Теоретическая информатика III, осень 2018 г. Формальные языки и действия над ними. Регулярные выражения. Детерминированные и недетерминированные конечные автоматы. Детерминизация недетерминированных автоматов, преобразование регулярных выражений к автоматам*

Александр Охотин

29 сентября 2018 г.

Содержание

1 Формальные языки и действия над ними

Символьная строка — это самое естественное для человека представление данных. Символы из заранее заданного набора идут один за другим — такова, например, человеческая речь как последовательность фонем, таковы книги, таковы файлы на компьютере — и так можно представить любые данные.

Сперва задаётся конечное множество *символов* Σ, называемое *алфавитом.* В абстрактных примерах элементы Σ обозначаются строчными латинскими буквами из начала алфавита (a, b, ...).

Строка (англ. string) над алфавитом Σ — это конечная последовательность $w = a_1 \ldots a_\ell$, где $\ell \geqslant 0$, и $a_1, \ldots, a_\ell \in \Sigma$ — символы.[1] Строки обычно обозначаются латинскими буквами

*Краткое содержание лекций, прочитанных студентам 2-го курса СПбГУ, обучающимся по программе «математика», в осеннем семестре 2018–2019 учебного года. Страница курса: `http://users.math-cs.spbu.ru/~okhotin/teaching/tcs3_2018/`.

[1] Чистые математики любят называть строки «словами» (words); в информатике же это наименование не используется, и в этом курсе его не будет. В очень старых учебниках строки иногда называют «цепочками» — но так говорить уж точно не стоит.

w (дубль-вэ[2]), u, v, x, y и z. Например, $w = abb$ — это 3-символьная строка над алфавитом, содержащим символы a и b.

Число символов в строке называется её *длиной*, $|w| = \ell$. Существует единственная строка длины 0, называемая *пустой строкой* и обозначаемая через ε. Множество всех строк над алфавитом Σ обозначается через Σ^*.

Основная операция над строками — *конкатенация*, то есть, приписывание одной строки вслед за другой[3]. Если $u = a_1 \dots a_m$ и $v = b_1 \dots b_n$ — две строки, то их конкатенация — это строка $u \cdot v = uv = a_1 \dots a_m b_1 \dots b_n$.

Пример 1. *У строки $abab$ восемь различных подстрок: ε, a, b, ab, ba, aba, bab, $abab$. У неё пять префиксов (ε, a, ab, aba, $abab$) и пять суффиксов (ε, b, ab, bab, $abab$).*

Обращение строки w, обозначаемое через w^R — это та же самая строка, записанная в обратном порядке: если $w = a_1 \dots a_\ell$, то $w^R = a_\ell \dots a_1$. В частности, $\varepsilon^R = \varepsilon$.

Множества строк называют «формальными языками» или просто «языками». Если Σ — алфавит, то Σ^* — множество всех строк над ним, и языком называется всякое подмножество Σ^*.

Пусть $K, L \subseteq \Sigma^*$. Так как языки — это множества, для них определены обычные действия над множествами.

$$K \cup L = \{\, w \mid w \in K \text{ или } w \in L \,\} \qquad \text{(объединение } K \text{ и } L\text{)}$$
$$K \cap L = \{\, w \mid w \in K \text{ и } w \in L \,\} \qquad \text{(пересечение } K \text{ и } L\text{)}$$

Для языка L, определённого над алфавитом Σ, его *дополнением* называется дополнение до Σ^* — то есть, язык $\overline{L} = \Sigma^* \setminus L$.

$$\overline{L} = \{\, w \mid w \in \Sigma^*, \quad w \notin L \,\} \qquad \text{(дополнение } L\text{)}$$

Конкатенация двух языков, K и L — это язык, состоящий из всех возможных конкатенаций строки из K и строки из L.

$$KL = K \cdot L = \{\, uv \mid u \in K \text{ и } v \in L \,\} \qquad \text{(конкатенация } K \text{ и } L\text{)}$$

Операции конкатенации и объединения обладают свойством *дистрибутивности*, то есть, $K(L \cup M) = KL \cup KM$ и $(K \cup L)M = KM \cup LM$ для всех языков $K, L, M \in \Sigma^*$. Формальные языки образуют *полукольцо* с конкатенацией в качестве умножения и объединением в качестве сложения.

Так как конкатенация языков считается умножением, конкатенация k экземпляров одного и того же языка называется её *k-й степенью*.

$$L^k = \underbrace{L \cdot \ldots \cdot L}_{k \text{ раз}} = \{\, w_1 \dots w_k \mid w_1, \dots, w_k \in L \,\}$$

В частности, $L^0 = \{\varepsilon\}$ для всякого языка L, что соответствует свойству $x^0 = 1$ для чисел.

Следующее действие над языком L — *повторение 0 и более раз* — задаёт множество всех строк, получаемых конкатенаций любого числа любых строк из L. Эта операция выражается следующим образом.

$$L^* = \bigcup_{k=0}^{\infty} L^k = \{\, w_1 \dots w_k \mid k \geqslant 0,\ w_1, \dots, w_k \in L \,\}$$

[2]На худой конец, «дабл-ю». Но, ради Бога, ни в коем случае не «омега»! Омега (ω, Ω) — это совсем другая буква, и ею обозначают совсем другие объекты!

[3]К сожалению, общепринятого русского названия нет — это, конечно, позор на весь мир, но что делать?

Операция повторения была введена Клини [1951], и её часто называют *замыканием Клини*, или *звёздочкой Клини*, или просто «звёздочкой».[4]

2 Регулярные выражения

Строятся с помощью трёх операторов: *выбор*, *конкатенация*, *повторение*.

Формальное определение: сперва, какой вид может иметь регулярное выражение (его *синтаксис*), затем — определение языка, задаваемого всяким регулярным выражением.

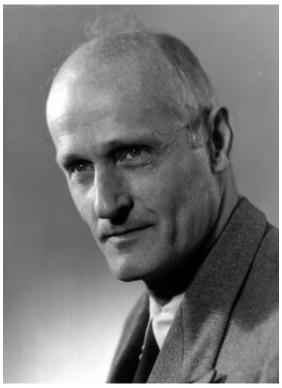

Рис. 1: Стивен Клини (1909–1994).

Определение 2.1 (Клини [1951]). *Регулярные выражения над алфавитом Σ определяются так.*

- *Символ для пустого множества $\varnothing$ — регулярное выражение.*
- *Всякий символ a, где $a \in \Sigma$ — регулярное выражение.*
- *Если α и β — регулярные выражения, то тогда $(\alpha \mid \beta)$, $(\alpha\beta)$ и $(\alpha)^*$ — тоже регулярные выражения.*

Всякое регуларное выражение α определяет язык над алфавитом Σ, обозначаемый через $L(\alpha)$. Символ для пустого множества определяет пустое множество.

$$L(\varnothing) = \varnothing$$

Всякий символ из Σ обозначает одноэлементное множество, состоящее из односимвольной строки.

$$L(a) = \{a\}$$

Оператор выбора задаёт объединение множеств.

$$L(\alpha \mid \beta) = L(\alpha) \cup L(\beta)$$

[4]Формальные языки с операциями объединения, конкатенации и звёздочки — это основной пример *алгебры Клини* — абстрактной алгебраической структуры, представляющей собою полукольцо, расширенное оператором замыкания, удовлетворяющим определённым аксиомам. В данном курсе это понятие не потребуется, да и сами полукольца не потребуются тоже.

Конкатенация регулярных выражений задаёт конкатенацию языков. Оператор итерации задаёт итерацию.

$$L(\alpha\beta) = L(\alpha) \cdot L(\beta)$$
$$L(\alpha^*) = L(\alpha)^*$$

Лишние скобки можно опускать, используя следующий порядок действий: сперва итерация, затем конкатенация, затем выбор. Например, регулярное выражение $(a \mid bc^*)d$ читается как $(a \mid (b(c^*)))d$.

Синтаксис регулярных выражений можно расширить лишними конструкциями: пустая строка (ε), повторение один и более раз (α^+), необязательная конструкция ($[\alpha]$, что означает "α или ничего"). Всё это можно выразить в терминах определения 2.1. Пустая строка: $\varnothing^* = \{\varepsilon\}$. Повторение один и более раз (α^+) — как $\alpha\alpha^*$. Необязательная конструкция $[\alpha]$ — как $\alpha \mid \varepsilon$, и в конечном счёте как $\alpha \mid \varnothing^*$. Например, $a^+b \mid \varepsilon$ — это сокращённая запись для $aa^*b \mid \varnothing^*$.

Пример 2. *Множество всех строк над алфавитом $\Sigma = \{a, b\}$, в которых третий символ с конца — a, задаётся регулярным выражением $(a \mid b)^*a(a \mid b)(a \mid b)$.*

Пример 3. *В языках программирования, **имена** обычно определяются как непустые последовательности из букв и цифр, начинающиеся с буквы.*

$$\underbrace{(\mathtt{a} \mid \ldots \mid \mathtt{z})}_{\text{любая буква}}\underbrace{(\mathtt{a} \mid \ldots \mid \mathtt{z} \mid \mathtt{0} \mid \ldots \mid \mathtt{9})}_{\text{любая буква или цифра}}{}^*$$

3 Детерминированные конечные автоматы (DFA)

Конечный автомат — простейшая модель вычисления в теоретической информатике, соответствующая вычислениям с конечной памятью.

Конечный автомат читает входную строку посимвольно слева направо, не возвращаясь назад. Закончив читать строку, автомат выдаёт ответ «да» или «нет».

Понятие *состояния*: конфигурация памяти всего автомата в некоторый момент. Можно сказать, что у автомата с 32 состояниями — 5 битов памяти; однако никакой битовой структуры на самом деле нет, на каждом шаге можно перейти из любого состояния памяти в любое. Число состояний конечно, отсюда название.

Конечный автомат начинает работу в определённом *начальном состоянии* q_0, головка видит самый левый символ строки. На каждом шаге текущее состояние и текущий символ определяют состояние в следующий момент, и головка переезжает на одну клетку направо. Состояние после чтения последнего, самого правого символа определяет, принимается строка или отвергается.

Определение 3.1 (Клини [1951]). ***Детерминированный конечный автомат*** *(deterministic finite automaton, DFA; мн. число: automata) — пятёрка $\mathcal{A} = (\Sigma, Q, q_0, \delta, F)$, со следующим значением компонентов.*

- Σ — *алфавит (конечное множество).*
- Q — *конечное множество состояний.*
- $q_0 \in Q$ — *начальное состояние.*
- *Функция переходов* $\delta\colon Q \times \Sigma \to Q$. *Если автомат находится в состоянии* $q \in Q$ *и читает символ* $a \in \Sigma$, *то его следующее состояние* — $\delta(q, a)$. *Функция переходов определена для всех* q *и* a.
- *Множество принимающих состояний* $F \subseteq Q$.

Для всякой входной строки $w = a_1 \dots a_\ell$, *где* $\ell \geqslant 0$ *и* $a_1, \dots, a_\ell \in \Sigma$, *вычисление — последовательность состояний* $p_0, p_1, \dots, p_{\ell-1}, p_\ell$, *где* $p_0 = q_0$, *и всякое следующее состояние* p_i, *где* $i \in \{1, \dots, \ell\}$, *однозначно определено как* $p_i = \delta(p_{i-1}, a_i)$.

$$p_0 \xrightarrow{a_1} p_1 \xrightarrow{a_2} \dots \xrightarrow{a_{\ell-1}} p_{\ell-1} \xrightarrow{a_\ell} p_\ell$$

Строка ***принимается****, если последнее состояние* p_ℓ *принадлежит множеству* F *— иначе* ***отвергается****.*

Язык, распознаваемый автоматом, обозначаемый через $L(\mathcal{A})$ *— это множество всех строк, которые он принимает.*

Дополнительное обозначение: если DFA начинает вычисление в состоянии $q \in Q$ и читает строку $w = a_1 \dots a_\ell$, то его состояние после её прочтения обозначается через $\delta(q, w)$. Формально, определение функции переходов расширяется до $\delta\colon Q \times \Sigma^* \to Q$, и определяется как $\delta(q, \varepsilon) = q$ и $\delta(q, aw) = \delta(\delta(q, a), w)$. В этих обозначениях язык, распознаваемый DFA, кратко определяется так.

$$L(\mathcal{A}) = \{\, w \in \Sigma^* \mid \delta(q_0, w) \in F \,\}$$

Автоматы удобно представлять в виде диаграмм переходов, таких как на рис. 2. Из каждого состояния по каждому символу выходит ровно одна стрелка. Начальное состояние обозначается стрелкой, ведущей из ниоткуда, принимающие состояния обведены в кружок.

Пример 4. *Язык* $a^*b^* = \{\, a^i b^j \mid i, j \geqslant 0 \,\}$, *определённый над алфавитом* $\Sigma = \{a, b\}$, *распознаётся конечным автоматом* $\mathcal{A} = (\Sigma, Q, q_0, \delta, F)$ *с состояниями* $Q = \{q_0, q_1, q_2\}$. *Начав работу в состоянии* q_0, *автомат остаётся в* q_0 *при прочтении* a, *и переходит в* q_1, *как только встретится первый символ* b. *Это обеспечивается следующими переходами.*

$$\delta(q_0, a) = q_0$$
$$\delta(q_0, b) = q_1$$

Далее, в состоянии q_1 *при чтении* b *автомат остаётся в этом же состоянии. Если же в состоянии* q_1 *когда-нибудь встретится* a, *это значит, что строка — не вида* a^*b^*, *и потому автомат переходит в состояние* q_2, *в котором он отвергнет эту строку.*

$$\delta(q_1, a) = q_2$$
$$\delta(q_1, b) = q_1$$

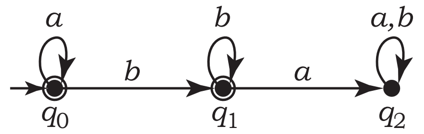

Рис. 2: DFA из примера 4, распознающий язык a^*b^*.

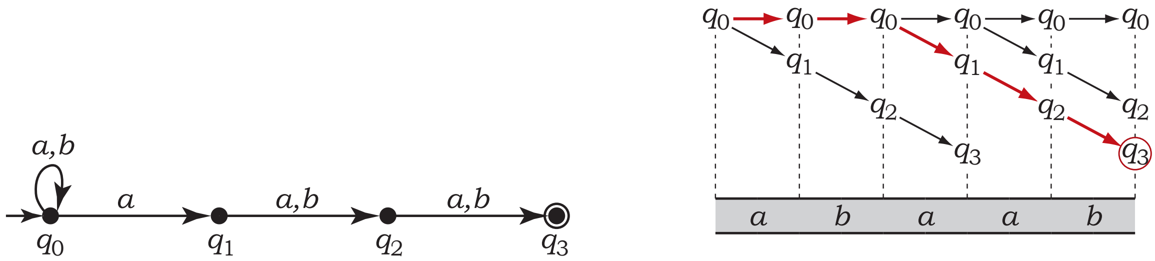

Рис. 3: (слева) NFA из примера 5, угадывающий третий символ с конца; (справа) четыре вычисления на строке $w = abaab$, из которых одно — принимающее.

Наконец, в состоянии q_2 автомат просто дочитывает строку до конца, хотя уже известно, что принята она не будет.

$$\delta(q_2, a) = \delta(q_2, b) = q_2$$

Множество принимающих состояний — $F = \{q_0, q_1\}$.

Автомат изображён на рис. 2.

4 Недетерминированные конечные автоматы (NFA)

Недетерминированный конечный автомат (nondeterministic finite automaton, NFA) — обобщение детерминированного автомата, в котором переход в данной конфигурации может быть определён не единственным образом. В некоторых состояниях и по некоторым символам, у NFA может быть определено несколько возможных переходов, и, соответственно, на одной строке может быть несколько различных вычислений. *Если хотя бы одно из этих вычислений — принимающее*, то тогда считается, что NFA принимает эту строку; в противном случае, *если все эти вычисления — отвергающие*, то строка отвергается.

Это абстрактное определение можно понимать так. Если в некоторой конфигурации допустимо несколько различных переходов, и при некотором выборе действия на этом шаге вычисление может увенчаться успехом, то можно сказать, что NFA обладает способностью «угадать» это правильное продолжение — своего рода «интуицией». Вообще, если существует последовательность недетерминированных решений, заканчивающаяся принятием, то тогда NFA «угадывает» эту последовательность — так, что строка отвергается только если она не может быть принята ни для какой последовательности догадок. Иными словами, *догадки автомата всегда правильны.*

В общем случае, такая способность угадывать правильное продолжение не обязательно будет иметь физическую реализацию. Однако для случая конечных автоматов будет показано, что NFA и DFA равномощны — то есть, способны распознавать один и тот же класс языков.

Пример 5 (ср. с примером 2)**.** *Язык всех строк над алфавитом $\Sigma = \{a, b\}$, в которых третий символ с конца — a, распознаётся NFA на рис. 3(левый).*

Его вычисления на строке $abaab$ представлены на рис. 3(правый); в их числе — одно принимающее вычисление.


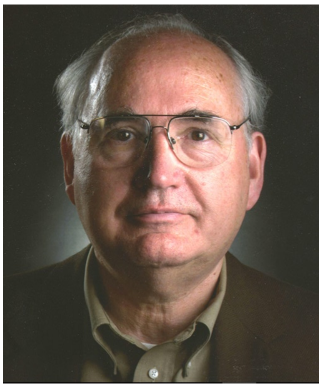

Рис. 4: Майкл Рабин (род. 1931), Дана Скотт (род. 1932).

Определение 4.1 (Рабин и Скотт [1959])**.** ***Недетерминированный конечный автомат*** *(NFA) — это пятёрка $\mathcal{B} = (\Sigma, Q, Q_0, \delta, F)$, со следующим значением компонентов.*

- *Σ — алфавит.*
- *Q — конечное множество состояний.*
- *$Q_0 \in Q$ — множество начальных состояний (автомат угадывает, в каком из них начать).*
- *Функция переходов $\delta\colon Q \times \Sigma \to 2^Q$ даёт множество возможных следующих состояний. Находясь в состоянии $q \in Q$ и читая символ $a \in \Sigma$, автомат может перейти в любое состояние из $\delta(q, a)$.*
- *Множество* ***принимающих состояний*** *$F \subseteq Q$.*

Вычисление на входной строке $w = a_1 \ldots a_\ell$ — это всякая последовательность состояний $p_0, p_1, \ldots, p_{\ell-1}, p_\ell$, где $p_0 \in Q_0$, и $p_i \in \delta(p_{i-1}, a_i)$. для всех $i \in \{1, \ldots, \ell\}$,

$$p_0 \xrightarrow{a_1} p_1 \xrightarrow{a_2} \ldots \xrightarrow{a_{\ell-1}} p_{\ell-1} \xrightarrow{a_\ell} p_\ell$$

Вычисление — ***принимающее****, если последнее состояние p_ℓ принадлежит множеству F; иначе оно называется* ***отвергающим****.*

Язык, распознаваемый автоматом, обозначаемый через $L(\mathcal{B})$ — это множество всех входных строк, на которых есть хотя бы одно принимающее вычисление.

В этих обозначениях. NFA в примере 5 имеет множество состояний $Q = \{q_0, q_1, q_2, q_3\}$, где q_0 — единственное начальное состояние ($Q_0 = \{q_0\}$), q_3 — единственное принимающее

$(F = \{q_3\})$, а функция переходов δ принимает следующие значения.

$$\begin{aligned} \delta(q_0, a) &= \{q_0, q_1\} \\ \delta(q_0, b) &= \{q_0\} \\ \delta(q_1, a) = \delta(q_1, b) &= \{q_2\} \\ \delta(q_2, a) = \delta(q_2, b) &= \{q_3\} \\ \delta(q_3, a) = \delta(q_3, b) &= \varnothing \end{aligned}$$

Тогда принимающее вычисление на $w = abaab$ обозначается так.

$$q_0 \xrightarrow{a} q_0 \xrightarrow{b} q_0 \xrightarrow{a} q_1 \xrightarrow{a} q_2 \xrightarrow{b} q_3$$

DFA может рассматриваться как особый случай NFA, с единственным начальным состоянием ($|Q_0| = 1$) и с единственным следующим состоянием в каждом переходе ($|\delta(q, a)| = 1$ для всех q и a).

5 Построение подмножеств (перевод NFA в DFA)

«Интуиция», которой обладает NFA, может быть механически воплощена в материальном мире.

NFA отличается от DFA тем, что может иметь несколько различных вычислений на одной и той же входной строке. Все эти вычисления проходят те же символы в том же порядке, и отличаются только в состояниях. Можно считать, что все они происходят одновременно. Например, как показано на рис. 3(правом), после чтения первых трёх символов есть возможных вычисления, находящиеся в состояниях q_0, q_1 и q_3, соответственно. DFA может вычислить *множество* этих состояний.

Для NFA на рис. 3(левом), моделирующий его работу DFA, читая ту же самую строку $w = abaab$ пройдёт через следующую последовательность состояний-подмножеств: $\{q_0\}$, $\{q_0, q_1\}$, $\{q_0, q_2\}$, $\{q_0, q_1, q_3\}$, $\{q_0, q_1, q_2\}$, $\{q_0, q_2, q_3\}$. Их нетрудно видеть на рис. 3(правом).

Следующая лемма даёт общее построение.

Лемма 1 («построение подмножеств», Рабин и Скотт [1959])**.** *Пусть $\mathcal{B} = (\Sigma, Q, Q_0, \delta, F)$ — произвольный NFA. Тогда существует DFA $\mathcal{A} = (\Sigma, 2^Q, Q_0, \delta', F')$, состояния которого — подмножества Q, который распознаёт тот же язык, что и $\mathcal{B}$. Его переход в каждом состоянии-подмножестве $S \subseteq Q$ по каждому символу $a \in \Sigma$ ведёт во множество состояний, достижимых по a из некоторого состояния их S.*

$$\delta'(S, a) = \bigcup_{q \in S} \delta(q, a)$$

Состояние-подмножество $S \subseteq Q$ — принимающее, если оно содержит хотя бы одно принимающее состояние NFA.

$$F' = \{\, S \mid S \subseteq Q,\ S \cap F \neq \varnothing \,\}$$

Общий вид утверждения: по одному вычислительному устройству строится второе, и поведение построенного устройства выражается через поведение исходного устройства. Доказательства подобных результатов обычно начинаются с *утверждения о правильности*

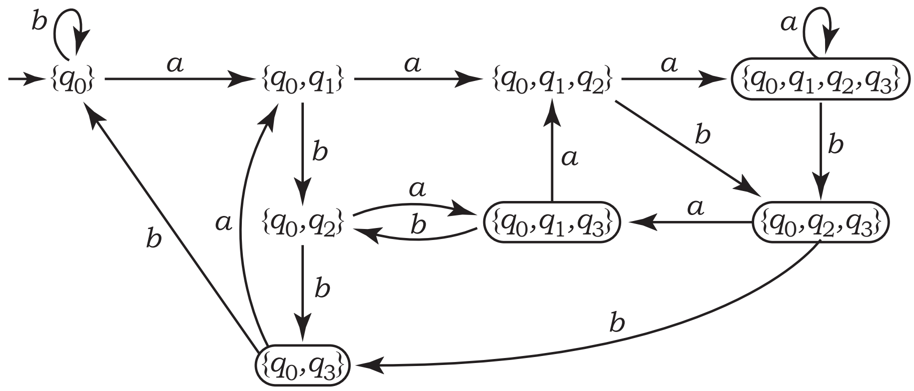

Рис. 5: DFA, моделирующий работу NFA из примера 5, полученный построением подмножеств (все состояния, содержащие q_3 — принимающие).

— подробного математического утверждения, описывающего, что новое устройство делает на каждом шаге, и как это связано с работой исходного устройства. Как правило, именно утверждение о правильности содержит в себе главный смысл построения, а его доказательство бывает неинтересным.

Доказательство.

Утверждение о правильности. *Для всякой строки $w \in \Sigma^*$, состояние-подмножество, достигаемое DFA по прочтении строки w, содержит элемент q тогда и только тогда, когда хотя бы одно из вычислений NFA на w заканчивается в состоянии q.*

Доказывается индукцией по длине строки w.

Далее из утверждения о правильности выводится, что построенный DFA принимает строку $w \in \Sigma^*$ тогда и только тогда, когда её принимает исходный NFA. □

Построение переводит NFA с n состояниями в DFA с 2^n состояниями-подмножествами. На практике, многие из них обычно бывают недостижимы. Поэтому алгоритм будет строить только состояния-подмножества, достижимые из уже построенных, начиная с Q_0.

Пример 6. *Построение подмножеств, применённое к NFA с 4 состояниями из примера 5, производит DFA с 8 достижимыми состояниями, представленный на рис 5.*

Можно заметить, каждое состояние-подмножество по существу кодирует три последних прочитанных символа (q_i принадлежит ему, если i-й символ с конца — это a).

В худшем случае построение оптимально по числу состояний: Лупанов [1963] построил, для всякого n, такой NFA над $\{a, b\}$ из n состояний, что всякий DFA для этого языка должен содержать хотя бы 2^n состояний.

Было показано, что DFA и NFA определяют одно и то же семейство языков, называемых *регулярными языками*. Следующий результат: что регулярные выражения определяют то же семейство.

6 Преобразование регулярных выражений в автоматы

Теорема 1 (Клини [1951])**.** *Язык распознается неким DFA тогда и только тогда, когда он определяется неким регулярным выражением.*

Рис. 6: Олег Лупанов (1932–2006).

Доказательство конструктивно в обе стороны.

Для преобразования регулярного выражения в автомат удобно ввести промежуточную модель.

Определение 6.1. Недетерминированный конечный автомат с ε-переходами *(ε-NFA) — пятёрка $\mathcal{C} = (\Sigma, Q, Q_0, \delta, F)$, где Σ, Q, Q_0 и F — как в NFA, а функция переходов имеет вид $\delta \colon Q \times (\Sigma \cup \{\varepsilon\}) \to 2^Q$. В состоянии $q \in Q$ автомат может или сделать обычный переход по символу $a \in \Sigma$ в любое состояние из $\delta(q, a)$, перемещая головку на символ вперёд—или же перейти в любое состояние из $\delta(q, \varepsilon)$, не читая ничего (*ε-переход*).*

Вычисление на w может потребовать более $|w|$ шагов. Строка принимается, если есть разбиение $w = u_1 \dots u_m$, где $u_i \in \Sigma \cup \{\varepsilon\}$, и последовательность состояний $r_0, \dots, r_m \in Q$, для которых: $r_0 = q_0$, всякое следующее r_i принадлежит $\delta(r_{i-1}, u_i)$, и $r_n \in F$.

Использование ε-переходов не увеличивает мощности конечных автоматов.

Лемма 2. *Для всякого ε-NFA существует NFA, использующий то же множество состояний и распознающий тот же язык.*

Доказательство. Пусть $\mathcal{C} = (\Sigma, Q, Q_0, \delta, F)$ — произвольный ε-NFA. Для всякого состояния $q \in Q$, множество состояний, достижимых из него за 0 и более ε-переходов, обозначается через $\varepsilon\text{-closure}(q) \subseteq Q$. Главная мысль построения: NFA будет проделывать произвольную последовательность ε-переходов за один шаг.

Первый вариант построения. Определяется NFA $\mathcal{B} = (\Sigma, Q, Q_0, \delta', F')$, всякий переход которого за один шаг выполняет любую последовательность ε-переходов в $\mathcal{C}$, и затем переход по одному символу.

$$\delta'(p, a) = \bigcup_{q \in \varepsilon\text{-closure}(p)} \delta(q, a) \qquad\qquad (p \in Q,\ a \in \Sigma)$$

Если $\mathcal{C}$ может придти по ε-переходам из $p \in Q$ в некоторое принимающее состояние, то p помечается как принимающее в $\mathcal{B}$.

$$F' = \{\, p \mid \varepsilon\text{-closure}(p) \cap F \neq \varnothing \,\}$$

Утверждение о правильности. *Исходный ε-NFA $\mathcal{C}$ может достигнуть состояния $q \in Q$ по прочтении строки $w \in \Sigma^*$ тогда и только тогда, когда построенный NFA $\mathcal{B}$, прочитав w, может достигнуть некоторого состояния p, где $q \in \varepsilon\text{-closure}(p)$.*

Второй вариант построения. Строится автомат $\mathcal{B}' = (\Sigma, Q, Q_0', \delta', F)$. На этот раз в автомате $\mathcal{B}'$ начальными будут все состояния, которые достижимы в $\mathcal{C}$ из его начальных состояний по ε-переходам.

$$Q_0' = \bigcup_{q_0 \in Q_0} \varepsilon\text{-closure}(q_0)$$

Всякий переход $\mathcal{B}$ по символу a начинается с перехода $\mathcal{C}$ по этому же символу, а вслед за тем выполняется любая последовательность ε-переходов.

$$\delta'(p, a) = \bigcup_{q \in \delta(p,a)} \varepsilon\text{-closure}(q) \qquad\qquad (p \in Q,\ a \in \Sigma)$$

Множество принимающих состояний остаётся тем же.

Утверждение о правильности. *Исходный ε-NFA $\mathcal{C}$ может достигнуть состояния $q \in Q$ по прочтении строки $w \in \Sigma^*$ тогда и только тогда, когда построенный NFA $\mathcal{B}'$, прочитав w, может достигнуть того же состояния состояния q.*

$\square$

Регулярные выражения легко выражаются в этой модели.

Лемма 3 («построение Томпсона»)**.** *Для всякого регулярного выражения α, существует ε-NFA $\mathcal{C}_\alpha$ с одним начальным и одним принимающим состояниями, распознающий язык, задаваемый α.*

Рис. 7: Кеннет Томпсон (род. 1943).

Доказательство. Индукция по структуре регулярного выражения, пять случаев представлены на рис. 8. $\square$

Известно также прямое преобразование регулярного выражения в DFA, но это по существу построение Томпсона и построение подмножеств, объединённые в одно маловразумительное определение.

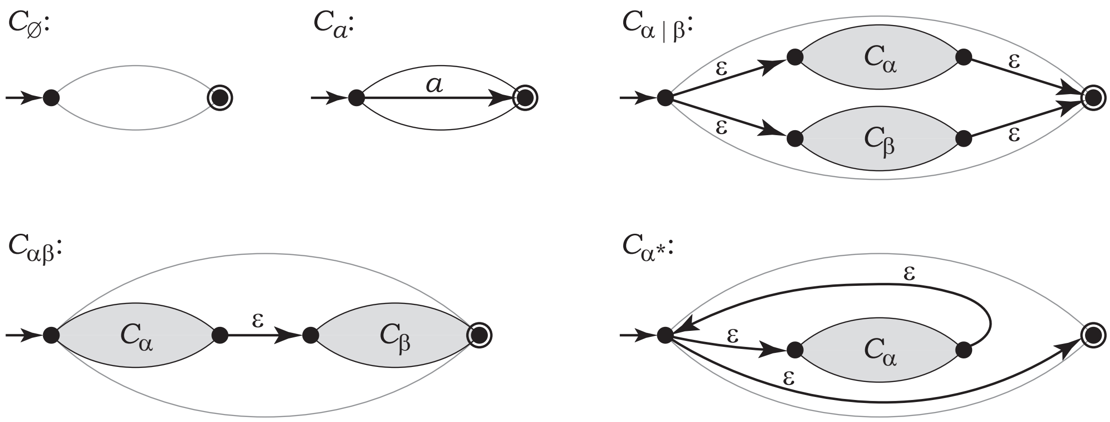

Рис. 8: Преобразование регулярного выражения в ε-NFA.

Список литературы

[1951] S. C. Kleene, “Representation of events in nerve nets and finite automata”, *RAND Research Memorandum* RM-704, 1951, 98 pp.

[1963] О. Б. Лупанов, “О сравнении двух типов конечных источников”, *Проблемы кибернетики*, 9 (1963), 321–326.

[1959] M. O. Rabin, D. Scott, “Finite automata and their decision problems”, *IBM Journal of Research and Development*, 3:2 (1959), 114–125.