Теоретическая информатика, осень 2020 г. Лекция 2. Преобразование NFA в DFA («построение подмножеств»). Действия над формальными языками, их реализация над конечными автоматами. Регулярные выражения. Преобразование регулярных выражений в автоматы. Преобразование автоматов в регулярные выражения. Конечные автоматы над односимвольным алфавитом. Нерегулярный язык, удовлетворяющий лемме о накачке*

Александр Охотин

10 сентября 2020 г.

Содержание

1 Преобразование NFA в DFA («построение подмножеств»)

«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.

---

*Краткое содержание лекций, прочитанных студентам 2-го курса факультета МКН СПбГУ в осеннем семестре 2020–2021 учебного года. Страница курса: `http://users.math-cs.spbu.ru/~okhotin/teaching/tcs_fl_2020/`.

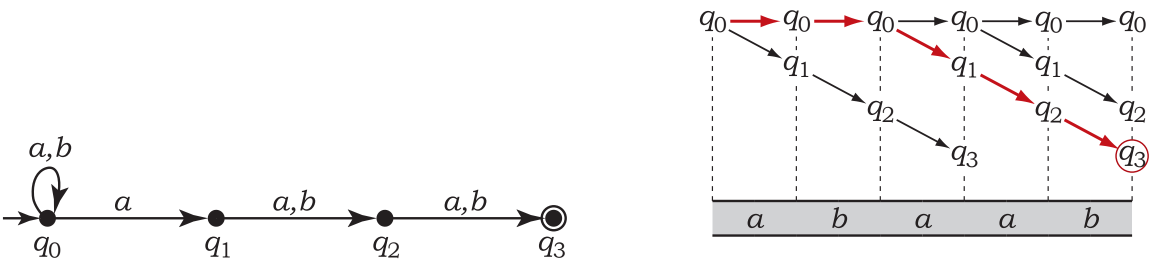


Рис. 1: (слева) NFA из прошлой лекции, угадывающий третий символ с конца; (справа) четыре вычисления на строке $w = abaab$, из которых одно — принимающее.

Например, как показано на рис. 1(правом), после чтения первых трёх символов есть три возможных вычисления, находящиеся в состояниях q_0, q_1 и q_3, соответственно. DFA может вычислить *множество* этих состояний.

Для NFA на рис. 1(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. 1(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* — подробного математического утверждения, описывающего, что новое устройство делает на каждом шаге, и как это связано с работой исходного устройства. Обыкновенно, 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.

Базовый случай: $w = \varepsilon$**.** Тогда DFA достигает своё начальное состояние-подмножество Q_0, и все вычисления NFA длины 0 начинаются и заканчиваются в состояниях из Q_0.

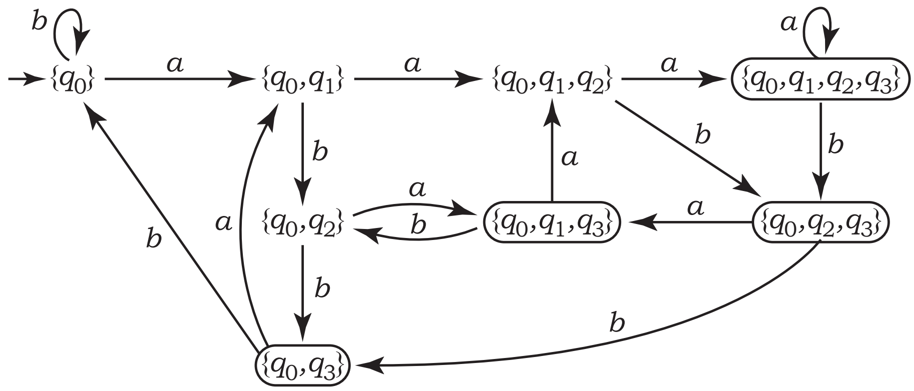


Рис. 2: DFA, моделирующий работу NFA из примера **??**, полученный построением подмножеств (все состояния, содержащие q_3 — принимающие).

Переход: $w = ua$, где $a \in \Sigma$. По предположению индукции, состояние-подмножество $S \subseteq Q$, в котором DFA заканчивает читать строку u, состоит ровно из тех состояний, в которые NFA может придти, прочитав u.

Тогда NFA может придти в состояние q, прочитав строку ua, в любом из состояний из $\bigcup_{q \in S} \delta(q, a)$ — а это и есть в точности состояние-подмножество, в которое DFA перейдёт из S по a.

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 1. *Построение подмножеств, применённое к NFA с 4 состояниями из примера* **??**, *производит DFA с 8 достижимыми состояниями, представленный на рис 2.*

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.

Доказать немного худшую нижнюю оценку 2^{n-1} легко.

Пример 2. *Для всякого $n \geqslant 2$, язык всех строк над алфавитом $\{a, b\}$, в которых $(n-1)$-й символ с конца — a, распознаётся NFA с n состояниями, однако всякий DFA для этого языка содержит не менее чем 2^{n-1} состояний.*

Доказательство стоит привести, как образец доказательства нижней оценки размера DFA для данного языка.

Доказательство. Построение NFA — это обобщение примера **??**.

Рис. 3: Олег Лупанов (1932–2006).

Пусть есть DFA $\mathcal{A} = (\{a, b\}, Q, q_0, \delta, F)$ с менее чем 2^{n-1} состояниями, который распознаёт тот же язык. Тогда существуют какие-то две различных строки длины $n-1$, прочитав которые, автомат приходит в одно и то же состояние $q \in Q$. Пусть эти строки отличаются в i-м символе, то есть, имеют вид uav и xby. Тогда строка $uava^{i-1}$ принадлежит языку, а строка $xbya^{i-1}$ — не принадлежит. Однако автомат заканчивает чтение этих двух строк в одном и том же состоянии $\delta(q, a^{i-1})$ — и потому или принимает обе, или отвергает обе. Получено противоречие. $\square$

Было показано, что DFA и NFA определяют одно и то же семейство языков. Такие языки называются *регулярными языками*.

2 Действия над формальными языками

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

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

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

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

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

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

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*.

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

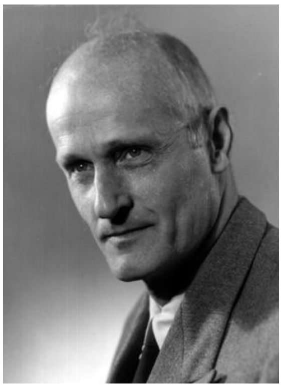


Рис. 4: Стивен Клини (1909–1994).

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.

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

Операция повторения была введена Клини [1951], и её часто называют *замыканием Клини*, или *звёздочкой Клини*, или просто «звёздочкой».[1]

3 Регулярные выражения

Регулярное выражение — это формула, задающая язык, построенная с помощью трёх операторов — *выбора*, *конкатенации* и *повторения* («звёздочки»), применённых к элементарным языкам $\varnothing$ и $\{a\}$, где a — символ алфавита.

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

Определение 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.

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

[1]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.

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

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

Оператор выбора задаёт объединение множеств.

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

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 $(a \mid (b(c^*)))d$ и задаёт, соответственно, язык $(\{a\} \cup (\{b\} \cdot (\{c\}^*))) \cdot \{d\}$.

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

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

Стоит заметить, что в инженерной практике программирования «регулярными выражениями» часто называют одно из расширений выражений из определения 1. В этих моделях определены неочевидные дополнительные операторы, позволяющие задавать некоторые нерегулярные синтаксические конструкции, но требующие значительного времени для анализа. В общем и целом, это весьма бестолковые модели, редко полезные на практике и не заслуживающие внимания с теоретической точки зрения.

Оказывается, что регулярные выражения равномощны конечным автоматам. Это свойство весьма ценно и с практической точки зрения, поскольку позволяет механически преобразовывать описания, подобные примеру 4, в программу, в конечные автоматы, разбирающие тексты.

4 Преобразование регулярных выражений в автоматы

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

Доказательство конструктивно в обе стороны.

Для преобразования регулярного выражения в автомат удобно ввести промежуточную модель.

Определение 2. 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 $w = u_1 \ldots u_m$, где $u_i \in \Sigma \cup \{\varepsilon\}$, и последовательность состояний $r_0, \ldots, r_m \in Q$, для которых: $r_0 \in Q_0$, 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, выполнять переход по символу $a \in \Sigma$, заодно проделывать последовательность ε-переходов.

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 $q \in \varepsilon\text{-closure}(p)$.*

Рис. 5: Кеннет Томпсон (род. 1943).

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.

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

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.*

□

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.*

Доказательство. Индукция по структуре регулярного выражения, пять случаев представлены на рис. 6. □

Пример 5. *На рис. 7 показано преобразование регулярного выражения $(ab \mid a)^*b$ — сперва, по лемме 3, в ε-NFA, собранный из кусков индукцией по структуре выражения, а затем в NFA по лемме 2, используя первый вариант построения.*

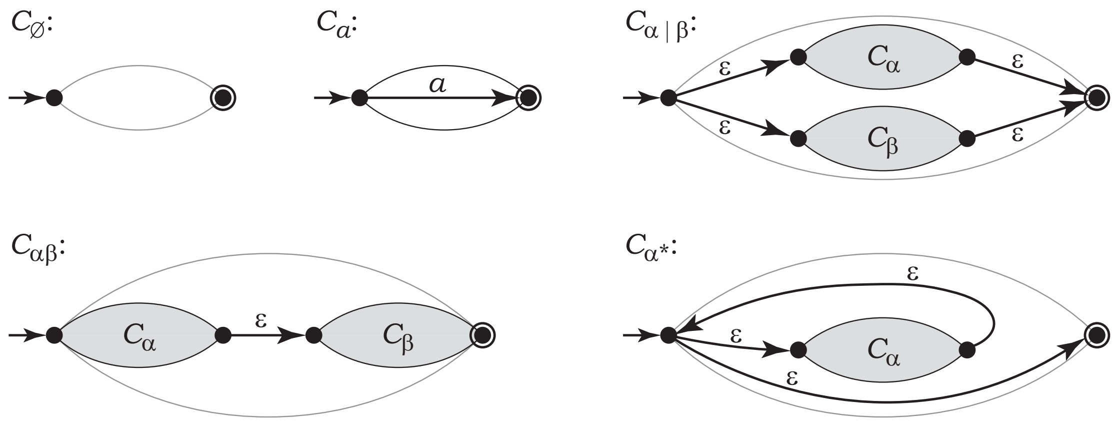


Рис. 6: Преобразование регулярного выражения в ε-NFA.

Упражнение 1. *Перевести полученный NFA в DFA с помощью построения подмножеств.*

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

5 Преобразование автоматов в регулярные выражения

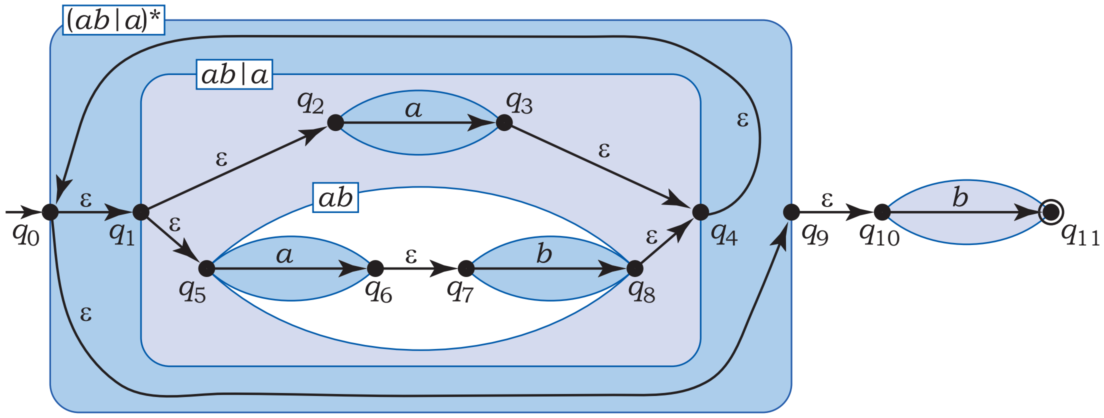
Чтобы завершить доказательство теоремы Клини, осталось для всякого DFA построить регулярное выражение, задающее тот же язык. Для доказательства удобно ввести следующую промежуточную модель.

Определение 3. Недетерминированный конечный автомат с переходами по регулярным выражениям *(RE-NFA) — это пятёрка $\mathcal{D} = (\Sigma, Q, q_0, R, q_f)$, где Σ и Q — как в NFA, $q_0 \in Q$ — единственное начальное состояние, $q_f \in Q$ — единственное принимающее состояние, а вместо функции переходов используется функция $R\colon Q \times Q \to RE(\Sigma)$, определяющая регулярное выражение над алфавитом Σ для всякой пары состояний. Переход из p в q возможен по любой строке, определяемой регулярным выражением $R(p, q)$.*

Формально RE-NFA принимает строку w, если существует её разбиение $w = u_1 \dots u_m$ и последовательность состояний $r_0, \dots, r_m \in Q$, где $r_0 = q_0$, всякая подстрока u_i определяется регулярным выражением $R(r_{i-1}, r_i)$, и $r_n = q_f$.

Лемма 4 (Бжозовский и Маккласки [1963])**.** *Для всякого RE-NFA существует регулярное выражение, которое определяет язык, распознаваемый этим RE-NFA.*

Доказательство. Построение ведётся методом *удаления состояний* (state elimination). Автомат постепенно преобразуется, на каждом шаге удаляется одно состояние, а регулярные выражения на оставшихся переходах постепенно усложняются. В итоге не остаётся никаких состояний — одно большое регулярное выражение.



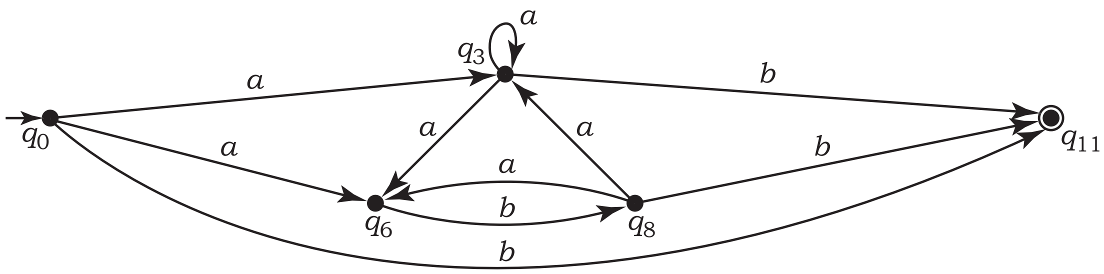


Рис. 7: Преобразование регулярного выражения $(ab \mid a)^*b$ (сверху) в ε-NFA, (снизу) и далее в NFA по первому варианту построения.

Сперва удобно обеспечить следующее свойство: в начальное состояние нельзя вернуться, а из принимающего состояния нельзя никуда перейти. Для этого достаточно добавить новое начальное и новое принимающие состояния, соединив их с имеющимися ε-переходами.

Далее, на каждом шаге, пусть состояние q — не начальное и не принимающее. Всякий раз, когда оно используется в каком-то вычислении, автомат приходит в него из некоторого состояния p, потом крутится в q ноль или более раз, и наконец покидает его, переходя в некоторое состояние r (которое может совпадать с p). Пусть регулярные выражения на этих переходах таковы: $\alpha = R(p, q)$, $\theta = R(q, q)$ и $\beta = R(q, r)$, как на рис. 9(левом). Тогда вычисление из p в r через q описывается регулярным выражением $\alpha\theta^*\beta$. Если текущее значение $R(p, r)$ — γ, то $R(p, r)$ можно переопределить как $\gamma \mid \alpha\theta^*\beta$, заменяя тем самым путь через q одним переходом. После того как это проделывается для всех пар из p и r, состояние q более не нужно и может быть удалено.

В итоге удаляются все состояния, кроме начального и принимающего, причём в начальное состояние так и не ведут никакие переходы, а из принимающего, соответственно, никакие переходы не ведут дальше. Поэтому единственный оставшийся переход ведёт из начального в принимающее состояние. Написанное на нём регулярное выражение — и есть искомое. □

6 Действия над регулярными языками

Вопрос о представимости действий над регулярными языками. Пусть есть операция над формальными языками, такая как их конкатенация, объединение, пересечение, и т.д. Имея некоторым образом представленные языки (автоматами, регулярными выражениями), нередко бывает нужно получить представление для результата применения этой операции к данным языкам. Для каждой операции прежде всего возникает вопрос, всегда ли это возможно? Если при применении операции к регулярным языком результат всегда регулярен, то

Рис. 8: Януш Бжозовский (1934–2019) и Эдвард Маккласки (1929–2016).

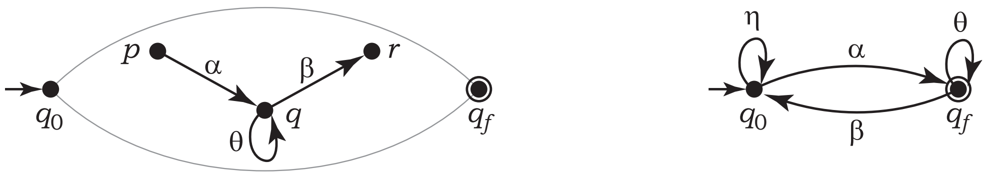


Рис. 9: Преобразование конечного автомата в регулярное выражение: (слева) удаление состояний; (справа) RE-NFA из 2 состояний.

говорится, что *регулярные языки замкнуты относительно операции*, или что *операция сохраняет класс регулярных языков*.

Регулярные языки замкнуты относительно почти всех очевидных операций над языками, а также относительно многочисленных неочевидных.

Замкнутость класса регулярных языков относительно дополнения доказывается очень простым построением. По данному DFA строится новый DFA с теми же состояниями и переходами, который проделывает то же вычисление, что и исходный, в конце строки приходят в то же самое состояние; но затем он примет, если исходный автомат отвергает, и отвергнет, если исходный автомат принимает. Для этого достаточно поменять местами принимающие и отвергающие состояния.

Утверждение 1. *Для всякого DFA $\mathcal{A} = (\Sigma, Q, q_0, \delta, F)$, DFA $\mathcal{A}' = (\Sigma, Q, q_0, \delta, Q \setminus F)$ распознаёт дополнение $L(\mathcal{A})$.*

Замкнутость класса регулярных языков относительно пересечения можно получить с помощью следующего построения.

Теорема 2 («прямое произведение автоматов»)**.** *Для всяких двух DFA $\mathcal{A} = (\Sigma, P, p_0, \eta, E)$ и $\mathcal{B} = (\Sigma, Q, q_0, \delta, F)$, пересечение $L(\mathcal{A})$ и $L(\mathcal{B})$ распознаётся DFA $\mathcal{C}$ со множеством состояний $P \times Q$.*

Доказательство. Для входной строки $w \in \Sigma^*$, новый DFA $\mathcal{C}$ одновременно выполняет вычисления $\mathcal{A}$ и $\mathcal{B}$ на той же самой строке w. Когда $\mathcal{C}$ находится в состоянии (p, q), где $p \in P$ и $q \in Q$, это значит, что моделируемое вычисление $\mathcal{A}$ находится в состоянии p, а вычисление $\mathcal{B}$ — в состоянии q. Поэтому автомат определяется как $\mathcal{C} = (\Sigma, P \times Q, (p_0, q_0), \pi, E \times F)$, где

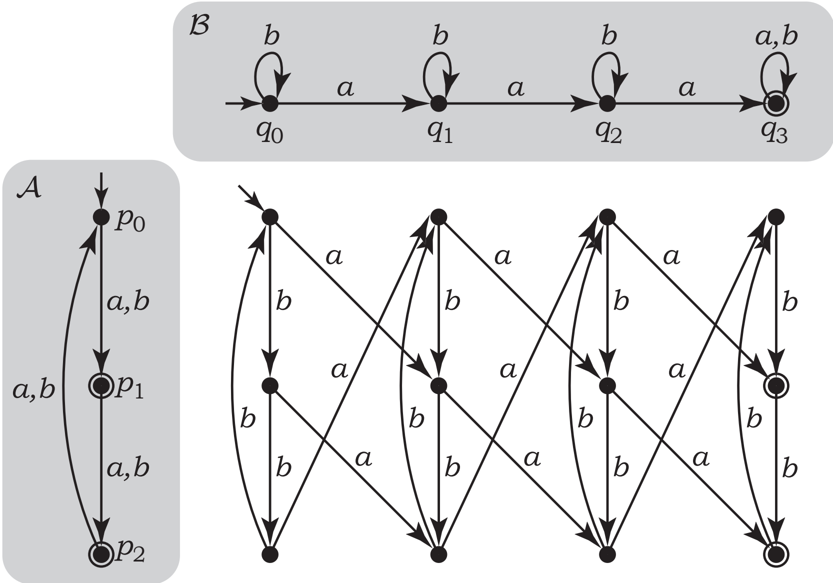


Рис. 10: Автоматы $\mathcal{A}$ и $\mathcal{B}$ из примера 6, и их прямое произведение — DFA, распознающий пересечение $L(\mathcal{A}) \cap L(\mathcal{B})$.

функция переходов π применяет функции переходов $\mathcal{A}$ и $\mathcal{B}$, каждую к своему компоненту пары.

$$\pi((p, q), a) = (\eta(p, a), \delta(q, a))$$

В конце вычисления строка принимается тогда и только тогда, когда каждый из двух моделируемых автоматов завершил работу в одном из своих принимающих состояний. $\square$

Пример 6. *Пусть $\mathcal{A}$ — DFA автомат с 3 состояниями, распознающий множество всех строк над алфавитом $\Sigma = \{a, b\}$, длина которых не делится на три, а $\mathcal{B}$ — DFA с 4 состояниями, распознающий множество всех строк над тем же алфавитом, которые содержат хотя бы три символа a. Эти автоматы и их прямое произведение, построенное в соответствии с теоремой 2, приведены на рис. 10.*

Точно такое же построение работает и для пересечения NFA. Построение нетрудно переделать, чтобы получить объединение DFA. Объединение NFA, равно как и конкатенация и звёздочка, делается проще — как при преобразовании регулярных выражений к автомату.

Пример нестандартной операции, относительно которой регулярные языки тоже замкнуты: *поэлементный квадратный корень*, $\sqrt{L} = \{\, w \mid ww \in L \,\}$. Сама по себе эта операция достаточно надуманна, однако пример интересен тем, что в нём используется важный метод построения конечных автоматов.

Теорема 3. *Для всякого DFA $\mathcal{A} = (\Sigma, Q, q_0, \delta, F)$ поэлементный квадратный корень $\sqrt{L(A)}$ распознаётся DFA $\mathcal{B}$ со множеством состояний $Q^Q = \{\, f \mid f\colon Q \to Q \,\}$.*

Для каждой строки $w \in \Sigma^*$, если $\mathcal{A}$ начинает своё вычисление на w в состоянии $q \in Q$, то пусть состояние, в котором он завершает чтение w, обозначается через $f_w(q)$. Тогда f_w — это функция $f_w\colon Q \to Q$, называемая *поведением $\mathcal{A}$ на w*.

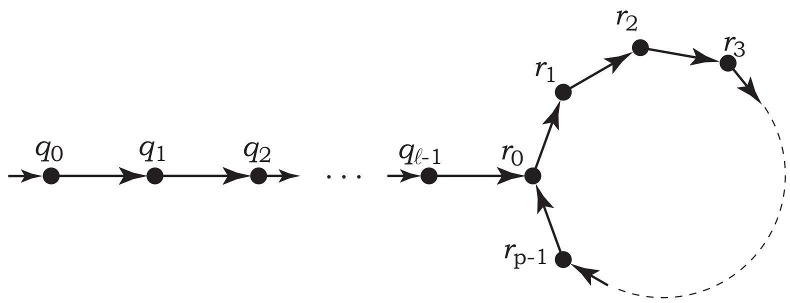


Рис. 11: Переходы DFA над односимвольным алфавитом.

Лемма 5. *Для всякого DFA $\mathcal{A} = (\Sigma, Q, q_0, \delta, F)$ существует DFA $\mathcal{B}$ со множеством состояний $Q^Q = \{ f \mid f\colon Q \to Q \}$, вычисляющий поведение $\mathcal{A}$ на прочитанной строке.*

Доказательство. Начальное состояние $\mathcal{B}$ — тождественная функция на множестве Q, то есть, поведение $\mathcal{A}$ на пустой строке.

Чтобы определить переходы, вводится обозначение $\delta_a\colon Q \to Q$ для *функции переходов $\mathcal{A}$ по каждому символу* $a \in \Sigma$ — то есть, $\delta_a(q) = \delta(q, a)$. Тогда поведение $\mathcal{A}$ на строке wa, где $w \in \Sigma^*$ и $a \in \Sigma$ — это композиция поведения на w и функции переходов по a. Тогда $\mathcal{B}$ вычисляет эту композицию на каждом шаге: $\delta'(f, a) = \delta_a \circ f$. □

Доказательство теоремы 3. Новый автомат $\mathcal{B}$, будучи запущенным на w, вычисляет поведение $\mathcal{A}$ на w. После этого достаточно определить множество принимающих состояний как $F' = \{ f \mid f(f(q_0)) \in F \}$. □

Маслов [1970] показал, что в худшем случае n^n состояний необходимы.

7 Конечные автоматы над односимвольным алфавитом

Переходы DFA над односимвольным алфавитом $\Sigma = \{a\}$ образуют конечный граф со степенью исхода 1, и потому последовательность переходов, начинающаяся в начальном состоянии, со временем переходит в *цикл*, как показано на рис. 11. Ноль или более состояний между начальным состоянием и циклом называются *хвостом* DFA. Длина цикла называется *периодом*. Соответственно, автомат полностью описывается длиной цикла (ℓ), периодом (p) и множеством принимающих состояний.

Односимвольные или *унарные* языки можно рассматривать как множества натуральных чисел, и регулярные унарные языки — это множества, представимые в виде объединения конечного множества арифметических прогрессий (принимающие состояния в цикле) и просто конечного множества (принимающие состояния в хвосте).

Как показано Любичем [1964] и Хробаком [1986], для преобразования унарного NFA с n состояниями в DFA достаточно и в худшем случае необходимо $g(n) + O(n^2)$ состояний, где $g(n)$ — *функция Ландау* (Ландау [1903]).

$$g(n) = \max\{ \operatorname{lcm}(p_1, \dots, p_k) \mid k \geqslant 1,\ p_1 + \dots + p_k \leqslant n \} = e^{(1+o(1))\sqrt{n \ln n}}$$

7.1 Нерегулярный язык, удовлетворяющий лемме о накачке

Условие леммы о накачке — это необходимое, но не достаточное условие регулярности языка. Нерегулярный язык в следующем примере удовлетворяет этому условию, однако прямого применения леммы о накачке недостаточно, чтобы доказать его регулярность.

Пример 7. *Язык* $L = \{ (ab)^n a^n \mid n \geqslant 1 \} \cup (\{a, b\}^* \setminus (ab)^+ a^+)$ *нерегулярен, однако он удовлетворяет условию леммы о накачке с константой* $p = 3$.

Доказательство. Для доказательства нерегулярности можно воспользоваться замкнутостью относительно пересечения. Пусть L регулярен. Тогда его пересечение с языком $(ab)^+ a^+$ также должно быть регулярно. Однако, это пересечение — это язык $\{ (ab)^n a^n \mid n \geqslant 1 \}$, который не удовлетворяет лемме о накачке и потому оказывается нерегулярным. Полученное противоречие доказывает нерегулярность языка L.

Чтобы проверить условие леммы о накачке с константой $p = 3$, для всякой строки из L длины не менее чем 3 необходимо построить её разложение вида xyz, для которого все строки вида $xy^i z$ принадлежат L. Разложение определяется в зависимости от первых трёх символов строки — так, чтобы ни одна из накачанных строк $xy^i z$ не попала в $(ab)^+ a^+$.

- Строка $abw \in L$, где $w \in \{a, b\}^*$, разбивается на $x = \varepsilon$, $y = a$, $z = bw$: тогда строка, полученная после накачки, начинается или с b, или с aa.
- Строка $bbw \in L$ разбивается на $x = \varepsilon$, $y = b$, $z = bw$: тогда после накачки она будет начинаться с b.
- Строка $satw \in L$, где $s, t \in \{a, b\}$ — её первый и третий символы, разбивается на $x = sa$, $y = t$, $z = w$: накачанная строка начинается с sa, и потому, как и во всех предыдущих случаях, принадлежит L.

Стало быть, искомое разложение xyz существует для любой строки из L длины хотя бы 3, и условие леммы о накачке выполняется. □

Список литературы

[1963] J. A. Brzozowski, E. J. McCluskey, "Signal flow graph techniques for sequential circuit state diagrams", *IEEE Transactions on Electronic Computers*, 12:2 (1963), 67–76.

[1986] M. Chrobak, "Finite automata and unary languages", *Theoretical Computer Science*, 47 (1986), 149–158; errata: 302:1–3 (2003), 497–498.

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

[1903] E. Landau, "Über die Maximalordnung der Permutationen gegebenen Grades" (О максимальном порядке перестановок данного числа элементов), *Archiv der Mathematik und Physik, Ser. 3*, 5 (1903), 92–103.

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

[1964] Ю. Любич, "Оценки для оптимальной детерминизации недетерминированных автономных автоматов", *Сибирский математический журнал*, 5:2 (1964), 337–355.

[1970] A. Н. Маслов, “Оценки числа состояний конечных автоматов”, *Доклады Академии наук СССР*, 194:6 (1970), 1266–1268.

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