции δ'^*.

Следовательно, $\delta'^*(q, x) \in F$ выполнено тогда и только тогда, когда $\delta^*(q, h(x)) \in F$, т. е. x принимается A' тогда и только тогда, когда $h(x)$ принимается A (что эквивалентно $h(x) \in L$). Тогда A' распознаёт язык $h^{-1}(L)$ и последний регулярен.

Пример

Пусть L над $\{0, 1\}^*$ регулярен. Докажем, что и язык $L_{\not 2}$, полученный из L удалением у каждого слова из L второго символа (если таковой имелся), также регулярен.

Пусть $g : \{0, 1, 0', 1'\} \to \{0, 1\}$, $g(0) = g(0') = 0$, $g(1) = g(1') = 1$. Рассмотрим язык $L' = g^{-1}(L)$. Слова в L' “такие же”, как и в L, но

только на буквы могут быть в произвольных местах навешены штрихи. К примеру, если $01 \in L$, то каждое из слов $01, 0'1, 01', 0'1'$ лежит в L'.

Теперь рассмотрим язык $L_{2'} = L' \cap L_{\text{mask}}$, где

$$L_{\text{mask}} = \mathsf{L}\left((0+1)\,(0'+1')\,(0+1)^* + (\varepsilon + 0 + 1)\right).$$

Пересечение отфильтровывает из L' только слова со штрихами и только над второй буквой (а также слова, длина которых меньше двух), потому что только такие слова лежат в L_{mask}. Заметим, что последний регулярен, так как задан регулярным выражением. Язык $L_{2'}$ состоит из таких же слов, как и L, только над вторым символом здесь стоит штрих.

Окончательно, пусть

$$h : \{0, 1, 0', 1'\} \to \{0, 1\},$$

$$h(0) = 0, h(1) = 1, h(0') = h(1') = \varepsilon.$$

Тогда язык $h(L_{2'})$ – это в точности $L_{\not 2}$.

Получаем, что

$$L_{\not 2} = h\left(g^{-1}(L) \cap \mathsf{L}\left((0+1)\,(0'+1')\,(0+1)^* + (\varepsilon + 0 + 1)\right)\right)$$

и регулярен, так как все операции сохраняют регулярность.

2. Необходимые и достаточные условия регулярности

2.1. Лемма о накачке

Пусть L регулярен, тогда его распознаёт некоторый ДКА A, пусть в нём $|Q|$ состояний. Предположим теперь, что в L есть "длинное слово", а именно такое слово w, что $|w| \geqslant |Q|$. Как тогда A ведёт себя на слове w? Как и всегда A читает букву за буквой и (если представить, что работа ДКА – это обход соответствующего графа) в течение работы посещает $|w| + 1 > |Q|$ вершину (одна начальная и $|w|$ после прочтения каждой буквы). Поскольку всего вершин в графе $|Q|$, по крайней мере одна вершина была посещена несколько раз, а значит при чтении слова возник цикл. Последовательность состояний, в которых был A в процессе работы над словом w выглядит так: $sq_i \dots q_j \dots q_F$, где q_F принимающее. Где-то "в середине" есть цикл, а значит последовательность состояний выглядит так:

1) ДКА прошёл от начального состояния до входа в цикл сколько-то состояний (может быть, нисколько, потому что начальное состояние может быть частью цикла);

2) ДКА прошёл по циклу (может быть, не единожды);

3) ДКА вышел из цикла и прошёл сколько-то состояний до принимающего состояния (может быть, нисколько, потому что последнее принимающее состояние может быть частью цикла).

Тогда слово w делится на три части $w = w_1w_2w_3$ (первая и третья могут быть пустыми) соответственно работе ДКА A. Рассмотрим теперь слово $w_1w_2w_2w_3$, на этом слове:

1) ДКА прошёл от начального состояния до входа в цикл сколько-то состояний по слову w_1, *как и ранее*;

2) ДКА прошёл по циклу по слову w_2, *как и ранее*;

2+) *ДКА ещё раз прошёл по циклу по слову* w_2, *остановившись ровно там же*;

3) ДКА вышел из цикла и прошёл сколько-то состояний до принимающего состояния по слову w_3, *как и ранее*.

Тогда ДКА A принимает слово $w_1w_2w_2w_3$. Аналогичными рассуждениями получаем, что A должен принять любое слово вида $w_1w_2^iw_3$. Более того, можно положить $i = 0$, выкинув внутренний цикл вовсе. Тем самым мы получаем следующее: если в регулярном языке есть "длинное слово", то в нём есть бесконечно много слов специфической структуры $w_1w_2^iw_3$. Этот результат называется *леммой о накачке* и является необходимым условием регулярности языка.

Теорема (Лемма о накачке)**.** *Пусть L – регулярный язык. Тогда существует натуральное число n такое, что для любого слова $w \in L$ с длиной $|w| \geqslant n$ существует разбиение на подслова $w = xyz$, удовлетворяющее условиям $|xy| \leqslant n$ и $|y| \geqslant 1$, такое, что для любого натурального m слово $xy^mz \in L$.*

Доказательство. Пусть L – регулярный язык, пусть ДКА A принимает L, пусть A имеет $|Q|$ состояний, положим $n = |Q|$. Пусть теперь слово $w \in L$ имеет длину $|w| \geqslant n$. Рассмотрим все префиксы w, префикс длиной k обозначим w_k, также обозначим $q_k = \delta^*(s, w_k)$. У слова w существует $|w|+1 > |Q|$ префиксов, так что из состояний q_k, $k = \overline{0 \dots |Q|}$, хотя бы два состояния совпадают. Пусть совпадают состояния q_i и q_j, $i < j$. Тогда $w_j = w_iy$ для некоторого y, отличного от ε. Более того,

$$q_i = q_j = \delta^*(s, w_j) = \delta^*(\delta^*(s, w_i), y) = \delta^*(q_i, y).$$

По индукции получим, что $q_i = \delta^*(q_i, y^m)$ для любого натурального m (в том числе для $m = 0$ по определению δ^*). Тогда, положив $x = w_i$,

получаем $w = w_j z = xyz$ для некоторого z, причём $\delta^*(q_i, z) \in F$ (потому что w принимается ДКА). Окончательно, тогда

$$\delta^*(s, xy^m z) = \delta^*(\delta^*(\delta^*(s, w_i), y^m), z) = \delta^*(\delta^*(q_i, y^m), z) = \delta^*(q_i, z) \in F.$$

Поскольку y – непустое слово, то $|y| \geqslant 1$, поскольку мы рассматриваем префиксы длины вплоть до $|Q|$, то $|xy| = |w_j| \leqslant n$.

Предыдущие рассуждения показывают, что в качестве n в условии леммы можно взять число состояний в (любом) ДКА, принимающем L. Условие $|y| \geqslant 1$ означает, что цикл непуст, условие $|xy| \leqslant n$, – что цикл начинается где-то на первых n символах слова. Строки x и z могут быть пустыми.

Как и любое необходимое условие лемма о накачке применяется прежде всего для доказательства отрицания посылки. А именно, если лемма о накачке не выполняется для L, то язык L нерегулярный.

Для записи отрицания посылки сперва запишем лемму в математической нотации:

$$(L \in \mathcal{REG}) \Rightarrow (\exists n \in \mathbb{N}\ \forall w \in L\,(|w| \geqslant n \Rightarrow \exists x, y, z\,(w = xyz \wedge |xy| \leqslant n \wedge |y| \geqslant 1 \wedge \forall i \in \mathbb{N}\ xy^i z \in L))).$$

Затем напишем контрапозицию:

$$(\forall n \in \mathbb{N}\ \exists w \in L\,(|w| \geqslant n \wedge \forall x, y, z\,((w = xyz \wedge |xy| \leqslant n \wedge |y| \geqslant 1) \Rightarrow \exists i \in \mathbb{N}\ xy^i z \notin L))) \Rightarrow (L \notin \mathcal{REG}).$$

Отметим, что для доказательства нерегулярности языка с помощью леммы о накачке недостаточно привести пример одного слова, необходимо привести последовательность слов: для каждой потенциальной длины n своё слово.

Пример нерегулярного языка

$L = \left\{0^j 1^j \,\middle|\, j \in \mathbb{N}\right\}$ нерегулярен. Докажем это с помощью леммы о накачке.

Пусть $n \in \mathbb{N}$ *произвольное*, *возъмём* в качестве w слово $0^n 1^n \in L$: мы не вольны выбирать длину, но при фиксированном n мы вольны выбрать слово. Длина w есть $2n \geqslant n$, теперь *рассмотрим произвольное* разбиение. Опять же, мы не вольны выбрать удобное для нас разбиение, напротив, мы должны провести рассуждение для каждого разбиения, подходящего под условие леммы. Пусть $w = xyz$, $|xy| \leqslant n$, $|y| \geqslant 1$. Поскольку $|xy| \leqslant n$, то $xy = 0^k$ для некоторого k. Поскольку $|y| \geqslant 1$, то

$y = 0^j$ для некоторого $j \geqslant 1$. Тогда $w = 0^{k-j}0^j z$. Тогда возьмём $i = 0$ (или любое натуральное, отличное от единицы), $xy^0z = xz = 0^{k-j}z$, в этом слове n единиц и $n - j < n$ нулей, так что это слово не принадлежит L. Из леммы о накачке заключаем, что L нерегулярный.

Недостаточное условие

Лемма о накачке не является достаточным условием, так что из выполнения её для языка нельзя сделать вывод о регулярности языка. Приведём пример: пусть $\Sigma = \{0, 1, \#\}$, пусть

$$L = \left\{\#0^j1^j \,\middle|\, j \in \mathbb{N}\right\} \cup \mathsf{L}(\varepsilon + \#\#\#^*) \cdot \{0,1\}^*.$$

Язык L удовлетворяет условию леммы о накачке: пусть дано слово $w \in L$ длиной $\geqslant 2$.

Если $w \in \{0,1\}^*$, то можно взять произвольное разбиение (например, $x = \varepsilon$, y равно первой букве слова, z – всё остальное), поскольку тогда $xy^mz \in \{0,1\}^* \subset L$.

Если $w \in \#\# \cdot \{0,1\}^*$, то возьмём $x = \varepsilon$, $y = \#\#$, z – всё остальное. Тогда $xy^0z \in \{0,1\}^* \subset L$, для $m > 0$ выполнено $xy^mz \in \#\#\#^*\{0,1\}^* \subset L$, поскольку число символов $\#$ не меньше двух.

Если $w \in \#\#\#\#^* \cdot \{0,1\}^*$, то возьмём $x = \#$, $y = \#$, z – всё остальное. Тогда для $m \in \mathbb{N}$ выполнено $xy^mz \in \#\#\#^*\{0,1\}^* \subset L$, поскольку опять-таки число символов $\#$ не меньше двух.

Если же $w \in \left\{\#0^j1^j \,\middle|\, j \in \mathbb{N}\right\}$, то возьмём $x = \varepsilon$, $y = \#$, z – всё остальное. Тогда для слова xy^mz при $m = 0$ оно принадлежит $\{0,1\}^* \subset L$, при $m = 1$ оно равно w, при $m > 1$ оно принадлежит $\#\#\#^* \cdot \{0,1\}^* \subset L$.

Получаем, что условия леммы о накачке выполнены. При этом L не является регулярным: если бы L был регулярным, то регулярным было бы и пересечение $L \cap \mathsf{L}(\#0^*1^*) = \{\#0^n1^n \mid n \in \mathbb{N}\}$. Однако этот язык нерегулярен, поскольку стирающий $\#$ морфизм переводит его в нерегулярный $\left\{0^j1^j \,\middle|\, j \in \mathbb{N}\right\}$.

2.2. Разделяющие суффиксы

Рассмотрим некоторый ДКА $A = (\Sigma, Q, s, F, \delta)$, распознающий регулярный язык L. Пусть два слова x и y таковы, что $\delta^*(s, x) = \delta^*(s, y)$. Тогда, для любого слова z выполнено

$$\delta^*(s, xz) = \delta^*(\delta^*(s, x), z) = \delta^*(\delta^*(s, y), z) = \delta^*(s, yz).$$

В том числе это означает, что $xz \in L$ тогда и только тогда, когда $yz \in L$. А значит, неформально, если ДКА A не может различить два слова x и y, то язык L не может различить никакую пару xz и yz.

Посмотрим теперь на это с другой точки зрения. Пусть L – язык, пусть x и y – два слова (не обязательно из этого языка). Пусть существует такое слово z, что **ровно одно** из слов xz, yz принадлежит L. Если так оказывается, что L распознаётся некоторым ДКА, то в этом ДКА слова xz и yz должны вести к различным состояниям (принимающему и непринимающему), а, значит, слова x и y должны вести к различным состояниям.

Слово z называется **разделяющим суффиксом** для пары x и y (относительно языка L), если ровно одно из слов xz, yz принадлежит L. **Разделяющим множеством** слов для языка L называется множество слов S (они могут лежать или не лежать в L, на это не накладывается ограничений) такое, что любые два различные слова из S имеют относительно L разделяющий суффикс.

Теорема. *Если для языка L существует бесконечное разделяющее множество, то L не является регулярным.*

Доказательство. Действительно, пусть бы L был регулярным, тогда он должен был бы распознаваться некоторым ДКА A с $|Q|$ состояниями. Но для L существует бесконечное разделяющее множество S. Множество S бесконечно, так что в нём существует подмножество S_0, состоящее из $|Q|+1$ слов, которое также является разделяющим множеством для L. Для любой пары слов из S_0 существует разделяющий суффикс, так что для любых различных $x, y \in S_0$ выполняется $\delta^*(s,x) \neq \delta^*(s,y)$. Значит, в A должно быть по крайней мере $|Q|+1$ состояние – хотя бы по одному состоянию $\delta^*(s,x)$ для каждого выбора $x \in S_0$. Получили противоречие с определением A и, соответственно, с регулярностью L.

Проведённые рассуждения также доказывают, что если для языка L существует разделяющее множество с N словами, то любой ДКА, распознающий L (если такой вообще есть), должен иметь не менее N состояний.

Пример нерегулярного языка

$L = \left\{0^j 1^j \,\middle|\, j \in \mathbb{N}\right\}$ нерегулярен. Докажем это с помощью разделяющих суффиксов.

Рассмотрим $S = \left\{0^j \,\middle|\, j \in \mathbb{N}\right\}$, очевидно, S бесконечно. Докажем, что S – разделяющее множество для L. Возьмём два различных слова из S: $x = 0^i$ и $y = 0^j$, $i \neq j$. Для них существует разделяющий суффикс $z = 1^i$, так как $xz = 0^i 1^i \in L$, а $yz = 0^j 1^i \notin L$. Вследствие произвольности выбора x и y каждая пара различных слов из S имеет разделяющий

суффикс. Следовательно, S – это бесконечное разделяющее множество для L, язык L нерегулярный.

Экспоненциальный разрыв между ДКА и НКА

Стандартный алгоритм перехода к подмножествам порождает по НКА с $|Q|$ состояниями эквивалентный ДКА с $2^{|Q|}$ состояниями. Такой ДКА может не быть минимальным (по числу состояний) для данного языка, но может и быть. Экспоненциальный разрыв между числом состояний НКА и эквивалентного ему ДКА в худшем случае неизбежен. Приведём пример.

Пусть L_n – язык над $\{0,1\}$ всех слов такой, что n-й символ с конца есть 1. Язык L_n распознаётся НКА B_n (без ε-переходов) с $n+1$ состоянием: $q_0, q_1, \ldots, q_n$, начальное состояние q_0, единственное принимающее q_n. Переходы определены так: $\delta(q_i, 0) = \delta(q_i, 1) = \{q_{i+1}\}$ для всех $1 \leqslant i < n$, из q_n переходов нет. Из q_0 определены переходы в q_0 по обоим символам и дополнительно переход в q_1 по символу 1.

Любое слово из L_n принимается НКА B_n: первые символы проматываются в состоянии q_0, затем происходит переход в q_1 по символу 1, затем последовательный переход по суффиксу до принимающего q_n. Никакие другие слова не принимаются, так как единственный маршрут из начального состояния в принимающее имеет суффикс $1(0+1)^{n-1}$.

Теперь рассмотрим два различных слова длиной n. Поскольку они различны, то они отличаются в некотором индексе. Пусть на позиции j у слова u стоит нуль, а у слова v стоит единица. Тогда u и v имеют разделяющий суффикс 0^{j-1}, в самом деле, $u0^{j-1}$ имеет на n-й позиции с конца нуль, так что $u0^{j-1} \notin L_n$, а $v0^{j-1}$ имеет на n-й позиции с конца единицу, так что $v0^{j-1} \in L_n$. Тогда всё множество слов $S = \{0,1\}^n$ является разделяющим множеством для языка L_n. S содержит 2^n слов, так что любой ДКА, распознающий L_n, содержит по крайней мере 2^n состояний.

3. Теорема Майхилла–Нероуда

3.1. Отношения эквивалентности

Отношение эквивалентности на множестве – это бинарное рефлексивное, транзитивное и симметричное отношение. Если мы возьмём множество S и отношение эквивалентности $\equiv$ на нём, то для каждого элемента $x \in S$ множество распадается на две кучки: все эле-

менты, эквивалентные x (там точно лежит сам x вследствие рефлексивности, может лежать ещё что-то, вплоть до всего S), и все прочие элементы. Все элементы, эквивалентные x, эквивалентны между собой и образуют **класс эквивалентности**. Элемент x в таком случае называется **представителем** класса, пишут $[x]_{\equiv} = \{y \in S \mid x \equiv y\}$. Если $x \equiv y$, то $[x]_{\equiv} = [y]_{\equiv}$, так что в некотором смысле конкретный представитель класса "неважен".

Фактормножество (при заданном отношении эквивалентности) – это множество всех классов эквивалентности (каждый класс эквивалентности – это один элемент фактормножества). Фактически, отношение эквивалентности разбивает всё множество S на кучки, в каждой кучке все элементы эквивалентны между собой и неэквивалентны никакому другому элементу из любой другой кучки. Кучек может быть конечное или бесконечное количество, каждая кучка может быть конечной или бесконечной, а фактормножество – это множество всех кучек, где каждая кучка – это единое целое.

Пусть на множестве S задано два отношения эквивалентности $\equiv_1$ и $\equiv_2$. Если для любых $x, y \in S$ из выполнения $x \equiv_1 y$ следует выполнение $x \equiv_2 y$, то говорят, что $\equiv_2$ грубее, чем $\equiv_1$, а $\equiv_1$ является уточнением или измельчением $\equiv_2$.

ДКА порождает отношение эквивалентности

Пусть дан ДКА A над Σ. Он порождает отношение эквивалентности $\equiv_A$ на множестве Σ^*:

$$x \equiv_A y \iff \delta^*(s, x) = \delta^*(s, y).$$

Это отношение эквивалентности **право-инвариантно**: если $x \equiv_A y$, то для любого z верно, что $xz \equiv_A yz$. Ещё раз отметим, $\equiv_A$ порождается конкретным автоматом A, а не языком, который этот автомат распознаёт. Фактормножество по $\equiv_A$ состоит не более, чем из $|Q|$ элементов (менее, если есть недостижимые состояния). Язык, распознаваемый A, есть объединение тех классов эквивалентности, слова в которых приходят в принимающее состояние A (т. е. $\delta^*(s, w) \in F$).

Язык порождает отношение эквивалентности

Пусть L – некоторый язык над Σ (это может быть произвольный язык, не обязательно регулярный). Язык L порождает собой отношение эквивалентности на множестве Σ^*, которое называется **отноше-**

нием эквивалентности Майхилла–Нероуда, – это отношение эквивалентности $\equiv_L$, где

$$x \equiv_L y \iff (\forall z \in \Sigma^* \; xz \in L \Leftrightarrow yz \in L).$$

Как и в случае $\equiv_A$, отношение $\equiv_L$ разбивает всё множество Σ^* на классы эквивалентности.

Два неэквивалентных по $\equiv_L$ слова имеют разделяющий суффикс (напоминаю, эти слова также могут принадлежать или не принадлежать L, на это не накладывается ограничений). Заметим, что язык L (не обязательно регулярный) представляет собой объединение некоторых классов эквивалентности Майхилла–Нероуда. В самом деле, если $x \in L$ и $x \equiv_L y$, то, подставляя $z = \varepsilon$, получаем, что $y \in L$. Также, отношение эквивалентности Майхилла–Нероуда право-инвариантно: если $x \equiv_L y$, то для любого z верно, что $xz \equiv_L yz$.

Лемма. *Отношение эквивалентности Майхилла–Нероуда (порождённое языком L) – это самое грубое из всех право-инвариантных отношений (определённых на Σ^*) таких, что L есть объединение некоторых классов эквивалентности.*

В самом деле, пусть $\equiv_?$ – такое отношение эквивалентности, пусть $x \equiv_? y$. Тогда из право-инвариантности следует, что для любого z верно, что $xz \equiv_? yz$. Поскольку L есть объединение классов эквивалентности по $\equiv_?$, то $\forall z \in \Sigma^* \; xz \in L \Leftrightarrow yz \in L$. Последнее по определению означает, что $x \equiv_L y$. То есть $x \equiv_? y$ влечёт $x \equiv_L y$.

В частности, если $x \equiv_A y$ для автомата A, распознающего регулярный язык L, то и $x \equiv_L y$.

3.2. Теорема Майхилла–Нероуда

Теперь мы готовы сформулировать основной результат, который является необходимым и достаточным условием регулярности языка.

Теорема (Майхилла–Нероуда)**.** *Пусть L – язык над Σ. Следующие утверждения об L равносильны.*

1) L регулярный язык (принимается некоторым ДКА).

2) L представляет собой объединение классов эквивалентности некоторого право-инвариантного отношения эквивалентности на Σ^* с конечным факторомножеством.

3) Отношение эквивалентности Майхилла–Нероуда $\equiv_L$, порождённое языком L, имеет конечное факторомножество.

4) Размер максимального разделяющего множества для L конечен.

Более того, в случае регулярности языка L размер максимального разделяющего множества равен размеру фактормножества по отношению эквивалентности Майхилла–Нероуда и одновременно равен числу состояний в минимальном ДКА, принимающем L.

Доказательство.

1) $\to$ 2)

Если L регулярный, он принимается некоторым ДКА A. Отношение эквивалентности $\equiv_A$ право-инвариантно, имеет конечное факторможество (число классов эквивалентности не превосходит числа состояний в A), и L есть объединение классов эквивалентности $\equiv_A$ (тех, что соответствуют принимающим состояниям A).

2) $\to$ 3)

Отношение эквивалентности Майхилла–Нероуда – самое грубое из отношений с требуемыми свойствами, так что для данного отношения $\equiv_?$ выполнено $x \equiv_? y \Rightarrow x \equiv_L y$. Каждый класс эквивалентности $[x]_?$ по отношению $\equiv_?$ является тогда подмножеством класса эквивалентности $[x]_L$ по отношению $\equiv_L$. Но тогда размер факторможества $\equiv_L$ не превосходит размера факторможества $\equiv_?$ и, в частности, является конечным.

3) $\leftrightarrow$ 4)

Из определений отношения эквивалентности Майхилла–Нероуда и разделяющего множества следует, что S разделяющее множество для L тогда и только тогда, когда оно содержит не более одного представителя из каждого класса эквивалентности по $\equiv_L$. Если факторможество конечно, то и любое S конечно. Если факторможество бесконечно, то можно выбрать по одному представителю и получить бесконечное S.

3) $\to$ 1)

Построим ДКА, распознающий L. Пусть $A = (\Sigma, Q, s, F, \delta)$, где

$Q = \{[x]_L \mid x \in \Sigma^*\}$,

$s = [\varepsilon]_L$,

$F = \{[x]_L \mid x \in L\}$,

$\delta([x]_L, a) = [xa]_L$. (Это определение функции перехода корректно, т. е. для разных представителей x и y класса $[x]_L$ строки xa и ya лежат в одном классе $[xa]_L$ вследствие право-инвариантности.)

Докажем, что $\delta^*([\varepsilon]_L, y) = [y]_L$ для всех y с помощью индукции по длине y. База $y = \varepsilon$: $\delta^*([\varepsilon]_L, \varepsilon) = [\varepsilon]_L = [y]_L$. Пусть утверждение верно для всех строк длины меньше $|y|$, пусть $y = xa$ (так удобнее, чем

предполагать $y = ax$). Тогда

$$\delta^*([\varepsilon]_L, y) = \delta^*([\varepsilon]_L, xa) = \delta^*(\delta^*([\varepsilon]_L, x), a) =$$
$$= \delta^*([x]_L, a) = \delta([x]_L, a) = [xa]_L = [y]_L.$$

Это означает, что слово y принимается A тогда и только тогда, когда $[y]_L \in F$, что то же самое, что $y \in L$.

Завершающее замечание о равенстве соответствующих чисел: мы доказали, что число классов эквивалентности Майхилла–Нероуда равно размеру максимального разделяющего множества. Также ранее мы показали, что размер любого разделяющего множества для L не больше, чем число состояний в любом ДКА, принимающем L. Кроме того, мы построили ДКА, число состояний в котором равно числу классов эквивалентности Майхилла–Нероуда. Поэтому все три числа равны друг другу.

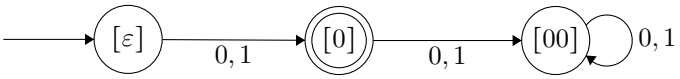
Примеры

Разберём несколько примеров. Работать будем над $\Sigma = \{0, 1\}$.

1) Пусть $L = \{0, 1\}$. Фактормножество отношения эквивалентности Майхилла–Нероуда имеет три элемента: $\{\{\varepsilon\}, \{0, 1\}, \{0, 1\}^* - \{\varepsilon, 0, 1\}\}$. В самом деле, 0 и 1 эквивалентны, так как при приписывании ε они оба принадлежат языку, а при приписывании чего угодно другого – не принадлежат. Все слова, кроме ε, 0 и 1 эквивалентны между собой, так как при приписывании чего угодно они все не принадлежат языку. Слово ε не эквивалентно 0 и 1, так как после приписывания 0 оно принадлежит языку, а 0 и 1 нет. По той же причине ε не эквивалентно ни одному слову в множестве $\{0, 1\}^* - \{\varepsilon, 0, 1\}$. 0 (и 1) не эквивалентен 00 (и любому представителю класса), так как после приписывания ε слово 0 принадлежит языку, а 00 нет.

Соответственно, максимальное разделяющее множество содержит по одному элементу из каждого класса эквивалентности и имеет, например, такой вид: $\{\varepsilon, 0, 00\}$.

Минимальный ДКА, распознающий L, выглядит так:



2) Рассмотрим теперь язык $L = \left\{0^j 1^j \,\middle|\, j \in \mathbb{N}\right\}$.

Заметим, что если слово не имеет вид $0^n 1^m$, где $m \leqslant n$, то оно не принадлежит языку и “это нельзя исправить” приписыванием к нему

суффикса, потому что все префиксы всех слов L имеют вышеобозначенный вид. Поэтому, все слова, не имеющие вид 0^n1^m, где $m \leqslant n$, эквивалентны между собой, все они не принадлежат языку и не принадлежат языку после дописывания любого суффикса. Будем обозначать этот класс эквивалентности $[1]$.

Пусть слово имеет структуру 0^n1^m, где $m \leqslant n$. Если в этом слове есть хотя бы одна единица (т. е. $m > 0$), то его можно дописать до слова из L единственным способом, поэтому есть лишь один суффикс, помещающий слово в L. Все слова такого вида с таким суффиксом эквивалентны. Это слова вида $0^{k+m}1^m$, где $m > 0$. Все такие слова при всех возможных m эквивалентны – приписывание 1^k помещает слово в язык, приписывание чего угодно другого нет. Получаем бесконечное количество классов вида $[0^k01]$, где $k \in \mathbb{N}$. Здесь $[0^k01] = \left\{0^{k+m}1^m \,\middle|\, m > 0\right\}$.

Остались без рассмотрения слова вида 0^n1^m, не содержащие единиц, т. е. слова вида 0^n. Все такие слова (мы уже делали это ранее) не эквивалентны друг другу – в самом деле, слова 0^i и 0^j различаются суффиксом 1^i. Кроме того, они не эквивалентны слову 1 (суффикс 1^i) и не эквивалентны слову 0^k01 (суффикс 011^i). Получаем ещё одно бесконечное семейство классов эквивалентности. В каждом классе находится только один элемент: $[0^k] = \{0^k\}$.

Всего получаем бесконечное множество классов: $[0^k]$ при $k \in \mathbb{N}$, $[0^k01]$ при $k \in \mathbb{N}$ и $[1]$. Заметим, что $L = [01] + [\varepsilon]$.

3.3. Минимизация ДКА

Пусть L – регулярный язык, пусть $A = (\Sigma, Q, s, F, \delta)$ – ДКА, построенный нами в доказательстве теоремы Майхилла–Нероуда. A распознаёт L и имеет N состояний, причём N – это минимально возможное число состояний для любого ДКА, принимающего L.

ДКА A единственный (с точностью до переименования состояний[1]) ДКА с N состояниями, принимающий L. В самом деле, пусть другой автомат $A' = (\Sigma, Q', s', F', \delta')$ распознаёт L и имеет N состояний. Отношение $\equiv_{A'}$ измельчает отношение $\equiv_L$, так что для фиксированного $q \in Q'$ язык всех слов w таких, что $\delta^*(s, w) = q$ есть непустое (из минимальности A') подмножество некоторого класса эквивалентности Майхилла–Нероуда. Но поскольку $|Q'| = N$, этот язык равен

[1] Формально, тут с точностью до изоморфизма. ДКА $A = (\Sigma, Q_A, s_A, F_A, \delta_A)$ и $B = (\Sigma, Q_B, s_B, F_B, \delta_B)$ изоморфны, если существует биекция $f : Q_A \to Q_B$ такая, что $f(s_A) = s_B$, $f(F_A) = F_B$, $f(\delta_A(q, a)) = \delta_B(f(q), a)$.

классу эквивалентности. Тогда состоянию $q' \in Q'$ автомата A' можно сопоставить состояние $[x]$ автомата A, где x таково, что $\delta'^*(s', x) = q'$.

Более того, теорема Майхилла–Нероуда позволяет нам получить алгоритм **минимизации** ДКА: получения минимального ДКА, эквивалентного данному.

Пусть регулярный язык L распознаётся ДКА $C = (\Sigma, Q, s, F, \delta)$ (не обязательно минимальным). Будем предполагать, что все состояния C достижимы. Такое предположение позволяет нам рассуждать следующим образом: "пусть q – состояние, тогда существует слово x такое, что $\delta^*(s, x) = q$". Это предположение не ограничивает общности, так как можно сперва удалить из C все недостижимые состояния. Также будем предполагать, что в C существует хотя бы одно (достижимое) принимающее состояние и хотя бы одно (достижимое) непринимающее состояние. В противном случае C тривиально распознаёт пустой язык или язык всех слов.

Рассмотрим два отношения эквивалентности: отношение $\equiv_C$ порождается автоматом, отношение (Майхилла–Нероуда) $\equiv_L$ порождается языком. При этом ранее мы получили, что $\equiv_L$ грубее $\equiv_C$. Если ДКА не минимален (состояний в нём больше, чем классов эквивалентности Майхилла–Нероуда), то существует два различных состояния p и q (а для них строки x: $\delta^*(s, x) = p$, y: $\delta^*(s, y) = q$) таких, что $x \equiv_L y$. Последнее означает, что $\delta^*(p, z) \in F$ тогда и только тогда, когда $\delta^*(q, z) \in F$ на любом слове z. Такие состояния p и q мы назовём **неразличимыми**. Формально мы называем два состояния p и q **различимыми** и пишем $p \not\sim q$, тогда и только тогда, когда существует слово z такое, что ровно одно из двух состояний $\delta^*(p, z)$ и $\delta^*(q, z)$ – принимающее. В противном случае, состояния p и q неразличимы, мы пишем $p \sim q$. Неразличимость является отношением эквивалентности (на состояниях ДКА).

Ещё раз: для данного ДКА C мы называем неразличимыми такие состояния p и q, что на каждом слове z происходит следующее. Если C начинает читать z из состояния p (а не из s как обычно), и если C начинает читать z из состояния q, то либо C завершает работу в принимающем состоянии в **обоих случаях**, либо C завершает работу в непринимающем состоянии в **обоих случаях**. Для минимизации ДКА мы теперь сделаем следующее: найдём все пары неразличимых состояний и "склеим" каждое максимальное множество попарно неразличимых состояний в одно состояние.

Для получения и обоснования алгоритмической реализации введём ещё несколько объектов. Назовём два состояния p и q **неразличимыми словом длиной** k и будем писать $p \overset{k}{\sim} q$, если для любого

слова z длиной $|z| \leqslant k$ состояния $\delta^*(p,z)$ и $\delta^*(q,z)$ – оба принимающие или оба непринимающие.

Отношение $\overset{k}{\sim}$ является отношением эквивалентности. При этом очевидно, что $p \overset{0}{\sim} q$ тогда и только тогда, когда они оба принимающие или оба непринимающие. Очевидно также, что если $p \overset{k+1}{\sim} q$, то $p \overset{k}{\sim} q$.

Кроме того, если отношения $\overset{k}{\sim}$ и $\overset{k+1}{\sim}$ совпадают, то они совпадают также с $\overset{k+j}{\sim}$ для любого j. В самом деле, если $p \overset{k+2}{\sim} q$, то $p \overset{k}{\sim} q$ и $p \overset{k+1}{\sim} q$. Пусть $p \overset{k+2}{\not\sim} q$, тогда существует строка z длиной $k+2$ (без ограничения общности не меньше), которая их различает: опять же без ограничения общности $\delta^*(p,z) \in F$, $\delta^*(q,z) \notin F$. Поскольку z имеет длину по крайней мере 2, то $z = aw$, где a буква. Подставляя разложение z, получаем $\delta^*(\delta(p,a),w) \in F$, $\delta^*(\delta(q,a),w) \notin F$, значит $\delta(p,a) \overset{k+1}{\not\sim} \delta(q,a)$. Поскольку отношения $\overset{k}{\sim}$ и $\overset{k+1}{\sim}$ совпадают, то $\delta(p,a) \overset{k}{\not\sim} \delta(q,a)$, а тогда существует строка y, где $|y| \leqslant k$, такая, что из состояний $\delta^*(\delta(p,a),y) = \delta^*(p,ay)$ и $\delta^*(\delta(q,a),y) = \delta^*(q,ay)$ ровно одно принимающее. Тогда $p \overset{k+1}{\not\sim} q$, тогда и $p \overset{k}{\not\sim} q$, поскольку $\overset{k}{\sim}$ и $\overset{k+1}{\sim}$ совпадают. Значит, все три отношения $\overset{k}{\sim}$, $\overset{k+1}{\sim}$, $\overset{k+2}{\sim}$ совпадают.

Отношение $\overset{0}{\sim}$ имеет два класса эквивалентности: F и $Q - F$ (это непустые множества по предположению). Каждое следующее $\overset{k}{\sim}$ либо добавляет новые неразличимые между собой состояния, увеличивая тем самым число классов эквивалентности, либо равно предыдущему, и тогда оно равно всем последующим, и равно отношению $\sim$. Общее число классов эквивалентности не может превысить числа состояний $|Q|$ в ДКА, так что в любом случае отношение $\sim$ равно отношению $\overset{|Q|-2}{\sim}$.

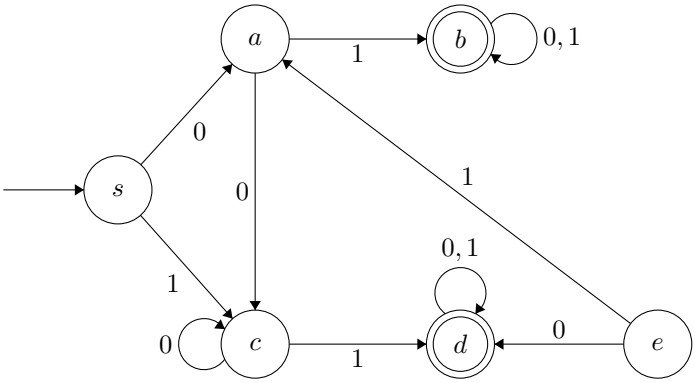
Теперь вычислим отношение $\sim$ с помощью рекуррентной функции для $\overset{k}{\sim}$, а именно определим булеву функцию: “два состояния различимы словом ограниченной длины” $\textsc{AreDist} : Q \times Q \times \mathbb{N} \to \{0,1\}$:

$$\textsc{AreDist}(p,q,k) = \begin{cases} (p \in F) \oplus (q \in F), & \text{если } k = 0, \\ \textsc{AreDist}(p,q,k-1) \vee \\ \bigvee\limits_{a \in \Sigma} \textsc{AreDist}(\delta(p,a),\delta(q,a),k-1), & \text{иначе.} \end{cases}$$

Здесь $\oplus$ – это сложение по модулю 2 или так называемое “исключающее или”.

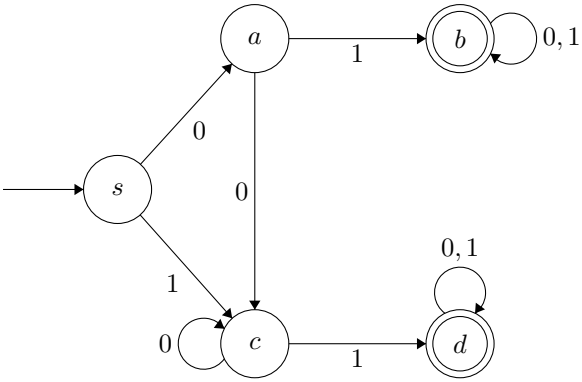
Пример

Рассмотрим такой ДКА



Построим минимальный ДКА, эквивалентный ему.

Шаг номер –1, препроцессинг. Удаляем из ДКА все недостижимые состояния, получаем эквивалентный ДКА с принимающими и непринимающими состояниями.



Шаг номер 0, вычисляем AreDist$(p, q, 0)$. Получаем такой результат (значения записаны в таблицу, 1 – различимы, 0 – неразличимы).

$k = 0$	s	a	b	c	d
s					
a	0				
b	1	1			
c	0	0	1		
d	1	1	0	1	

То есть состояния s, a, c неразличимы между собой, но различимы с состояниями b, d, которые также неразличимы между собой.

Шаг номер 1, вычисляем AreDist$(p, q, 1)$.

Состояния s и a становятся различимы, поскольку переход по единице ведёт в различимые c и b. Состояния s и c также становятся различимы, поскольку переход по единице ведёт в различимые c и d.

Состояния a и c остаются неразличимы: переход по нулю ведёт в неразличимые c и c, переход по единице ведёт в неразличимые b и d.

Состояния b и d остаются неразличимы: переходы ведут в неразличимые b и d.

Получаем

$k = 1$	s	a	b	c	d
s					
a	1				
b	1	1			
c	1	0	1		
d	1	1	0	1	

Шаг номер 2, вычисляем AreDist$(p, q, 2)$.

Состояния a и c остаются неразличимы: переход по нулю ведёт в неразличимые c и c, переход по единице ведёт в неразличимые b и d.

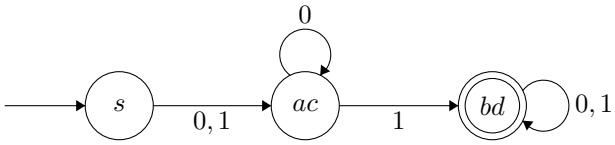
Состояния b и d остаются неразличимы: переходы ведут в неразличимые b и d.

Получаем

$k = 1$	s	a	b	c	d
s					
a	1				
b	1	1			
c	1	0	1		
d	1	1	0	1	

Переход шаг 1 → шаг 2 не изменил значений функции в таблице, так что дальнейшие вычисления не нужны.

Шаг номер *последний*. Состояния a и c неразличимы, так что мы объединяем их в одно состояние ac. Аналогично поступаем с b и d. Получаем автомат:



Автомат принимает язык $\mathsf{L}((0+1)0^*1(0+1)^*)$.

4. Синтаксический моноид

4.1. Моноид переходов ДКА

Пусть дан ДКА $A = (\Sigma, Q, s, F, \delta)$. Фиксируем некоторое слово w, если стандартным образом "скормить" это слово A, он перейдёт в состояние $\delta^*(s, w)$. Что будет, если начать с другого состояния? С точки зрения определения начальное состояние явно описано в пятёрке A, так что результат не будет иметь прямого отношения к тому, принимается слово w автоматом A или нет. Тем не менее, операция "скормить автомату слово w, начиная с состояния q", является вполне корректно определённой: результат этой операции – это просто $\delta^*(q, w)$.

Если мы будем "скармливать" одно и то же слово w различным состояниям $q_1, q_2, \ldots$ ДКА A, мы будем получать (в общем случае различные) состояния $p_1, p_2, \ldots$ Таким образом, слово w на автомате A определяет функцию.

Пусть A – ДКА над Σ, пусть $w \in \Sigma^*$. Определим функцию

$$\nu_w : Q \to Q \qquad \nu_w(q) = \delta^*(q, w).$$

Различные слова w порождают различные ν_w, однако поскольку Q конечно, то всевозможных функций $Q \to Q$ конечное число (их $|Q|^{|Q|}$), так что есть лишь конечное число различных функций среди всех ν_w (их вполне может быть меньше, чем $|Q|^{|Q|}$, потому что не каждая функция $Q \to Q$ обязана присутствовать среди ν_w).

Заметим, что по определению функция ν_ε действует по правилу $\nu_\varepsilon(q) = \delta^*(q, \varepsilon) = q$, так что функция ν_ε – это тождественная функция.

Возьмём две функции ν_u и ν_v. Поскольку эти функции – преобразования множества Q, можно определить композицию $\nu_u \circ \nu_v$, для которой выполнено

$$(\nu_u \circ \nu_v)(q) = \nu_u(\nu_v(q)) = \nu_u(\delta^*(q, v)) = \delta^*(\delta^*(q, v), u) = \\ = \delta^*(q, vu) = \nu_{vu}(q).$$

Следовательно, $(\nu_u \circ \nu_v) \circ \nu_w = \nu_{vu} \circ \nu_w = \nu_{wvu} = \nu_u \circ \nu_{wv} = \nu_u \circ (\nu_v \circ \nu_w)$.

Значит, множество всех функций ν_w по всем $w \in \Sigma^*$ с операцией композиции – это множество с "единицей" ν_ε и ассоциативной операцией умножения (композицией функций). Такая структура называется в алгебре **моноидом** (в отличие от группы, в которой ещё дополнительно требуется существование обратного для каждого элемента).

Для фиксированного ДКА A язык, который он принимает – это $\mathsf{L}(A)$, а моноид, порождённый операциями выше, – это $\mathcal{M}_A$, **моноид переходов** A.

Пусть теперь Σ – алфавит, тогда множество Σ^* является моноидом (с умножением – конкатенацией и пустым словом в качестве единицы), который называется **свободным моноидом** над Σ.

Сделаем такое отображение:

$$\varphi : \Sigma^* \to \mathcal{M}_A \qquad \varphi(w) = \nu_w.$$

Отображение φ является (гомо)морфизмом между двумя моноидами (оно сохраняет единицу и переводит произведение в произведение). Гомоморфизм φ "склеивает" все слова, у которых ν_w одинаковые (как функции) в одну эту функцию. Помимо прочего, такое отображение определяет отношение эквивалентности на всех словах над алфавитом Σ, а именно:

$$w \equiv_{\mathcal{M}_A} u \iff \varphi(w) = \varphi(u).$$

Теперь рассмотрим всевозможные ДКА, получающиеся из A заменой F на другое подмножество состояний, – каждый такой ДКА распознаёт (в общем случае) другой регулярный язык, но у всех таких ДКА один и тот же моноид переходов $\mathcal{M}_A$ (потому что он не зависит от F вообще). Все слова из Σ^* разбиваются функцией φ на $|\mathcal{M}_A|$ классов эквивалентности, каждый из классов есть $\varphi^{-1}(\nu)$ для некоторого $\nu \in \mathcal{M}_A$ – полный прообраз элемента – множество всех слов, которые порождают одну и ту же функцию ν.

Пусть функция ν_w для ДКА A такова, что $\nu_w(s) \in F$. Это по определению означает, что $\nu_w(s) = \delta^*(s, w) \in F$, т. е. что A принимает w.

Значит, A принимает w тогда и только тогда, когда $\nu_w(s) \in F$. В свою очередь это означает, что

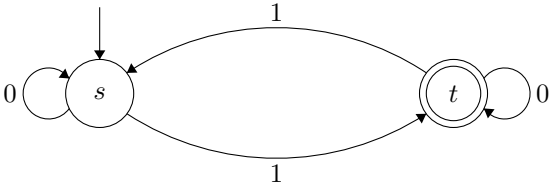
$$\mathsf{L}(A) = \varphi^{-1}\left(\{\nu_w \in \mathcal{M}_A \,|\, \nu_w(s) \in F\}\right).$$

Здесь аргументом φ^{-1} выступает целое множество – те функции (т. е. слова), которые переводят начальное состояние в принимающее. Следовательно, язык, принимаемый A, определяется некоторым *подмножеством* моноида переходов $\mathcal{M}_A$ и морфизмом φ. Смена подмножества отвечает смене принимающих состояний у того ДКА, из которого получился моноид переходов.

Пример

Рассмотрим ДКА $A = (\{0,1\}, \{s,t\}, s, \{t\}, \delta)$, где

$$\delta(s,0) = s,\ \delta(s,1) = t,\ \delta(t,0) = t,\ \delta(t,1) = s.$$



ДКА A принимает язык

$$\mathsf{L}(0^*10^*(10^*10^*)^*)$$

всех слов, содержащих нечётное число единиц.

Здесь два состояния, поэтому теоретически в моноиде переходов может быть четыре функции, задаваемые правилами:

$$f_1(s) = s,\ f_1(t) = s;$$

$$f_2(s) = s,\ f_2(t) = t;$$

$$f_3(s) = t,\ f_3(t) = s;$$

$$f_4(s) = t,\ f_4(t) = t.$$

При этом очевидно, $\nu_\varepsilon = f_2$, $\nu_1 = f_3$, $\nu_0 = f_2$. Поскольку f_2 – тождественная функция, а $f_3 \circ f_3 = f_2$, то моноид переходов содержит только два элемента: ν_ε и ν_1. При этом

$$\nu_\varepsilon \circ \nu_\varepsilon = \nu_1 \circ \nu_1 = \nu_\varepsilon,\ \nu_\varepsilon \circ \nu_1 = \nu_1 \circ \nu_\varepsilon = \nu_1.$$

Отображение φ переводит все слова с чётным числом единиц в ν_ε, все слова с нечётным числом единиц в ν_1.

В моноиде $\mathcal{M}_A = \{\nu_\varepsilon, \nu_1\}$ можно выделить четыре различных подмножества: $\varnothing, \{\nu_\varepsilon\}, \{\nu_1\}, \mathcal{M}_A$. Язык, принимаемый A, соответствует третьему их них, $\{\nu_1\}$, поскольку $\{\nu_1\} = \{\nu \in \mathcal{M}_A \mid \nu(s) \in F\}$.

Если же взять другие подмножества, мы получим "другие версии" A, отличающиеся друг от друга выбором множества принимающих состояний F. Все такие ДКА будут распознавать некоторые регулярные языки, а именно:

$$\varnothing = \varphi^{-1}(\varnothing),$$

$$\mathsf{L}(0^*(10^*10^*)^*) = \varphi^{-1}(\{\nu_\varepsilon\}),$$

$$\mathsf{L}(A) = \mathsf{L}(0^*10^*(10^*10^*)^*) = \varphi^{-1}(\{\nu_1\}),$$

$$\Sigma^* = \varphi^{-1}(\mathcal{M}_A).$$

4.2. Синтаксический моноид языка

Напомним, язык L (над Σ) порождает собой отношение эквивалентности Майхилла–Нероуда на множестве Σ^* – это отношение эквивалентности $\equiv_L$, где

$$x \equiv_L y \iff (\forall z \in \Sigma^*\ xz \in L \Leftrightarrow yz \in L).$$

Отношение эквивалентности Майхилла–Нероуда учитывает только приписывание слов справа (ведь ДКА читает слова слева направо). Можно рассмотреть симметричное отношение, назовём его $\equiv_L^L$:

$$x \equiv_L^L y \iff (\forall u, v \in \Sigma^*\ uxv \in L \Leftrightarrow uyv \in L).$$

У этого нового отношения есть важное свойство: пусть $x \equiv_L^L x'$ и $y \equiv_L^L y'$. Тогда
1) $xy \equiv_L^L x'y$ (подставим вместо v суффикс yv в определение $x \equiv_L^L x'$),
2) $x'y \equiv_L^L x'y'$ (подставим вместо u префикс ux' в определение $y \equiv_L^L y'$),
3) $xy \equiv_L^L x'y'$ по транзитивности, поскольку это отношение эквивалентности.

Почему это важно: благодаря этому свойству классы эквивалентности по отношению $\equiv_L^L$ можно перемножать:

$$[x]_L^L \cdot [y]_L^L = [xy]_L^L.$$

Это возможно, поскольку класс $[xy]_L^L$ корректно определён: каких бы представителей из классов $[x]_L^L$ и $[y]_L^L$ мы ни взяли, всевозможные

их произведения будут эквивалентны между собой и, значит, будут принадлежать одному классу эквивалентности.

Множество классов эквивалентности по отношению $\equiv_L^L$ является моноидом (с единицей $[\varepsilon]_L^L$) и называется **синтаксическим моноидом языка** L. Мы обозначим его $\mathcal{S}_L$.

Поскольку на отношение $\equiv_L^L$ накладывается больше требований, в общем случае оно измельчает $\equiv_L$, так что классов эквивалентности у него может быть больше. Для отношения эквивалентности Майхилла–Нероуда язык L есть объединение некоторых классов эквивалентности. Поскольку $\equiv_L^L$ измельчает $\equiv_L$, то L также есть объединение некоторых классов эквивалентности (в общем случае большего их количества) по $\equiv_L^L$. Это значит, что

$$\exists X \subseteq \mathcal{S}_L \quad L = \bigcup_{[\cdot]_L^L \in X} [\cdot]_L^L.$$

Кроме того, поскольку отношение $\equiv_L^L$ измельчает $\equiv_L$, то в силу теоремы Майхилла–Нероуда, если синтаксический моноид языка конечен, то язык регулярен (фактормножество по $\equiv_L^L$ конечно, значит фактормножество по $\equiv_L$ тем более конечно).

Определим **синтаксический морфизм** $\sigma : \Sigma^* \to \mathcal{S}_L$, отправляющий слово в соответствующий этому слову класс эквивалентности:

$$\sigma(w) = [w]_L^L.$$

Для языка L существует $X \subseteq \mathcal{S}_L$, что

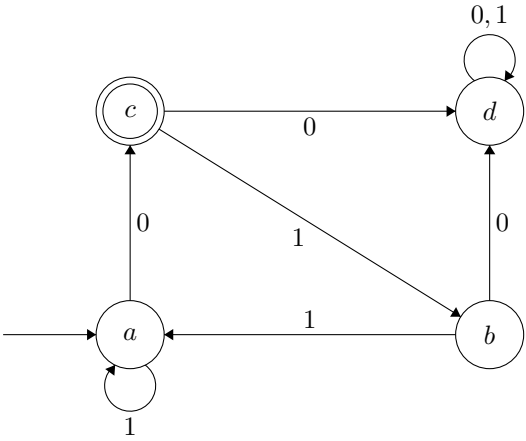
$$L = \sigma^{-1}(\{x \in \mathcal{S}_L \mid x \in X\}) = \sigma^{-1}(X).$$

Множество X при этом можно положить равным $X = \sigma(L)$, поскольку именно у этих элементов $\mathcal{S}$ существует прообраз в языке L. Окончательно получим

$$L = \sigma^{-1}(\sigma(L)).$$

Пример

Рассмотрим такой ДКА



Этот автомат принимает некоторый язык L, причём это минимальный автомат для этого языка – так что его состояния суть классы эквивалентности Майхилла–Нероуда.

Проверим последнее (и заодно минимальность ДКА) – четыре состояния отвечают четырём классам:

1) a есть класс $[\varepsilon]_L$;
2) b есть класс $[01]_L$;
3) c есть класс $[0]_L$;
4) d есть класс $[00]_L$.

Очевидно, 0 неэквивалентен ни ε, ни 01, ни 00 – так как он лежит в языке, а остальные нет (суффикс ε). Слово 00 не эквивалентно ни ε, ни 01 по суффиксу 10. Слово ε не эквивалентно слову 01 по суффиксу 0. Следовательно, это действительно четыре различных класса эквивалентности, так что этот автомат действительно минимален для языка L. Сам язык при этом равен классу $[0]_L$.

Построим теперь моноид переходов для этого автомата. Теоретически здесь могло бы быть $4^4 = 256$ классов. Мы получаем, однако, 12 классов эквивалентности.

Класс $[\varepsilon]_L$ расщепился на $[\varepsilon]_L^L$, $[1]_L^L$, $[11]_L^L$, $[011]_L^L$, $[1011]_L^L$.

Класс $[01]_L$ расщепился на $[01]_L^L$, $[101]_L^L$, $[1101]_L^L$.

Класс $[0]_L$ расщепился на $[0]_L^L$, $[10]_L^L$, $[110]_L^L$ – эти три класса образуют язык L.

Класс $[00]_L$ перешёл целиком в класс $[00]_L^L$.

	a	b	c	d
ε	a	b	c	d
0	c	d	d	d
1	a	a	b	d
00	d	d	d	d
01	b	d	d	d
10	c	c	d	d
11	a	a	a	d
011	a	d	d	d
101	b	b	d	d
110	c	c	c	d
1011	a	a	d	d
1101	b	b	b	d

Элементы моноида переходов. В левой колонке стоит w, а справа в столбце с меткой x стоит значение $\nu_w(x)$.

	ε	0	1	00	01	10	11	011	101	110	1011	1101
ε	ε	0	1	00	01	10	11	011	101	110	1011	1101
0	0	00	01	00	00	00	011	00	00	0	00	01
1	1	10	11	00	101	110	11	1011	1101	110	11	1101
00	00	00	00	00	00	00	00	00	00	00	00	00
01	01	00	011	00	00	0	011	00	01	0	011	01
10	10	00	101	00	00	00	1011	00	00	10	00	101
11	11	110	11	00	1101	110	11	11	1101	110	11	1101
011	011	0	011	00	01	0	011	011	01	0	011	01
101	101	00	1011	00	00	10	1011	00	101	10	1011	101
110	110	00	1101	00	00	00	11	00	00	110	00	1101
1011	1011	10	1011	00	101	10	1011	1011	101	10	1011	101
1101	1101	00	11	00	00	110	11	00	1101	110	11	1101

Таблица умножения для моноида переходов. Элементы моноида заданы представителями, для элемента x в строке и элемента y в столбце в таблице стоит произведение xy.

В частности, $\varepsilon \equiv_L 1$, поскольку

$$\delta^*(a, 1x) = \delta^*(\delta(a, 1), x) = \delta^*(a, x).$$

Но при этом $\varepsilon \not\equiv_L^L 1$ (префикс 0 и суффикс ε), поскольку $0\varepsilon\varepsilon \in L$, а $01\varepsilon \notin L$.

4.3. Распознавание языка моноидом

Пусть M – произвольный моноид. Скажем, что язык $L \subseteq \Sigma^*$ **распознаётся моноидом** M, если существует морфизм $\psi : \Sigma^* \to M$ такой, что

$$L = \psi^{-1}(\psi(L)).$$

Что здесь написано: язык L обладает такой структурой – если $w \in L$ и при этом $\psi(w) = m$, то все такие слова u, что $\psi(u) = m$, тоже должны лежать в L.

Очевидно, любой язык распознаётся своим собственным синтаксическим моноидом (требуемым морфизмом будет синтаксический морфизм), поскольку L есть объединение некоторых классов эквивалентности по $\equiv_L^L$.

В контексте регулярных языков нас будет интересовать возможность распознавания языка некоторым **конечным** моноидом.

4.3.1. Регулярный язык распознаётся конечным моноидом

Теорема. *Если язык L регулярный, то он распознаётся некоторым конечным моноидом.*

Доказательство. В самом деле, пусть L регулярен, пусть ДКА A его принимает, рассмотрим моноид переходов $\mathcal{M}_A$. Этот моноид конечен, кроме того отображение φ обладает свойством

$$L = \mathsf{L}(A) = \varphi^{-1}\left(\{\nu_w \in \mathcal{M}_A \,|\, \nu_w(s) \in F\}\right).$$

Условие $\nu_w(s) \in F$ эквивалентно условию: "A принимает w" и, значит, условию: $w \in L$. Значение ν_w есть просто $\varphi(w)$, и значит множество

$$\{\nu_w \in \mathcal{M}_A \,|\, \nu_w(s) \in F\}$$

есть просто $\{\varphi(w) \,|\, w \in L\}$, т. е. $\varphi(L)$.

Получается, что $L = \varphi^{-1}(\varphi(L))$ и тогда конечный моноид $\mathcal{M}_A$ распознаёт L.

4.3.2. Язык, распознаваемый конечным моноидом, регулярен

Теорема. *Если язык L распознаётся некоторым конечным моноидом, то он является регулярным.*

Доказательство. Пусть L (над Σ) распознаётся некоторым конечным моноидом M с некоторым морфизмом $\varphi : \Sigma^* \to M$.

Как и всегда будем использовать обозначение $\varphi(A)$ для $A \subseteq \Sigma^*$, имея в виду

$$\varphi(A) = \{\varphi(x) \,|\, x \in A\}.$$

Рассмотрим ДКА $A = (\Sigma, \varphi(\Sigma^*), \varphi(\varepsilon), \varphi(L), \delta)$, где δ действует по правилу:

$$\delta(\varphi(w), a) = \varphi(wa), \qquad w \in \Sigma^*, a \in \Sigma.$$

Состояниями автомата являются всевозможные элементы $\varphi(w) \in M$ (весь моноид M, если морфизм φ сюръективен), начальным состоянием является $\varphi(\varepsilon) \in M$, принимающими состояниями являются такие элементы M, что их прообраз содержит слова из языка L (и вследствие того, что M распознаёт L, состоит только из таких слов).

Докажем, что для любого $w \in \Sigma^*$ выполнено $\delta^*(\varphi(\varepsilon), w) = \varphi(w)$. База индукции: $\delta^*(\varphi(\varepsilon), \varepsilon) = \varphi(\varepsilon)$ по определению δ^*. Шаг индукции:

$$\delta^*(\varphi(\varepsilon), wa) = \delta(\delta^*(\varphi(\varepsilon), w), a) = \delta(\varphi(w), a) = \varphi(wa),$$

второе равенство здесь – предположение индукции. Значит, действительно $\delta^*(\varphi(\varepsilon), w) = \varphi(w)$, а значит слово w принимается A тогда и только тогда, когда $\varphi(w) \in \varphi(L)$, т. е. $w \in \varphi^{-1}(\varphi(L))$. Но поскольку L распознаётся M с морфизмом φ, $L = \varphi^{-1}(\varphi(L))$, так что w принимается A тогда и только тогда, когда $w \in L$, так что L – регулярный язык.

4.3.3. Основной результат

Соберём результаты предыдущих пунктов вместе.

Теорема. *Язык L регулярен тогда и только тогда, когда он распознаётся некоторым конечным моноидом.*

В приложениях удобно пользоваться эквивалентным определением распознаваемости. Язык $L \subseteq \Sigma^*$ **распознаётся моноидом** M, если существует морфизм $\psi : \Sigma^* \to M$ и подмножество $N \subseteq M$ такие, что

$$L = \psi^{-1}(N).$$

В самом деле, из исходного определения следует это определение после обозначения $N = \psi(L)$. Обратно, если $L = \psi^{-1}(N)$ для некоторого N, то $\psi(L) \subseteq N$ (иначе существует такой $x \in L$, что $\psi(x) \notin N$, но тогда $x \notin \psi^{-1}(N)$, что противоречит $x \in L$). Тогда

$$L \subseteq \psi^{-1}(\psi(L)) \subseteq \psi^{-1}(N) = L,$$

так что выполнено исходное определение. Множество N может быть больше, чем $\psi(L)$, но только на элементы z, для которых $\psi^{-1}(z) = \varnothing$.

Пример

Пусть L – регулярный язык, докажем, что язык $\sqrt{L} = \{w \,|\, ww \in L\}$ также регулярный.

Поскольку L регулярен, он распознаётся некоторым конечным моноидом M с морфизмом φ, причём $L = \varphi^{-1}(\varphi(L))$. Введём обозначение $P = \varphi(L)$. P – это некоторое подмножество M (то самое, прообраз которого под действием φ равен L).

Пусть $Q = \left\{x \in M \,\middle|\, x^2 \in P\right\} \subseteq M$. Тогда

$$\varphi^{-1}(Q) \overset{1}{=} \{w \in \Sigma^* \,|\, \varphi(w) \in Q\} \overset{2}{=} \left\{w \in \Sigma^* \,\middle|\, \varphi(w)^2 \in P\right\} \overset{3}{=}$$
$$\overset{3}{=} \left\{w \in \Sigma^* \,\middle|\, \varphi\left(w^2\right) \in P\right\} \overset{4}{=} \left\{w \in \Sigma^* \,\middle|\, w^2 \in L\right\} \overset{5}{=} \sqrt{L}.$$

Здесь первое равенство – это определение φ^{-1}, второе равенство – это определение Q, третье равенство верно, поскольку φ морфизм, четвёртое – это определение P, пятое – это определение $\sqrt{L}$.

Значит, для языка $\sqrt{L}$ существует конечный моноид M, его подмножество Q и морфизм φ такие, что $\sqrt{L} = \varphi^{-1}(Q)$. Значит (по эквивалентному определению распознавания языка моноидом), язык $\sqrt{L}$ распознаётся конечным моноидом и тогда является регулярным.

4.4. Моноид для минимального ДКА

Рассмотрим теперь регулярный язык L и минимальный автомат A, который его принимает.

Теорема. *Моноид переходов минимального ДКА есть синтаксический моноид соответствующего регулярного языка.*

Доказательство. Действительно, по определению

$$x \equiv_L^L y \Longleftrightarrow (\forall u, v \in \Sigma^* \; uxv \in L \Leftrightarrow uyv \in L)\,.$$

Пусть $p = \delta^*(s, u)$, перебор всевозможных $u \in \Sigma^*$ соответствует перебору всех $p \in Q$ (поскольку все они достижимы из-за минимальности ДКА). Условие $uxv \in L$ эквивалентно $\delta^*(s, uxv) \in F$, так что получаем

$$x \equiv_L^L y \Longleftrightarrow (\forall v \in \Sigma^* \forall p \in Q \quad \delta^*(p, xv) \in F \Leftrightarrow \delta^*(p, yv) \in F)\,.$$

Два состояния p и q ДКА различимы тогда и только тогда, когда существует слово z такое, что ровно одно из двух состояний $\delta^*(p, z)$

и $\delta^*(q,z)$ принимающее. У минимального ДКА все состояния различимы между собой. Но $\delta^*(p,xv) = \delta^*(\delta^*(p,x),v)$ и аналогично для yv. Получаем, что

$$x \equiv_L^L y \iff \forall p \in Q\,(\forall v \in \Sigma^* \quad \delta^*(\delta^*(p,x),v) \in F \Leftrightarrow \delta^*(\delta^*(p,y),v) \in F)\,.$$

Значит, состояния $\delta^*(p,x)$ и $\delta^*(p,y)$ неразличимы, значит (в силу минимальности ДКА) это одно и то же состояние, $\delta^*(p,x) = \delta^*(p,y)$, т. е.

$$x \equiv_L^L y \iff \forall p \in Q \quad \delta^*(p,x) = \delta^*(p,y) \iff \nu_x = \nu_y,$$

последняя эквивалентность выполнена в силу того, что эти функции совпадают на любых аргументах.

Значит, $x \equiv_L^L y$ эквивалентно $\nu_x = \nu_y$ для моноида переходов, что и требовалось доказать. В частности, это означает, что синтаксический моноид регулярного языка конечен.

5. Модификации конечных автоматов

5.1. Автомат Мура

Все конечные автоматы, рассматриваемые до сих пор, являлись распознавателями языков: по данному слову они отвечали "да" или "нет", выделяя из множества всех слов над данным алфавитом некоторое подмножество – язык. Автомат Мура является строковым *преобразователем*: он получает на вход слово, а выдаёт в качестве ответа другое слово.

Формальное определение (одно из нескольких эквивалентных) таково: автомат Мура – это шестёрка

$$M = (\Sigma, \Delta, Q, s, \delta, \tau), \quad \text{где}$$

Σ – это **входной алфавит** автомата,
Δ – это **выходной алфавит** автомата,
Q – это конечное множество **состояний**,
$s \in Q$ – это **начальное состояние**,
$\delta : Q \times \Sigma \to Q$ – это **функция переходов**,
$\tau : Q \to \Delta$ – это **функция вывода**.

Автомат M получает на вход слово x, начинает свою работу в состоянии s и читает x букву за буквой. Во время чтения символа a из состояния p происходит следующее: автомат переходит в состояние $\delta(p,a)$ и **печатает** символ $\tau(\delta(p,a))$.

Заметим, что у автомата нет принимающих или отвергающих состояний, так что он не выделяет некоторый язык из Σ^*.

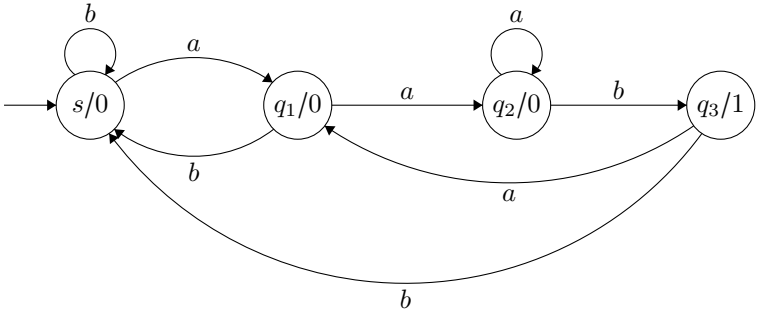
Расширим δ до $\delta^* : Q \times \Sigma^* \to Q$ точно так же, как это сделано для ДКА. Определим далее $\tau^* : Q \times \Sigma^* \to \Delta^*$:

$$\tau^*(q, w) = \begin{cases} \varepsilon, & \text{если } w = \varepsilon, \\ \tau(\delta(q, a)) \cdot \tau^*(\delta(q, a), x), & \text{если } w = ax. \end{cases}$$

Результатом работы или **выходом** автомата Мура M на входе x называется слово $\tau^*(s, x)$.

Пример

Пусть $\Sigma = \{a, b\}$, $\Delta = \{0, 1\}$, $Q = \{s, q_1, q_2, q_3\}$. Метка на состоянии q в диаграмме ниже имеет вид $q/\tau(q)$.



На слове $baabaaaba$ автомат выдаст

$$\begin{aligned}
&\tau^*(s, baabaaaba) = \\
&= \tau(\delta(s,b))\tau^*(\delta(s,b), aabaaaba) = \tau(s)\tau^*(s, aabaaaba) = \\
&= 0\tau(\delta(s,a))\tau^*(\delta(s,a), abaaaba) = 0\tau(q_1)\tau^*(q_1, abaaaba) = \\
&= 00\tau(\delta(q_1,a))\tau^*(\delta(q_1,a), baaaba) = 00\tau(q_2)\tau^*(q_2, baaaba) = \\
&= 000\tau(\delta(q_2,b))\tau^*(\delta(q_2,b), aaaba) = 000\tau(q_3)\tau^*(q_3, aaaba) = \\
&= 0001\tau(\delta(q_3,a))\tau^*(\delta(q_3,a), aaba) = 0001\tau(q_1)\tau^*(q_1, aaba) = \\
&= 00010\tau(\delta(q_1,a))\tau^*(\delta(q_1,a), aba) = 00010\tau(q_2)\tau^*(q_2, aba) = \\
&= 000100\tau(\delta(q_2,a))\tau^*(\delta(q_2,a), ba) = 000100\tau(q_2)\tau^*(q_2, ba) = \\
&= 0001000\tau(\delta(q_2,b))\tau^*(\delta(q_2,b), a) = 0001000\tau(q_3)\tau^*(q_3, a) = \\
&= 00010001\tau(\delta(q_3,a))\tau^*(\delta(q_3,a), \varepsilon) = 00010001\tau(q_1)\tau^*(q_1, \varepsilon) = \\
&= 000100010\varepsilon = 000100010.
\end{aligned}$$

После прочтения каждой подстроки aab автомат выдаёт 1.

5.1.1. Язык автомата Мура

Языком автомата Мура M назовём множество

$$L(M) = \{\tau^*(s, w) \,|\, w \in \Sigma^*\} \subseteq \Delta^*.$$

Теорема. *Язык автомата Мура является регулярным языком.*

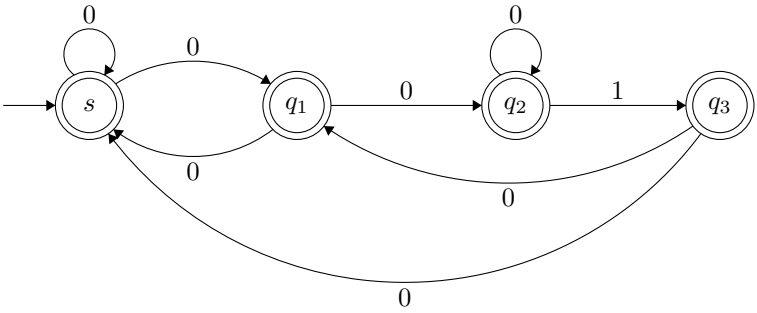
Доказательство. В самом деле, пусть дан автомат Мура:

$$M = (\Sigma, \Delta, Q, s, \delta, \tau),$$

построим для него НКА (без ε-переходов) M', распознающий $L(M)$. Положим $M' = (\Delta, Q, s, Q, \delta')$ (все состояния будут принимающими). Здесь для $q \in Q$ и $b \in \Delta$

$$\delta'(q, b) = \{\delta(q, a) \,|\, a \in \Sigma, \tau(\delta(q, a)) = b\}.$$

В частности, пример выше преобразуется в такой НКА:



Докажем следующее утверждение: для любых $p, q \in Q$, для любого $x \in \Delta^*$ верно, что

$$q \in \delta'^*(p, x) \iff \exists w \in \Sigma^* \;\; (\delta^*(p, w) = q \wedge \tau^*(p, w) = x)\,,$$

по индукции по длине слова x.

База: $x = \varepsilon$: слева $p = q$, справа $w = \varepsilon, p = q$.

Шаг: утверждение верно для всех y таких, что $|y| < |x|$. Пусть $x = by$.

$$\begin{aligned}
&q \in \delta'^*(p, x) \iff \\
&\exists r \in \delta'(p, b) \;\; q \in \delta'^*(r, y) \iff \\
&\exists a \in \Sigma \;\; \tau(\delta(p, a)) = b \;\; q \in \delta'^*(\delta(p, a), y) \iff \\
&\exists a \in \Sigma \;\; \tau(\delta(p, a)) = b \;\; \exists z \in \Sigma^* \;\; (\delta^*(\delta(p, a), z) = q \wedge \tau^*(\delta(p, a), z) = y) \iff \\
&\exists a \in \Sigma \; \exists z \in \Sigma^* \;\; (\delta^*(p, az) = q \wedge \tau^*(p, az) = by) \iff \\
&\exists w \in \Sigma^* \;\; (\delta^*(p, w) = q \wedge \tau^*(p, w) = x)\,.
\end{aligned}$$

Первая эквивалентность следует из определения δ'^*, вторая – из определения δ', третья – шаг индукции, четвёртая – из определения δ^* и τ^*, пятая – при выборе $w = az$.

Из этого утверждения при $p = s$ следует утверждение теоремы.

5.2. Двусторонний конечный автомат

Рассмотренные нами модели ДКА, НКА (а также ОКА) не могут прочесть буквы входного слова несколько раз: фактически по прочтении буквы уничтожаются. Однако вполне разумной выглядит идея, что входное слово раз и навсегда записано на некотором постоянном носителе, а автомат может "гулять" по слову и менять свои состояния. Это естественным образом приводит к (одному из многих-многих вариаций) определению двустороннего (детерминированного) конечного автомата. Неформально: автомат состоит из *ленты, разбитой на клетки,* и бегающей по ленте *головки.* На ленте написано входное слово, по одной букве в клетке. Головка читает символ в клетке, находясь при этом в каком-то *состоянии.* По прочтении символа головка меняет состояние и *сдвигается по ленте*: влево или вправо. При этом возникают специфические вопросы следующего вида: как обрабатывать случаи, когда головка "пытается выйти за пределы слова", как определить остановку и тому подобные. Существуют различные, более или менее эквивалентные способы работы с этим, мы представим один из них.

Двусторонний детерминированный конечный автомат (сокращённо 2ДКА) D – это семёрка:

$$D = (\Sigma, \vdash, \dashv, Q, s, t, \delta), \quad \text{где}$$

Σ – это алфавит автомата,

$\vdash \notin \Sigma$, $\dashv \notin \Sigma$ – это левый и правый **турникеты**, которые обозначают левую и правую границы входного слова,

Q – это конечное множество состояний,

$s \in Q$ – это начальное состояние,

$t \in Q$ – это принимающее состояние (достаточно иметь одно),

$\delta : Q \times (\Sigma + \{\vdash, \dashv\}) \to Q \times \{-1, +1\}$ – это функция переходов,

причём на функцию переходов накладываются следующие ограничения:

$$\forall q \in Q \quad \delta(q, \vdash) = (\cdot, +1), \qquad \delta(q, \dashv) = (\cdot, -1),$$

$$\forall b \in \Sigma + \{\vdash\} \quad \delta(t, b) = (t, +1), \qquad \delta(t, \dashv) = (t, -1).$$

Ограничения считаются частью определения, в том смысле что те семёрки, функции в которых не удовлетворяют требованиям выше, не считаются корректными 2ДКА. Первое ограничение (точка в первом аргументе означает произвольное состояние без ограничений) означает, что турникеты "отталкивают" головку автомата, не позволяя ему выйти за пределы ленты. Второе ограничение означает, что, попав в состояние t, D останется в состоянии t и, более того, дойдёт до правого турникета (а потом будет «болтаться» между правым турникетом и предыдущим символом).

Символы -1 и $+1$ в определении функции перехода указывают на движение головки: соответственно влево и вправо.

Несложно понять, что так определённый 2ДКА не останавливается в стандартном смысле, он уходит в бесконечный цикл: возможно в принимающем состоянии, а возможно и нет. Следовательно, для определения понятия "автомат принимает слово" необходимо использовать нечто иное, нежели чем проверка состояния при остановке.

Пусть на вход 2ДКА подано слово $x \in \Sigma^*$. Будем считать, что $x = x_1 x_2 \dots x_{|x|}$, где каждый $x_j \in \Sigma$ – буква. Тогда на ленте автомата написано слово $x_0 x_1 \dots x_{|x|} x_{|x|+1}$, где $x_0 = \vdash$, $x_{|x|+1} = \dashv$. Подчеркнём, что это неформальное понятие, формально у автомата нет никакой ленты.

Конфигурацией 2ДКА D на входе x называется пара

$$(q, j) \in Q \times \{0, \dots, |x| + 1\}.$$

Фактически, конфигурация – это "мгновенный снимок" работы 2ДКА, показывающий в каком состоянии находится 2ДКА и какой символ входа он читает. **Начальной конфигурацией** называется $(s, 0)$.

Конфигурация (p, i) **влечёт** конфигурацию $(q, i \pm 1)$ на слове x, если $\delta(p, x_i) = (q, \pm 1)$. Мы пишем $(p, i) \to_x (q, j)$ для обозначения, что конфигурация (p, i) влечёт конфигурацию (q, j). Каждый *шаг* работы 2ДКА – переход от одной конфигурации к другой по отношению "влечёт".

Определим по индукции для двух конфигураций α и β отношение: «**влечёт за** n **шагов**», писать будем $\alpha \to_x^n \beta$:

– $\alpha \to_x^0 \beta$ тогда и только тогда, когда $\alpha = \beta$,

– для $n > 0$ $\alpha \to_x^n \beta$ выполнено тогда и только тогда, когда

$$\exists \gamma \left(\alpha \to_x^{n-1} \gamma\right) \wedge \left(\gamma \to_x \beta\right).$$

Определим отношение: «α **финально влечёт** β» или «β **достижима** из α»:

$$\alpha \to_x^* \beta \iff \exists n \in \mathbb{N} \quad \alpha \to_x^n \beta,$$

т. е. из α можно получить β за некоторое конечное число шагов.

Наконец, D **принимает** слово x, если $(s, 0) \to_x^* (t, |x| + 1)$. Язык, принимаемый 2ДКА D, – это множество $L(D)$ всех слов, принимаемых D. Заметим, что D принимает слово w тогда и только тогда, когда на этом слове достигается (из начальной) хотя бы некоторая конфигурация вида $(t, \cdot)$ – это следует из ограничений на функцию перехода.

Заметим, что из определения следует, что если D принимает слово x, то в некоторый момент головка D находится над правым турникетом – это техническая особенность используемого определения, которая будет полезна в дальнейшем.

Можно было бы добавить в определение 2ДКА ещё одно состояние r с аналогичными t свойствами, интерпретируемое как "отвергнуть". Такая возможность показывает, что у 2ДКА теоретически можно ввести несколько корректных "режимов" завершения работы.

Пример

Пусть $D = (\{a, b\}, \vdash, \dashv, \{q_0, q_1, q_2, p_0, p_1, t, r\}, q_0, t, \delta)$, где функция перехода δ задана таблицей ниже. Прочерками обозначены переходы, не встречающиеся при работе автомата, так что для работы 2ДКА не имеет значения, что там написано (формально там необходимо проставить что-то, мы не делаем этого для облегчения восприятия описания автомата).

δ	$\vdash$	a	b	$\dashv$
q_0	$(q_0, +1)$	$(q_1, +1)$	$(q_0, +1)$	$(p_0, -1)$
q_1	—	$(q_2, +1)$	$(q_1, +1)$	$(r, -1)$
q_2	—	$(q_0, +1)$	$(q_2, +1)$	$(r, -1)$
p_0	$(t, +1)$	$(p_0, -1)$	$(p_1, -1)$	—
p_1	$(r, +1)$	$(p_1, -1)$	$(p_0, -1)$	—
t	$(t, +1)$	$(t, +1)$	$(t, +1)$	$(t, -1)$
r	$(r, +1)$	$(r, +1)$	$(r, +1)$	$(r, -1)$

На входе *aababbb* автомат D последовательно проходит по конфигурациям $(q_0, 0)$, $(q_0, 1)$, $(q_1, 2)$, $(q_2, 3)$, $(q_2, 4)$, $(q_0, 5)$, $(q_0, 6)$, $(q_0, 7)$, $(q_0, 8)$, $(p_0, 7)$, $(p_1, 6)$, $(p_0, 5)$, $(p_1, 4)$, $(p_1, 3)$, $(p_0, 2)$, $(p_0, 1)$, $(p_0, 0)$, $(t, 1)$, $(t, 2)$, $(t, 3)$, $(t, 4)$, $(t, 5)$, $(t, 6)$, $(t, 7)$, $(t, 8)$, что означает, что автомат принимает слово.

На входе *aababa* автомат D последовательно проходит по конфигурациям $(q_0, 0)$, $(q_0, 1)$, $(q_1, 2)$, $(q_2, 3)$, $(q_2, 4)$, $(q_0, 5)$, $(q_0, 6)$, $(q_1, 7)$, $(r, 6)$

и далее всегда остаётся в состоянии r, что означает, что автомат не принимает слово.

Несложно понять, что автомат D принимает те и только те слова, в которых число символов a кратно трём, а число символов b чётно.

5.2.1. Эквивалентность ДКА

Теорема. *Язык L принимается некоторым 2ДКА тогда и только тогда, когда L является регулярным.*

Доказательство. Мы знаем, что регулярность языка эквивалентна распознаванию языка ДКА. Очевидно, что если ДКА A принимает язык L, то существует 2ДКА, который принимает L: двустороннему автомату достаточно не делать переходов вправо, а после прочтения входа перейти в принимающее или в уникальное "отвергающее" состояние.

Пусть теперь язык L принимается 2ДКА D. Пусть вход для 2ДКА D имеет вид xy с префиксом x и суффиксом y. Какую информацию о строке x можно "пронести" в строку y? Автомат начинает свою работу, читая левый турникет, далее он может на некотором шаге пересечь границу между x и y или же не пересечь её никогда (тогда он не принимает вход). Если автомат пересекает границу, то он пересекает её в некотором состоянии p. Заметим, что это состояние не зависит от y, поскольку ни один символ y ещё не был прочитан автоматом, так что суффикс y никак не влиял на поведение автомата.

Введём функцию $T_x : (Q + \{@\}) \to (Q + \{\perp\})$. Символы @ и $\perp$ – просто удобные символы для обозначения поведения в начале работы и зацикливания.

Определим сначала T_x в точке @. Если 2ДКА D, начиная работу на слове xy, пересекает границу между x и y в состоянии p, то положим $T_x(@) = p$. Если D никогда не пересекает границу между x и y, то $T_x(@) = \perp$.

Далее, определим T_x для каждого $q \in Q$. Пусть $T_x(@) \neq \perp$, тогда автомат, работая на слове xy после первого попадания в y, может вернуться в x (а может и не вернуться). Если он возвращается в x, он возвращается туда, пересекая границу между x и y (на этот раз справа налево) в некотором состоянии q. После этого произойдёт одно из двух:

1) автомат вернётся обратно в y в некотором состоянии p,

2) автомат никогда не вернётся в y.

В первом случае положим $T_x(q) = p$, а во втором $T_x(q) = \perp$.

Более точно, $T_x(q)$ – это то состояние, в котором автомат пересечёт правую границу слова x слева направо, если он начнёт работу над *правым* символом x в состоянии q (или $\perp$, если пересечения не произойдёт).

Заметим также, что $T_x(q)$ зависит от x и q, но не от y. Задав функцию T_x, мы полностью описываем всю информацию, которую автомат D может пронести сквозь правую границу слова x.

Всего различных функций T_x существует $(|Q|+1)^{|Q|+1}$. Более того, если $T_x = T_y$ для двух слов x и y и D принимает xz, то D принимает yz. В самом деле, последовательность состояний, в которых D пересекает левую границу z совпадают у xz и yz – слева направо из-за $T_x = T_y$, а справа налево из-за общего суффикса z. Автомат принимает слово тогда и только тогда, когда он читает правый турникет в состоянии t. Поскольку последовательности состояний одинаковы, то если это верно для xz, то верно и для yz.

Получаем

$$T_x = T_y \Longrightarrow \forall z \in \Sigma^* \ \left(xz \in L(D) \Leftrightarrow yz \in L(D)\right).$$

Следовательно, если у двух слов равны их T-функции, то эти слова эквивалентны по отношению эквивалентности Майхилла–Нероуда. Поскольку всего вариантов T-функций конечное число, то фактормножество по отношению эквивалентности Майхилла–Нероуда конечно, а, значит, язык $L(D)$ является регулярным.

5.3. Альтернирующий конечный автомат

В отличие от ДКА НКА "принимает слово, если может", а точнее, реализует дизъюнкцию (квантор существования): НКА реализует дерево вычислений и если дизъюнкция на всех листьях (в смысле того, что принимающий лист – это 1, непринимающий – 0) равна 1, то НКА принимает слово.

Можно обобщить такой подход на более сложные булевы операции. Для начала приведём несколько определений. Пусть M – конечное множество. Через $\mathcal{B}^+(M)$ обозначим множество **положительных булевых функций** с элементами M как переменными. Элементы $\mathcal{B}^+(M)$ – это булевы функции, использующие только связки $\wedge$ и $\vee$, кроме того, мы будем считать, что в этом множестве лежат функции "тождественная истина" (обозначим её $\top$) и "тождественная ложь" (обозначим её $\perp$).

Пусть $\varphi \in \mathcal{B}^+(M)$, пусть $S \subseteq M$. Через $\varphi(S)$ обозначим значение функции φ на наборе переменных такого вида: если $q \in S$, то положим $q = 1$ в формуле, если $q \notin S$, то положим $q = 0$ в формуле.

Теперь определим альтернирующий конечный автомат (АКА). Это кортеж из пяти элементов:

$$G = (\Sigma, Q, s, F, \delta),$$

где первые четыре компоненты стандартны для конечных автоматов:

Σ – это алфавит АКА,

Q – это конечное множество состояний,

$s \in Q$ – это начальное состояние,

$F \subseteq Q$ – это множество принимающих состояний.

Функция переходов δ имеет сигнатуру $\delta : Q \times \Sigma \to \mathcal{B}^+(Q)$.

Как обычно, расширим функцию переходов на слова, чтобы определить термин "АКА принимает слово". Функция

$$\delta^* : \mathcal{B}^+(Q) \times \Sigma^* \to \mathcal{B}^+(Q)$$

определяется рекурсивно. Для любого слова w и любой функции, отличной от одной переменной (эта функция является тождественной или же дизъюнкцией/конъюнкцией некоторых функций):

$$\delta^*(\varphi \wedge \psi, w) = \delta^*(\varphi, w) \wedge \delta^*(\psi, w),$$

$$\delta^*(\varphi \vee \psi, w) = \delta^*(\varphi, w) \vee \delta^*(\psi, w),$$

$$\delta^*(\top, w) = \top,$$

$$\delta^*(\bot, w) = \bot .$$

Если же булева функция – это одна переменная q, то $w = \varepsilon$ или $w = ax$, где $a \in \Sigma$:

$$\delta^*(q, \varepsilon) = q,$$

$$\delta^*(q, ax) = \delta^*(\delta(q, a), x).$$

Скажем, что G принимает слово w, если $(\delta^*(s, w))(F) = 1$.

Пример

Пусть $G = (\{a, b, c\}, \{q_0, q_1, q_2\}, q_0, \{q_1\}, \delta)$, функция перехода задана таблицей ниже:

δ	a	b	c
q_0	$q_0 \vee q_1$	$\bot$	$q_1 \wedge q_2$
q_1	$\bot$	$q_0 \vee q_1$	q_1
q_2	$\top$	q_2	$(q_1 \vee q_2) \wedge q_0$

Рассмотрим работу этого автомата на слове $abbcb$:

$$
\begin{aligned}
&\delta^*(q_0, abbcb) = \\
&= \delta^*(\delta(q_0, a), bbcb) = \\
&= \delta^*(q_0 \vee q_1, bbcb) = \\
&= \delta^*(q_0, bbcb) \vee \delta^*(q_1, bbcb) = \\
&= \delta^*(\delta(q_0, b), bcb) \vee \delta^*(\delta(q_1, b), bcb) = \\
&= \delta^*(\bot, bcb) \vee \delta^*(q_0 \vee q_1, bcb) = \\
&= \bot \vee (\delta^*(q_0, bcb) \vee \delta^*(q_1, bcb)) = \\
&= \bot \vee (\delta^*(\delta(q_0, b), cb) \vee \delta^*(\delta(q_1, b), cb)) = \\
&= \bot \vee (\delta^*(\bot, cb) \vee \delta^*(q_0 \vee q_1, cb)) = \\
&= \bot \vee (\bot \vee (\delta^*(q_0, cb) \vee \delta^*(q_1, cb))) = \\
&= \bot \vee (\bot \vee (\delta^*(\delta(q_0, c), b) \vee \delta^*(\delta(q_1, c), b))) = \\
&= \bot \vee (\bot \vee (\delta^*(q_1 \wedge q_2, b) \vee \delta^*(q_1, b))) = \\
&= \bot \vee (\bot \vee ((\delta^*(q_1, b) \wedge \delta^*(q_2, b)) \vee \delta^*(\delta(q_1, b), \varepsilon))) = \\
&= \bot \vee (\bot \vee ((\delta^*(\delta(q_1, b), \varepsilon) \wedge \delta^*(\delta(q_2, b), \varepsilon)) \vee \delta^*(q_0 \vee q_1, \varepsilon))) = \\
&= \bot \vee (\bot \vee ((\delta^*(q_0 \vee q_1, \varepsilon) \wedge \delta^*(q_2, \varepsilon)) \vee (\delta^*(q_0, \varepsilon) \vee \delta^*(q_1, \varepsilon)))) = \\
&= \bot \vee (\bot \vee (((\delta^*(q_0, \varepsilon) \vee \delta^*(q_1, \varepsilon)) \wedge q_2) \vee (q_0 \vee q_1))) = \\
&= \bot \vee (\bot \vee (((q_0 \vee q_1) \wedge q_2) \vee (q_0 \vee q_1))) .
\end{aligned}
$$

Теперь вычислим $(\delta^*(q_0, abbcb))(F)$:

$$
\begin{aligned}
[\bot \vee (\bot \vee (((q_0 \vee q_1) \wedge q_2) \vee (q_0 \vee q_1)))]\,(q_0 = 0, q_1 = 1, q_2 = 0) = \\
= ((0 \vee 1) \wedge 0) \vee (0 \vee 1) = 1,
\end{aligned}
$$

так что G принимает слово $abbcb$.

Эквивалентность ДКА

Мы не будем здесь строго доказывать следующий результат: АКА принимают регулярные языки и только их. В самом деле, по АКА можно построить эквивалентный ДКА, состояния в котором – это классы эквивалентных положительных функций (функции эквивалентны,

если на всех наборах аргументов они выдают одинаковые значения, например q и $q \wedge q$). Отметим, что переход к ДКА – это дважды экспоненциальное увеличение числа состояний.

Заключение

В учебно-методическом пособии исследуются структурные свойства регулярных языков, необходимые и достаточные условия регулярности. Отдельное внимание уделяется теореме Майхилла–Нероуда, приведены начальные сведения о синтаксическом моноиде языка. Приводятся также некоторые автоматные обобщения стандартных моделей вычислений для регулярных языков. Рассмотренные конструкции снабжены примерами. Литература в конце пособия рекомендуется для углубления и расширения знаний в контексте теории регулярных языков, а также для знакомства с приложениями.

Список литературы

1. *Хопкрофт Д., Мотвани Р., Ульман Д.* Введение в теорию автоматов, языков и вычислений / пер. с англ. 2-е изд. Москва : Издательский дом «Вильямс», 2002. 528 с.
2. *Рубцов А.А.* Заметки и задачи о регулярных языках и конечных автоматах. Москва : МФТИ, 2019. 112 с.
3. *Голубенко Д.А., Саватеев Ю.В.* Языки, автоматы и грамматики. Москва : МЦНМО, 2023. 304 с.
4. *Hopcroft J.E., Ullman J.D.* Introduction to Automata Theory, Languages, and Computation. Addison-Wesley, 1979. 418 p.
5. *Kozen D.* Automata and Computability. New York : Springer, 1997. 400 p.
6. *Lewis H.R., Papadimitriou C.H.* Elements of the Theory of Computation. Prentice-Hall, 1998. 361 p.
7. *Linz P.* An Introduction to Formal Languages and Automata. Jones and Bartlett Publishers, 2006. 415 p.
8. *Shallit J.O.* A Second Course in Formal Languages and Automata Theory. Cambridge University Press, 2009. 240 p.
9. *Sipser M.* Introduction to the Theory of Computation. Cengage Learning. 3rd edition. 2012. 504 p.