Теоретическая информатика, осень 2020 г. Лекция 1. Введение в предмет ТИ: машины Тьюринга, неразрешимые задачи. Формальные языки. Детерминированные конечные автоматы и их ограничения. Регулярная лемма о накачке. Недетерминированные конечные автоматы*

Александр Охотин

5 сентября 2020 г.

Содержание

Предисловие

Теоретическая информатика — это область науки, находящаяся на стыке математики и информатики. С одной стороны, теоретическая информатика — это полноценный раздел математики, вдохновлённый задачами обработки информации. В этой области точно так же дают определения, формулируют и доказывают теоремы — и в них заключены определённые математические идеи, проявляется характерная для этой области разновидность математического мышления. Это мышление может оказаться нужным при решении задач в разных областях математики, и потому теоретическая информатика — это полезная составляющая образования всякого математика.

С другой стороны, теоретическая информатика изучает всю совокупность математических идей, лежащих в основе технологий обработки информации. Изучая и развивая эти идеи, можно понять существующие технологии и изобрести новые. Потому теоретическая информатика — это важная часть фундаментального образования для специалистов в области информатики.

*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/`.

1 Строки и языки

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*, из которых составляются строки. Это множество называется *алфавитом* и обозначается буквой Σ. В абстрактных примерах элементы Σ обозначаются строчными латинскими буквами из начала алфавита (a, b, ...).

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

Количество вхождений символа a в строку w обозначается через $|w|_a$.

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 $u \cdot v = uv = a_1 \dots a_m b_1 \dots b_n$. Конкатенация — это *умножение* строк, а пустая строка, соответственно, *единица*. Соответственно, конкатенация нескольких экземпляров одной строки — это возведение в степень, обозначаемое стандартно: $w^0 = \varepsilon$, $w^1 = w$, $w^2 = ww$, и т.д.

Строка u называется *префиксом* строки w, если $w = uv$ для некоторой строки v. Аналогично, v — *суффикс* w, если $w = uv$. Строка y — *подстрока* w, если $w = xyz$.

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 (reversal) 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, $\varepsilon^R = \varepsilon$. Любопытно, что эта операция встречается в теории музыки, где называется «ракоход».

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

2 Формализация понятия вычисления: машина Тьюринга

Машина Тьюринга: математическая модель вычислений, производимых человеком с помощью карандаша и бумаги, руководствуясь некоторым конечным набором формальных правил.

Предмет вычисления: или вычислить функцию $f \colon \Sigma^* \to \Sigma^*$, то есть, по данной на входе строке w вычислить строку $f(w)$, или распознать принадлежность данной на входе строки множеству $L \subseteq \Sigma^*$ и дать ответ «да» или «нет».

---

[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.

[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 поделать?

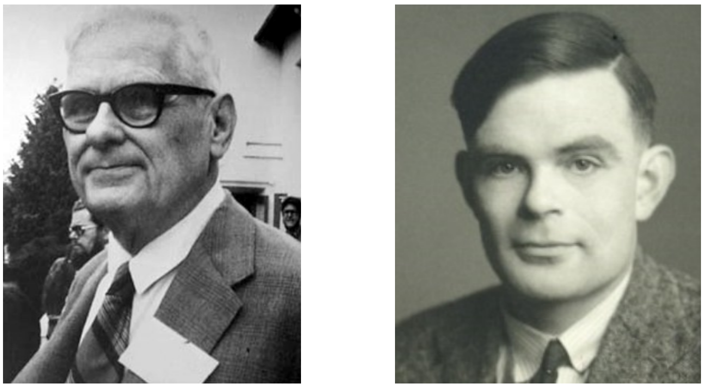

Рис. 1: Алонзо Чёрч (Church) (1903–1995), Алан Тьюринг (Turing) (1912–1954).

В определении машины Тьюринга «бумага» — одномерная, то есть, строка символов произвольной длины, записанная на *ленте*. Лента состоит из *клеток*, и в каждой записывается один символ. Лента полагается неограниченной, т.е., на ней всегда можно найти клетки для записи новых символов, и справа и слева от уже используемых клеток. «Карандаш» становится *головкой*, которая ездит по ленте и в каждый момент времени видит символ в одной клетке. На каждом шаге головка может написать на месте текущего символа любой другой и переместиться на одну клетку вправо или влево. Правила, которыми руководствуется машина, называются её *функцией переходов*.

Тезис Чёрча–Тьюринга: интуитивно вычислимые функции — это в точности функции, вычислимые на машине Тьюринга.

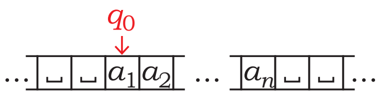

Рис. 2: Лента машины Тьюринга в начальный момент работы на входной строке $w = a_1 \ldots a_n$ (начальная конфигурация $q_0 a_1 \ldots a_n$).

Определение 1 (Тьюринг [1937])**.** *Машина Тьюринга — это семёрка* $\mathcal{M} = (\Sigma, \Gamma, Q, q_0, \delta, q_{acc}, q_{rej})$*, компоненты которой имеют следующий вид.*

- *Конечное множество* Σ *—* ***входной алфавит****.*
- *Другое конечное множество* Γ *—* ***рабочий алфавит****, содержащий все символы, допустимые на ленте, причём* $\Sigma \subset \Gamma$*, Рабочий алфавит содержит особый символ — пробел:* $\textvisiblespace \in \Gamma$, $\textvisiblespace \notin \Sigma$.
- *Конечное множество* Q *— множество* ***состояний****.*
- *Есть* ***начальное состояние*** $q_0 \in Q$.
- ***Функция переходов*** $\delta\colon (Q \setminus \{q_{acc}, q_{rej}\}) \times \Gamma \to Q \times \Gamma \times \{-1, +1\}$ *определяет поведение машины на каждом шаге. Если машина находится в состоянии* $q \in Q$ *и обозревает символ* $a \in \Gamma$*, то* $\delta(q, a)$ *— это тройка* (q', a', d)*, где* $q' \in Q$ *— новое состояние,* a' *— символ, записываемый на ленте вместо* a*, и* $d \in \{-1, +1\}$ *направление перемещения головки*[4].

- *Если машина переходит в* ***принимающее состояние*** $q_{acc} \in Q$ *или в* ***отвергающее состояние*** $q_{rej} \in Q$*, то она останавливается.*

Лента бесконечна в обе стороны, начальная конфигурация: состояние q_0*, головка смотрит на первый символ входной строки.*

Конфигурация *машины Тьюринга — это строка вида* $\alpha \boldsymbol{q} a \beta$*, где* $\alpha, \beta \in \Gamma^*$*,* $a \in \Gamma$ *и* $q \in Q$*, означающая, что машина находится в состоянии* q*, головка обозревает указанный символ* a*, и на ленте записаны символы* $\alpha a \beta$*, окружённые бесконечным числом пробелов в обоих направлениях.*

На данной входной строке $w \in \Sigma^*$ *машина начинает своё вычисление в* ***начальной конфигурации*** $\boldsymbol{q_0} w$*, то есть, в состоянии* q_0 *и с символами* w *на ленте, окружённые пробелами в обоих направлениях* $(\ldots \text{␣␣} w \text{␣␣} \ldots)$*.*

На каждом шаге, если машина находится в конфигурации $\alpha \boldsymbol{q} a \beta$*, то её конфигурацию на следующем шаге однозначно определена в соответствии со значением функции переходов для текущего состояния* $q \in Q$ *и текущего символа* $a \in \Gamma$*. Если головка едет налево, следующая конфигурация имеет вид* $\alpha \boldsymbol{q'} b a' \beta$*, и это определяется так.*

$$\alpha b \boldsymbol{q} a \beta \vdash \alpha \boldsymbol{q'} b a' \beta, \qquad \text{если } \delta(q, a) = (q', a', -1)$$

Если же машина находится у левого края непустого содержимого ленты, то в следующей конфигурации дописывается новый пробел.

$$q a \beta \vdash \boldsymbol{q'} \text{␣} a' \beta, \qquad \text{если } \delta(q, a) = (q', a', -1)$$

Если машина едет направо, случай самого правого символа также рассматривается отдельно.

$$\begin{aligned} \alpha \boldsymbol{q} a b \beta &\vdash \alpha a' \boldsymbol{q'} b \beta, && \text{если } \delta(q, a) = (q', a', +1) \\ \alpha \boldsymbol{q} a &\vdash \alpha a' \boldsymbol{q'} \text{␣}, && \text{если } \delta(q, a) = (q', a', +1) \end{aligned}$$

Таким образом однозначно определяется конечная или бесконечная последовательность конфигураций, называемая ***вычислением*** *машины на строке* w*. Вычисление может или* ***остановиться*** *на некотором шаге, в том смысле, что машина перейдёт в принимающее или отвергающее состояние, или же оно может продолжаться бесконечно, в каковом случае говорится, что машина* ***зацикливается****. Строка* ***принимается*** *машиной Тьюринга* $\mathcal{M}$*, если машина останавливается на ней в принимающем состоянии. Множество,* распознаваемое *машиной — это множество всех строк, которые она принимает.*

$$L(\mathcal{M}) = \{\, w \mid \boldsymbol{q} w \vdash \ldots \vdash \alpha \boldsymbol{q_{acc}} a \beta \text{ для некоторых } \alpha, \beta, a \,\}$$

Множество L *разрешимо*, если существует машина Тьюринга, останавливающаяся на любом входе и распознающая L.

Функцию переходов $\delta \colon Q \times \Gamma \to Q \times \Gamma \times \{-1, +1\}$ обычно записывают в виде таблицы, где строки соответствуют состояниям, столбцы — символам рабочего алфавита, и в каждой клетке написана тройка вида *(новое состояние, записываемый символ, направление перемещения головки)*.

Пример 2. *Пусть $M = (\Sigma, \Gamma, Q, q_0, \delta, q_{acc}, q_{rej})$ — машина Тьюринга со входным алфавитом $\Sigma = \{a, b\}$, рабочим алфавитом $\Gamma = \{a, b, \text{\textvisiblespace}\}$ и множеством состояний $Q = \{q_0, q_1, q_2, q_3, q_{acc}, q_{rej}\}$, и с функцией переходов, задаваемых следующей таблицей.*

	a	b	$\text{\textvisiblespace}$
q_0	$(q_1, \text{\textvisiblespace}, +1)$	$(q_{rej}, b, +1)$	$(q_{acc}, \text{\textvisiblespace}, +1)$
q_1	$(q_1, a, +1)$	$(q_1, b, +1)$	$(q_2, \text{\textvisiblespace}, -1)$
q_2	$(q_{rej}, a, +1)$	$(q_3, \text{\textvisiblespace}, -1)$	$(q_{rej}, \text{\textvisiblespace}, +1)$
q_3	$(q_3, a, -1)$	$(q_3, b, -1)$	$(q_0, \text{\textvisiblespace}, +1)$

Эта машина останавливается на любом входе и принимает строки из множества $\{\, a^n b^n \mid n \geqslant 0 \,\}$.

В состоянии q_0 машина стирает первое a входной строки и затем движется направо до конца в состоянии q_1. Достигнув конца, машина стирает последнее b в состоянии q_2 и возвращается в начало строки в состоянии q_3. Тогда машина переходит в состояние q_0 и продолжает делать то же самое, пока или все символы входной строки будут стёрты (в этом случае машина принимает), или же будет получена строка неправильного вида, то есть, начинающаяся с b или заканчивающаяся на a. Последнее рано или поздно произойдёт для всякой входной строки не вида $a^n b^n$.

Вот так выглядит вычисление на строке $aabb$.

$$\ldots \text{\textvisiblespace}\ \text{\textvisiblespace}\ \overset{\overset{q_0}{\downarrow}}{a}\ a\ b\ b\ \text{\textvisiblespace}\ \text{\textvisiblespace} \ldots$$

$$\ldots \text{\textvisiblespace}\ \text{\textvisiblespace}\ \text{\textvisiblespace}\ \overset{\overset{q_1}{\downarrow}}{a}\ b\ b\ \text{\textvisiblespace}\ \text{\textvisiblespace} \ldots$$

$$\ldots \text{\textvisiblespace}\ \text{\textvisiblespace}\ \text{\textvisiblespace}\ a\ \overset{\overset{q_1}{\downarrow}}{b}\ b\ \text{\textvisiblespace}\ \text{\textvisiblespace} \ldots$$

$$\ldots \text{\textvisiblespace}\ \text{\textvisiblespace}\ \text{\textvisiblespace}\ a\ b\ \overset{\overset{q_1}{\downarrow}}{b}\ \text{\textvisiblespace}\ \text{\textvisiblespace} \ldots$$

$$\ldots \text{\textvisiblespace}\ \text{\textvisiblespace}\ \text{\textvisiblespace}\ a\ b\ b\ \overset{\overset{q_1}{\downarrow}}{\text{\textvisiblespace}}\ \text{\textvisiblespace} \ldots$$

$$\ldots \text{\textvisiblespace}\ \text{\textvisiblespace}\ \text{\textvisiblespace}\ a\ b\ \overset{\overset{q_2}{\downarrow}}{b}\ \text{\textvisiblespace}\ \text{\textvisiblespace} \ldots$$

$$\ldots \text{\textvisiblespace}\ \text{\textvisiblespace}\ \text{\textvisiblespace}\ a\ \overset{\overset{q_3}{\downarrow}}{b}\ \text{\textvisiblespace}\ \text{\textvisiblespace}\ \text{\textvisiblespace} \ldots$$

$$\ldots \text{\textvisiblespace}\ \text{\textvisiblespace}\ \text{\textvisiblespace}\ \overset{\overset{q_3}{\downarrow}}{a}\ b\ \text{\textvisiblespace}\ \text{\textvisiblespace}\ \text{\textvisiblespace} \ldots$$

$$\ldots \text{\textvisiblespace}\ \text{\textvisiblespace}\ \overset{\overset{q_3}{\downarrow}}{\text{\textvisiblespace}}\ a\ b\ \text{\textvisiblespace}\ \text{\textvisiblespace}\ \text{\textvisiblespace} \ldots$$

$$\ldots \text{\textvisiblespace}\ \text{\textvisiblespace}\ \text{\textvisiblespace}\ \overset{\overset{q_0}{\downarrow}}{a}\ b\ \text{\textvisiblespace}\ \text{\textvisiblespace}\ \text{\textvisiblespace} \ldots$$

$$\ldots \text{\textvisiblespace}\ \text{\textvisiblespace}\ \text{\textvisiblespace}\ \text{\textvisiblespace}\ \overset{\overset{q_1}{\downarrow}}{b}\ \text{\textvisiblespace}\ \text{\textvisiblespace}\ \text{\textvisiblespace} \ldots$$

$$\ldots \text{\textvisiblespace}\ \text{\textvisiblespace}\ \text{\textvisiblespace}\ \text{\textvisiblespace}\ b\ \overset{\overset{q_1}{\downarrow}}{\text{\textvisiblespace}}\ \text{\textvisiblespace}\ \text{\textvisiblespace} \ldots$$

$$\ldots \text{\textvisiblespace}\ \text{\textvisiblespace}\ \text{\textvisiblespace}\ \text{\textvisiblespace}\ \overset{\overset{q_2}{\downarrow}}{b}\ \text{\textvisiblespace}\ \text{\textvisiblespace}\ \text{\textvisiblespace} \ldots$$

$$\ldots \text{\textvisiblespace}\ \text{\textvisiblespace}\ \text{\textvisiblespace}\ \overset{\overset{q_3}{\downarrow}}{\text{\textvisiblespace}}\ \text{\textvisiblespace}\ \text{\textvisiblespace}\ \text{\textvisiblespace}\ \text{\textvisiblespace} \ldots$$

$$\ldots \text{\textvisiblespace}\ \text{\textvisiblespace}\ \text{\textvisiblespace}\ \text{\textvisiblespace}\ \overset{\overset{q_0}{\downarrow}}{\text{\textvisiblespace}}\ \text{\textvisiblespace}\ \text{\textvisiblespace}\ \text{\textvisiblespace} \ldots$$

В этот момент строка принимается.

В данном примере машина работает неожиданно медленно — за время $O(n^2)$, где n — длина входной строки. Однако можно доказать, что любой язык программирования высокого уровня можно скомпилировать в машину Тьюринга, причём не более чем с полиномиальным замедлением времени работы по сравнению с тем, как работает обычная программа. Это показывает, что машина Тьюринга и в наши дни остаётся правильной математической моделью вычисления.

Упражнение 1. *Построить машину Тьюринга, распознающую множество палиндромов над алфавитом* $\Sigma = \{a, b\}$, *то есть* $\mathrm{PAL} = \{\, w \mid w \in \{a, b\}^*,\ w = w^R \,\}$.

Можно доказать, что всякая одноленточная машина Тьюринга, распознающая множество палиндромов, работает за время $\Omega(n^2)$.

Полезность машины Тьюринга в том, что для неё можно математически доказать вещи, доказательство которых на таких моделях вычисления, как, скажем, язык C++, будет слишком трудоёмким. Действительно, всякое доказательство, говорящее о «любой программе» на C++, будет вынуждено включать в себя даже не стандарт языка C++, а целиком компилятор, запрограммированный в виде математических умозаключений — в то время как для машины Тьюринга всё умещается в определение 1.

Упражнение 2. *Построить машину Тьюринга, распознающую множество степеней двойки в унарной записи — то есть, язык* $L = \{\, a^{2^n} \mid n \geqslant 0 \,\} = \{a, aa, aaaa, aaaaaaaa, \ldots\}$ *над алфавитом* $\Sigma = \{a\}$.

Множество, не распознаваемое никакой машиной Тьюринга Можно построить язык, который не может распознать ни одна машина Тьюринга. Основная идея построения — подать на вход машине Тьюринга её собственную запись.

Пусть $M = (\Sigma, \Gamma, Q, q_0, \delta, q_{acc}, q_{rej})$ — машина Тьюринга. Её можно записать математическими значками на бумаге, как в примере 2; ничего не мешает записать её и в виде символьной строки — как может быть записана любая информация.

Описание машины Тьюринга M — это строка $\sigma(M) \in \{0, 1\}^*$, определяемая следующим образом. Пусть множество состояний — $Q = \{q_1, \ldots, q_n\}$ (если состояния назывались как-то иначе, их можно переименовать без ущерба для распознаваемого множества). Пусть входной алфавит — $\Sigma = \{a_1, \ldots, a_\ell\}$, рабочий алфавит — $\Gamma = \{a_1, \ldots, a_\ell, a_{\ell+1}, \ldots, a_m\}$, последний символ — пробел, $a_m = \llcorner\!\lrcorner$; начальное состояние — q_1, принимающее состояние — $q_{acc} = q_n$, отвергающее состояние — $q_{rej} = q_{n-1}$. Тогда машина кодируется следующим образом, где символ произведения означает несколько строк, записанных одна за одной.

$$\sigma(M) = 1^\ell 0 1^m 0 1^n 0 \Big(\prod_{\delta(q_i, a_j) = (q_{i'}, a_{j'}, d)} 1^i 0 1^j 0 1^k 0 1^r 0 1^{d+2} 0 \Big)$$

Под *размером* машины Тьюринга M, обозначаемом через $|M|$, понимается длина её записи, $|\sigma(M)|$.

Далее, всякой машине Тьюринга M, входной алфавит которой содержит хотя бы два символа, можно дать на входе *её собственное описание* $\sigma(M)$. Машина может принять своё собственное описание или не принять. Соответственно, определяются множества машин, принимающих свои собственные описания (L_1) и не принимающих (L_0).

$$L_1 = \{\, \sigma(M) \mid \sigma(M) \in L(M) \,\}$$
$$L_0 = \{\, \sigma(M) \mid \sigma(M) \notin L(M) \,\}$$

Множества L_0 и L_1 интересны тем, что они формализуют задачу анализа программ в её простейшем виде. И оказывается, что для полноценной модели вычисления эта задача всегда будет неразрешима.

Теорема 1. *Множество L_0 не распознаётся никакой машиной Тьюринга.*

Доказательство. Используется, по существу, диагональный метод Кантора.

Делается предположение, что существует машина Тьюринга M_0, распознающая множество $L(M_0) = L_0$. Примет ли машина M_0 свою собственную запись $\sigma(M_0)$?

- Если примет, то $\sigma(M_0) \in L(M_0) = L_0$, и потому, согласно определению L_0, строка $\sigma(M_0)$ не лежит в $L(M_0)$ — противоречие.
- Если же M_0 не примет $\sigma(M_0)$, то $\sigma(M_0) \notin L_0$, и потому строка $\sigma(M_0)$ должна лежать в $L(M_0)$ — точно такое же противоречие.

Стало быть, такой машины Тьюринга M_0 не существует. $\square$

С этого результата теоретическая информатика заявила о себе как дисциплина; поэтому было естественным начать курс именно с него.

3 Детерминированные конечные автоматы (DFA)

Конечный автомат — это 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*: содержимое 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 2 (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 $w = a_1 \ldots a_\ell$, где $\ell \geqslant 0$ и $a_1, \ldots, a_\ell \in \Sigma$, вычисление — последовательность состояний $p_0, p_1, \ldots, p_{\ell-1}, p_\ell$, где $p_0 = q_0$, и всякое следующее состояние p_i, где $i \in \{1, \ldots, \ell\}$, однозначно определено как $p_i = \delta(p_{i-1}, a_i)$.

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

*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 $w = a_1 \ldots a_\ell$, 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. 3. Из каждого состояния по каждому символу выходит ровно одна стрелка. Начальное состояние обозначается стрелкой, ведущей из ниоткуда, принимающие состояния обведены в кружок.

Пример 3. *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.

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

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. 3.

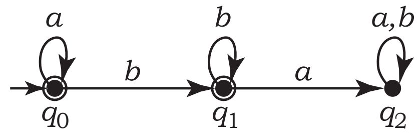

Рис. 3: DFA из примера 3, распознающий язык a^*b^*.

Упражнение 3. *Построить детерминированный конечный автомат, распознающий язык* $\{\, w \mid w \in \{a,b\}^*,\ |w|_a \not\equiv 0 \pmod 3 \,\}$ *всех строк, в которых число символов q не делится на 3.*

4 Доказательства нерегулярности

Пример 4. *Язык* $L = \{\, a^n b^n \mid n \geqslant 0 \,\}$ *не распознаётся никаким конечным автоматом.*

Доказательство. От противного: пусть он распознаётся некоторым DFA $\mathcal{A} = (\{a,b\}, Q, q_0, \delta, F)$. Для всякой строки a^i, где $i \geqslant 0$, пусть $q_i = \delta(q_0, a^i)$ — состояние DFA по прочтении этой строки.

Пусть $n = |Q|$. Тогда, читая первую половину строки $w = a^n b^n$, автомат пройдёт через последовательность из $n+1$ состояний $p_0, \ldots, p_n$, а после, читая вторую половину, ещё через состояния $p_{n+1}, \ldots, p_{2n}$. Так как строка принадлежит языку L, последнее состояние должно быть принимающим: $p_{2n} \in F$.

Поскольку всего различных состояний n, в последовательности $p_0, \ldots, p_n$ будет пара одинаковых состояний: то есть, существуют числа i и j, где $0 \leqslant i < j \leqslant n$, для которых p_i совпадает с p_j. Тогда вычисление на строке $w' = a^{n-(j-i)} b^n$ имеет вид $p_0, \ldots, p_i, p_{j+1}, \ldots, p_n, \ldots, p_{2n}$, то есть, автомат не замечает, что из строки вырезали кусок. Получается, что автомат принимает строку w', которая не лежит в L — противоречие. $\square$

Это доказательство обобщается до нижеследующей леммы. Надо сказать, что формулировка леммы сложнее её доказательства, однако в дальнейшем будут представлены более интересные утверждения с похожими формулировками, и этот простой случай поможет к ним подготовиться.

Лемма 1 (Рабин и Скотт [1959]: лемма о накачке для регулярных языков)**.** *Для всякого регулярного языка* $L \subseteq \Sigma^*$ *существует такая константа* $p \geqslant 1$, *что для всякой строки* $w \in L$ *длины не менее чем* p, *существует разложение* $w = xyz$, *где* y *непусто,* $|xy| \leqslant p$, *и* $xy^kz \in L$ *для всех* $k \geqslant 0$.

Доказательство. Пусть $A = (\Sigma, Q, q_0, \delta, F)$ — DFA, распознающий L, и пусть $p = |Q|$. Для произвольной строки $w \in L$, удовлетворяющей $|w| \geqslant p$, пусть $r_0, r_1, \ldots, r_{|w|} \in Q$ — вычисление A на w, то есть, всякое состояние r_i достигается по прочтении первых i символов w. Рассматривается начальный участок этой последовательности, $r_0, r_1, \ldots, r_p$. Так как этот участок содержит более чем $|Q|$ состояний, среди них есть пара одинаковых, $r_i = r_j$, где $0 \leqslant i < j \leqslant p$.

Обозначая первые i символов w через x, последние $|w|-j$ символов w через z, и средний участок w через y, получается разложение $w = xyz$, где $|y| = j - i$. Пусть $q = r_i = r_j$. Тогда

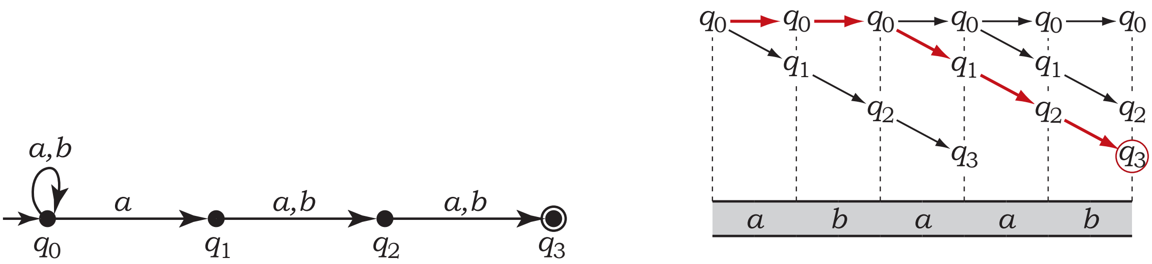

Рис. 4: (слева) NFA из примера 5, угадывающий третий символ с конца; (справа) четыре вычисления на строке $w = abaab$, из которых одно — принимающее.

$\delta(q_0, x) = q$, $\delta(q, y) = q$ и $\delta(q, z) = r_{|w|} \in F$. Всякая строка вида $xy^k z$ тогда принимается автоматом A, поскольку он посещает состояние q по прочтении каждого префикса вида xy^k. $\square$

Так как условия леммы о накачке выполняются для всякого регулярного языка, если для некоторого языка они не выполняются, то, следовательно, этот язык нерегулярен.

5 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.*

Важно: Недетерминизм не означает случайности, недетерминированный автомат никогда не делает случайного выбора! Использование случайности в алгоритме вполне реализуемо в физическом мире, в то время как недетерминизм, с его способностью угадать всегда правильно, остаётся абстрактной моделью.

Пример 5. *Язык всех строк над алфавитом $\Sigma = \{a, b\}$, в которых третий символ с конца — a, распознаётся NFA на рис. 4(левый).*

Его вычисления на строке abaab представлены на рис. 4(правый); в их числе — одно принимающее вычисление.

 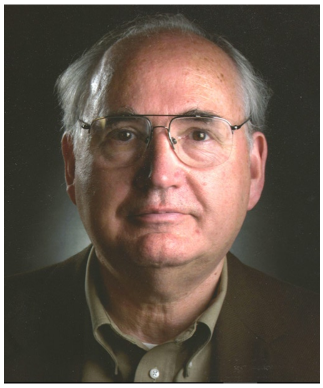

Рис. 5: Майкл Рабин (род. 1931), Дана Скотт (род. 1932).

Определение 3 (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.*
- *$Q_0 \subseteq Q$ — 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).

Упражнение 4. *Построить детерминированный конечный автомат, распознающий язык из примера 5 — язык всех строк, в которых третий символ с конца — a.*

Разумно задаться вопросом: что необходимо хранить в памяти на каждом шаге, чтобы в момент окончания строки ответить на вопрос о том, какой символ был прочитан три шага назад?

В общем случае, способность угадывать правильное продолжение вычисления 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.

Список литературы

[1936] A. Church, “An unsolvable problem of elementary number theory”, *American Journal of Mathematics*, 58 (1936), 345–363.

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

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

[1937] A. M. Turing, “On computable numbers, with an application to the Entscheidungsproblem”, *Proceedings of the London Mathematical Society, Series 2*, 42:1 (1937), 230–265.