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: основные конструкции

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: основные конструкции : учебно-методическое 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. – 48 с.

Содержит ряд теоретических сведений, входящих в состав курса теории алгоритмов и моделей вычислений для второкурсников физтех-школы прикладной математики и информатики МФТИ. В настоящем пособии разобраны основные конструкции, с помощью которых определяются и распознаются регулярные языки, доказана теорема Клини. 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».

УДК 519.713(075)
ББК 22.18я73

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

Содержание

Введение

Настоящее пособие написано на основе лекций по курсу “Теория алгоритмов и моделей вычислений”, читаемых автором студентам второго курса ФПМИ МФТИ. Основным объектом для изучения являются специфического вида алгоритмы – конечные автоматы, которые берут на вход текст, читают его слева направо один раз и выдают в ответ один бит информации: 1 или 0, “хороший” пришёл текст или “плохой”.

0 Слова и языки

Абстрактные тексты – это строки конечной длины, составленные из символов наперёд заданного множества. Начнём поэтому с изучения строк и операций над ними. Нам придётся использовать некоторые термины и понятия, восходящие к теории множеств: функция, упорядоченная пара, кортеж и т.п. В большинстве своём эти понятия интуитивно корректны – в том смысле, что интуиция достаточно точно описывает все необходимые свойства объекта. В этом разделе некоторые определения и доказательства нарочно будут сделаны излишне подробными и формальными – не ради определений и доказательств, а ради иллюстрации техники и принципиальной возможности доказывать некоторые леммы строгим образом. Натуральные числа у нас будут начинаться с нуля.

0.1 Слова

Алфавит – это конечное *множество*. Типичные алфавиты: латинский $\{a, \ldots, z\}$, унарный $\{0\}$, бинарный $\{0, 1\}$ и т.д. Допускается также пустой алфавит $\varnothing$.

Для дальнейших определений зафиксируем алфавит и обозначим его символом Σ. Для удобства (и вообще возможности нормально работать с алфавитом) предположим (здесь и всюду далее), что фигурные скобки $\{, \}$, знаки препинания и прочие служебные символы не входят в Σ. Также в Σ не входит символ ε, который зарезервирован для отдельного объекта. *Элементы* алфавита называются **буквами** или **символами**.

Пустой строкой или **пустым словом** назовём абстрактный объект ε. Пустое слово не является буквой. Интуитивно пустая строка не

содержит символов, её приписывание к другим строкам ничего не меняет.

Множество всех слов или **строк** над алфавитом Σ обозначается Σ^* и определяется следующей индуктивной процедурой.

Пусть $S_0 = \{\varepsilon\}$. Пусть теперь для каждого натурального $i > 0$ множество S_i определено так: S_i – это множество всех *упорядоченных пар* (a, x), где $a \in \Sigma$, $x \in S_{i-1}$. Множество всех слов:

$$\Sigma^* = \bigcup_{i=0}^{\infty} S_i.$$

Элементы множества всех слов называются, естественно, **словами** или **строками** (над алфавитом Σ).

Для *упрощения записи* строки записываются сплошной последовательностью символов, при этом ε в непустой строке обычно опускается. То есть вместо $(\mathsf{e}, (\mathsf{x}, (\mathsf{a}, (\mathsf{m}, (\mathsf{p}, (\mathsf{l}, (\mathsf{e}, \varepsilon))))))))$ пишется просто $\mathsf{example}$.

Поясним ещё раз смысл определения: поскольку множество строк есть объединение отдельных множеств S_i, то каждая строка лежит в некотором S_i, так что строками являются лишь те объекты, которые можно получить из *базового* случая ε путем *конечного* применения "правила" упорядоченной пары. Это в том числе означает, что любое слово "получено" из пустого слова за конечное число шагов (конечно же, это число равно длине слова). Такая структура позволяет далее записывать индуктивные определения: каждая строка есть либо ε, либо $(a, x) \in S_j$ для некоторой строки $x \in S_{j-1}$.

Ещё раз явно пропишем, Σ^* – это множество всех конечных строк над алфавитом Σ, при этом Σ^* само по себе является бесконечным множеством (если только Σ непусто, если же $\Sigma = \varnothing$, то $\Sigma^* = \{\varepsilon\}$).

Конкатенация

Фундаментальной операцией над двумя строками является конкатенация. **Конкатенацией** двух строк x и y называется строка $z = x \cdot y$, которая определяется рекурсивно:

$$x \cdot y = \begin{cases} y, & \text{если } x = \varepsilon, \\ (a, w \cdot y), & \text{если } x = (a, w). \end{cases}$$

Отметим структуру определения: поскольку каждая строка по определению лежит в некотором S_j, то каждая строка удовлетворяет по крайней мере одному из двух случаев "если". Понятно, что она удовлетворяет ровно одному из двух случаев. Поскольку натуральные индексы j у множеств S_j уменьшаются на один, то индуктивное определение за конечное число шагов "дойдёт" до базового случая пустого слова.

Неформально, конкатенация x и y – это строка, полученная «приписыванием» слова y справа к слову x.

Конкатенация с пустой строкой слева не меняет строку по определению. Докажем, что конкатенация с пустой строкой справа также не меняет строку. Сперва для примера развернём рекурсию на

$$ab \cdot \varepsilon = (a,(b,\varepsilon)) \cdot \varepsilon = (a,(b,\varepsilon) \cdot \varepsilon) = (a,(b,\varepsilon \cdot \varepsilon)) = (a,(b,\varepsilon)) = ab.$$

Хорошо, если этот пример ясен, тем более хорошо, если ясно, как обобщить этот пример на доказательство общего факта. Тем не менее всегда необходимо помнить, что частный пример может иллюстрировать мысль или технику, но не является доказательством. Приведём далее доказательство.

Лемма. Для любого слова w выполняется $w \cdot \varepsilon = w$.

Доказательство мы будем проводить по индукции. Будем пользоваться структурной индукцией – индукцией по длине рекурсивного определения строки, т.е. по количеству шагов, которое требуется для получения слова из пустого слова (фактически, это то же самое, что длина слова, но определения длины у нас ещё не было). В наших определениях строки структурная индукция выглядит так: пусть для некоторого свойства P одновременно верны два утверждения:

1) свойство P выполнено для пустой строки,

2) верна импликация (если свойство P выполнено для строки w, то оно выполнено для строки (a,w) (здесь $a \in \Sigma$)),

то тогда свойство P выполнено для всех строк.

Применим теперь структурную индукцию для доказательства леммы. Свойство $P(w)$ – это свойство $w \cdot \varepsilon = w$.

1) $P(\varepsilon)$ есть утверждение $\varepsilon \cdot \varepsilon = \varepsilon$. Оно выполнено по определению конкатенации (первый случай в определении).

2) Пусть $w \cdot \varepsilon = w$, тогда $aw \cdot \varepsilon \overset{1}{=} (a, w) \cdot \varepsilon \overset{2}{=} (a, w \cdot \varepsilon) \overset{3}{=} (a, w) \overset{4}{=} aw$. Здесь первое и четвёртое равенства – это упрощённая запись строки без упорядоченной пары, второе равенство выполнено по определению конкатенации, а третье – по предположению индукции.

Заключаем, что для всех w верно $w \cdot \varepsilon = w$.

Длиной строки x называется число $|x| \in \mathbb{N}$:

$$|x| = \begin{cases} 0, & \text{если } x = \varepsilon, \\ 1 + |w|, & \text{если } x = (a, w). \end{cases}$$

Естественно, длина строки – это количество символов в ней. Кроме того, длина строки – это ещё один удобный (и, возможно, более привычный) способ проводить индуктивные доказательства.

Лемма. Конкатенация ассоциативна, т.е. для любых слов w, x, y выполняется

$$(w \cdot x) \cdot y = w \cdot (x \cdot y).$$

Докажем лемму по индукции по длине строки (только по слову w, индукция по другим словам не требуется).

База индукции: если $|w| = 0$, то есть $w = \varepsilon$, то

$$(w \cdot x) \cdot y \overset{1}{=} (\varepsilon \cdot x) \cdot y \overset{2}{=} x \cdot y \overset{3}{=} \varepsilon \cdot (x \cdot y) \overset{4}{=} w \cdot (x \cdot y).$$

Первое равенство здесь выполняется, потому что $w = \varepsilon$, второе по определению конкатенации строк ε и x, третье по определению конкатенации строк ε и $x \cdot y$, четвёртое – потому что $w = \varepsilon$.

Пусть теперь $(z \cdot x) \cdot y = z \cdot (x \cdot y)$ для всех $|z| < |w|$. Пусть $w = (a, z) = az$, тогда

$$\begin{aligned} (w \cdot x) \cdot y \overset{1}{=} ((a, z) \cdot x) \cdot y \overset{2}{=} (a, z \cdot x) \cdot y \overset{3}{=} (a, (z \cdot x) \cdot y) \overset{4}{=} \\ \overset{4}{=} (a, z \cdot (x \cdot y)) \overset{5}{=} (a, z) \cdot (x \cdot y) \overset{6}{=} w \cdot (x \cdot y). \end{aligned}$$

Первое равенство выполняется, потому что $w = (a, z)$. Второе – по определению конкатенации. Третье – по определению конкатенации (на этот раз внешней конкатенации со строкой y). Четвёртое верно вследствие индуктивной гипотезы. Здесь мы неявно воспользовались тем, что из $w = (a, z)$ следует, что $|z| < |w|$. Пятое снова верно по определению конкатенации, а шестое – потому что $w = (a, z)$.

Лемма об ассоциативности конкатенации доказана.

Несложно также показать, что, $a{\cdot}x = (a, x)$ для символа a и строки x (здесь не нужна индукция, хватит двух применений определения). Всё это вместе позволяет записывать конкатенации строк сплошной записью, не заботясь о расстановке скобок и опуская символ $\cdot$ для упрощения записи. Также будем отождествлять символ алфавита a и строку (a, ε).

Мы показали, что ε действительно является "единицей по умножению" – нейтральным элементом для конкатенации. По аналогии с определением натуральной степени числа определим теперь w^i для слова w и натурального i:

$$w^i = \begin{cases} \varepsilon, & \text{если } i = 0, \\ ww^{i-1}, & \text{если } i > 0. \end{cases}$$

Слово y называется **подстрокой** или **подсловом** слова x, если существуют такие слова z_1 и z_2, что $x = z_1 y z_2$. Слово y называется **префиксом** слова x, если существует такое слово z, что $x = yz$. Слово y называется **суффиксом** слова x, если существует такое слово z, что $x = zy$.

Слово y называется **подпоследовательностью** слова x, если выполняется хотя бы одно из условий:

1) $y = \varepsilon$,

2) $x = aw$, где a – буква, а y подпоследовательность w,

3) $x = aw$, $y = au$, где a – буква, а u подпоследовательность w.

Последнее определение немного искусственное, однако смысл его весьма прозрачен особенно в сравнении с подсловом. Подслово получено из слова путём удаления некоторых символов в начале и некоторых в конце, то есть подслово – "непрерывный кусок" исходного слова. Подпоследовательность получена из слова путём удаления каких-то символов где-то в слове. Например, для слова ababa слово ba является подсловом и подпоследовательностью (и суффиксом заодно), а слово aaa является подпоследовательностью, но не является подсловом.

Префиксы, суффиксы, подслова и подпоследовательности слова называются **собственными**, если они не совпадают со всем словом. Пустое слово ε является префиксом и суффиксом (а значит, подсловом и подпоследовательностью) любого слова.

0.2 Языки

Язык (**формальный язык**) над алфавитом Σ – это подмножество множества всех слов Σ^*. К примеру, над алфавитом $\{0,1\}$ языками являются $\varnothing$, $\{\varepsilon\}$, $\{0\}$, Σ, Σ^*, множество всех бинарных строк, в которых пять единиц и чётное число нулей и т.д.

Отдельно отметим:

1) $\varnothing$ является языком над любым алфавитом (в том числе пустым алфавитом). Это пустой язык, в нём нет ни одного слова.

2) $\{\varepsilon\}$ также является языком над любым алфавитом (в том числе пустым алфавитом). Это непустой язык, он состоит из одного слова, это пустое слово. При этом ε – это не язык, это одно слово, которое не содержит букв.

Прежде всего, языки – это множества, так что любые теоретико-множественные операции на языках определены естественным образом. Операцию объединения $\cup$ иногда будем обозначать плюсом $+$, а операцию разности множеств $\setminus$ минусом $-$ (с этими обозначениями нужно работать аккуратно, поскольку $A + A - A \neq A - A + A$ для непустого A). Для языка A его дополнение до множества всех слов Σ^* будем обозначать чертой сверху, т.е. $\overline{A} = \Sigma^* - A$.

Поскольку языки – это специфические множества, на них можно ввести ещё несколько полезных операций, перенеся их с операций над строками.

Конкатенация двух языков L и M – это язык

$$L \cdot M = \{xy \in \Sigma^* \mid x \in L, y \in M\}.$$

То есть мы берём каждое слово из L и конкатенируем его с каждым словом из M, все получившиеся слова объединяем в один язык.

Мы здесь (и далее в подобных ситуациях) неявно предполагаем, что L и M определены над одним алфавитом Σ. Это всегда можно предполагать без потери общности, так как для языка L над Σ_L и языка M над Σ_M можно взять общий алфавит $\Sigma = \Sigma_L + \Sigma_M$.

Степень языка определяется аналогично степени строки:

$$L^i = \begin{cases} \{\varepsilon\}, & \text{если } i = 0, \\ L \cdot L^{i-1}, & \text{если } i > 0. \end{cases}$$

Как и в случае строк, мы иногда будем опускать символ конкатенации $\cdot$ для удобства записи.

Итерация языка L – это язык

$$L^* = \bigcup_{i=0}^{\infty} L^i.$$

Неформально, L^* принадлежат всевозможные конкатенации из нуля или более слов из L. Кроме того, полезным бывает рекурсивное определение: L^* – это минимальный по включению язык из Σ^*, для которого $w \in L^*$ тогда и только тогда, когда или $w = \varepsilon$, или $w = xy$ для $x \in L$, $y \in L^*$.

Заметим, что Σ^* – это множество всех слов, если считать Σ языком над Σ (с отождествлением букв и строк длины один), а звёздочку – итерацией. Это показывает, что обозначение Σ^*, используемое нами ранее без пояснений, корректно в смысле определения итерации.

Следующие свойства введённых операций тривиально следуют из определений:

1) $\varnothing \cdot L = L \cdot \varnothing = \varnothing$,

2) $\{\varepsilon\} \cdot L = L \cdot \{\varepsilon\} = L$,

3) $\varnothing^* = \{\varepsilon\}^* = \{\varepsilon\}$.

Иногда отдельно выделяют операцию "плюс": $L^+ = L \cdot L^*$. Понятно, что $L^* = L^+ + \{\varepsilon\}$.

1 Регулярные языки

Зафиксируем алфавит Σ и рассмотрим множество всех языков над Σ – это так называемый *класс* языков $\mathcal{ALL}_\Sigma = 2^{\Sigma^*}$. Из этого класса мы выделим подкласс – множество языков, которое будем изучать более подробно.

Класс регулярных языков (над алфавитом Σ) $\mathcal{REG}_\Sigma$ – это **минимальное по включению подмножество** 2^{Σ^*}, обладающее следующими пятью свойствами:

1) $\varnothing \in \mathcal{REG}_\Sigma$, т.е. пустой язык принадлежит классу регулярных языков,

2) $\forall a \in \Sigma \quad \{a\} \in \mathcal{REG}_\Sigma$, т.е. язык, состоящий из одного однобуквенного слова, принадлежит классу регулярных языков,

3) $\forall X, Y \in \mathcal{REG} \quad X + Y \in \mathcal{REG}_\Sigma$, т.е. объединение двух языков из этого класса принадлежит этому классу,

4) $\forall X, Y \in \mathcal{REG} \quad XY \in \mathcal{REG}_\Sigma$, т.е. конкатенация двух языков из этого класса принадлежит этому классу,

5) $\forall X \in \mathcal{REG} \quad X^* \in \mathcal{REG}_\Sigma$, т.е. итерация языка из этого класса принадлежит этому классу.

Индекс Σ можно опустить, если он ясен из контекста. Языки, принадлежащие $\mathcal{REG}$, называются **регулярными языками**.

Можно дать также альтернативное определение, явно описывающее "минимальность по включению". Пусть (проследите за количеством скобок):

$$\mathcal{S}_0 = \{\varnothing\} + \bigcup_{a \in \Sigma} \{\{a\}\}.$$

Пусть далее для любого $i > 0$:

$$\mathcal{S}_i = \mathcal{S}_{i-1} + \{X + Y \mid X, Y \in \mathcal{S}_{i-1}\} + \{XY \mid X, Y \in \mathcal{S}_{i-1}\} + \{X^* \mid X \in \mathcal{S}_{i-1}\}.$$

Тогда $\mathcal{REG} = \bigcup_{i=0}^{\infty} \mathcal{S}_i$. То есть класс регулярных языков строится из счётной "лестницы" классов – на каждом следующем уровне находятся языки с предыдущего уровня и языки, полученные одним применением объединения, конкатенации или итерации к языкам с предыдущего уровня.

Из определения следует, что $\{\varepsilon\}$ регулярен как итерация $\varnothing$, язык, состоящий из одного непустого слова, регулярен как конкатенация языков, содержащих соответствующие однобуквенные слова. Далее, любой конечный язык регулярен, поскольку является конечным объединением языков, содержащих одно слово. Язык всех слов Σ^* регулярен, поскольку является итерацией регулярного языка Σ (алфавит конечен, а значит, регулярен).

Каждый регулярный язык получен конечным применением операций объединения, конкатенации и итерации к базовым случаям – пустому множеству и языку из одного однобуквенного слова. Типично, регулярные языки описываются с помощью какой-то компактной нотации, которая является "инструкцией" по тому, как получить регулярный язык из его определяющих свойств 1) – 5). Мы рассмотрим две такие инструкции: деревья разбора и регулярные выражения.

1.1 Деревья

Один из способов записи регулярного языка – его дерево разбора, которое определяется опять-таки рекурсивно. Деревом разбора является размеченное корневое дерево (т.е. дерево с выделенным корнем и метками на вершинах) одного из следующих видов:

1) одна вершина (корень и при этом лист) с меткой $\varnothing$,

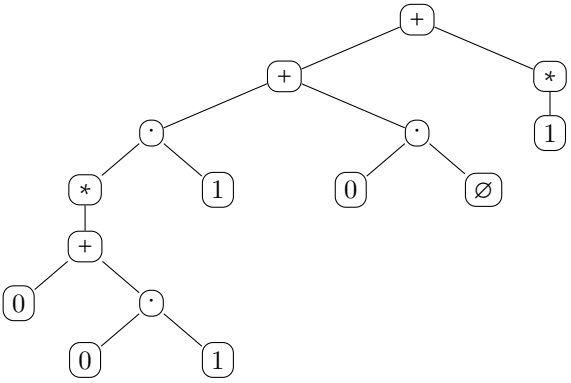
2) одна вершина (корень и при этом лист) с меткой a, где $a \in \Sigma$,

3) корневое дерево с меткой + и двумя потомками, каждый из которых при этом является деревом разбора,

4) корневое дерево с меткой $\cdot$ и двумя потомками, каждый из которых при этом является деревом разбора,

5) корневое дерево с меткой $*$ и одним потомком, являющимся деревом разбора.

Приведём пример: для языка $((\{0\} + \{0\} \cdot \{1\})^* \cdot \{1\} + \{0\} \cdot \varnothing) + \{1\}^*$ его деревом разбора является



1.2 Регулярные выражения

Более простой инструкцией для описания регулярного языка служит регулярное выражение. Фиксируем, как и ранее, алфавит Σ.

Языком регулярных выражений над Σ называется множество

REG_Σ, определяемое как наименьший по включению язык *над другим алфавитом* $\Sigma + \{(,), \cdot, +, ^*, \varnothing\}$, который

1) содержит слово $\varnothing$,

2) содержит слово a для каждой буквы $a \in \Sigma$,

3) вместе с любыми двумя словами x и y содержит слово $(x + y)$,

4) вместе с любыми двумя словами x и y содержит слово $(x \cdot y)$,

5) вместе с любым словом x содержит слово x^*.

Слова из REG_Σ называются **регулярными выражениями** над Σ. Каждому регулярному выражению r соответствует язык над Σ, определяемый этим выражением, который мы обозначим $\mathsf{L}(r)$. Определение этого объекта достаточно естественно:

1) если $r = \varnothing$, то $\mathsf{L}(r) = \varnothing$,

2) если $r = a$ для $a \in \Sigma$, то $\mathsf{L}(r) = \{a\}$,

3) если $r = (x + y)$ для некоторых регулярных выражений x и y, то $\mathsf{L}(r) = \mathsf{L}(x) + \mathsf{L}(y)$,

4) если $r = (x \cdot y)$ для некоторых регулярных выражений x и y, то $\mathsf{L}(r) = \mathsf{L}(x) \cdot \mathsf{L}(y)$,

5) если $r = x^*$ для регулярного выражения x, то $\mathsf{L}(r) = \mathsf{L}(x)^*$.

При построении регулярного выражения ненужные скобки чаще всего опускают. Приоритет операций следующий: итерация, конкатенация, объединение (аналогично приоритету степень, умножение, сложение). Кроме того, мы будем использовать символ ε вместо $\varnothing^*$.

Пример: регулярное выражение (над алфавитом $\{0, 1\}$)

$$(((0 \cdot (0 \cdot 1)^*) + ((\varnothing^* + 0) + 1)^*) + (((1 \cdot 1^*) \cdot 1) \cdot 1))$$

записывается упрощённо в виде

$$0(01)^* + (\varepsilon + 0 + 1)^* + 11^*11$$

и задаёт регулярный язык

$$\big(\{0\} \cdot (\{0\} \cdot \{1\})^*\big) + (\{\varepsilon\} + \{0\} + \{1\})^* + (\{1\} \cdot \{1\}^* \cdot \{1\} \cdot \{1\})\,.$$

Подвыражение B регулярного выражения A – это такое регулярное выражение, что его дерево разбора является поддеревом дерева разбора A.

Помимо прочего, пример выше $0(01)^* + (\varepsilon + 0 + 1)^* + 11^*11$ задаёт язык всех слов над алфавитом $\{0, 1\}$, который описывается также и регулярным выражением $(0+1)^*$. Это означает, что для регулярного языка может существовать несколько способов быть порождённым по свойствам 1) – 5) и соответственно несколько (и даже бесконечно много) регулярных выражений, описывающих его.

Два регулярных выражения r и s называются **эквивалентными**, если они задают один и тот же язык, т.е. $\mathsf{L}(r) = \mathsf{L}(s)$.

Отметим ещё раз, что порядок операций важен: регулярное выражение 000^* задаёт язык всех строк над $\{0\}$, содержащих по крайней мере два нуля, а выражение $(000)^*$ – язык всех строк над $\{0\}$, содержащих кратное трём число нулей.

1.3 Не все языки регулярны

Напоследок отметим, что не все языки регулярны, т.е. (если $\Sigma \neq \varnothing$) $\mathcal{ALL}_\Sigma \neq \mathcal{REG}_\Sigma$. Наиболее простой способ это заметить – соображения мощности множеств.

Каждое регулярное выражение – это слово над конечным алфавитом (исходный алфавит Σ и служебные символы). Слов каждой фиксированной длины конечное число, а всего слов счётное число, так что регулярных выражений не более чем счётное число. Соответственно и регулярных языков не более чем счётное число.

Множество всех вообще языков несчётно, поскольку это множество всех подмножеств Σ^*, а само Σ^* счётно (если только $\Sigma \neq \varnothing$).

Значит, нерегулярные языки существуют, и, более того, в некотором смысле “большинство” языков нерегулярны.

2 Детерминированные конечные автоматы

Перейдём теперь к рассмотрению отдельно выделенных моделей вычислений – специфического вида алгоритмов. Мы будем рассматривать различные виды конечных автоматов – абстрактных вычислительных устройств, умеющих по входному слову выдавать бинарный ответ.

2.1 Определение ДКА

Сосредоточим сперва своё внимание на *детерминированном конечном автомате* (ДКА). Общая концепция такова: ДКА имеет конечное множество *состояний* (о состояниях можно думать как о конечной оперативной памяти автомата) и, получая на вход слово w, читает буквы этого слова одну за другой слева направо. В результате прочтения каждой следующей буквы ДКА меняет своё состояние (по правилам именно этого конкретного ДКА, у разных ДКА разные правила). После прочтения слова целиком ДКА завершает работу в каком-то конкретном состоянии. Некоторые из состояний ДКА помечены как "хорошие", прочие помечены как "плохие". Если ДКА закончил работу на слове w в "плохом" состоянии, то это слово "плохое", а если в "хорошем", то "хорошее". Это позволяет ДКА раскладывать все слова над данным алфавитом в две кучки и, таким образом, выделять из множества всех слов некоторое "хорошее" подмножество – язык.

Перейдём теперь к формальным определениям: **детерминированный конечный автомат** (ДКА) A – это *упорядоченная пятёрка (кортеж)*:

$$A = (\Sigma, Q, s, F, \delta), \quad \text{где}$$

Σ – это конечное множество, называемое **алфавитом** или **входным алфавитом** ДКА. Здесь нет неожиданностей, это алфавит, в котором работает ДКА, слова, распознаваемые этим ДКА, состоят из букв этого алфавита. Говорят, что A – это автомат над Σ.

Q – это конечное множество, называемое множеством **состояний**. Его элементы – состояния ДКА. Мы считаем, что $\Sigma \cap Q = \varnothing$ для удобства. Ещё раз подчеркнём обязательность конечности Q.

$s \in Q$ – это **начальное состояние** ДКА. В этом состоянии ДКА начинает работу. Помимо прочего, это означает, что Q непустое множество.

$F \subseteq Q$ – это **множество принимающих состояний**. Если по прочтении слова ДКА остановился в состоянии из F, слово принимается, а если нет, то нет. F может быть пустым, может совпадать с Q и может быть чем угодно "посередине". Начальное состояние s может быть или не быть принимающим.

$\delta : Q \times \Sigma \to Q$ – это всюду определённая **функция переходов**. Она определяет правило, по которому ДКА переходит между состояниями.

Само по себе определение ДКА не содержит никаких указаний на то, как ДКА работает, или на то, как мы интерпретируем результат его работы (и даже что вообще является результатом). В этом смысле определение абстрактной вычислительной машины недостаточно для задания модели вычислений. Необходимо также описать, что именно мы считаем вычислением. Сперва сделаем это неформально.

Пусть ДКА A на вход было подано слово w. A начинает работу в состоянии s и читает первый символ слова w (если таковой существует). Пусть этот символ есть a, тогда A переходит в состояние $\delta(s, a)$. Далее, A читает следующий символ и так далее до тех пор, пока последний символ слова w не будет прочитан. После того, как непрочитанных символов не осталось (если вход был пустым словом, то это происходит сразу же), A останавливается в некотором состоянии q. Если $q \in F$, то говорят, что A *принимает* слово w, если $q \notin F$ – *отвергает* слово w.

Теперь определим работу ДКА формально. Расширим функцию δ, позволив ей работать со строками (а не только с отдельными буквами). Для этого введём для ДКА A функцию $\delta^* : Q \times \Sigma^* \to Q$ по правилу:

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

Скажем, что ДКА $A = (\Sigma, Q, s, F, \delta)$ **принимает** слово w, если $\delta^*(s, w) \in F$ (в противном случае A **отвергает** слово w). Множество всех слов, которые A принимает, – это язык, **принимаемый** или **распознаваемый** ДКА A. Мы обозначим этот язык $\mathsf{L}(A)$.

Конкретный ДКА A распознаёт какой-то конкретный язык $\mathsf{L}(A)$, с другой стороны, для одного языка L может существовать множество ДКА, распознающих его (ровно как может не существовать ни единого). Два ДКА A_1 и A_2 называются **эквивалентными**, если они распознают один и тот же язык, т.е. $\mathsf{L}(A_1) = \mathsf{L}(A_2)$.

Понятно, что $\delta^*(q, a) = \delta(q, a)$ для любого символа a. Несложно

также доказать, что для любого состояния q и любых слов x, y выполнено $\delta^*(q, xy) = \delta^*(\delta^*(q, x), y)$: для $x = \varepsilon$ это выполняется по определению, по индукции затем для $x = az$ получаем

$$\delta^*(\delta^*(q,x),y) = \delta^*(\delta^*(q,az),y) = \delta^*(\delta^*(\delta(q,a),z),y) \overset{!!}{=} \delta^*(\delta(q,a),zy) = \\ = \delta^*(q,azy) = \delta^*(q,xy),$$

здесь $\overset{!!}{=}$ выполняется по предположению индукции.

Пусть дан ДКА над Σ. Его состояние q называется **достижимым**, если существует слово $w \in \Sigma^*$ такое, что $\delta^*(s, w) = q$, т.е. в состояние q можно попасть из s с помощью некоторой последовательности символов. В противном случае состояние q называется **недостижимым**. Недостижимые состояния "лишние" в следующем смысле: пусть $A = (\Sigma, Q, s, F, \delta)$ – некоторый ДКА, пусть Q_u – множество его недостижимых состояний. Тогда ДКА $A' = (\Sigma, Q - Q_u, s, F - Q_u, \delta')$, где $\delta'(q, a) = \delta(q, a)$ на любых аргументах, эквивалентен A. Докажем сперва, что определение корректно. Во-первых, s – достижимо, поскольку $\delta^*(s, \varepsilon) = s$, во-вторых, если q достижимо, то и $\delta(q, a)$ достижимо для любого $a \in \Sigma$, так что значение $\delta'(q, a)$ существует. Далее, если слово принимается A', то оно принимается и A, поскольку δ расширяет δ'. Если слово принимается A, то последовательность состояний, в которых находится A, состоит только из достижимых состояний (они есть $\delta^*(s, v)$ для всех v – префиксов принимаемого слова w), так что такая же последовательность существует и в A'. Поскольку $\delta' = \delta$ на всех аргументах, переходы будут происходить в том же порядке и A' примет слово.

Иногда в литературе функции перехода δ в ДКА позволяется по определению быть частичной, т.е. быть неопределённой на некоторых парах (состояние, буква). В случае попытки перехода по неопределённой паре такой автомат тут же останавливается и автоматически отвергает входное слово. Мы не будем считать такие автоматы ДКА. Несложно понять, что для любого автомата с, возможно, частичной функцией переходов существует ДКА, который распознаёт тот же самый язык. Действительно, пусть автомат (это автомат, но это не ДКА!) $B = (\Sigma, Q, s, F, \delta)$ имеет частичную функцию перехода, тогда определим ДКА:

$$B' = \left(\Sigma, Q \cup \{q_{\frown}\}, s, F, \delta'\right),$$

где $q_{\frown}$ – дополнительное состояние, в которое будут "сливаться" все недостающие переходы:

$$\delta'(q,a) = \begin{cases} \delta(q,a), & \text{если } \delta(q,a) \text{ определено,} \\ q_{\frown}, & \text{если } \delta(q,a) \text{ не определено.} \end{cases}$$

В том числе $\delta(q_{\text{☹}}, a) = q_{\text{☹}}$ для любой буквы a.

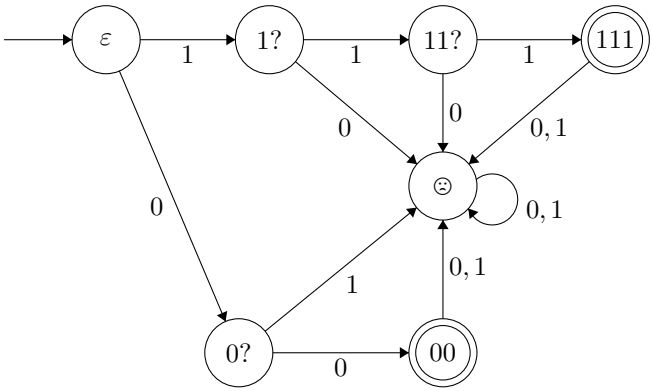
Пусть ДКА $A = (\Sigma, Q, s, F, \delta)$ принимает язык L. Тогда для любого $x \in L$ выполнено $\delta^*(s, x) \in F$. Кроме того, для любого $y \notin L$ выполнено $\delta^*(s, y) \in Q - F$. Сделаем в ДКА все принимающие состояния непринимающими, а непринимающие – принимающими, т.е. определим автомат $\bar{A} = (\Sigma, Q, s, Q - F, \delta)$. ДКА $\bar{A}$ принимает все слова $y \notin L$ и отвергает все слова $x \in L$, а значит, $\bar{A}$ принимает язык $\overline{L}$.

Примеры

Для наглядности ДКА можно представлять в виде размеченного орграфа с (возможно) петлями и некоторыми дополнительными пометками. Вершины орграфа – это состояния ДКА, из каждой вершины выходит $|\Sigma|$ рёбер с метками – символами алфавита. Стандартными обозначениями для начального состояния является входящая из ниоткуда стрелка, а для принимающих состояний – двойной кружок вместо обычного. Если $\delta(q_1, a) = q_2$ для нескольких символов a, для них обычно рисуют одну стрелку с несколькими пометками сразу.

1. Пусть сперва $A_\varnothing = (\Sigma, Q, s, \varnothing, \delta)$. Вне зависимости от алфавита, состояний и функции перехода условие $F = \varnothing$ обеспечивает, что ни на каком слове w не выполнится условие $\delta^*(s, w) \in F$, так что $A_\varnothing$ не принимает никакое слово и распознаёт, соответственно, пустой язык.

2. Рассмотрим более содержательный пример. ДКА A_{2w} над алфавитом $\{0, 1\}$ зададим графически.



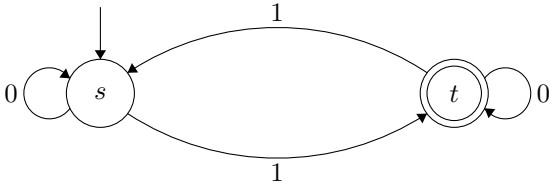
Проверьте, что этот ДКА распознаёт язык из двух слов $\{111, 00\}$. Заметим, что названия состояний (метки внутри кружков) в общем случае произвольны и не обязаны отражать какие-то свойства этих состояний (хотя это и удобно – примерно так же, как и осмысленные названия переменных при написании программного кода).

3. Последний пример, пусть

$$A_{odd} = (\{0,1\}, \{s,t\}, s, \{t\}, \delta),$$

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

Этот ДКА изображается так:



Сперва приведём пример того, как A_{odd} работает. На слове 110100 происходит следующее:

$$\begin{aligned}
&\delta^*(s, 110100) = \delta^*(\delta(s,1), 10100) = \\
&= \delta^*(t, 10100) = \delta^*(\delta(t,1), 0100) = \\
&= \delta^*(s, 0100) = \delta^*(\delta(s,0), 100) = \\
&= \delta^*(s, 100) = \delta^*(\delta(s,1), 00) = \\
&= \delta^*(t, 00) = \delta^*(\delta(t,0), 0) = \\
&= \delta^*(t, 0) = \delta^*(\delta(t,0), \varepsilon) = \\
&= \delta^*(t, \varepsilon) = t.
\end{aligned}$$

Поскольку $t \in F = \{t\}$, то A_{odd} принимает слово 110100. Проверьте, что A_{odd} не принимает (отвергает) слова 0011, 000, ε.

Несложно понять, что A_{odd} принимает те и только те слова, которые содержат нечётное число единиц. Покажем для примера, как это может быть доказано. Пусть $|w|_0$ и $|w|_1$ обозначают число нулей и единиц в слове соответственно. Докажем по индукции следующие утверждения:

$$\delta^*(s,w) = \begin{cases} s, & \text{если } |w|_1 \text{ чётно,} \\ t, & \text{если } |w|_1 \text{ нечётно,} \end{cases} \qquad \delta^*(t,w) = \begin{cases} t, & \text{если } |w|_1 \text{ чётно,} \\ s, & \text{если } |w|_1 \text{ нечётно.} \end{cases}$$

База для $w = \varepsilon$ выполнена по определению δ^*. Предположение индукции таково: формулы выше выполнены для любого слова x с длиной меньше $|w|$. Тогда есть четыре варианта:

1) $w = 1x$, $|w|_1$ чётно, тогда $|x|_1$ нечётно, а $\delta^*(s, w) = \delta^*(\delta(s, 1), x) = = \delta^*(t, x) = s$, последнее равенство верно по предположению индукции. Также выполнено $\delta^*(t, w) = \delta^*(\delta(t, 1), x) = \delta^*(s, x) = t$.

Случаи 2) $w = 1x$, $|w|_1$ нечётно, 3) $w = 0x$, $|w|_1$ чётно, 4) $w = 0x$, $|w|_1$ нечётно рассматриваются аналогично.

Отсюда (после рассмотрения всех четырёх случаев) напрямую следует, что A_{odd} распознаёт язык слов над $\{0, 1\}$, содержащих нечётное число единиц.

2.2 Произведение ДКА

Пусть теперь ДКА $A_1 = (\Sigma, Q_1, s_1, F_1, \delta_1)$ принимает язык L_1, ДКА $A_2 = (\Sigma, Q_2, s_2, F_2, \delta_2)$ принимает язык L_2 над одним и тем же алфавитом Σ. Построим новый ДКА A (над тем же алфавитом). Состояния A – это пары состояний, переходы определяются переходами A_1 и A_2, множество принимающих состояний определим отдельно.

Конструкция **произведения** двух ДКА $A = A_1 \times A_2$ строится следующим образом: $A = (\Sigma, Q, s, F, \delta)$, где

$Q = Q_1 \times Q_2$,

$s = (s_1, s_2)$,

F – *определим позднее*,

$\delta : (Q_1 \times Q_2) \times \Sigma \to Q_1 \times Q_2$, $\delta((p, q), a) = (\delta_1(p, a), \delta_2(q, a))$.

Докажем по индукции, что $\delta^*((p, q), w) = (\delta_1^*(p, w), \delta_2^*(q, w))$ (определение функции δ^* для ДКА дано выше):

если $w = \varepsilon$, то $\delta^*((p, q), \varepsilon) = (p, q) = (\delta_1^*(p, \varepsilon), \delta_2^*(q, \varepsilon))$, равенства здесь выполнены по определению функций $\delta^*, \delta_1^*, \delta_2^*$,

если $w = ax$, то

$$\delta^*((p,q),w) = \delta^*((p,q),ax) = \delta^*(\delta((p,q),a),x) =$$
$$= \delta^*((\delta_1(p,a),\delta_2(q,a)),x) \overset{!!}{=} (\delta_1^*(\delta_1(p,a),x),\delta_2^*(\delta_2(q,a),x)) =$$
$$= (\delta_1^*(p,ax),\delta_2^*(q,ax)) = (\delta_1^*(p,w),\delta_2^*(q,w)),$$

здесь $\overset{!!}{=}$ выполняется по предположению индукции.

Слово w по определению принимается ДКА A тогда и только тогда, когда $\delta^*((s_1,s_2),w) \in F$, т.е. тогда и только тогда, когда

$$(\delta_1^*(s_1,w),\delta_2^*(s_2,w)) \in F.$$

Выбирая различные определения для F, мы можем распознавать различные языки с помощью конструкции произведения.

Для **пересечения** языков L_1 и L_2 эта конструкция использует (что интуитивно понятно) конъюнкцию:

$$F = \{(p,q) \,|\, p \in F_1 \wedge q \in F_2\}.$$

В этом случае $(\delta_1^*(s_1,w),\delta_2^*(s_2,w)) \in F$ выполняется тогда и только тогда, когда $\delta_1^*(s_1,w) \in F_1$ **и** $\delta_2^*(s_2,w) \in F_2$, т.е. A_1 принимает w **и** A_2 принимает w. Таким образом, построен ДКА для пересечения двух языков по двум ДКА для этих языков.

Для **объединения** языков L_1 и L_2 эта конструкция использует (по аналогии) дизъюнкцию:

$$F = \{(p,q) \,|\, p \in F_1 \vee q \in F_2\}.$$

В этом случае $(\delta_1^*(s_1,w),\delta_2^*(s_2,w)) \in F$ выполняется тогда и только тогда, когда $\delta_1^*(s_1,w) \in F_1$ **или** $\delta_2^*(s_2,w) \in F_2$, т.е. A_1 принимает w **или** A_2 принимает w. Таким образом, построен ДКА для объединения двух языков по двум ДКА для этих языков.

Для **разности** языков L_1 и L_2 эта конструкция использует конъюнкцию с разностью:

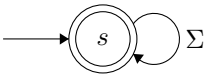
$$F = \{(p,q) \,|\, p \in F_1 \wedge q \notin F_2\}.$$

В этом случае $(\delta_1^*(s_1,w),\delta_2^*(s_2,w)) \in F$ выполняется тогда и только тогда, когда $\delta_1^*(s_1,w) \in F_1$ **и** $\delta_2^*(s_2,w) \notin F_2$, т.е. A_1 принимает w **и** при этом A_2 не принимает w, что отвечает языку $L_1 - L_2$.

Итого, верна следующая лемма.

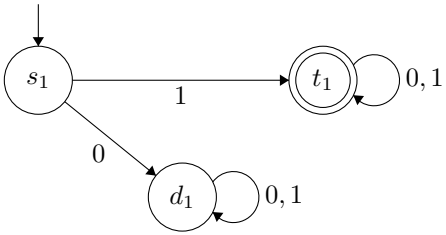
Лемма. Если языки (над общим алфавитом Σ) L_1 и L_2 распознаются некоторыми ДКА A_1 и A_2, то существуют ДКА, которые распознают $L_1 \cap L_2$, $L_1 \cup L_2$, $L_1 - L_2$, $\overline{L_1} = \Sigma^* - L_1$.

Последнее было известно и ранее, конструкция произведения позволяет получить этот результат ещё раз, поскольку язык Σ^* распознаётся ДКА A_{Σ^*}

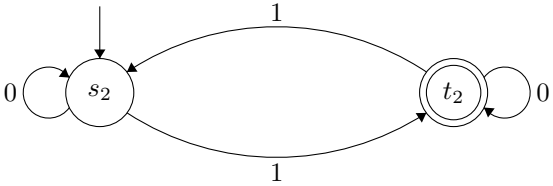


Пример

Пусть A_1 (над $\{0,1\}$) принимает язык всех слов, начинающихся с 1, пусть A_2 принимает слова с нечётным количеством единиц.

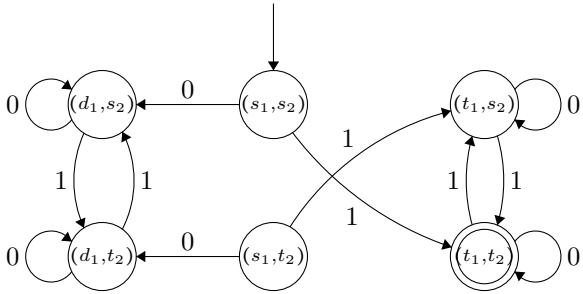


A_1 распознаёт слова, начинающиеся с 1

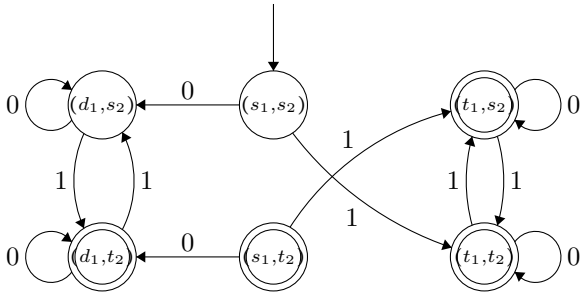


A_2 распознаёт слова с нечётным количеством 1

Пересечение языков $\mathsf{L}(A_1) \cap \mathsf{L}(A_2)$ тогда распознаётся следующим ДКА:



Объединение же $\mathsf{L}(A_1) \cup \mathsf{L}(A_2)$ распознаётся следующим ДКА:



2.3 Теорема Клини

Внимательный читатель, вероятно, заметил, что выше мы не ввели весьма естественное определение примерно такого вида: язык L называется **???**, если он распознаётся некоторым ДКА. По смыслу вместо **???** должно стоять что-то типа "**автоматным**". Избыточность такого термина показывает следующая теорема.

Теорема (Клини). Для языка L над алфавитом Σ следующие два условия равносильны:

1) L регулярный язык,

2) L распознаётся некоторым ДКА.

Доказательство этой теоремы будет дано позднее (оно будет использовать дополнительную конструкцию – недетерминированный конечный автомат). Пока же, пользуясь теоремой как данностью, отметим, что регулярные языки замкнуты не только относительно конкатенации, объединения и итерации (по определению), но и относительно произвольных теоретико-множественных операций.

3 Недетерминированные конечные автоматы

Рассмотрим теперь недетерминированную модель вычислений.

3.1 Определения

Недетерминированный конечный автомат (НКА) без ε-переходов B – это *упорядоченная пятёрка (кортеж)*:

$$B = (\Sigma, Q, s, F, \delta), \quad \text{где}$$

Σ – это конечное множество, называемое алфавитом или **входным алфавитом** НКА. *Так же, как у ДКА.*

Q – это конечное множество, называемое множеством **состояний**. Его элементы – состояния НКА. *Так же, как у ДКА.*

$s \in Q$ – это **начальное состояние** НКА. *Так же, как у ДКА.*

$F \subseteq Q$ – это **множество принимающих состояний**. *Так же, как у ДКА.*

$\delta : Q \times \Sigma \to 2^Q$ – это **функция переходов**. *Не так, как у ДКА!*

Единственное отличие в определении НКА от определения ДКА содержится в функции переходов: $\delta(q, a)$ – теперь это некоторое подмножество множества состояний Q. Это подмножество может быть одноэлементным, пустым, совпадать с Q или быть чем угодно между.

Пусть НКА B на вход было подано слово w. B начинает работу и поддерживает на каждом шаге *множество текущих состояний* (ДКА поддерживал одно текущее состояние). Исходно множество текущих состояний равно $\{s\}$. По прочтении НКА B символа a происходит

следующее: B заменяет множество текущих состояний Q_C на множество $\bigcup_{q \in Q_C} \delta(q, a)$. После прочтения последнего символа B останавливается с некоторым множеством текущих состояний Q_F. Если $Q_F \cap F \neq \varnothing$, то говорят, что B **принимает** слово w, в противном случае – **отвергает** слово w. Множество всех слов, которые B принимает, – это язык, **принимаемый** или **распознаваемый** НКА B.

Определим работу НКА формально. Снова расширим функцию δ до $\delta^* : Q \times \Sigma^* \to 2^Q$ по правилу:

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

НКА $B = (\Sigma, Q, s, F, \delta)$ **принимает** слово w тогда и только тогда, когда $\delta^*(s, w) \cap F \neq \varnothing$ (в противном случае B **отвергает** слово w). Множество слов, принимаемых НКА B, – это язык $\mathsf{L}(B)$, **принимаемый** или **распознаваемый** этим НКА.

Определение функции δ^* будет более похоже на его аналог для ДКА, если "перегрузить" обозначения: пусть для множества $C \subseteq Q$ выражение $\delta^*(C, w)$ означает $\bigcup_{r \in C} \delta^*(r, w)$. Тогда

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

Ещё раз отметим, определение НКА и определение языка, принимаемого НКА, – это два различных определения. Только вместе они позволяют что-то утверждать о языках, распознаваемых недетерминированными автоматами. Мы могли бы оставить определение НКА, но сменить определение языка, распознаваемого НКА (вместо $\delta^*(s, w) \cap F \neq \varnothing$ использовать $\delta^*(s, w) = F$ или что-то подобное), тогда мы получили бы некоторый другой (в общем случае) язык для каждого НКА.

Аналогично ДКА, НКА также удобно представлять в виде размеченного орграфа. Теперь на количество ребёр и на метки не накладывается ограничений: из вершины может выходить несколько рёбер, помеченных одним и тем же символом, или вовсе ни одного.

Если отождествить одноэлементное множество $\{q\}$ и состояние q, то ДКА можно считать частным случаем НКА. Мы будем поступать так в дальнейшем. Также мы будем называть два конечных автомата (детерминированных или недетерминированных) **эквивалентными**, если они принимают один и тот же язык.

ε-переходы

Стандартным и очень удобным соглашением является добавление в НКА так называемых ε-переходов (эпсилон-переходов).

Недетерминированный конечный автомат с ε-переходами (сокращение "НКА" не изменилось) B – это *упорядоченная пятёрка*:

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

где отличие от НКА без ε-переходов состоит лишь в функции переходов:

$$\delta : Q \times (\Sigma + \{\varepsilon\}) \to 2^Q.$$

Неформально говоря, НКА может пройти по ε-переходу "бесплатно", не читая символы входной строки.

ε-замыканием (эпсилон-замыканием) состояния q назовём множество $C \subseteq Q$, которое есть наименьшее (по включению) подмножество Q такое, что выполняются оба условия:

1) $q \in C$,

2) $\forall p \in C\ \delta(p, \varepsilon) \subseteq C$.

Мы будем обозначать ε-замыкание состояния q через $\varepsilon_r(q)$. В случае НКА без ε-переходов мы будем считать, что $\delta(q, \varepsilon) = \varnothing$ и соответственно $\varepsilon_r(q) = \{q\}$. Иначе говоря, $\varepsilon_r(q)$ – это множество состояний, в которые можно попасть из состояния q, не прочитав ни одного символа входного слова.

Для НКА с ε-переходами определим функцию $\delta^* : Q \times \Sigma^* \to 2^Q$:

$$\delta^*(q, w) = \begin{cases} \varepsilon_r(q), & \text{если } w = \varepsilon, \\ \bigcup_{p \in \varepsilon_r(q)} \bigcup_{r \in \delta(p,a)} \delta^*(r, x), & \text{если } w = ax. \end{cases}$$

Последнюю запись проще понять, если снова "перегрузить" обозначения: пусть для множества состояний D обозначение $\varepsilon_r(D)$ означает $\bigcup_{q \in D} \varepsilon_r(q)$ и аналогично $\delta(D, a) = \bigcup_{q \in D} \delta(q, a)$, $\delta^*(D, a) = \bigcup_{q \in D} \delta^*(q, a)$. Тогда определение δ^* выглядит так:

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

Как и ранее, НКА с ε-переходами принимает слово w тогда и только тогда, когда $\delta^*(s, w) \cap F \neq \varnothing$.

Очевидно, НКА с ε-переходами может распознать любой язык, который может распознать НКА без ε-переходов. Верно и обратное: для НКА с ε-переходами B можно построить эквивалентный НКА без ε-переходов B'.

В самом деле, пусть $B = (\Sigma, Q, s, F, \delta)$. Тогда $B' = (\Sigma, Q, s, F', \delta')$, где

$$F' = \{q \in Q \mid \varepsilon_r(q) \cap F \neq \varnothing\}, \quad \delta'(q, a) = \delta(\varepsilon_r(q), a).$$

Тогда для любого состояния q и любой строки x выполняется

$$\delta^*(q, x) \cap F = \varnothing \Leftrightarrow \delta'^*(q, x) \cap F' = \varnothing,$$

что нетрудно проверить индукцией по длине x. В случае $x = \varepsilon$

$$\delta^*(q, \varepsilon) \cap F = \varepsilon_r(q) \cap F,$$

$$\delta'^*(q, \varepsilon) \cap F' = \{q\} \cap F'.$$

Правая часть второго тождества непуста в том и только в том случае, когда $q \in F'$, что, в свою очередь, выполнено тогда и только тогда, когда $\varepsilon_r(q) \cap F \neq \varnothing$ по определению F'.

Пусть $\delta^*(q, y) \cap F = \varnothing \Leftrightarrow \delta'^*(q, y) \cap F' = \varnothing$ выполнено на любом слове y длины меньше $|x|$. Тогда

$$\delta^*(q, ay) \cap F = \delta^*(\delta(\varepsilon_r(q), a), y) \cap F = \delta^*(\delta'(q, a), y) \cap F = \\ = \bigcup_{p \in \delta'(q,a)} (\delta^*(p, y) \cap F),$$

$$\delta'^*(q, ay) \cap F' = \delta'^*(\delta'(q, a), y) \cap F' = \bigcup_{p \in \delta'(q,a)} (\delta'^*(p, y) \cap F').$$

Правые части пусты или непусты одновременно в силу предположения индукции.

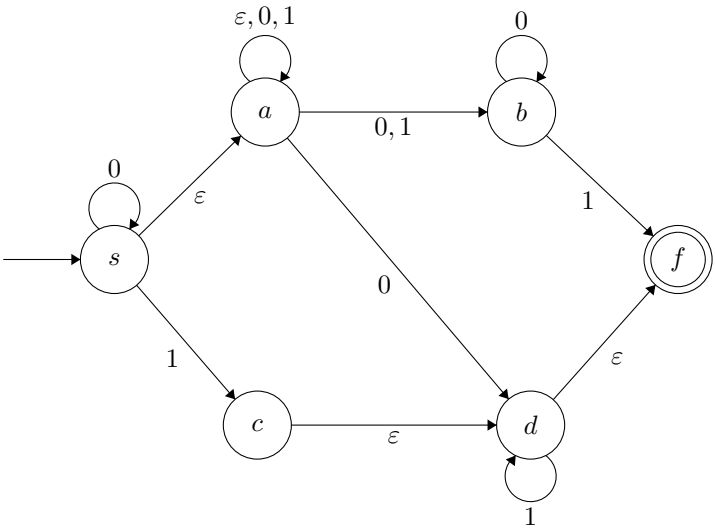
Следовательно,

$$\delta^*(s, x) \cap F \neq \varnothing \Leftrightarrow \delta'^*(s, x) \cap F' \neq \varnothing,$$

т.е. B принимает x тогда и только тогда, когда B' принимает x.

Пример

Рассмотрим такой НКА B_0:



Пусть автомат получает на вход слово 101. Отметим сперва, что 101 можно прочесть по пути $s \xrightarrow{\varepsilon} a \xrightarrow{1} b \xrightarrow{0} b \xrightarrow{1} f$ или, например, $s \xrightarrow{\varepsilon} a \xrightarrow{1} a \xrightarrow{0} d \xrightarrow{1} d \xrightarrow{\varepsilon} f$, – существование любого из таких путей автоматически означает, что B принимает 101. Кроме того, если перейти по первому символу 1 в состояние c, то возможности дочитать суффикс 01 не существует, так что эта ветвь вычислений завершается непринятием слова (что не мешает автомату принять слово).

Теперь будем рассуждать более формально:

$$\delta^*(s, 101) \overset{1}{=} \delta^*(\delta(\varepsilon_r(s), 1), 01) \overset{2}{=} \delta^*(\delta(\{s, a\}, 1), 01) \overset{3}{=} \delta^*(\{a, b, c\}, 01) \overset{4}{=}$$
$$\overset{4}{=} \delta^*(\delta(\varepsilon_r(\{a, b, c\}), 0), 1) \overset{5}{=} \delta^*(\delta(\{a, b, c, d, f\}, 0), 1) \overset{6}{=} \delta^*(\{a, b, d\}, 1) \overset{7}{=}$$
$$\overset{7}{=} \delta^*(\delta(\varepsilon_r(\{a, b, d\}), 1), \varepsilon) \overset{8}{=} \delta^*(\delta(\{a, b, d, f\}, 1), \varepsilon) \overset{9}{=}$$
$$\overset{9}{=} \delta^*(\{a, b, d, f\}, \varepsilon) \overset{10}{=} \varepsilon_r(\{a, b, d, f\}) \overset{11}{=} \{a, b, d, f\}.$$

Расшифруем по очереди:

(1) Это определение функции δ^*.

(2) Это вычисление $\varepsilon_r(s) = \varepsilon_r(\{s\}) = \{s, a\}$, так как единственный ε-переход из s ведёт в a (а состояние s всегда лежит в $\varepsilon_r(s)$).

(3) Это вычисление перехода $\delta(\{s, a\}, 1)$. Поскольку $\delta(s, 1) = \{c\}$, и при этом $\delta(a, 1) = \{a, b\}$, то ответ здесь $\{a, b, c\}$.

(4) Это определение функции δ^*.

(5) Это вычисление $\varepsilon_r(\{a, b, c\})$. Поскольку $\varepsilon_r(a) = \{a\}$, $\varepsilon_r(b) = \{b\}$ (несмотря на отсутствие ε-переходов само состояние всегда лежит в своём собственном ε_r), $\varepsilon_r(c) = \{c, d, f\}$, то ответ здесь $\{a, b, c, d, f\}$.

(6) Это вычисление перехода $\delta(\{a, b, c, d, f\}, 0)$. Поскольку выполнено $\delta(a, 0) = \{a, b, d\}$, $\delta(b, 0) = \{b\}$, $\delta(c, 0) = \varnothing$, $\delta(d, 0) = \varnothing$, $\delta(f, 0) = \varnothing$, то $\delta(\{a, b, c, d, f\}, 0) = \{a, b, d\}$ как объединение.

(7) Это определение функции δ^*.

(8) Это вычисление $\varepsilon_r(\{a, b, d\})$. Поскольку $\varepsilon_r(a) = \{a\}$, $\varepsilon_r(b) = \{b\}$, $\varepsilon_r(d) = \{d, f\}$, то ответ здесь $\{a, b, d, f\}$.

(9) Это вычисление перехода $\delta(\{a, b, d, f\}, 1)$. Поскольку $\delta(a, 1) = \{a, b\}$, $\delta(b, 1) = \{f\}$, $\delta(d, 1) = \{d\}$, $\delta(f, 1) = \varnothing$, то $\delta(\{a, b, d, f\}, 1) = \{a, b, d, f\}$.

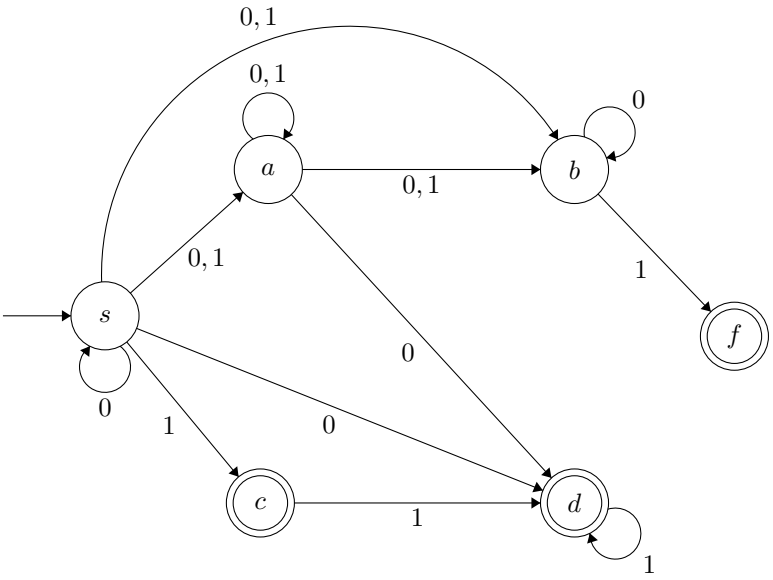
(10) Это определение функции δ^* – на этот раз первый случай в определении.

(11) Это вычисление $\varepsilon_r(\{a, b, d, f\})$. Поскольку $\varepsilon_r(a) = \{a\}$, $\varepsilon_r(b) = \{b\}$, $\varepsilon_r(d) = \{d, f\}$, $\varepsilon_r(f) = \{f\}$, то ответ здесь $\{a, b, d, f\}$.

Окончательно $\{a, b, d, f\} \cap F = \{a, b, d, f\} \cap \{f\} = \{f\} \neq \varnothing$, так что автомат B принимает слово 101.

В качестве заключения отметим, что если слово w содержит нули, то B принимает его, скажем, так: переходит в a по ε, затем "проматывает" все буквы слова w до последнего нуля в состоянии a, переходит по последнему нулю в состояние d, затем проматывает все единицы (если от слова что-то осталось, то это только единицы) в состоянии d и по ε переходит в принимающее состояние f. Если слово w состоит только из единиц, то B также принимает w: по первой единице он переходит в c, затем сразу в d по ε, затем проматывает все единицы слова в состоянии d и снова по ε переходит в принимающее состояние f. Единственное слово, которое B не принимает, – это слово ε, у него нет "последнего нуля" или "первой единицы" для работы предыдущих рассуждений, а формально $\delta^*(s, \varepsilon) = \varepsilon_r(s) = \{s, a\} \cap \{f\} = \varnothing$.

Построим для B_0 эквивалентный НКА без ε-переходов. Применение конструкции выше даёт такой НКА



Одно принимающее состояние

Наличие или отсутствие ε-переходов позволяет удобно переходить между различными вариантами определений НКА и быстро доказывать различные полезные факты.

Для каждого НКА существует эквивалентный ему НКА с одним принимающим состоянием. В самом деле, достаточно добавить в НКА одно новое состояние, добавить ε-переходы из каждого исходного принимающего состояния в новое, сделать старые состояния не принимающими, а новое – единственным принимающим.

Подобная конструкция позволяет всегда, когда это удобно, считать, что принимающее состояние у НКА (с ε-переходами) ровно одно.

Множество начальных состояний

По аналогии с множеством принимающих состояний можно позволить НКА иметь несколько (или нисколько) начальных состояний. Тогда в определении вместо состояния s будет стоять множество S, а слово w будет приниматься автоматом тогда и только тогда, когда

$$\delta^*(S, w) \cap F = \left(\bigcup_{s \in S} \delta^*(s, w) \right) \cap F \neq \varnothing,$$

т.е. когда в графе будет существовать некоторый маршрут по буквам слова из некоторого начального состояния в некоторое принимающее.

Понятно, что для любого НКА с множеством начальных состояний существует эквивалентный обычный НКА – необходимо аналогично предыдущему случаю добавить новое состояние, которое назначить уникальным начальным, и ε-переходы в старые начальные состояния. После этого, если необходимо, можно убрать ε-переходы стандартным способом.

Различные начальное и уникальное принимающее состояния

Очевидно из предыдущего – добавим новое уникальное принимающее состояние, ε-переходы из каждого исходного принимающего состояния в новое, сделаем старые принимающие состояния непринимающими. Значит, для любого НКА существует эквивалентный ему НКА с единственным начальным состоянием и единственным, отличным от начального, принимающим состоянием.

Кроме того, добавлением нового начального состояния несложно добиться следующего: начальное состояние недостижимо ни из какого другого состояния (в него нет стрелочек), принимающее состояние не имеет ни одной выходящей стрелочки (из него ничего кроме него не достижимо).

3.2 НКА эквивалентны ДКА

После отождествления состояния q и одноэлементного множества $\{q\}$ ДКА можно считать частным случаем НКА без ε-переходов – в любых переходах множество значений функции δ всегда содержит ровно одно состояние.

Несложно также понять, что для любого НКА существует ДКА, который распознаёт тот же язык. В самом деле, НКА совершает детерминированные переходы между множествами состояний: если дано множество состояний Q_0 и символ слова a, то значение $\delta(Q_0, a)$ определено однозначно.

Более формально, пусть дан НКА $B = (\Sigma, Q, s, F, \delta)$. Построим ДКА $A = (\Sigma, Q', s', F', \delta')$, где

$Q' = 2^Q$ – множество всех подмножеств Q,

$s' = \{s\}$ (или $s' = S$, если НКА позволено иметь целое множество S начальных состояний),

$F' = \{A \subset Q \,|\, \varepsilon_r(A) \cap F \neq \varnothing\}$ (если в НКА нет ε-переходов, то можно считать $\varepsilon_r(q) = q$),

$\delta'(q', a) = \delta(\varepsilon_r(q'), a)$ для $q' \in Q'$ и $a \in \Sigma$.

Тогда A принимает слово w тогда и только тогда, когда это делает B по определению. Докажем это по индукции – доказывать будем утверждение

$$\forall p \subseteq Q \ \forall x \in \Sigma^* \ \left(\delta'^*(p, x) \in F' \Leftrightarrow \delta^*(p, x) \cap F \neq \varnothing\right).$$

База индукции: $x = \varepsilon$. Левая часть (это ДКА) преобразуется в $p \in F'$, правая часть (это НКА, возможно, с ε-переходами) преобразуется в $\varepsilon_r(p) \cap F \neq \varnothing$. Левая и правая части эквивалентны по определению F'.

Шаг. Предполагаем, что утверждение индукции верно на строках длины меньше $|x|$, пусть также $x = aw$. Левая часть:

$$\delta'^*(p, aw) = \delta'^*(\delta'(p, a), w) = \delta'^*(\delta(\varepsilon_r(p), a), w).$$

Правая часть

$$\delta^*(p, aw) = \delta^*(\delta(\varepsilon_r(p), a), w).$$

Если обозначить $g = \delta(\varepsilon_r(p), a)$, получим, что левая часть преобразуется в $\delta'^*(g, w) \in F'$, правая часть преобразуется в $\delta^*(g, w) \cap F \neq \varnothing$ – части эквивалентны по предположению индукции. Утверждение доказано. Подставляя $\{s\}$ вместо p, получаем эквивалентность автоматов A и B.

Отметим, что по НКА с $|Q|$ состояниями переход к подмножествам порождает ДКА с $2^{|Q|}$ состояниями. Такой ДКА может не быть минимальным (по числу состояний) для этого языка, но может и быть. Экспоненциальный разрыв между числом состояний НКА и эквивалентного ему ДКА в худшем случае неизбежен. Позже мы докажем этот факт.

3.3 Смысл недетерминированности

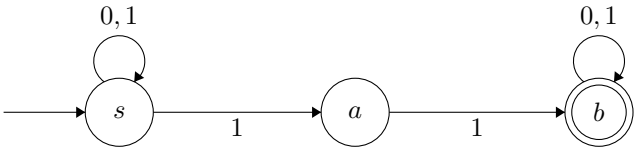
Из определений следует, что НКА принимает слово w, если "может", в том смысле, что у него есть хотя бы один способ так пройти по состояниям, читая буквы слова, чтоб закончить в принимающем состоянии. Это означает, что НКА в своей работе реализует *квантор существования* – если *существует* способ принять слово, то слово будет принято.

ДКА был "естественной" моделью вычислений – любая реалистичная вычислительная процедура фактически реализует ДКА. Как устроен НКА? Как он "магическим образом" умудряется выбирать удачный путь при заданном слове? Это не математические вопросы, тем не менее полезно поразмышлять о том, как можно *думать* о работе НКА. Подходящих способов несколько.

1) НКА удачлив. Как только НКА B читает символ a слова w из состояния q, он всегда выбирает (из всех возможностей) ту самую, которая позволит ему в конце концов принять слово w, если только это возможно в принципе. Как он выбирает? Он выбирает некоторую возможность наугад, но ему всегда везёт, потому что НКА очень удачлив. Здесь квантор существования таков: если удачный выбор существует, он будет сделан.

2) Параллелизм. Как только НКА B читает символ a слова w из состояния q, он создаёт несколько (по числу переходов) веток параллельных вычислений. Если переходов нет, ветка аварийно завершает работу. После этого некто, находящийся снаружи от НКА, смотрит на все ветки сразу (на все их результаты). Если хотя бы одна ветка привела к принимающему состоянию (это квантор существования), некто объявляет, что НКА принял слово. Заметим, что сам НКА "не знает", какая из его веток приняла слово.

3) Верификатор. Это понятие сперва поясним на конкретном примере. НКА, представленный ниже, принимает язык $(0+1)^*11(0+1)^*$, т.е. язык всех слов, содержащих 11 в качестве подслова.



Если мы хотим верифицировать (проверить) тот факт, что слово w принадлежит языку, то нам достаточно продемонстрировать произвольный

маршрут (последовательность вершин) по НКА, начинающийся в s, заканчивающийся в b, с рёбрами, помеченными символами w. Мы скажем, что этот маршрут является **сертификатом принадлежности** слова w языку L. Для конкретного слова $w = 01001101$ таким сертификатом (здесь мы опускаем начальное состояние, в котором находится НКА до прочтения букв) является слово $ssssabbb$.

В этом смысле НКА проверяет имеющееся доказательство принадлежности строки w языку L. Каждый раз, когда НКА нужно сделать выбор, он смотрит в сертификат, в котором написано, куда нужно сделать переход для того, чтоб принять слово.

В данном примере первой букве 0 слова соответствует первая буква s сертификата, значит, НКА должен после прочтения буквы перейти (в данном случае остаться) в состояние s. НКА может *детерминированно* проверить, что такой переход (из начального s в необходимое s по символу 0) существует. Далее аналогично по другим буквам слова. Для слова $w = 111$ существует целых два сертификата: sab и abb. Для слова $w = 010$ не существует ни единого сертификата в следующем смысле. Если кто-то попробует предъявить сертификат, скажем ssb, то НКА поймёт, что это некорректный сертификат, поскольку какой-то из переходов невозможен (или последнее состояние не принимающее) – в данном случае по последнему нулю нельзя перейти из s в b.

Соображения о детерминированности всех переходов при применении сертификата позволяют представлять работу НКА как работу ДКА над словом w при наличии доступа к сертификату того, что $w \in L$.

4 Теорема Клини

Мы уже доказали, что распознавание некоторого языка ДКА, НКА с ε-переходами или без таковых – эквивалентные условия. Уточним и докажем теперь теорему.

Теорема (Клини). Для языка L над алфавитом Σ следующие условия эквивалентны:

1) L регулярный язык (т.е. задаётся некоторым регулярным выражением),

2) L распознаётся некоторым конечным автоматом (КА).

4.1 Построение автомата по регулярному выражению

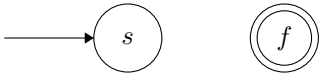
Лемма. Для каждого регулярного языка L над Σ (заданного некоторым регулярным выражением) существует НКА (с ε-переходами) B, который принимает язык L. При этом B удовлетворяет дополнительным ограничениям:

1) у него единственное начальное состояние s, в которое не ведёт ни один переход по стрелке,

2) у него единственное принимающее состояние f, отличное от начального, из которого нет переходов ни по одной стрелке.

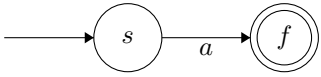
Доказывать лемму будем по (структурной) индукции. Базовых случаев два.

1) Регулярное выражение есть $\varnothing$. Тогда соответствующий (пустой) язык принимается НКА



Формально, это НКА $(\Sigma, \{s, f\}, s, \{f\}, \delta)$, где $\delta(q, a) = \varnothing$ на любых значениях аргументов. Очевидно, такой НКА и правда принимает пустой язык.

2) Регулярное выражение есть a, где $a \in \Sigma$. Соответствующий язык $L = \{a\}$ принимается НКА

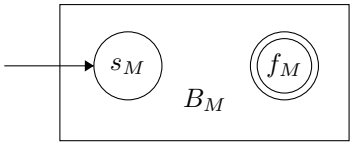


Формально, это НКА $(\Sigma, \{s, f\}, s, \{f\}, \delta)$, где $\delta(s, a) = \{f\}$, а на всех прочих значениях аргументов δ выдаёт $\varnothing$. Очевидно, такой НКА и правда принимает язык $\{a\}$.

Шаг индукции: пусть регулярное выражение, описывающее наш регулярный язык L, получено применением n операций объединения, конкатенации и итерации к базовым случаям. Пусть предположение индукции (существует НКА, распознающий соответствующий язык) верно для всех

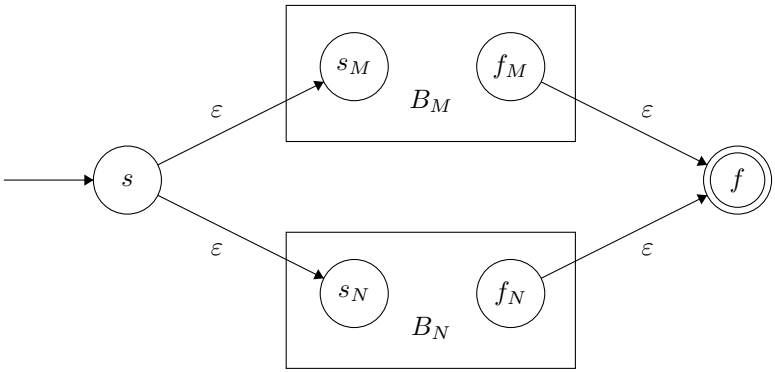
языков, определяемых регулярными выражениями с числом операций, меньших n. Регулярный язык L тогда есть объединение двух регулярных языков, конкатенация двух регулярных языков или итерация регулярного языка. Во всех случаях в "компонентах" менее n операций в каждой, так что для них выполняется индуктивное предположение.

Пусть язык M таков, что для него выполняется индуктивное предположение, тогда он распознаётся некоторым НКА B_M, который мы будем схематично изображать следующим образом:



Начальное состояние B_M – это s_M, принимающее – f_M, а "внутри" он устроен некоторым образом, своим для каждого языка M. Предположение индукции гарантирует единственность принимающего состояния и отсутствие стрелок из f_M или в s_M.

3) Если L есть объединение M и N (для которых выполняется индуктивное предположение), то L распознаётся НКА



Формально, если M и N распознаются соответственно НКА

$$B_M = (\Sigma_M, Q_M, s_M, \{f_M\}, \delta_M), \quad B_N = (\Sigma_N, Q_N, s_N, \{f_N\}, \delta_N),$$

то НКА для объединения есть $B = (\Sigma_M + \Sigma_N, Q_M + Q_N + \{s, f\}, s, \{f\}, \delta)$, где

$\delta(s, \varepsilon) = \{s_M, s_N\}$,

$\delta(q, a) = \delta_M(q, a)$ для $q \in Q_M - \{f_M\}$, $a \in \Sigma_M + \{\varepsilon\}$,

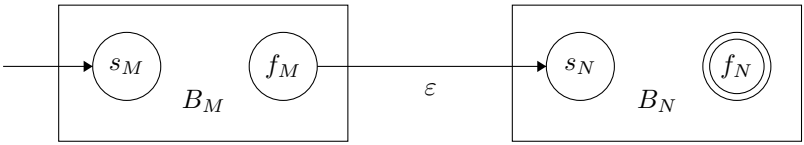
$\delta(q, a) = \delta_N(q, a)$ для $q \in Q_N - \{f_N\}$, $a \in \Sigma_N + \{\varepsilon\}$,

$\delta(f_M, \varepsilon) = \delta(f_N, \varepsilon) = \{f\}$,

все прочие значения δ – пустые множества.

Любой маршрут из s в f в B проходит либо через s_M, либо через s_N, причём верен только один из этих вариантов. Затем он проходит (каким-то образом) по, соответственно, автоматам B_M или B_N, затем по ε-переходу в f. Значит, путь по слову w из s в f в B существует тогда и только тогда, когда существует путь по слову w из s_M в f_M в B_M или из s_N в f_N в B_N. Это означает, что B принимает $M + N$.

4) Если L есть конкатенация M и N, то L распознаётся НКА



Формально, если M и N распознаются соответственно НКА

$$B_M = (\Sigma_M, Q_M, s_M, \{f_M\}, \delta_M), \quad B_N = (\Sigma_N, Q_N, s_N, \{f_N\}, \delta_N),$$

то НКА для конкатенации есть $B = (\Sigma_M + \Sigma_N, Q_M + Q_N, s_M, \{f_N\}, \delta)$, где

$\delta(f_M, \varepsilon) = \{s_N\}$,

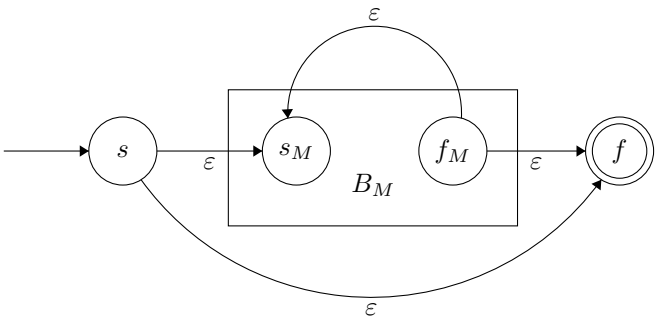
$\delta(q, a) = \delta_M(q, a)$ для $q \in Q_M - \{f_M\}$, $a \in \Sigma_M + \{\varepsilon\}$,

$\delta(q, a) = \delta_N(q, a)$ для $q \in Q_N$, $a \in \Sigma_N + \{\varepsilon\}$,

все прочие значения δ – пустые множества.

Любой маршрут из s_M в f_N в B проходит (каким-то образом) по автомату B_M до состояния f_M, затем в s_N, затем по автомату B_N до f_N. Значит, путь по слову w из s в f в B существует тогда и только тогда, когда существует путь по слову u из s_M в f_M в B_M и путь по слову v из s_N в f_N в B_N, причём $w = uv$. Это означает, что B принимает $M \cdot N$.

5) Наконец, если L – итерация M, то L распознаётся НКА



Формально, если M распознаётся НКА

$$B_M = (\Sigma_M, Q_M, s_M, \{f_M\}, \delta_M),$$

то НКА для итерации есть $B = (\Sigma_M, Q_M + \{s, f\}, s, \{f\}, \delta)$, где

$\delta(s, \varepsilon) = \delta(f_M, \varepsilon) = \{s_M, f\}$,

$\delta(q, a) = \delta_M(q, a)$ для $q \in Q_M - \{f_M\}$, $a \in \Sigma_M + \{\varepsilon\}$,

все прочие значения δ – пустые множества.

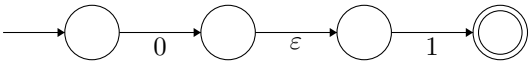
Любой маршрут из s в f в B либо есть прямой ε-переход, либо начинается с перехода в s_M. Во втором случае этот маршрут сначала сколько-то (может быть нуль) раз переходит в f_M, оттуда обратно в s_M, затем один раз (последний) переходит в f_M и оттуда по ε-переходу в f. Значит, путь по слову w из s в f в B существует тогда и только тогда, когда $w = \varepsilon$ или существует разбиение $w = w_1 \dots w_k$ такое, что каждый $w_j \in M$. Это означает, что B принимает M^*.

Пример

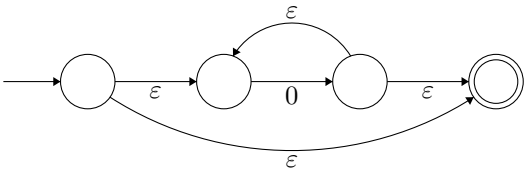
Пусть дано регулярное выражение $\varepsilon + (010^* + 1)^*1$, построим НКА по алгоритму выше.

Автоматы, распознающие языки, задаваемые регулярными выражениями 0 и 1, имеют вид

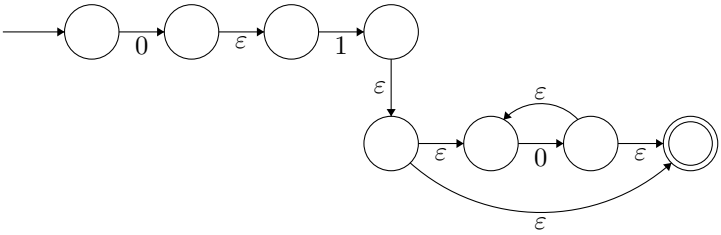
Автомат, распознающий язык, задаваемый регулярным выражением 01, имеет вид



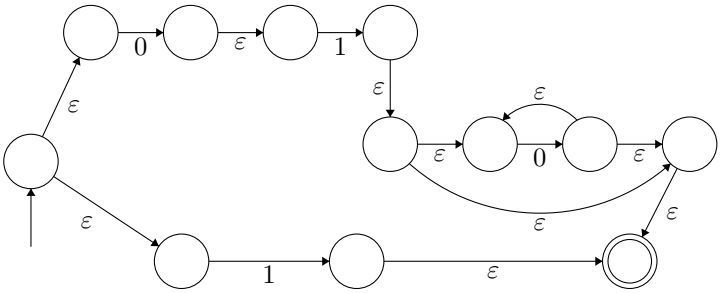
Автомат, распознающий язык, задаваемый регулярным выражением 0^*, имеет вид



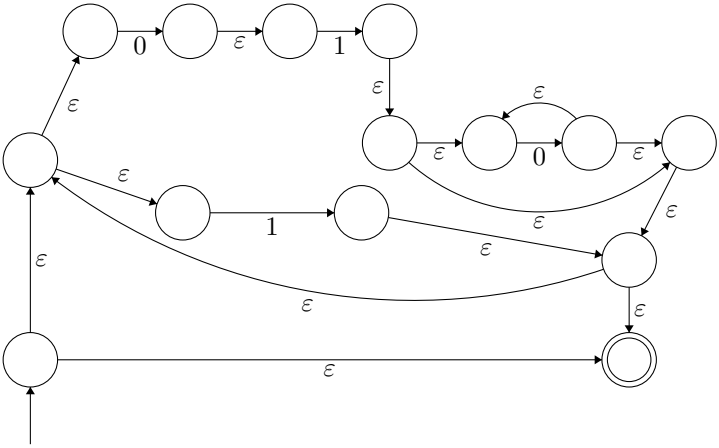
Автомат, распознающий язык, задаваемый регулярным выражением 010^*, имеет вид



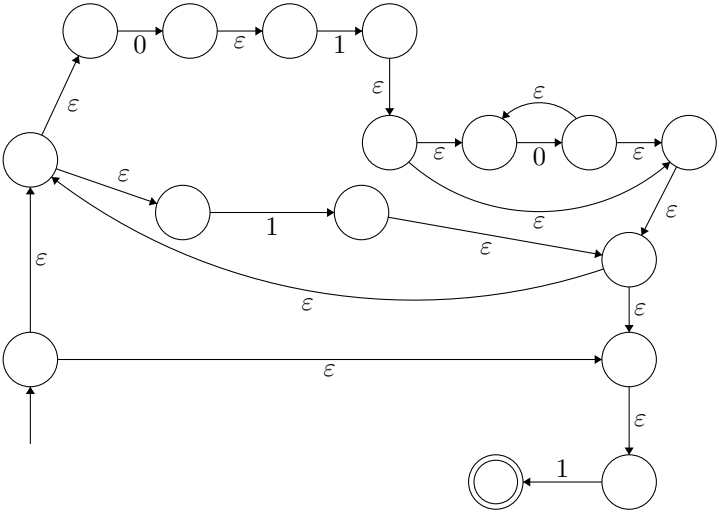
Автомат, распознающий язык, задаваемый регулярным выражением $010^* + 1$, имеет вид



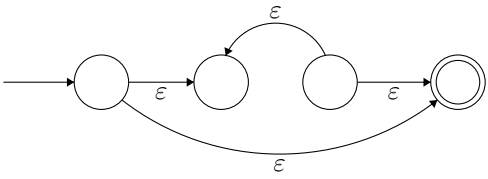
Автомат, распознающий язык, задаваемый регулярным выражением $(010^* + 1)^*$, имеет вид



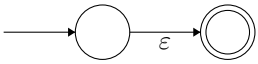
Автомат, распознающий язык, задаваемый регулярным выражением $(010^* + 1)^*1$, имеет вид



Автомат, распознающий язык, задаваемый регулярным выражением ε, имеет вид (напоминаю, что с точки зрения регулярного выражения ε – это сокращение для $\varnothing^*$)

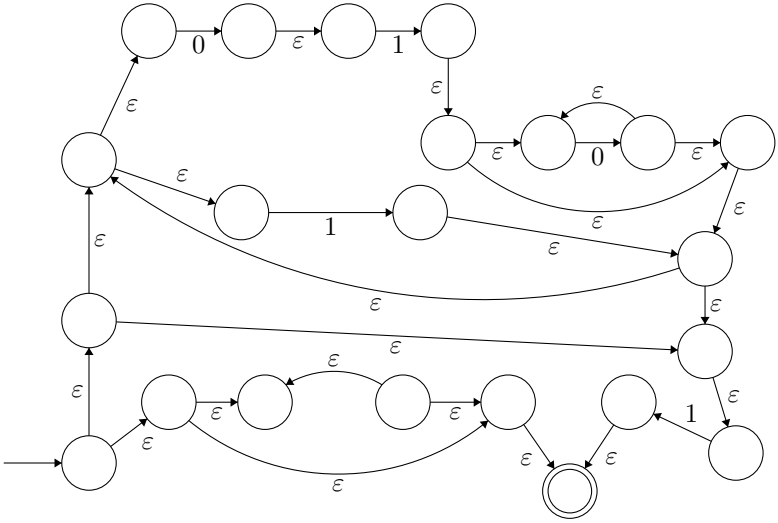


Конечно, его можно упростить до



Мы не будем так делать в этом конкретном примере, однако такие (и другие подобные) упрощения возможны.

Наконец, автомат, распознающий язык, задаваемый исходным регулярным выражением $\varepsilon + (010^* + 1)^*1$, имеет вид



4.2 Построение регулярного выражения по автомату: сжатие состояний

Заменим функцию перехода $\delta : Q \times \Sigma \to Q$ на функцию, которая по паре состояний выдаёт набор букв, по которым возможен переход между

этой парой (даже для ДКА это в общем случае подмножество Σ).

Обобщённый КА (ОКА) – это пятёрка $R = (\Sigma, Q, s, t, d)$, где

Σ – алфавит,

Q – конечное множество состояний автомата,

$s \in Q$ – начальное состояние,

$t \in Q - \{s\}$ – принимающее состояние – мы будем считать, что оно единственно для упрощения дальнейших рассуждений,

$d : (Q - \{t\}) \times (Q - \{s\}) \to \mathrm{REG}_\Sigma$ – функция перехода. Здесь REG_Σ – язык всех регулярных выражений над алфавитом Σ.

ОКА принимает слово w, если существует $k \geqslant 1$, что выполнено:

1) слово w представляется в виде конкатенации подслов:

$$w = w_1 \dots w_k,$$

2) существует конечная последовательность состояний ОКА

$$(q_0, \dots, q_k),$$

причём $q_0 = s$, $q_k = t$, все остальные q_j не равны ни s, ни t,

3) для каждого $1 \leqslant i \leqslant k$ подслово w_i принадлежит языку, определяемому регулярным выражением $d(q_{k-1}, q_k)$.

Как обычно, язык, принимаемый или распознаваемый ОКА, – это множество слов, принимаемых им. ОКА эквивалентен другому конечному автомату, если принимает тот же язык.

Каждый ДКА и каждый НКА можно трансформировать в эквивалентный ему ОКА: в самом деле, достаточно добавить новое начальное состояние и новое принимающее состояние. Затем добавить ε-переходы из нового начального состояния в старое начальное, из всех старых принимающих состояний в новое принимающее. Если из состояния p в состояние q вело ранее несколько рёбер, нужно заменить их на один переход по объединению меток на рёбрах.

Докажем, что каждый ОКА принимает регулярный язык (в смысле, существует регулярное выражение, описывающее этот язык). Сделаем

это с помощью техники **сжатия состояний**. Пусть дан ОКА R. Если в нём два (меньше быть не может) состояния, то он принимает регулярный язык, поскольку в нём есть всего один переход по регулярному выражению $d(s,t)$ – ОКА распознаёт именно язык $\mathsf{L}(d(s,t))$.

Пусть R имеет $n > 2$ состояний. Пусть $q \in Q - \{s,t\}$. Построим ОКА $R' = (\Sigma, Q', s, t, d')$, эквивалентный R такой, что его множество состояний $Q' = Q - \{q\}$. Пусть $p \in Q' - \{t\}$, $r \in Q' - \{s\}$. Положим

$$d'(p,r) = d(p,r) + d(p,q) \cdot d(q,q)^* \cdot d(q,r).$$

Пусть R принимает слово w. Если для разбиения $w = w_1 \dots w_k$, существующего по определению, последовательность состояний $(q_0, \dots, q_k)$ не содержит q, то R' принимает слово w с помощью той же последовательности переходов, поскольку регулярное выражение $d'(p,r)$ содержит в качестве подвыражения $d(p,r)$ (как одну из компонент объединения). Если же последовательность $(q_0, \dots, q_k)$ содержит q, рассмотрим принимающий маршрут в ОКА. Он имеет вид

$$\dots p \xrightarrow{u_i} q \xrightarrow{u_{i+1}} q \dots \xrightarrow{u_{i+k}} q \xrightarrow{u_{i+k+1}} r \dots$$

здесь $u_i u_{i+1} \dots u_{i+k} u_{i+k+1}$ – подстрока w, причём $k \in \mathbb{N}$ (и может быть нулём), $u_i \in d(p,q)$, $u_{i+1} \dots u_{i+k} \in d(q,q)^*$, $u_{i+k+1} \in d(q,r)$ (здесь я пишу "принадлежит регулярному выражению", имея в виду принадлежность языку, определяемому этим регулярным выражением). Тогда

$$u_i u_{i+1} \dots u_{i+k} u_{i+k+1} \in d(p,q) \cdot d(q,q)^* \cdot d(q,r)$$

и можно заменить кусок маршрута на

$$\dots p \xrightarrow{u_i u_{i+1} \dots u_{i+k} u_{i+k+1}} r \dots$$

Повторяя замену кусков до тех пор, пока состояние q не будет полностью удалено из маршрута, получим новый принимающий маршрут, уже в автомате R', поскольку регулярное выражение $d'(p,r)$ содержит в качестве подвыражения $d(p,q) \cdot d(q,q)^* \cdot d(q,r)$.

Обратно, если R' принимает w, то в принимающем маршруте в переходе $d'(p,r)$ соответствующая подстрока либо удовлетворяет выражению $d(p,r)$ и тогда такой же кусок маршрута существует в R, либо она удовлетворяет $d(p,q) \cdot d(q,q)^* \cdot d(q,r)$ и тогда в R существует маршрут через состояние q, который также примет подстроку.

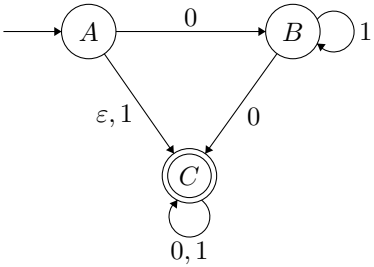
Для получения по ДКА или НКА регулярного выражения теперь нужно представить данный автомат в виде ОКА, а затем последовательно (в любом порядке) сжать все состояния ОКА, получив ОКА с двумя

состояниями s и t. Поскольку сжатие сохраняет эквивалентность автоматов, то регулярное выражение $d(s,t)$ – искомое.

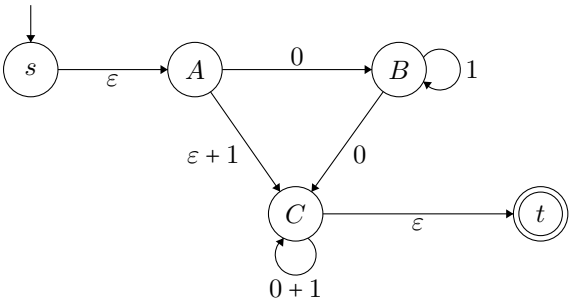
Тем самым, эквивалентность условий регулярности языка доказана в обе стороны, а значит, доказана и теорема Клини.

Пример

Пусть дан следующий НКА над $\{0,1\}$ (конечно же, он принимает все слова – ε-переход из A в C и затем проматывание всего слова в принимающем C).



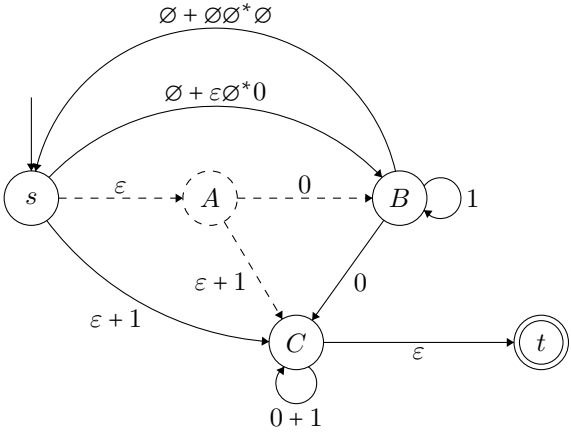
Прежде всего приведём его к необходимому виду – ОКА. Все переходы, не показанные на рисунке, – переходы по пустому множеству (регулярному выражению $\varnothing$).



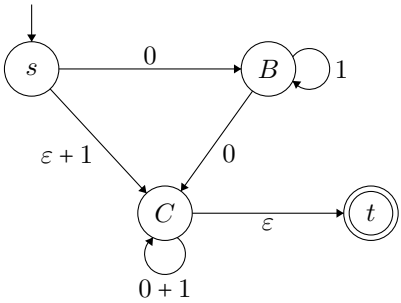
Сожмём состояние A (пунктиром обозначены удаляемые состояние и переходы). Для примера: переход

$$d'(s,B) = d(s,B) + d(s,A) \cdot d(A,A)^* \cdot d(A,B).$$

Исходно $d(s,B) = \varnothing$, $d(s,A) = \varepsilon$, $d(A,A) = \varnothing$, $d(A,B) = 0$. Получаем $d'(s,B) = \varnothing + \varepsilon\varnothing^*0 = 0$. Аналогично получаем $d'(B,s) = \varnothing$, для прочих пар расписывать пустые множества не будем. Новые непустые переходы – это $d'(s,B) = 0$ и $d'(s,C) = \varepsilon + 1$.



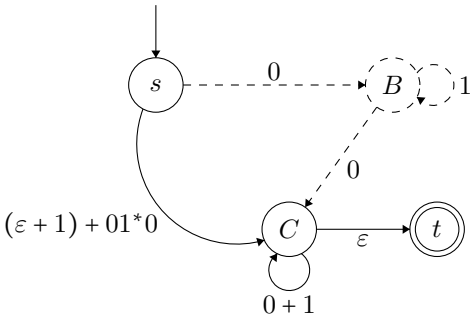
Избавимся от пустых множеств, ε в конкатенации и удалим состояние A.



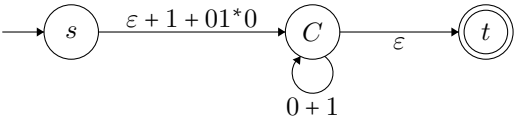
Сожмём теперь состояние B:

$$d'(s,C) = d(s,C) + d(s,B) \cdot d(B,B)^* \cdot d(B,C) = (\varepsilon + 1) + 01^*0,$$

прочие переходы не изменились.

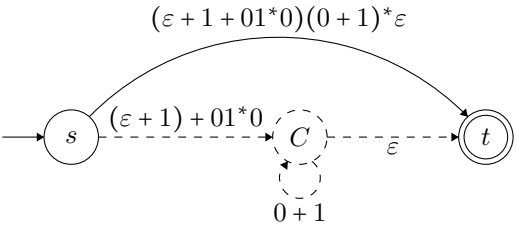


Удалим B

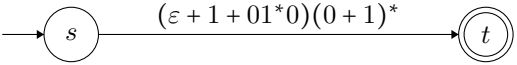


Сожмём состояние C:

$$d'(s,t) = d(s,t) + d(s,C) \cdot d(C,C)^* \cdot d(C,t) = \varnothing + (\varepsilon + 1 + 01^*0)(0+1)^*\varepsilon.$$



Избавимся от пустых множеств, ε в конкатенации и удалим состояние C, получив ОКА с двумя состояниями.



Искомое регулярное выражение $(\varepsilon + 1 + 01^*0)(0+1)^*$. Оно эквивалентно $(0+1)^*$ и описывает язык всех слов над $\{0,1\}$.

Заключение

Настоящее пособие представляет собой введение в теорию регулярных языков. Определяются и описываются стандартные способы задания регулярного языка: детерминированные и недетерминированные конечные автоматы, а также регулярные выражения. Рассмотренные конструкции снабжены примерами. Доказана теорема об эквивалентности различных представлений регулярного языка. 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.

Учебное издание

Шестаков Сергей Алексеевич

Регулярные языки: основные конструкции

Учебно-методическое пособие

Редакторы: *Н. Е. Кобзева*, *И. А. Волкова*
Компьютерная верстка *Н. Е. Кобзевой*

Подписано в печать 07.07.2025. Формат 60×84 $^{1}/_{16}$.
Усл. печ. л. 3,0. Уч.-изд. л. 2,1. Тираж 20 экз. Заказ № 108.

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