Условные обозначения

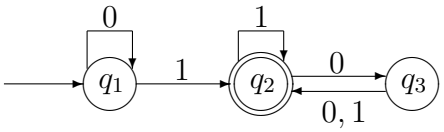
Далее в тексте будем использовать следующие термины и обозначения

- алфавитом Σ будет называться произвольное непустое множество;
- строкой w над алфавитом Σ будет называться конечный упорядоченный набор элементов алфавита:
$$w = x_1 x_2 \dots x_n,\ x_i \in \Sigma,\ i = \overline{1, n};$$
- пустая строка, т.е. строка, не содержащая ни одного символа, будет обозначаться через ε;
- через $\square$ будет обозначаться пустой символ;
- через $|w|$ будет обозначаться длина строки w, т.е. число символов, из которых состоит w;
- языком A будет называться любое множество строк;
- через $\overline{A}$ будет обозначаться дополнение языка A до множества всех возможных строк над выбранным алфавитом Σ;
- через $A \cap B$ будет обозначаться пересечение двух языков A и B;
- через $A \cup B$ будет обозначаться объединение двух языков A и B;
- через $A \times B$ будет обозначаться декартово произведение двух множеств A и B;
- через $\mathcal{P}(A)$ будет обозначаться множество всех подмножеств некоторого множества A.
- через оператор $\langle \cdot \rangle$ будет обозначаться описание некторого объекта в виде строки символов. Например, для МТ M через $\langle M \rangle$ будет обозначено её некоторая символьная запись (через «четвёрки», «пятёрки» или на конкретном языке программирования).

1 Конечные автоматы

Описание языков как множеств строк конечной длины обычно осуществляется посредством двух подходов: либо при помощи построения набора формальных правил, позволяющих построить произвольную строку языка, либо при помощи конструирования автоматов – вычислительных моделей, которые, выполняя некоторую последовательность действий (программу), позволяют определить, принадлежит ли строка языку. Первым рассмотренным классом языков будет являться класс регулярных языков, для описания которых используются достаточно простая вычислительная модель – конечный автомат. Поведение такого автомата определяется тем, в каком состоянии он находится и какой символ входной строки в данный момент считывает. В зависимости от этих факторов автомат переходит в следующее состояние, где производится чтение очередного символа. Графически такая программа иллюстрируется при помощи диаграммы состояний, где при помощи стрелок переходов указывается в какое состояние должен перейти автомат в зависимости от считанного символа. Итог работы автомата зависит от того, в каком состоянии он завершит работу: допускающие состояния обозначаются посредством двойного кружка.

Пример 1. Рассмотрим диаграмму состояний автомата M_1:



Построим последовательность состояний, через которые M_1 пройдёт на строке $w = 1101$:

$$q_1 \longrightarrow q_2 \longrightarrow q_2 \longrightarrow q_3 \longrightarrow q_2.$$

Строка w будет допущена, так как автомат завершит работу в допускающем состоянии q_2.

Определение. *Детерминированным конечным автоматом (ДКА)* называется пятёрка $M = (Q, \Sigma, \delta, q_0, F)$, где

1. Q – конечное непустое множество состояний;
2. Σ – алфавит;
3. $\delta\colon Q \times \Sigma \to Q$ – переходная функция;
4. $q_0 \in Q$ – начальное состояние;
5. $F \subset Q$ – множество допускающих состояний.

Пример 2. Опишем ДКА M_1 в виде пятёрки $(Q, \Sigma, \delta, q_0, F)$:

1. $Q = \{q_1, q_2, q_3\}$;
2. $\Sigma = \{0, 1\}$;
3.

δ	0	1
q_1	q_1	q_2
q_2	q_3	q_2
q_3	q_2	q_2

4. $q_0 = q_1$;
5. $F = \{q_2\}$.

Определение. Пусть $M = (Q, \Sigma, \delta, q_0, F)$ – детерминированный конечный автомат, $w = x_1 x_2 \dots x_n$ – строка над алфавитом Σ. Тогда говорят, что M *допускает* w, если существуют $r_1, \dots, r_{n+1} \in Q$ такие, что

1. $r_1 = q_0$;
2. $\delta(r_i, x_i) = r_{i+1}$, $i = \overline{1, n}$;
3. $r_{n+1} \in F$.

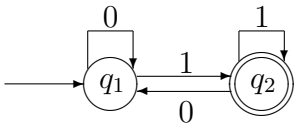
Определение. Через $L(M)$ обозначим множество всех строк, которые допускает ДКА M. Если для некоторого языка A верно, что $L(M) = A$, то говорят, что M *распознаёт* A.

Пример 3. Опишем язык, который распознаёт автомат M_1:

$$L(M_1) = \{w\colon w \text{ содержит хотя бы одну 1 и заканчивается либо на 1,}$$

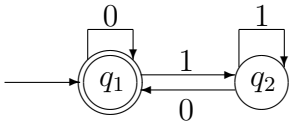
$$\text{либо на чётное число 0}\}.$$

Пример 4. Опишем язык, который распознаёт автомат M_2:



$$L(M_2) = \{w\colon w \text{ заканчивается на } 1\}.$$

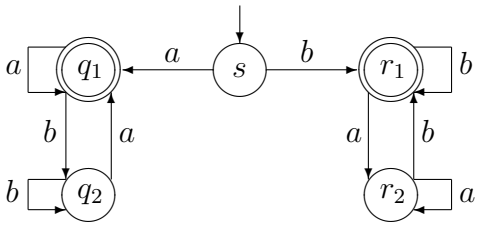
Пример 5. *Опишем язык, который распознаёт автомат M_3:*



$$L(M_3) = \{w\colon w \textit{ не заканчивается на } 0\}.$$

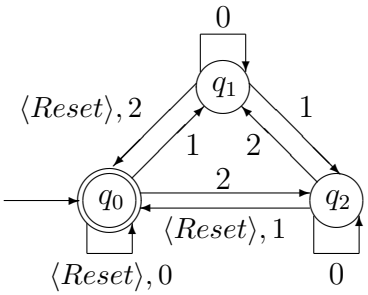
Заметим, что $\varepsilon \in L(M_3)$.

Пример 6. Опишем язык, который распознаёт автомат M_4:



$$L(M_4) = \{w\colon w \text{ заканчивается и начинается одинаковым символом}\}.$$

Пример 7. Опишем язык, который распознаёт автомат M_5:

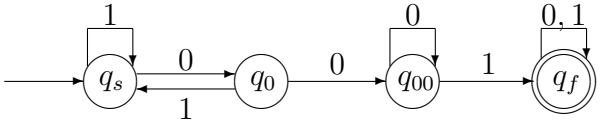


$$L(M_5) = \{w\colon \text{сумма цифр в } w \text{ после последнего } \langle Reset\rangle \text{ кратна } 3\}.$$

Определение. Язык A называется *регулярным*, если существует ДКА, который распознаёт его.

Пример 8. Построим автомат, который распознаёт язык

$$A = \{w\colon w \text{ содержит } 001 \text{ в качестве подстроки}\}.$$



Теорема 1. *Пусть A, B – регулярные языки. Тогда*

1. *$A \cap B$ – регулярный язык;*
2. *$\overline{A}$ – регулярный язык.*

Доказательство. Согласно определению существуют ДКА $M_1 = (Q_1, \Sigma, \delta_1, q_{01}, F_1)$ и $M_2 = (Q_2, \Sigma, \delta_2, q_{02}, F_2)$, распознающие языки A и B соответственно. Построим ДКА $M' = (Q_1 \times Q_2, \Sigma, \delta, \{(q_{01}, q_{02})\}, F_1 \times F_2)$, который будет симулировать вычисления M_1 и M_2 на входной строке одновременно. Переходная функция δ определяется следующим образом:

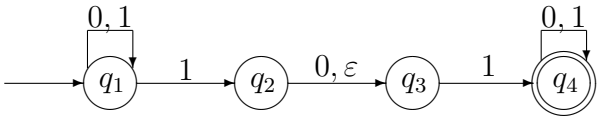
$$\delta((q_1, q_2), a) = (\delta_1(q_1, a), \delta_2(q_2, a)).$$

Тогда M' допускает входную строку тогда и только тогда, когда M_1 и M_2 допускают его, т.е. $L(M') = L(M_1) \cap L(M_2) = A \cap B$. Откуда в силу определения следует, что $A \cap B$ – регулярный язык.

Аналогичным образом для доказательства пункта 2 достаточно рассмотреть ДКА $M'' = (Q_1, \Sigma, \delta_1, q_{01}, Q_1 \setminus F_1)$, который допускает строку тогда и только тогда, когда M_1 её не допускает. □

2 Неопределённые вычисления

Рассмотрим диаграмму состояний некоторого автомата N_1:



Заметим, что данная диаграмма не соответствует ДКА, так как для состояния q_1 существует две выходящие стрелки по символу 1, для состояния q_2 нет выходящей стрелки по символу 1, но есть выходящая стрелка по ε. Такого рода автоматы будем называть недетерминированными.

Вычисления недетерминированных автоматов над произвольной строкой w можно интерпретировать так, что каждый раз, когда возникает неоднозначность перехода, автомат совершает все возможные переходы параллельно и продолжает вычисления аналогично ДКА на каждой из ветвей. Если в какой-либо из ветвей вычислений автомат не может совершить переход, то данная ветвь отвергается. Строка считается допущенной, если хотя бы одна ветвь вычислений завершается в допускающем состоянии.

Определение. *Недетерминированным конечным автоматом (НКА)* называется пятёрка $N = (Q, \Sigma, \delta, q_0, F)$, где

1. Q – конечное непустое множество состояний;
2. Σ – алфавит;
3. $\delta\colon Q \times \Sigma_\varepsilon \to \mathcal{P}(Q)$ – переходная функция;
4. $q_0 \in Q$ – начальное состояние;
5. $F \subset Q$ – множество допускающих состояний,

где $\Sigma_\varepsilon = \Sigma \cup \{\varepsilon\}$.

Определение. Пусть $N = (Q, \Sigma, \delta, q_0, F)$ – НКА, $w = x_1 x_2 \dots x_n$ – строка над алфавитом Σ. Тогда говорят, что N *допускает* w, если существуют $r_1, \dots, r_{n+1} \in Q$ такие, что

1. $r_1 = q_0$;
2. $r_{i+1} \in \delta(r_i, x_i)$, $i = \overline{1, n}$;

3. $r_{n+1} \in F$.

Определение. Через $L(N)$ обозначим множество всех строк, которые допускает НКА N. Если для некоторого языка A верно, что $L(N) = A$, то говорят, что N *распознаёт* A.

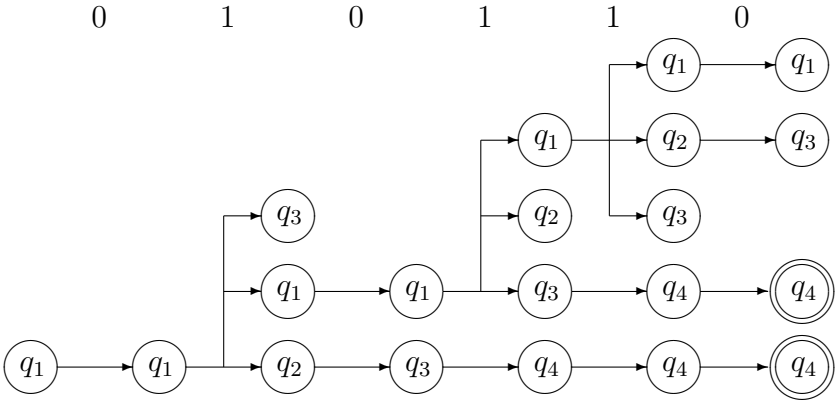
Пример 9. Опишем НКА N_1 в виде пятёрки $(Q, \Sigma, \delta, q_0, F)$:

1. $Q = \{q_1, q_2, q_3, q_4\}$;
2. $\Sigma = \{0, 1\}$;
3.

δ	0	1	ε
q_1	$\{q_1\}$	$\{q_1, q_2\}$	$\varnothing$
q_2	$\{q_3\}$	$\varnothing$	$\{q_3\}$
q_3	$\varnothing$	$\{q_4\}$	$\varnothing$
q_4	$\{q_4\}$	$\{q_4\}$	$\varnothing$

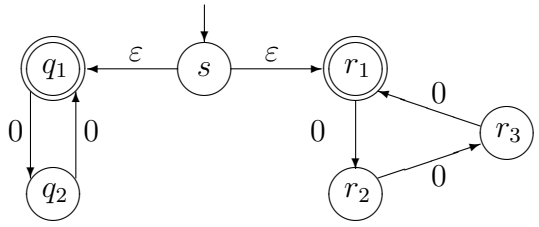
4. $q_0 = q_1$;
5. $F = \{q_4\}$.

Пример 10. Представим историю вычислений НКА N_1 на строке $w = 010110$ в виде дерева:



Две ветви вычислений N_1 завершаются в допускающем состоянии q_4, откуда следует, что N_1 допускает w.

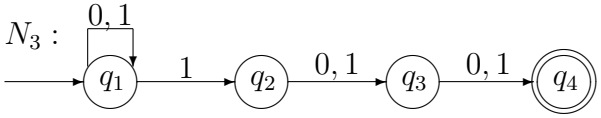
Пример 11. Опишем язык, который распознаёт автомат N_2:

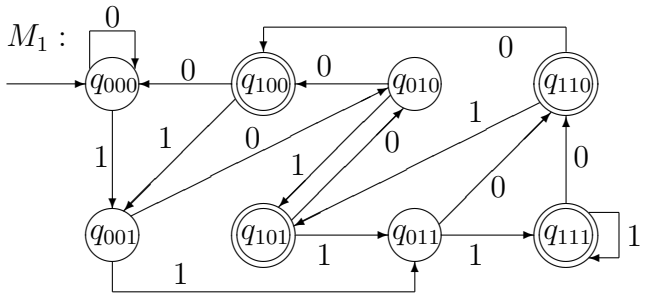


$$L(N_2) = \{0^k \colon k \in \mathbb{N} \text{ – кратно 2 или 3}\}.$$

Пример 12. Построим НКА N_3 и ДКА M_1, которые распознают язык

$$A = \{w \colon w \text{ содержит 1 в третьей позиции с конца}\}.$$





В данном случае недетерминированная и детерминированные модели эквивалентны в смысле равенства $L(N_3) = L(M_1)$.

В действительности результат аналогичный примеру 12 возможен для любого регулярного языка. Хотя эквивалентный ДКА, как правило, будет иметь более сложное описание, чем НКА.

Теорема 2. *Для любого НКА N существует ДКА M такой, что $L(N) = L(M)$.*

Доказательство. Пусть НКА $N = (Q, \Sigma, \delta, q_0, F)$. Построим ДКА $M = (Q', \Sigma, \delta', q_0', F')$ следующим образом:

$$Q' = \mathcal{P}(Q),\ q_0' = \{q_0\},\ F' = \{R \in Q' \colon R \cap F \neq \varnothing\},$$

Введём обозначения

$$E(R) = \{q \in Q \colon q \text{ может быть достигнуто из } R \text{ за 0 или более } \varepsilon\text{-переходов}\},$$

$$\delta'(R, a) = \bigcup_{r \in R} E(\delta(r, a)),\ R \subset Q,\ a \in \Sigma.$$

Тогда по построению $L(N) = L(M)$. □

Следствие 1. *Язык A регулярный тогда и только тогда, когда существует НКА N, который распознает его.*

Для работы с языками, как правило, требуются операции отличные от стандартных бинарных операций из теории множеств. Рассмотрим следующую группу операций.

Определение. *Регулярными операциями* над языками A и B называются

1. *объединение*:
$$A \cup B = \{w \colon w \in A \text{ или } w \in B\};$$
2. *конкатенация*:
$$A \circ B = \{w_1 w_2 \colon w_1 \in A \text{ и } w_2 \in B\};$$
3. *замыкание Клини*:
$$A^* = \{w_1 \dots w_n \colon n \in \mathbb{N},\ w_i \in A,\ i = \overline{1, n}\} \cup \{\varepsilon\}.$$

Пример 13. Пусть $A = \{0, 1\}$, $B = \{a, b\}$. Тогда

$$A \cup B = \{0, 1, a, b\},$$

$$A \circ B = \{0a, 1a, 0b, 1b\},$$

$$A^* = \{\varepsilon, 0, 1, 00, 01, 10, 11, 000, \dots\}.$$

Важным свойством регулярных операций является то, что они не выводят из класса регулярных языков. Докажем данные утверждения на основе теоремы 2 и следствия 1.

Теорема 3. *Пусть A и B – регулярные языки, определённые над одним алфавитом Σ. Тогда $A \cup B$ – регулярный язык.*

Доказательство. Согласно следствию 1 существуют НКА $N_1 = (Q_1, \Sigma, \delta_1, q_{01}, F_1)$ и $N_2 = (Q_2, \Sigma, \delta_2, q_{02}, F_2)$, распознающие языки A и B соответственно. Построим НКА $N = (\{q_0\} \cup Q_1 \cup Q_2, \Sigma, \delta, q_0, F_1 \cup F_2)$, который будет симулировать вычисления N_1 и N_2 на входной строке в виде различных ветвей вычислений. Здесь q_0 некоторое новое начальное состояние, не принадлежащее ни Q_1, ни Q_2, переходная функция δ определяется следующим образом:

$$\delta(q,a) = \begin{cases} \delta_1(q,a), & q \in Q_1, \\ \delta_2(q,a), & q \in Q_2, \\ \{q_{01}, q_{02}\}, & q = q_0,\ a = \varepsilon, \\ \varnothing, & q = q_0,\ a \neq \varepsilon. \end{cases}$$

Тогда N допускает входную строку тогда и только тогда, когда N_1 или N_2 допускают его, т.е. $L(N) = L(N_1) \cup L(N_2) = A \cup B$. Откуда в силу следствия 1 следует, что $A \cup B$ – регулярный язык. □

Теорема 4. *Пусть A и B – регулярные языки, определённые над одним алфавитом Σ. Тогда $A \circ B$ – регулярный язык.*

Доказательство. Согласно следствию 1 существуют НКА $N_1 = (Q_1, \Sigma, \delta_1, q_{01}, F_1)$ и $N_2 = (Q_2, \Sigma, \delta_2, q_{02}, F_2)$, распознающие языки A и B соответственно. Построим НКА $N = (Q_1 \cup Q_2, \Sigma, \delta, q_{01}, F_2)$, который, симулируя вычисления N_1 на входной строке, будет при попадании в допускающее состояние N_1 порождать параллельную ветвь вычислений, симулирующую N_2 на остатке строки. Переходная функция δ определяется следующим образом:

$$\delta(q,a) = \begin{cases} \delta_1(q,a), & q \in Q_1 \setminus F_1, \\ \delta_1(q,a), & q \in F_1,\ a \neq \varepsilon, \\ \delta_1(q,a) \cup \{q_{02}\}, & q \in F_1,\ a = \varepsilon, \\ \delta_2(q,a), & q \in Q_2. \end{cases}$$

Тогда N допускает входную строку w тогда и только тогда, когда справедливо представление $w = xy$, где N_1 допускает x, а N_2 допускает y. Т.е. $L(N) = L(N_1) \circ L(N_2) = A \circ B$. Откуда в силу следствия 1 следует, что $A \circ B$ – регулярный язык. □

Теорема 5. *Пусть A – регулярный язык. Тогда A^* – регулярный язык.*

Доказательство. Согласно следствию 1 существует НКА $N_1 = (Q_1, \Sigma, \delta_1, q_{01}, F_1)$, распознающий язык A. Построим НКА $N = (Q_1 \cup \{q_0\}, \Sigma, \delta, q_0, F_1 \cup \{q_0\})$, который симулирует вычисления N_1 на входной строке, создавая каждый раз параллельную ветвь вычислений при попадании в допускающее состояние N_1. Здесь q_0 некоторое новое начальное состояние, не принадлежащее Q_1, введённое для того, чтобы допустимой оказалась пустая строка ε. Переходная функция δ определяется следующим образом:

$$\delta(q,a) = \begin{cases} \delta_1(q,a), & q \in Q_1 \setminus F_1, \\ \delta_1(q,a), & q \in F_1,\ a \neq \varepsilon, \\ \delta_1(q,a) \cup \{q_{01}\}, & q \in F_1,\ a = \varepsilon, \\ \{q_{01}\}, & q = q_0,\ a = \varepsilon, \\ \varnothing, & q = q_0,\ a \neq \varepsilon, \end{cases}$$

Тогда N допускает входную строку w тогда и только тогда, когда $w = \varepsilon$ или справедливо представление $w = w_1 w_2 \dots w_n$, где N_1 допускает $w_1, w_2, \dots, w_n$. Т.е. $L(N) = L(N_1)^* = A^*$. Откуда в силу следствия 1 следует, что A^* – регулярный язык. □

3 Регулярные выражения

Введённые ранее регулярные операции можно использовать для конструирования языков посредством регулярных выражений.

Определение. Будем говорить, что R – *регулярное выражение* над алфавитом Σ, если R является одним из следующих выражений:

1. a, где $a \in \Sigma_\varepsilon$;
2. $\varnothing$;
3. $R_1 \cup R_2$, $R_1 \circ R_2$, R_1^*, где R_1, R_2 – регулярные выражения.

Замечание. Знак конкатенации $\circ$ в регулярных выражениях будем опускать при возможности. Также обозначим

$$R^+ = RR^*,\ R^k = \underbrace{R \circ \ldots \circ R}_{k}.$$

Регулярные операции упорядочиваются по возрастанию приоритета в следующем порядке: $\cup, \circ, ^*$. Под $L(R)$ будем понимать язык, порождённый регулярным выражением R.

Пример 14. Опишем сконструированные с помощью регулярных выражений над алфавитом $\Sigma = \{0, 1\}$ языки.

1. $(0 \cup 1)0^* = \{w\colon w$ состоит из строк, которые начинается с 0 или 1 и далее содержат произвольное число $0\}$;
2. $(0 \cup 1)^* = \{w\colon w$ состоит из 0 и $1\}$;
3. $0^*10^* = \{w\colon w$ содержит ровно одну $1\}$;
4. $\Sigma^*1\Sigma^* = \{w\colon w$ содержит хотя бы одну $1\}$;
5. $\Sigma^*001\Sigma^* = \{w\colon w$ содержит 001 в качестве подстроки$\}$;
6. $(\Sigma\Sigma)^* = \{w\colon w$ состоит из чётного числа символов$\}$;
7. $0\Sigma^*0 \cup 1\Sigma^*1 \cup 1 \cup 0 = \{w\colon w$ начинается и заканчивается одинаковым символом$\}$.

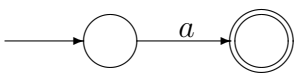
Замечание. Для любого регулярного выражения R всегда справедливы равенства

$$R \cup \varnothing = R;\quad R \circ \varnothing = \varnothing;\quad R \cup \varepsilon \neq R;\quad R \circ \varepsilon = R;\quad \varnothing^* = \varepsilon.$$

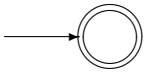
Продемонстрируем эквивалентность ДКА, НКА и регулярных выражений в смысле их возможностей описания регулярных языков.

Лемма 1. *Пусть R – регулярное выражение. Тогда $L(R)$ – регулярный язык.*

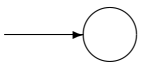
Доказательство. Построим НКА, распознающий $L(R)$ рекуррентным образом в соответствии с определением регулярного выражения. Если $R = a$, то НКА, распознающий $L(R)$, имеет вид



Если $R = \varepsilon$, то НКА, распознающий $L(R)$, имеет вид

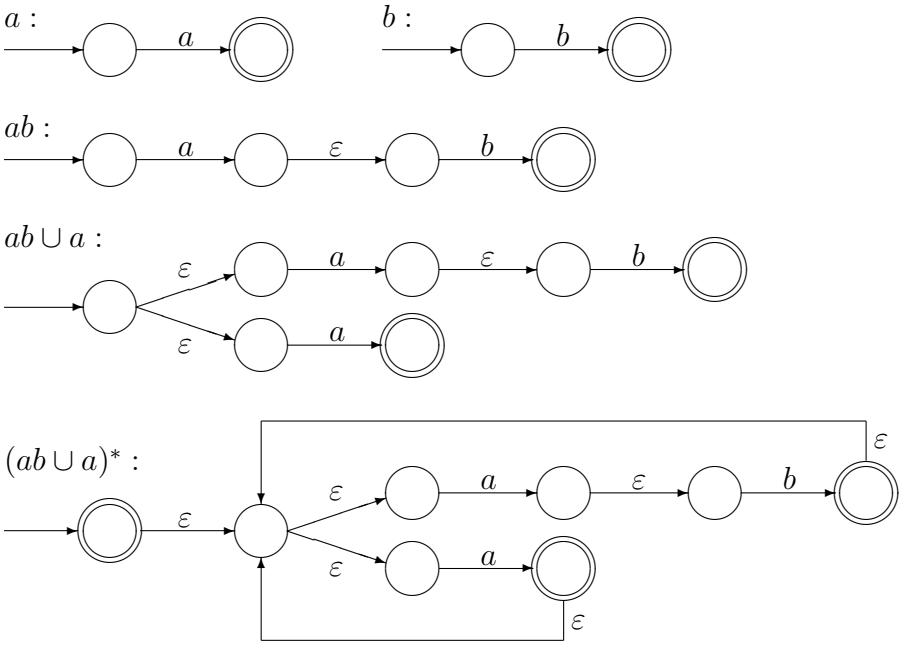


Если $R = \varnothing$, то НКА, распознающий $L(R)$, имеет вид



Т.е. в силу следствия 1 $L(R)$ в этих случаях является регулярным. Если R имеет вид $R_1 \cup R_2$, $R_1 \circ R_2$ или R_1^* для некоторых регулярных выражений R_1, R_2, то согласно теоремам 3, 4 и 5 по индукции $L(R)$ – также регулярный язык. □

Пример 15. Построим НКА, распознающий язык $(ab \cup a)^*$ в соответствии с подходом, приведённым в доказательстве леммы 1.



Для доказательства строгой эквивалентности НКА и регулярных выражений введём понятие обобщённого недетерминированного конечного автомата (ОНКА), у которого в диаграмме состояний над стрелками переходов будут записаны не отдельные символы, а регулярные выражения. При этом у ОНКА из начального состояния есть переходы во все остальные состояния, в единственное допускающее состояние есть переход из всех состояний, нет ни одной выходящей стрелки из допускающего состояния, у всех остальных состояний есть одна выходящая стрелка в каждое состояние, кроме начального.

Определение. *Обобщённым недетерминированным конечным автоматом (ОНКА)* называется пятёрка $G = (Q, \Sigma, \delta, q_{start}, q_{accept})$, где

1. Q – конечное непустое множество состояний;
2. Σ – алфавит;
3. $\delta\colon (Q \setminus \{q_{accept}\}) \times (Q \setminus \{q_{start}\}) \to \mathcal{R}$ – переходная функция;
4. $q_{start} \in Q$ – начальное состояние;
5. $q_{accept} \in Q$ – допускающее состояние,

где $\mathcal{R}$ – класс всех регулярных выражений над Σ.

Определение. Говорят, что ОНКА $G = (Q, \Sigma, \delta, q_{start}, q_{accept})$ *допускает строку* w, если допустимо представление $w = w_1 \dots w_n$, $w_i \in \Sigma^*$, $i = \overline{1,n}$ и существуют $q_0, \dots, q_n \in Q$ такие, что

1. $q_0 = q_{start}$;
2. $q_n = q_{accept}$;
3. $w \in L(\delta(q_{i-1}, q_i))$, $i = \overline{1, n}$.

Замечание. ОНКА является более мощной вычислительной моделью, чем ДКА, поскольку любой ДКА может быть преобразован в эквивалентный ему ОНКА следующим образом:

1. добавить в ДКА новое начальное состояние, соединённое с предыдущим допускающими состояниями через выходящую стрелку с ε;
2. добавть новое допускающее состояние, соединённое со всеми предыдущими через входящие стрелки с ε;
3. если над какой-либо стрелкой записано несколько символов, то объединить их через операцию $\cup$;
4. добавить все недостающие стрелки с символом $\varnothing$.

Лемма 2. *Пусть A – регулярный язык. Тогда существует регулярное выражение R такое, что $L(R) = A$.*

Доказательство. Пусть A – регулярный язык. Тогда по определению существует ДКА, распознающий его, который можно преобразовать в эквивалентный ему ОНКА $G = (Q, \Sigma, \delta, q_{start}, q_{accept})$ Далее преобразуем G согласно следующему алгоритму:

1. пусть k – число состояний G;
2. если $k = 2$, то завершить алгоритм;
3. если $k > 2$, то выбрать произвольное состояние $q_{rip} \in Q \setminus \{q_{start}, q_{accept}\}$ и построить ОНКА $G' = (Q \setminus \{q_{rip}\}, \Sigma, \delta', q_{start}, q_{accept})$, где для любых $q_i \in Q \setminus \{q_{accept}, q_{rip}\}$, $q_j \in Q \setminus \{q_{start}, q_{rip}\}$ верно представление

   $$\delta'(q_i, q_j) = R_1 R_2^* R_3 \cup R_4, \;\; R_1 = \delta(q_i, q_{rip}), \;\; R_2 = \delta(q_{rip}, q_{rip}), \;\; R_3 = \delta(q_{rip}, q_j), \;\; R_4 = \delta(q_i, q_j).$$

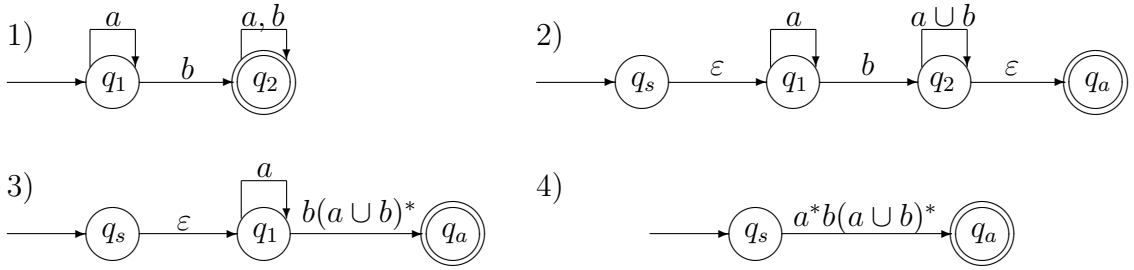
4. положить G равным G' и перейти к шагу 1.

По построению для каждого шага указанного алгоритма $A = L(G) = L(G')$. При этом алгоритм завершается, когда в ОНКА G остаётся два состояния, т.е. $A = L(G) = L(\delta(q_{start}, q_{accept}))$, т.е. регулярное выражение $\delta(q_{start}, q_{accept})$ порождает язык A. □

Из лемм 1 и 2 следует ещё одно необходимое и достаточное условие регулярности языка.

Теорема 6. *Язык A регулярный тогда и только тогда, когда существует регулярное выражение R такое, что $A = L(R)$.*

Пример 16. Продемонстрируем алгоритм преобразования ДКА в регулярное выражение, представленный в доказательстве леммы 2. Стрелки с символом $\varnothing$ для простоты изображения диаграммы опустим.



Доказанные следствие 1 и теорема 6 позволяют построить различные описания для регулярного языка, но с точки зрения проверки языка на регулярность они оказываются не конструктивными. Для этой цели, как правило, применяется следующее необходимое условие.

Теорема 7 (лемма о накачке)**.** *Пусть A – регулярный язык. Тогда существует число $p \in \mathbb{N}$, называемое длиной накачки, такое, что для любой строки $w \in A$, $|w| \geqslant p$ существует разбиение $w = xyz$, где*

1. *$xy^iz \in A$, $i \in \mathbb{N} \cup \{0\}$;*
2. *$|y| > 0$;*
3. *$|xy| \leqslant p$.*

Доказательство. Пусть $M = (Q, \Sigma, \delta, q_0, F)$ – ДКА, распознающий A, p – число состояний M. Пусть $s = s_1 \dots s_n \in A$, где $n > p$. Если таковой строки не существует, то утверждение теоремы 7 выполнены автоматически. Иначе существует последовательность состояний $r_1, \dots, r_{n+1} \in Q$, через которые проходит M при вычислении на s. Поскольку длина входной строки больше числа состояний, то найдутся $j, l = \overline{1, n+1}$ такие, что $j < l \leqslant p + 1 \leqslant n + 1$, $r_j = r_l$. Теперь положим $x = s_1 \dots s_{j-1}$, $y = s_j \dots s_{l-1}$, $z = s_l \dots s_n$.

Тогда по построению и определению $xy^iz \in A$, $i \in \mathbb{N} \cup \{0\}$. При этом $|y| > 0$, так как $j \neq l$, и $|xy| \leqslant p$, так как $l \leqslant p + 1$. □

Пример 17. Пусть $B = \{0^n1^n \colon n \geqslant 0\}$. Предположим, что B регулярный. Тогда в силу теоремы 7 существует длина накачки $p \in \mathbb{N}$. Выберем $s = 0^p1^p$. При этом $s = xyz$, где в силу пунктов 2 и 3 теоремы 7 $y = 0^k$, $k > 0$, а в силу пункта 1 должно выполняться включение $xy^2z = 0^{p+k}1^p \in B$, что неверно по определению B. Следовательно, B не является регулярным языком.

Пример 18. Пусть $C = \{w \colon w$ содержит одинаковое число 0 и $1\}$. Доказательство нерегулярности C строится аналогично примеру 17, если выбрать $s = 0^p1^p$.

Пример 19. Пусть $D = \{1^{n^2} \colon n \geqslant 0\}$. Предположим, что D регулярный. Тогда в силу теоремы 7 существует длина накачки $p \in \mathbb{N}$. Выберем $s = 1^{p^2}$. При этом $s = xyz$, где в силу пунктов 2 и 3 теоремы 7 $y = 1^k$, $0 < k \leqslant p$, а в силу пункта 1 должно выполняться включение $xy^2z = 1^{p^2+k} \in D$, что неверно, поскольку верно неравенство $p^2 < p^2 + k < p^2 + 2p + 1 = (p + 1)^2$. Следовательно, D не является регулярным языком.

Пример 20. Пусть $E = \{0^i1^j \colon 0 < j < i\}$. Предположим, что E регулярный. Тогда в силу теоремы 7 существует длина накачки $p \in \mathbb{N}$. Выберем $s = 0^{p+1}1^p$. При этом $s = xyz$, где в силу пунктов 2 и 3 теоремы 7 $y = 0^k$, $0 < k \leqslant p$, а в силу пункта 1 должно выполняться включение $xz = 0^{p+1-k}1^p \in B$, что неверно, поскольку $p + 1 - k \leqslant p$. Следовательно, E не является регулярным языком.

4 Контекстно-свободные грамматики

Введём более широкий класс языков, чем регулярные языки, которые называются контекстно-свободными. Для их описания используем такую конструкцию, как грамматика – набора правил подстановки, которые используются для вывода строк. Рассмотрим набор правил

$$\begin{gathered} G_1 \colon A \to 0A1, \\ A \to B, \\ B \to \#, \end{gathered}$$

где каждая подстановка может быть применена произвольное число раз, подстановки применяются к переменным A, B, вывод считается завершённым, если в строке не остаётся переменных, вывод любой строки начинается с переменной, указанной в левой части первого правило. Например, вывод строки 000#111 в грамматике G_1 можно представить следующим образом:

$$A \Rightarrow 0A1 \Rightarrow 00A11 \Rightarrow 000A111 \Rightarrow 000B111 \Rightarrow 000\#111.$$

Можно заметить, что G_1 порождает язык $\{0^n\#1^n \colon n \geqslant 0\}$.

Определение. *Контекстно-свободной (КС) грамматикой* называется четвёрка $G = (V, \Sigma, R, S)$, где

1. V – конечное непустое множество переменных;
2. Σ – алфавит, такой что $\Sigma \cap V = \varnothing$;
3. $R \subset V \times (\Sigma \cup V)^*$ – конечное множество правил;
4. $S \in V$ – начальная переменная.

Определение. Пусть $u, v, w \in (\Sigma \cup V)^*$, $A \in V$. Говорят, что *uAv порождает uwv* в грамматике $G = (V, \Sigma, R, S)$:

$$uAv \Rightarrow uwv,$$

если $(A \to w) \in R$. Говорят, что *u выводится v* в грамматике G:

$$u \overset{*}{\Rightarrow} v,$$

если существуют $u_1, \ldots, u_k \in (\Sigma \cup V)^*$, $k \geqslant 0$, такие что

$$u \Rightarrow u_1 \Rightarrow \ldots \Rightarrow u_k \Rightarrow v.$$

Определение. *Языком грамматики* $G = (V, \Sigma, R, S)$ называется множество строк, которые выводятся из начальной переменной:

$$L(G) = \{w \in \Sigma^* \colon S \overset{*}{\Rightarrow} w\}.$$

Язык A называется *контекстно-свободным (КС)*, если существует КС-грамматика G такая, что $A = L(G)$.

Пример 21. Построим грамматику $G_2 = (V, \Sigma, R, S)$, которая описывает корректно записанное математическое выражение с операциями сложения и умножения:

$$\begin{aligned} G_2 \colon\ & \langle EXPR\rangle \to \langle EXPR\rangle + \langle TERM\rangle, \\ & \langle EXPR\rangle \to \langle TERM\rangle, \\ & \langle TERM\rangle \to \langle TERM\rangle \times \langle FACTOR\rangle, \\ & \langle TERM\rangle \to \langle FACTOR\rangle, \\ & \langle FACTOR\rangle \to (\langle EXPR\rangle), \\ & \langle FACTOR\rangle \to a, \\ & \langle FACTOR\rangle \to b. \end{aligned}$$

Верно, что $a + a \times b \in L(G_2)$, $(a + a) \times a \in L(G_2)$. Также справедливы равенства

$$S = \langle EXPR\rangle,$$

$$\Sigma = \{+, \times, a, b, (,)\},$$

$$V = \{\langle EXPR\rangle, \langle TERM\rangle, \langle FACTOR\rangle\}.$$

Замечание. Для краткости записи будем объединять правила с одинаковой левой частью, в одну строчку, разделяя правые части с помощью специального символа «|». Например, грамматика G_2 примет краткий вид

$$\begin{aligned} G_2\colon\ & \langle EXPR\rangle \to \langle EXPR\rangle + \langle TERM\rangle | \langle TERM\rangle, \\ & \langle TERM\rangle \to \langle TERM\rangle \times \langle FACTOR\rangle | \langle FACTOR\rangle, \\ & \langle FACTOR\rangle \to (\langle EXPR\rangle) | a | b. \end{aligned}$$

Лемма 3. *Пусть $M = (Q, \Sigma, \delta, q_0, F)$ – ДКА. Тогда существует КС-грамматика $G = (V, \Sigma, R, S)$ такая, что $L(M) = L(G)$.*

Доказательство. Грамматику построим по следующим правилам:

1. для каждого состояния $q_i \in Q$ добавить в V переменную R_i;
2. для каждого $q_i \in F$ добавить в R правило $R_i \to \varepsilon$;
3. для каждого значения переходной функции $\delta(q_i, a) = q_j$ добавить в R правило $R_i \to aR_j$;
4. положить $S = R_0$.

По построению $L(M) = L(G)$. □

Одна из проблем анализа КС-грамматик заключается в неоднозначности того, какое число подстановок необходимо произвести для вывода строки заданной длины. Для решения данного вопроса при работе с КС-грамматиками зачастую удобно представлять их в упрощённом виде. Одной из таких упрощённых форм является нормальная форма Хомского, полезная при построении алгоритмов обработки КС-грамматик.

Определение. Говорят, что грамматика $G = (V, \Sigma, R, S)$находится в *нормальной форме Хомского*, если каждое правило имеет один из следующих двух видов:

$$A \to BC, \quad A \to a,$$

где $A, B, C \in V$, $a \in \Sigma$, $B, C \neq S$; также допустимо правило $S \to \varepsilon$.

Теорема 8. *Для любого КС-языка A существует КС-грамматика G в нормальной форме Хомского такая, что $A = L(G)$.*

Доказательство. По определению существует некоторая КС-грамматика $G' = (V, \Sigma, R, S)$, порождающая A. Представим процесс преобразования грамматики G' к эквивалентной ей грамматике G, которая находится в нормальной форме Хомского:

1. добавить новую переменную S_0 в V и правило $S_0 \to S$ в R, сделать S_0 начальной переменной G;
2. для всех переменных $A \in V \setminus \{S_0\}$ таких, что существует правило $A \to \varepsilon$, удалить данное правило из R и для каждого правила вида $B \to uAv$ добавить правило $B \to uv$, правила вида $B \to A$ заменить на правила $B \to \varepsilon$, где $B \in V$, $u, v \in (\Sigma \cup V)^*$;
3. повторять шаг 2 до тех пор, пока в R есть правила вида $A \to \varepsilon$;
4. для каждого правила вида $A \to B$, для всех правил вида $B \to u$ добавить в R правило $A \to u$, исходное правило $A \to B$ удалить, где $A, B \in V$, $u \in (\Sigma \cup V \setminus \{A\})^*$;
5. повторять шаг 4 до тех пор, пока в R есть правила вида $A \to B$;

6. каждое правило вида $A \to u_1 \dots u_k$, где $k \geqslant 3$, $u_1, \dots, u_k \in \Sigma \cup V$, заменить на набор правил
$$A \to u_1 A_1,\ A_1 \to u_2 A_2,\ \dots A_{k-2} \to u_{k-1} u_k$$
и добавить в V новые переменные $A_1, \dots, A_{k-2}$;

7. каждое правило вида $A \to uB$ ($A \to Bu$), где $B \in V$, $u \in \Sigma$ заменить на пару правил $A \to UB$ ($A \to BU$) и $U \to u$, также добавить новую переменную U в V;

8. каждое правило вида $A \to u_1 u_2$, где $u_1 u_2 \in \Sigma$ заменить на тройку правил $A \to U_1 U_2$, $U_1 \to u_1$, $U_2 \to u_2$, также добавить новые переменные U_1, U_2 в V.

По построению $L(G) = L(G')$. При этом в новой грамматике G шаг 1 исключает возможность появляния начальной переменной в правой части правил, шаги 2 и 3 исключают существование правил, в правой части которых находится пустая строка, шаги 4 и 5 исключают правила, в которых в правой части находится одна переменная, шаг 6 исключает правила, в которых в правой части находится больше двух символов, шаги 7 и 8 гарантируют, что в правой части правил будет либо две переменных, либо один терминальный символ. Т.е. G находится в нормальной форме Хомского. □

Пример 22. Проведём процесс преобразования КС-грамматики G_3 к нормальной форме Хомского, как это проводится в доказательстве теоремы 8:

$$G_3\colon S \to ASA|aB,\ A \to B|S,\ B \to b|\varepsilon,$$

шаг 1: $S_0 \to S,\ S \to ASA|aB,\ A \to B|S,\ B \to b|\varepsilon,$

шаг 2: $S_0 \to S,\ S \to ASA|aB|a,\ A \to B|S|\varepsilon,\ B \to b,$

шаг 3: $S_0 \to S,\ S \to ASA|SA|AS|S|aB|a,\ A \to B|S,\ B \to b,$

шаги 4-5: $S_0 \to ASA|SA|AS|aB|a,\ S \to ASA|SA|AS|aB|a,\ A \to b|ASA|SA|AS|aB|a,\ B \to b,$

шаг 6: $S_0 \to AA_1|SA|AS|aB|a,\ S \to AA_1|SA|AS|aB|a,\ A \to b|AA_1|SA|AS|aB|a,\ B \to b,\ A_1 \to SA,$

шаги 7-8: $S_0 \to AA_1|SA|AS|UB|a,\ S \to AA_1|SA|AS|UB|a,\ A \to b|AA_1|SA|AS|UB|a,\ B \to b,\ A_1 \to SA,\ U \to a.$

По определению любая строка $w \neq \varepsilon$, выведенная в грамматике в нормальной форме Хомского, потребует $2n-1$ подстановку, где $|w| = n$. Это связано с тем, что каждая подстановка либо увеличивает число переменных в строке на 1, либо заменяет одну переменную на терминальный символ.

5 Автомат с магазинной памятью

Аналогично регулярным языкам, которые допускают описание с помощью набора грамматических правил и некоторого автомата, контекстно-свободные языки также предполагают описание через вычислительную модель, которая эквивалентна КС-грамматикам. Такой вычислительной моделью являются автоматы с магазинной памятью. Под магазинной памятью понимается стек, который автомат при проведении вычислений может

использовать для добавления и удаления символов. При этом в любой момент времени у автомата есть доступ только к верхнему символу в стеке. Чтобы прочитать какой-либо другой символ, автомату необходимо удалить все вышестоящие. По умолчанию автомат с магазинной памятью предполагается недетерминированным.

Определение. *Автоматом с магазинной памятью (МП-автоматом)* называется шестёрка $M = (Q, \Sigma, \Gamma, \delta, q_0, F)$, где

1. Q – конечное непустое множество состояний;
2. Σ – входящий алфавит;
3. Γ – алфавит стека;
4. $\delta\colon Q \times \Sigma_\varepsilon \times \Gamma_\varepsilon \to \mathcal{P}(Q \times \Gamma_\varepsilon)$ – переходная функция;
5. $q_0 \in Q$ – начальное состояние;
6. $F \subset Q$ – множество допускающих состояний.

Определение. Пусть $M = (Q, \Sigma, \Gamma, \delta, q_0, F)$ – МП-автомат, $w \in \Sigma^*$. Тогда говорят, что M *допускает* w, если существуют представление $w = x_1 x_2 \dots x_n$, набор состояний $r_1, \dots, r_{n+1} \in Q$ и история содержимого стека $s_1, \dots, s_{n+1}$ такие, что

1. $x_i \in \Sigma_\varepsilon$, $i = \overline{1, n}$;
2. $r_1 = q_0$, $s_1 = \varepsilon$;
3. $(r_{i+1}, b) \in \delta(r_i, x_i, a)$, $i = \overline{1, n}$, где $s_i = at$, $s_{i+1} = bt$, $a, b \in \Gamma_\varepsilon$, $t \in \Gamma^*$;
4. $r_{n+1} \in F$.

Множество всех строк, которые допускает МП-автомат M обозначается $L(M)$.

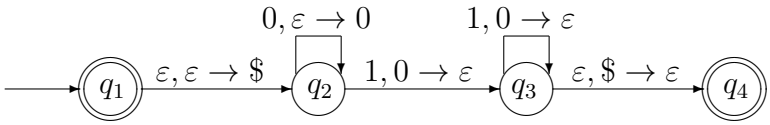
Пример 23. Построим МП-автомат M_1, распознающий язык $A = \{0^n 1^n\colon n \geqslant 0\}$. Вначале автомат M_1 добавляет в стек специальный символ, обозначающий конец стека, затем подсчитывает число 0, записывая их в стек. После первой считанной 1 автомат сравнивает их число с числом считанных 0, удаляя их из стека. Вычисления прерываются, либо если к окончанию строки в стеке находится 0, либо если элементы стека закончились до окончания строки, либо если после 1 в какой-либо момент автомат считал 0. Иначе автомат допускает строку. Формальное описание МП-автомата имеет следующий вид:

1. $Q = \{q_1, q_2, q_3, q_4\}$;
2. $\Sigma = \{0, 1\}$;
3. $\Gamma = \{0, \$\}$;
4.

δ	0			1			ε		
стек	0	\$	ε	0	\$	ε	0	\$	ε
q_1	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\{(q_2, \$)\}$
q_2	$\varnothing$	$\varnothing$	$\{(q_2, 0)\}$	$\{(q_3, \varepsilon)\}$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$
q_3	$\varnothing$	$\varnothing$	$\varnothing$	$\{(q_3, \varepsilon)\}$	$\varnothing$	$\varnothing$	$\{(q_4, \varepsilon)\}$	$\varnothing$	$\varnothing$
q_3	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$

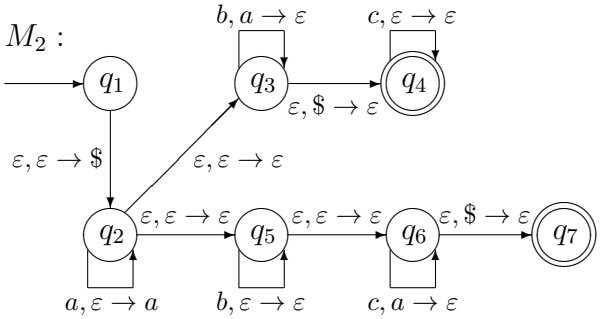
5. $q_0 = q_1$;
6. $F = \{q_1, q_4\}$.

Аналогично конечным автоматам МП-автоматы можно представить в виде диаграммы состояний



При построении диаграммы состояний МП-автомата над стрелками, кроме читаемого символа, также через запятую указывается выполняемое действие со стеком. Обозначение $a, b \to c$ предполагает, что переход выполняется, если из входной строки считан символ a и на вершине стека находится символ b, который заменяется на символ c.

Пример 24. Построим диаграмму состояний МП-автомата, распознающего язык $A = \{a^i b^j c^k : i = j \text{ или } i = k\}$.

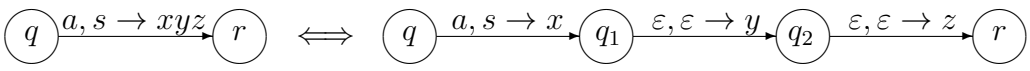


Лемма 4. *Пусть A – КС-язык. Тогда существует МП-автомат M такой, что $L(M) = A$.*

Доказательство. По определению существует КС-грамматика $G = (V, \Sigma, R, S)$, которая порождает A. Построим МП-автомат $M = (Q, \Sigma, \Gamma, \delta, q_0, F)$ эквивалентный G. Опишем принцип работы M:

1. поместить на вершину стека \$ и S;
2. в цикле повторять следующие действия:
   (a) если на вершине стека переменная $A \in V$, то недетерминировано заменить её на все её возможные подстановки в правилах R;
   (b) если на вершине стека терминальный символ $a \in \Sigma$, то удалить его, считать из входной строки один символ и сравнить его с a; при несовпадении отвергнуть;
   (c) если на вершине стека \$, то перейти в допускающее состояние.

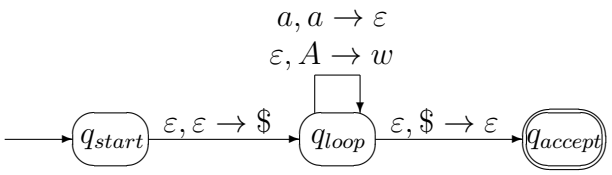
Построим формальное описание автомата M. Для сокращения рассуждений будет предполагать, что добавление в стек строки производится по следующему принципу:



Положим $Q = \{q_{start}, q_{loop}, q_{accept}\} \cup E$, где E – множество состояний, используемых для сокращения диаграммы состояний при добавлении в стек строки, $q_0 = q_{start}$, $F = \{q_{accept}\}$, $\Gamma = \Sigma \cup V \cup \{\$\}$. Определим переходную функцию:

$$\begin{aligned}
&\delta(q_{start}, \varepsilon, \varepsilon) = \{(q_{loop}, S\$)\},\\
&\delta(q_{loop}, \varepsilon, A) = \{(q_{loop}, w) : A \to w \in R\}, \quad A \in V,\\
&\delta(q_{loop}, a, a) = \{(q_{loop}, \varepsilon)\}, \qquad\qquad\quad a \in \Sigma,\\
&\delta(q_{loop}, \varepsilon, \$) = \{(q_{accept}, \varepsilon)\}.
\end{aligned}$$

Во всех прочих случаях значение переходной функции определяется как $\varnothing$. Также можно представить МП-автомат M в виде диаграммы:

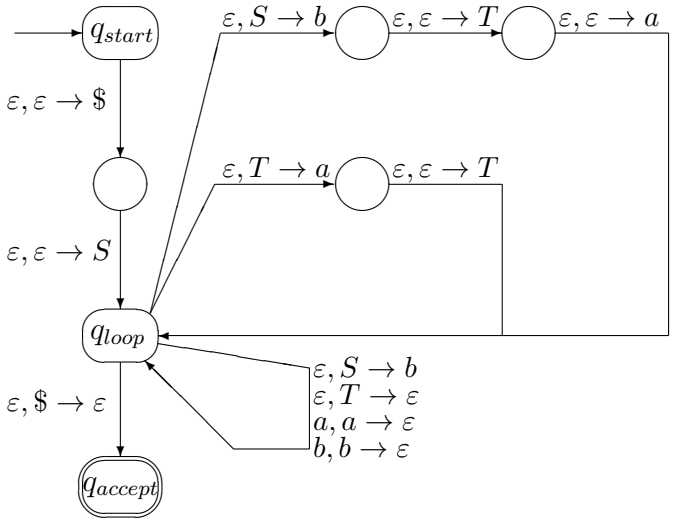


По построению $L(G) = L(M)$. □

Пример 25. Пусть задана КС-грамматика:

$$G\colon S \to aTb|b,$$
$$T \to Ta|\varepsilon.$$

Построим диаграмму состояний эквивалентного грамматике G МП-автомата M аналогично лемме 4:



Лемма 5. *Пусть для некоторого языка A существует МП-автомат M такой, что $A = L(M)$. Тогда A – КС-язык.*

Доказательство. Пусть $M = (Q, \Sigma, \Gamma, \delta, q_0, F)$. Построим эквивалентную КС-грамматику $G = (V, \Sigma, R, S)$, которая порождает A. Без ограничения общности можно полагать, что $F = \{q_{accept}\}$, M опустошает стек перед тем, как допустить строку и каждый переход по диаграмме состояний либо помещает символ в стек, либо удаляет его из стека, но не одновременно.

Пусть $V = \{A_{pq}\colon p, q \in Q\}$, $S = A_{q_0 q_{accept}}$. Построим набор правил R, симулирующих работу автомата M:

1. для всех $p, q, r, s \in Q$, $t \in \Gamma$ и $a, b \in \Sigma_\varepsilon$ таких, что $(r, t) \in \delta(p, a, \varepsilon)$ и $(q, \varepsilon) \in \delta(s, b, t)$ добавить в R правило $A_{pq} \to aA_{rs}b$;

2. для всех $p, q, r \in Q$ добавить в R правило $A_{pq} \to A_{pr}A_{rq}$;

3. для всех $p \in Q$ добавить в R правило $A_{pp} \to \varepsilon$.

□

Теорема 9. *Язык A контекстно-свободный тогда и только тогда, когда существует МП-автомат M такой, что $A = L(M)$.*

Доказательство. Доказательство следует из определения КС-языка и лемм 4 и 5. □

Следствие 2. *Пусть A – регулярный язык. Тогда A – контекстно-свободный язык.*

Доказательство. Доказательство следует из леммы 3 и теоремы 4. □

Теорема 9 позволяет построить различные описания для КС-языка, но с точки зрения проверки, не является ли некоторый язык контекстно-свободным, они оказываются не конструктивными. Для этой цели, как правило, применяется следующее необходимое условие.

Теорема 10 (лемма о накачке для КС-языков)**.** *Пусть A – КС-язык. Тогда существует число $p \in \mathbb{N}$, называемое длиной накачки, такое что для любой строки $w \in A$, $|w| \geqslant p$ существует разбиение $w = uvxyz$, где*

1. $uv^ixy^iz \in A$, $i \in \mathbb{N} \cup \{0\}$;
2. $|vy| > 0$;
3. $|vxy| \leqslant p$.

Доказательство. Пусть $G = (V, \Sigma, R, S)$ – КС-грамматика, $L(G) = A$. В силу теоремы 8 будем полагать, что G находится в нормальной форме Хомского. Через m обозначим число различных переменных в V. Положим $p = 2^{m+1}$. Рассмотрим вывод строки $w \in A$ в виде дерева разбора, в котором каждая подстановка с связана с поддеревом: переменной в левой части использованного для подстановки правила ставится в соответствие родительская вершина, а переменным и терминальным символам в правой части правила ставятся в соответствие дочерние вершины. Листьями в таком дереве являются только терминальные символы. Высотой дерева разбора назовём максимальное число переменных на пути от корня дерева к листу.

Если положить $|w| \geqslant p$, то любое дерево разбора будет иметь высоту не менее $m + 1$. Выберем путь от корня дерева к листу, имеющий наибольшую длину, т.е. длину не меньше, чем $m + 1$. Тогда в этом пути найдутся переменные, которые встречаются по крайней мере дважды. Выберем такую дублирующуюся переменную T, для которой путь от корня дерева до предпоследнего её появления максимален.

1. Следовательно, справедливы следующие выводы для некоторых $u, v, x, y, z \in \Sigma^*$:

   $$S \overset{*}{\Rightarrow} uTz \overset{*}{\Rightarrow} uvTyz \overset{*}{\Rightarrow} uvxyz = w.$$

   Откуда следует, что в грамматике G допустимы выводы $T \overset{*}{\Rightarrow} vTy$ и $T \overset{*}{\Rightarrow} x$. Тогда $uv^ixy^iz \in T$ для всех $i \in \mathbb{N} \cup \{0\}$.

2. Справедливо неравенство $|vy| > 0$, так как в противоположном случае оказывается возможен вывод $T \overset{*}{\Rightarrow} T$, что противоречит определению нормальной формы Хомского.

3. Также в силу максимальной удалённости T от корня дерева верно, что высота поддерева разбора вывода $T \overset{*}{\Rightarrow} vxy$ не превосходит $m + 1$. Поскольку дерево разбора является бинарным, то число листьев в соответствующем поддереве не более 2^{m+1}, или $|vxy| \leqslant 2^{m+1} = p$.

□

Пример 26. Пусть $A = \{a^nb^nc^n \colon n \geqslant 0\}$. Предположим, что A контекстно-свободный. Тогда в силу теоремы 10 существует длина накачки $p \in \mathbb{N}$. Выберем $s = a^pb^pc^p$. При этом $s = uvxyz$, где в силу пунктов 3 теоремы 10 либо $vxy = a^ib^j$, либо $vxy = b^ic^j$ для некоторых $i, j \geqslant 0$, $i + j \leqslant p$. В силу пункта 2 верно неравенство $|vy| > 0$, а в силу пункта 1 должно выполняться включение $uv^2xy^2z \in A$, что неверно по определению языка A. Следовательно, A не является КС-языком.

Пример 27. Пусть $B = \{a^i b^j c^k \colon 0 \leqslant i \leqslant j \leqslant k\}$. Предположим, что A контекстно-свободный. Тогда в силу теоремы 10 существует длина накачки $p \in \mathbb{N}$. Выберем $s = a^p b^p c^p$. При этом $s = uvxyz$, и в силу пунтка 1 теоремы 10 должно выполняться включение $uv^2xy^2z \in A$. Откуда следует, что v и y состоят из одинаковых символов. Рассмотрим все возможные случаи отдельно, учитывая, что в силу пункта 2 теоремы 10 $i + j > 0$:

1. если $v = a^i$, $y = a^j$, то $uv^2xy^2z = a^{p+i+j}b^p c^p \notin B$;
2. если $v = a^i$, $y = b^j$, то $uv^2xy^2z = a^{p+i}b^{p+j}c^p \notin B$;
3. если $v = a^i$, $y = c^j$, то либо $uv^2xy^2z = a^{p+i}b^p c^{p+j} \notin B$, либо $uv^0xy^0z = a^{p-i}b^p c^{p-j} \notin B$;
4. если $v = b^i$, $y = b^j$, то $uv^2xy^2z = a^p b^{p+i+j}c^p \notin B$;
5. если $v = b^i$, $y = c^j$, то $uv^0xy^0z = a^p b^{p-i}c^{p-j} \notin B$;
6. если $v = c^i$, $y = c^j$, то $uv^0xy^0z = a^p b^p c^{p-i-j} \notin B$.

Следовательно, A не является КС-языком.

Пример 28. Пусть $C = \{ww \colon w \in \{0,1\}^*\}$. Предположим, что C контекстно-свободный. Тогда в силу теоремы 10 существует длина накачки $p \in \mathbb{N}$. Выберем $s = 0^p 1^p 0^p 1^p$. При этом $s = uvxyz$, и в силу пунтка 3 теоремы 10 должно выполняться условие $|vxy| \leqslant p$. Если v и y оба лежат в первой половине строки s, то в строке uv^2xy^2z вторая половину строки начинается с 1, а первая – с 0, т.е. $uv^2xy^2z \notin C$. Если v и y оба лежат во второй половине строки s, то в строке uv^2xy^2z первая половина строки заканчивается с 0, а вторая – с 1, т.е. $uv^2xy^2z \notin C$. Следовательно, vxy расположено в середине s. Рассмотрим все возможные случаи отдельно, учитывая, что в силу пункта 2 теоремы 10 $i + j > 0$:

1. если $v = 1^i$, $y = 0^j$, то $uv^2xy^2z = 0^p 1^{p+i} 0^{p+j} 1^p \notin C$;
2. если $v = 1^i 0^k$, $y = 0^j$, $k \neq 0$, то $uv^2xy^2z = 0^p 1^p 0^k 1^i 0^{p+j} 1^p \notin C$;
3. если $v = 1^i$, $y = 1^k 0^j$, $k \neq 0$, то $uv^2xy^2z = 0^p 1^{p+i} 0^j 1^k 0^p 1^p \notin C$.

Следовательно, в силу пункта 3 теоремы 10 C не является КС-языком.

6 Машина Тьюринга

Рассмотрим более сложную вычислительную модель, позволяющую распознавать значительно больший класс языков, которая называется машина Тьюринга. Данная модель предполагает, что

1. входная строка записана на бесконечной рабочей ленте;
2. пишущая головка машины Тьюринга может перемещаться по ленте влево и вправо, производя чтение и запись;
3. вычисления завершаются немедленно при переходе машины Тьюринга в допускающее или отвергающее состояние.

Определение. *Машиной Тьюринга (MT)* называется семёрка $M = (Q, \Sigma, F, \delta, q_0, q_{accept}, q_{reject})$, где

1. Q – конечное непустое множество состояний;
2. Σ – входной алфавит, не содержащий пустого символа: $\square \notin \Sigma$;

3. Γ – рабочий алфавит такой, что $\Sigma \cup \{\square\} \subset \Gamma$;

4. $\delta \colon Q \times \Gamma \to Q \times \Gamma \times \{L, R\}$ – переходная функция;

5. $q_0 \in Q$ – начальное состояние;

6. $q_{accept} \in Q$ – допускающее состояние;

7. $q_{reject} \in Q \setminus \{q_{accept}\}$ – отвергающее состояние.

Определение. *Конфигурацией МТ* $M = (Q, \Sigma, F, \delta, q_0, q_{accept}, q_{reject})$ называется совокупность текущего содержимого ленты, состояния и положения головки МТ. Для $u, v \in \Gamma^*$, $q \in Q$ через uqv обозначается конфигурации МТ, находящейся в состоянии q, у которой на ленте написано uv при этом головка расположена на первом символе v.

Начальной конфигурацией M на строке $w \in \Sigma^*$ называется конфигурация $q_0 w$, *допускающей* или *отвергающей конфигурацией M* называется любая конфигурация uqv, где $q = q_{accept}$ или $q = q_{reject}$ соответственно.

Определение. Будем говорить, что для МТ $M = (Q, \Sigma, F, \delta, q_0, q_{accept}, q_{reject})$, $a, b, c \in \Gamma$, $u, v \in \Gamma^*$, $q_i, q_j \in Q$

1. конфигурация uaq_ibv *порождает* конфигурацию uq_jacv, если $\delta(q_i, b) = (q_j, c, L)$;

2. конфигурация uaq_ibv *порождает* конфигурацию $uacq_jv$, если $\delta(q_i, b) = (q_j, c, R)$.

Определение. Говорят, что МТ $M = (Q, \Sigma, F, \delta, q_0, q_{accept}, q_{reject})$ допускает (отвергает) строку $w \in \Sigma^*$, если существует набор конфигураций $C_1, \ldots, C_k$ такой, что

1. C_1 – начальная конфигурация M на w;

2. C_i порождает C_{i+1}, $i = \overline{1, k-1}$;

3. C_k – допускающая (отвергающая) конфигурация.

Определение. Множество всех строк, которые допускает M обозначается $L(M)$ и называется *языком, распознаваемым M*.

Язык называется *распознаваемым по Тьюрингу* или просто *распознаваемым*, если существует МТ, которая распознаёт его.

Замечание. Согласно определению и принципам проведения вычислений, в отличие от конечных автоматов и МП-автоматов, машина Тьюринга может не завершить вычисления на входной строке за конечное число тактов (переходов между последовательными конфигурациями). Возможна ситуация, при которой машина Тьюринга зацикливается, т.е. некоторая конфигурация uqv последовательно порождает саму себя. Т.е. из того, что МТ M не допускает некоторую строку w, не следует, что M её отвергнет. Данный факт делает существенным выделение более узкого класса языков, чем распознаваемые по Тьюрингу.

Определение. МТ, которая либо допускает, либо отвергает любую входную строку (но не зацикливается), называется *решателем*.

Язык называется *разрешимым по Тьюрингу* или просто *разрешимым*, если существует решатель, который распознаёт его.

Замечание. Как правило, машины Тьюринга описывают в виде «четвёрок» или «пятёрок» – специального набора команд, которые определяют переходную функцию МТ. Здесь для краткости записи будет использоваться более высокоуровневое описание машин Тьюринга посредством словесного описания действий, которые она должна совершить с входной строкой.

Пример 29. Построим МТ, которая распознаёт язык

$$A = \{w\#w \colon w \in \{0,1\}^*\}.$$

$M_1 =$ *на входе* w :

1. *Проверить, удовлетворяет ли* w *формату* $\{0,1\}^*\#\{0,1\}^*$, *если нет, то отвергнуть.*
2. *Найти крайний левый символ, расположенный слева от* $\#$, *который является 0 или 1.*
3. *Найти крайний левый символ, расположенный справа от* $\#$, *который является 0 или 1.*
4. *Если символы, найденные на шагах 1 и 2 совпадают, то заменить их на символ* x *и перейти к шагу 1.*
5. *Если данные символы не совпадают, то отвергнуть.*
6. *Если данные символы не были найдены и в строке все символы заменены на* x, *то допустить.*
7. *Иначе отвергнуть.*

Пример 30. Построим МТ, которая решает язык

$$B = \{0^{2^n} : n \geqslant 0\}.$$

$M_2 =$ *на входе* w :

1. *Проверить, имеет ли* w *формат* 0^*, *если нет, то отвергнуть.*
2. *Если остался единственный 0, то допустить.*
3. *Удалить каждый второй 0.*
4. *Если 0 было нечётное число, то отвергнуть, иначе перейти к шагу 2.*

Пример 31. Построим МТ, которая решает язык

$$C = \{a^i b^j c^k \colon i \cdot j = k,\ i,j,k \geqslant 1\}.$$

$M_3 =$ *на входе* w :

1. *Проверить, имеет ли* w *формат* $a^+b^+c^+$, *если нет, то отвергнуть.*
2. *Вернуться в начало строки.*
3. *Удалить один символ* a *и столько же символов* c, *сколько на ленте записано символов* b.
4. *Если символов* c *не хватило, то отвергнуть.*
5. *Повторить шаги 3 и 4, пока символы* a *или* c *на ленте не закончатся.*
6. *Если они закончились одновременно, то допустить, иначе отвергнуть.*

С точки зрения класса распознаваемых языков, существуют различные разновидности вычислительных моделей, эквивалентных машине Тьюринга. Рассмотрим некоторые из них.

Многоленточная МТ. Предполагается, что у данной МТ несколько рабочих лент и пишущих головок. При этом вычисления производятся одновременно на всех лентах. Основное отличие от обычной МТ в смысле описания заключается в определении переходной функции:

$$\delta \colon Q \times \Gamma^k \to Q \times \Gamma^k \times \{L,R\}^k,$$

где k – число лент МТ.

Теорема 11. *Для каждой многоленточной МТ существует эквивалентная ей одноленточная МТ.*

Доказательство. Пусть M – некоторая k-ленточная МТ. Построим одноленточную МТ M', симулирующую M. Организуем содержимое ленты M' следующим образом. Если i-й ленте M в некоторый момент времени соответствует конфигурация $u_i q a_i v_i$, где $u_i, v_i \in \Gamma^*$, $a_i \in \Gamma$, $q \in Q$, $i = \overline{1,k}$, то на ленте МТ M', симулирующей такт M будет записана строка $\#u_1\dot{a}_1v_1\# \dots \#u_k\dot{a}_kv_k\#$. Здесь постановка точки над символом a_i обозначает метку, определяющую положение головки МТ M на i-й ленте, специальный символ $\#$ обозначает разделение областей ленты M', отводящихся для симуляции различных лент M. Далее опишем работу M'.

$M' =$ *на входе* $w = w_1w_2 \dots w_n$:

1. *Представить входные данные в виде* $\#\dot{w}_1w_2 \dots w_n\#\dot{\square}\# \dots \#\dot{\square}\#$.
2. *По очереди производить вычисления на каждой из виртуальных лент, симулируя* M.
3. *Если в некоторый момент виртуальная головка МТ при движении вправо (влево) попадает на символ* $\#$*, то сдвинуть его и всё последующее содержимое ленты на одну ячейку вправо, вписав на исходное место символ* $\dot{\square}$*, и продолжить вычисления.*

По построению $L(M') = L(M)$. □

Следствие 3. *Язык распознаваем по Тьюрингу тогда и только тогда, когда некоторая многоленточная МТ распознаёт его.*

Недетерминированная МТ (НМТ). Предполагается, что данная МТ производит недетерминированные вычисления, т.е. допускает переход из текущей конфигурации в некоторое конечное множество производных конфигураций, вычисления на которых продолжаются параллельно. При этом НМТ допускает входную строку w, если хотя бы одна из ветвей вычислений завершается допускающей конфигурацией. Переходная функция НМТ имеет вид

$$\delta\colon Q \times \Gamma \to \mathcal{P}(Q \times \Gamma \times \{L, R\}).$$

Теорема 12. *Для каждой НМТ существует эквивалентная ей детерминированная МТ.*

Доказательство. Пусть N – некоторая НМТ. Построим четырёхленточную МТ M, симулирующую N. Пусть l – максимальное число недетерминированных ветвлений, которое N может сделать на одном такте. Тогда любую ветвь вычислений за k тактов можно однозначно описать набором чисел $i_1, \dots, i_k = \overline{1,l}$. Организуем содержимое лент M следующим образом. На первой ленте M перманентно будет хранится входная строка w, на второй ленте будет симулироваться работа N на выбранной ветви параллельных вычислений, на третьей будет записано число k симулируемых последовательно тактов N, на четвёртой будет хранится описание выбранной ветви недетерминированных вычислений в виде k чисел. Далее опишем работу M.

$M =$ *на входе* $w = w_1w_2 \dots w_n$:

1. *Скопировать содержимое 1-й ленты на 2-ю ленту, записать на 3-ю ленту число 1, на 4-ю ленту описание ветви вычислений в виде 1.*
2. *Симулировать вычисление ветви указанной на 4-й ленте такое число тактов, какое указано на 3-й ленте.*
3. *Если при симуляции строка была допущена, то допустить, иначе перейти к шагу 4.*
4. *Записать на 4-й ленте лексикографически следующее описание ветви вычислений и перейти к шагу 2.*
5. *Если таковых описаний заданной длины не осталось, то увеличить число на 3-й ленте на 1, записать на 4-й ленте первое лексикографичеки описание ветви вычислений и перейти к шагу 2.*

По построению $L(N) = L(M)$. С учётом теоремы 11 это эквивалентно существованию одноленточной МТ M' такой, что $L(N) = L(M')$. □

Следствие 4. *Язык распознаваем по Тьюрингу тогда и только тогда, когда некоторая НМТ распознаёт его.*

Перечислитель. Под перечислителем понимается МТ, совмещённая с печатающим устройством. Предполагается, что в любой момент перечислитель может отправить на печать текущее содержимое своей рабочей ленты и продолжить вычисления. Под языком перечислителя E понимается множество строк, которые E печатает.

Теорема 13. *Язык распознаваем по Тьюрингу тогда и только тогда, когда существует перечислитель, перечисляющий его.*

Доказательство. Пусть перечислитель E перечисляет некоторый язык A. Построим следующую МТ:
$M =$ *на входе* w :

1. *Запустить* E.
2. *Каждый раз, когда* E *печатает строчку, сравнить её с* w.
3. *Если строка совпала с* w, *то допустить.*

Тогда $L(M) = A$, т.е. A распознаваемый.

Пусть A – распознаваемый язык, M – МТ, которая распознаёт A. Поскольку Σ – конечное множество, то Σ^* является счётным и допускает представление

$$\Sigma^* = \{s_1, s_2, s_3, \ldots\}.$$

Построим перечислитель E:
$E =$ *игнорирует вход* :

1. *Повторить шаги 2 и 3 для всех* $i = 1, 2, 3, \ldots$.
2. *По очереди симулировать* M *на каждой строке* $s_1, \ldots, s_i$ *ровно* i *тактов.*
3. *Если какая-либо строка была допущена, то напечатать его.*

Тогда E перечисляет язык A. □

Построение такой вычислительной модели, как машина Тьюринга, позволило формализовать понятие алгоритма. Суть этого вопроса можно рассмотреть на примере исследования 10-й проблемы Гильберта, которая в 1900 г. была сформулирована следующим образом: требуется построить алгоритм, который определял бы, имеет ли многочлен с целыми коэффициентами и целыми показателями степеней целочисленные корни. Проблематика заключалась в том, что изначально предполагалось, что такой алгоритм есть, и его просто необходимо найти. Однако из-за отсутствия формального определения термина «алгоритм» доказать или опровергнуть его существование было невозможно.

Определение. *Алгоритмом* называется МТ.

В таком случае 10-я проблема Гильберта сводилась к вопросу, является ли разрешимым язык

$$D = \{p\colon p \text{ полином с целыми степенями и коэффициентами,}$$

$$\text{обладающий целочисленным корнем}\}.$$

В 1970 г. было доказано, что язык D не является разрешимым по Тьюрингу, хотя он и является распознаваемым. Построим МТ, которая распознаёт D:
$M =$ *на входе* p :

1. *Проверить, является ли* p *полином с целыми степенями и коэффициентами.*
2. *Последовательно перебрать все целочисленные векторы, вычисляя на них значение* p.
3. *Если* p *обратится в 0 при очередной подстановке, то допустить.*

M не является решателем D, так как в случае полинома с целыми коэффициентами $p \notin D$ МТ M никогда не завершит свои вычисления.

7 Разрешимость

Рассмотрим примеры различных разрешимых по Тьюрингу языков и построим их решатели.

Замечание. Для доказательства последующих утверждений важным является существование универсальной МТ – такой МТ S, которая может полностью симулировать МТ M на входе w, если $\langle M, w\rangle$ записано на ленте S. Как следствие, универсальная МТ также способна симулировать и более простые вычислительные модели такие, как конечные автоматы и МП-автоматы.

Через A_{DFA} обозначим задачу проверки, допускает ли заданный ДКА M строку w:

$$A_{DFA} = \{\langle M, w\rangle \colon M \text{ – ДКА, который допускает } w\}.$$

Теорема 14. *A_{DFA} разрешим по Тьюрингу.*

Доказательство. Построим решатель A_{DFA}:
$M_1 =$ *на входе* x :

1. *Проверить, верно ли $x = \langle M, w\rangle$, где M – ДКА, если нет, то отвергнуть.*
2. *Симулировать M на w и ответить то же самое, что и M.*

□

Через A_{NFA} обозначим задачу проверки, допускает ли заданный НКА M строку w:

$$A_{NFA} = \{\langle N, w\rangle \colon N \text{ – НКА, который допускает } w\}.$$

Теорема 15. *A_{NFA} разрешим по Тьюрингу.*

Доказательство. Построим решатель A_{NFA}:
$M_2 =$ *на входе* x :

1. *Проверить, верно ли $x = \langle N, w\rangle$, где N – НКА, если нет, то отвергнуть.*
2. *Конвертировать НКА N в эквивалентную ей ДКА M согласно теореме 2.*
3. *Симулировать M_1 из теоремы 14 на $\langle M, w\rangle$ и ответить то же самое, что и M_1.*

□

Через A_{REX} обозначим задачу проверки, порождается ли строка w заданным регулярным выражением R:

$$A_{REX} = \{\langle R, w\rangle \colon R \text{ – регулярное выражение, которое порождает } w\}.$$

Теорема 16. *A_{REX} разрешим по Тьюрингу.*

Доказательство. Построим решатель A_{NFA}:
$M_3 =$ *на входе* x :

1. *Проверить, верно ли $x = \langle R, w\rangle$, где R – регулярное выражение, если нет, то отвергнуть.*
2. *Конвертировать R в эквивалентную ему НКА N согласно лемме 1.*
3. *Симулировать M_2 из теоремы 15 на $\langle N, w\rangle$ и ответить то же самое, что и M_2.*

□

Замечание. Можно заметить, что в описаниях МТ M_1, M_2, M_3 первым пунктом всегда следует проверка корректности ввода данных. Будем полагать, что все последующие МТ данную проверку выполняют автоматически и отвергают те строки, которые её не проходят. Корректный формат входных данных будет указываться непосредственно в описании входа.

Через E_{DFA} обозначим задачу проверки, распознаёт ли ДКА M хотя бы одну строку:

$$E_{DFA} = \{\langle M\rangle \colon M \text{ – ДКА и } L(M) = \varnothing\}.$$

Теорема 17. *E_{DFA} разрешим по Тьюрингу.*

Доказательство. Построим решатель E_{DFA}:
$M_4 =$ *на входе* $\langle M\rangle$, *где* M *– ДКА* :

1. *Пометить начальное состояние.*
2. *Пометить все состояния достижимые из уже помеченных.*
3. *Если в шаге 2 не было помечено новых состояний, то перейти к шагу 4, иначе повторить шаг 2.*
4. *Если не помечено ни одного допускающего состояния, то допустить, иначе отвергнуть.*

□

Через EQ_{DFA} обозначим задачу проверки, распознают ли ДКА M' и M'' один и тот же язык:

$$EQ_{DFA} = \{\langle M', M''\rangle \colon M', M'' \text{ – ДКА и } L(M') = L(M'')\}.$$

Теорема 18. *EQ_{DFA} разрешим по Тьюрингу.*

Доказательство. Построим решатель EQ_{DFA}:
$M_5 =$ *на входе* $\langle M', M''\rangle$, *где* M', M'' *– ДКА* :

1. *Построить согласно теоремам 1 и 3 НКА N такой, что* $L(N) = \left(L(M') \cap \overline{L(M'')}\right) \cup \left(L(M'') \cap \overline{L(M')}\right)$.
2. *Конвертировать N в эквивалентный ДКА M согласно теореме 2.*
3. *Запустить M_4 из теоремы 17 на $\langle M\rangle$ и ответить то же самое.*

По построению N распознаёт только те строки, которые распознаёт только один из автоматов M' и M''. В таком случае язык N будет пустым тогда и только тогда, когда языки M' и M'' совпадают, т.е. M_4 решает EQ_{DFA}. □

Через A_{CFG} обозначим задачу проверки, порождает ли КС-грамматика G строку w:

$$A_{CFG} = \{\langle G, w\rangle \colon G \text{ – КС-грамматика и } w \in L(G)\}.$$

Теорема 19. *A_{CFG} разрешим по Тьюрингу.*

Доказательство. Построим решатель A_{CFG}:
$M_6 =$ *на входе* $\langle G, w\rangle$, *где* G *– КС-грамматика* :

1. *Преобразовать G в нормальную форму Хомского G' согласно теореме 8.*
2. *Если $|w| = 0$, то перебрать все выводы в G' в 1 подстановку.*
3. *Если $|w| = n > 0$, то перебрать все выводы за $2n - 1$ подстановку.*
4. *Если какая-нибудь выведенная строка совпадает с w, то допустить, иначе отвергнуть.*

Поскольку по определению нормальной формы Хомского любая строка длины $n > 0$ потребует для вывода ровно $2n - 1$ подстановку, то M_6 решает A_{CFG}. □

Через E_{CFG} обозначим задачу проверки, порождает ли КС-грамматика G хотя бы одну строку:

$$E_{CFG} = \{\langle G\rangle \colon G \text{ – КС-грамматика и } L(G) = \varnothing\}.$$

Теорема 20. *E_{CFG} разрешим по Тьюрингу.*

Доказательство. Построим решатель E_{CFG}:
$M_7 =$ *на входе* $\langle G\rangle$, *где* G *– КС-грамматика* :

1. *Пометить в G все терминальные символы.*
2. *Пометить в G все переменные A, для которых существует правило $A \to U_1 \dots U_k$, где $U_1, \dots, U_k$ уже помечены.*
3. *Если в шаге 2 не было помечено новых переменных, то перейти к шагу 4, иначе повторить шаг 2.*
4. *Если не помечена начальная переменная, то допустить, иначе отвергнуть.*

□

Теорема 21. *Любой КС-язык разрешим по Тьюрингу.*

Доказательство. Пусть A – КС-язык. Тогда по определению существует КС-грамматика G, порождающая A. Построим решатель A:
$M_8 =$ *на входе* w :

1. *Запустить M_6 из теоремы 19 на входе $\langle G, w\rangle$ и ответить то же самое.*

□

Приведём пример неразрешимого языка. Через A_{TM} обозначим задачу проверки, допускает ли МТ M строку w:

$$A_{TM} = \{\langle M, w\rangle \colon M \text{ – МТ и } w \in L(M)\}.$$

Теорема 22. *A_{TM} неразрешим по Тьюрингу.*

Доказательство. Предположим, что существует M' – решатель A_{TM}. Построим на основе M' МТ D, которая проверяет не допускает ли МТ M своё собственное описание:
$D =$ *на входе* $\langle M\rangle$, *где* M *– МТ* :

1. *Запустить M' на входе $\langle M, \langle M\rangle\rangle$.*
2. *Если M' допускает, то отвергнуть, иначе допустить.*

Тогда по построению вход $\langle D\rangle$ при запуске на D будет допущен в том и только в том случае, когда D не допускает $\langle D\rangle$. Получаем противоречие, т.е. A_{TM} неразрешим. □

Теорема 23. *Существуют нераспознаваемые по Тьюрингу языки.*

Доказательство. Для доказательства теоремы сопоставим множество всех возможных МТ и множество всех языков. Поскольку множество всех строк Σ^* является счётным, то множество всех языков $\mathcal{P}(\Sigma^*)$ обладает мощностью континуума. При этом каждая МТ M допускает своё описание в виде некоторой строки $\langle M\rangle \in \Sigma^*$, откуда следует, что всего существует не более чем счётное число различных МТ. Поскольку каждая МТ распознаёт единственный язык, то невозможно построить сюръективное отображение из множества всех МТ на более мощное множество всех языков $\mathcal{P}(\Sigma^*)$. Т.е. существует языки, которые не распознаёт ни одна МТ. □

Теорема 24. *Язык A разрешим по Тьюрингу тогда и только тогда, когда A и $\overline{A}$ распознаваемы по Тьюрингу.*

Доказательство. Пусть A разрешим, и M – МТ, которая решает его. Тогда МТ M', которая симулирует M на своём входе и отвечает противоположное, решает $\overline{A}$.

Пусть A и $\overline{A}$ распознаваемы, M' и M'' – МТ, которые распознают их соответственно. Построим решатель A:
$M =$ *на входе* w :

1. *Запустить M' и M'' на входе w параллельно.*
2. *Если M' допустит, то допустить.*
3. *Если M'' допустит, то отвергнуть.*

Поскольку $w \in A \cup \overline{A}$, то за конечное число тактов либо M', либо M'' допустит w. Т.е. вычисления M завершатся за конечное число тактов. □

Теорема 23 только постулирует существования нераспознаваемых языков. При этом комбинирование теорем 22 и 24 позволяет построить пример нераспознаваемого языка.

Следствие 5. *$\overline{A_{TM}}$ нераспознаваем по Тьюрингу.*

Доказательство. В силу теорем 22 и 24 достаточно показать, что A_{TM} распознаваем. Распознавателем A_{TM} будет являться любая универсальная МТ, которая, получая на вход $\langle M, w\rangle$, симулирует вычисления МТ M на входе w. □

8 Сведéние

Под сведéнием понимается возможность свести решение одной задачи к решению другой. Продемонстрируем эффективность данной методики на исследовании вопросов разрешимости задачи останова. Через $HALT_{TM}$ обозначим задачу проверки, завершает ли МТ M вычисления на строке w:

$$HALT_{TM} = \{\langle M, w\rangle \colon M \text{ – МТ, которая завершает вычисления на входе } w\}.$$

Теорема 25. *$HALT_{TM}$ неразрешим по Тьюрингу.*

Доказательство. Сведем вопрос разрешимости A_{TM} к разрешимости $HALT_{TM}$. Предположим, что существует M' – решатель $HALT_{TM}$. Построим на основе M' следующую МТ:
$M_1 =$ *на входе* $\langle M, w\rangle$, *где* M *– МТ*:

1. *Запустить* M' *на* $\langle M, w\rangle$.
2. *Если* M' *отвергает, то отвергнуть, иначе перети к шагу 3.*
3. *Запустить* M *на* w *и ответить то же самое.*

По построению M_1 отвергает $\langle M, w\rangle$, если M отвергает или зацикливается на w, и допускает, если M допускает w. Таким образом, M_1 решает A_{TM}. Получаем противоречие, т.к. A_{TM} неразрешим в силу теоремы 22. Следовательно, $HALT_{TM}$ также неразрешим. □

Через E_{TM} обозначим задачу проверки, допускает ли МТ M хотя бы одну строку w:

$$E_{TM} = \{\langle M\rangle \colon M \text{ – МТ и } L(M) = \varnothing\}.$$

Теорема 26. *E_{TM} неразрешим по Тьюрингу.*

Доказательство. Предположим, что существует M' – решатель E_{TM}. Построим на основе M' следующую МТ:
$M_2 =$ *на входе* $\langle M, w\rangle$, *где* M *– МТ*:

1. *Построить МТ* $M_0 =$ *на входе* x :
    1. *Если* $x \neq w$, *то отвергнуть.*
    2. *Иначе запустить* M *на* w *и ответить то же самое.*
2. *Запустить* M' *на* $\langle M_0\rangle$ *и ответить обратное.*

Здесь M_0 может зациклится на своем шаге 2, но в рамках алгоритма M_2 симуляции M_0 не происходит, M_0 необходимо только сконструировать, что является тривиальной задачей. При этом M_0 устроена так, что она может допустить только w в силу своего шага 1, и только в том случае, если M допускает w, в силу своего шага 2. Т.е. $L(M_0) = \varnothing$ в том и только в том случае, когда M не допускает w. Следовательно, M_2 решает A_{TM}. Получаем противоречие, т.к. A_{TM} неразрешим в силу теоремы 22. Тогда E_{TM} также неразрешим. □

Через $REGULAR_{TM}$ обозначим задачу проверки, распознаёт ли МТ M регулярный язык:

$$REGULAR_{TM} = \{\langle M\rangle \colon M \text{ – МТ, } L(M) \text{ – регулярный язык}\}.$$

Теорема 27. *$REGULAR_{TM}$ неразрешим по Тьюрингу.*

Доказательство. Предположим, что существует M' – решатель $REGULAR_{TM}$. Построим на основе M' следующую МТ:
$M_3 =$ *на входе* $\langle M, w\rangle$, *где* M – *МТ*:
1. *Построить МТ* $M_0 =$ *на входе* x:
    1. *Если* $x = 0^n1^n$, $n \geqslant 0$, *то допустить.*
    2. *Иначе запустить* M *на* w *и ответить то же самое.*
2. *Запустить* M' *на* $\langle M_0\rangle$ *и ответить то же самое.*

МТ M_0 устроена так, что она либо допускает все строки Σ^*, если M допускает w, либо допускает нерегулярный язык $\{0^n1^n\colon n \geqslant 0\}$, если M не допускает w. Т.е. $L(M_0)$ регулярен в том и только в том случае, когда M допускает w. Таким образом, M_3 решает A_{TM}. Получаем противоречие, т.к. A_{TM} неразрешим в силу теоремы 22. Следовательно, $REGULAR_{TM}$ также неразрешим. □

Через EQ_{TM} обозначим задачу проверки, распознают ли МТ N_1 и N_2 один и тот же язык:

$$EQ_{TM} = \{\langle N_1, N_2\rangle\colon N_1, N_2 \text{ – МТ и } L(N_1) = L(N_2)\}.$$

Теорема 28. *EQ_{TM} неразрешим по Тьюрингу.*

Доказательство. Предположим, что существует M' – решатель EQ_{TM}. Построим на основе M' следующую МТ:
$M_4 =$ *на входе* $\langle M\rangle$, *где* M – *МТ*:
1. *Построить МТ* $M_0 =$ *на входе* x:
    1. *Отвергнуть.*
2. *Запустить* M' *на* $\langle M, M_0\rangle$ *и ответить то же самое.*

Поскольку $L(M_0) = \varnothing$ по построению, то M' допускает $\langle M, M_0\rangle$ тогда и только тогда, когда $L(M) = \varnothing$. Таким образом, M_4 решает E_{TM}. Получаем противоречие, т.к. E_{TM} неразрешим в силу теоремы 26. Следовательно, EQ_{TM} также неразрешим. □

Определение. *Допускающей (отвергающей) историей вычислений МТ* M называется последовательность конфигураций $C_1, \ldots, C_n$ МТ M, где C_1 – начальная конфигурация, C_i порождает C_{i+1}, $i = \overline{1, n-1}$, C_n – допускающая (отвергающая) конфигурация.

Через ALL_{CFG} обозначим задачу проверки, порождает ли КС-грамматика G все возможные строки:

$$ALL_{CFG} = \{\langle G\rangle\colon G \text{ – КС-грамматика и } L(G) = \Sigma^*\}.$$

Теорема 29. *ALL_{CFG} неразрешим по Тьюрингу.*

Доказательство. Построим КС-грамматику $G_{M,w}$, которая порождает Σ^* тогда и только тогда, когда МТ M не допускает w. Т.е. $G_{M,w}$ будет порождать все строки, кроме допускающей истории вычислений M на w. Согласно теореме 9 вместо КС-грамматики $G_{M,w}$ можно построить эквивалентный ей МП-автомат D.

D будет недетерминировано производить следующие четыре проверки:

1. если входная строка не является историей вычислений M вида $\#C_1\#C_2^{\mathcal{R}}\#\ldots\#C^k$, где через $w^{\mathcal{R}}$ обозначена строка w записанная в обратном направлении, то D допускает вход;

2. если первая конфигурация во входной строке не является начальной конфигурацией M на w, то D допускает вход;

3. если последняя конфигурация во входной строке не является допускающей конфигурацией M, то D допускает вход;

4. если для каких-либо двух последовательно записанных во входной строке конфигураций вторая не порождает первую, то D допускает вход.

Последнее действие D выполняет, записывая первую конфигурацию в стек и сравнивая её посимвольно с второй конфигурацией на основе переходной функции M. Именно для этого требовалось чередовать записи конфигураций в прямом и обратном направлении, поскольку запись строки в стек МП-автомата разворачивает её.

Таким образом, D отвергнет только допускающую историю вычислений M на w, если такая существует. Следовательно, решатель ALL_{CFG} позволит решить A_{TM}, что невозможно в силу теоремы 22. □

Формализуем приём сведения, который был использован для доказательства приведённых ранее теорем 25–29.

Определение. Отображение $f\colon \Sigma^* \to \Sigma^*$ называется *вычислимой функцией*, если существует МТ, которая, начиная вычисления с w на ленте, завершает их с $f(w)$ на ленте.

Пример 32. Операция сложения является вычислимой функцией в следующем смысле:

$$f(\langle m, n\rangle) = \langle m + n\rangle,\ m, n \in \mathbb{N}.$$

Определение. Говорят, что *язык A m-сводим к языку B*:

$$A \leqslant_m B,$$

если существует вычислимая функция $f\colon \Sigma^* \to \Sigma^*$, называемая *редукцией или сведéнием от A к B*, такая, что для любого $w \in \Sigma^*$ включение верно $w \in A$ тогда и только тогда, когда $f(w) \in B$.

Замечание. Также из определения следует, что редукция от A к B является одновременно редукцией от $\overline{A}$ к $\overline{B}$. Т.е. соотношение $A \leqslant_m B$ эквивалентно соотношению $\overline{A} \leqslant_m \overline{B}$.

Как продемонстрировано в доказательстве теоремы 25, $A_{TM} \leqslant_m HALT_{TM}$. В доказательстве теоремы 28 также продемонстрировано, что $E_{TM} \leqslant_m EQ_{TM}$. Понятие m-сводимости позволяет обобщить рассмотренный ранее метод исследования языков на разрешимость и распознаваемость.

Теорема 30. *Пусть $A, B \subset \Sigma^*$, $A \leqslant_m B$, B разрешим по Тьюрингу. Тогда A также разрешим.*

Доказательство. Пусть M' решает B, вычислимая функция f – редукция от A к B. Построим на основе M' следующую МТ:
$M =$ *на входе w :*
1. *Вычислить $f(w)$.*
2. *Запустить M' на $f(w)$ и ответить то же самое.*

По определению m-сводимости M решает A. □

Следствие 6. *Пусть $A, B \subset \Sigma^*$, $A \leqslant_m B$, A неразрешим по Тьюрингу. Тогда B также неразрешим.*

Теорема 31. *Пусть $A, B \subset \Sigma^*$, $A \leqslant_m B$, B распознаваем по Тьюрингу. Тогда A также распознаваем.*

Доказательство. Представленная в доказательстве теоремы 30 МТ M распознаёт A в условиях теоремы 31. □

Следствие 7. *Пусть $A, B \subset \Sigma^*$, $A \leqslant_m B$, A нераспознаваем по Тьюрингу. Тогда B также нераспознаваем.*

Теорема 32. *Ни EQ_{TM}, ни $\overline{EQ_{TM}}$ не распознаваемы по Тьюрингу.*

Доказательство. Построим редукцию от A_{TM} к $\overline{EQ_{TM}}$:
$F =$ *на входе* $\langle M, w\rangle$, *где* M – *МТ*:
1. *Построить МТ* $M' =$ *на входе* x:
    1. *Отвергнуть.*
2. *Построить МТ* $M'' =$ *на входе* x:
    1. *Запустить* M *на* w *и ответить то же самое.*
3. *Вывести на ленту* $\langle M', M''\rangle$.

По построению в описании F верно, что

$$L(M') = \varnothing,\ L(M'') = \begin{cases} \varnothing, & w \notin L(M), \\ \Sigma^*, & w \in L(M). \end{cases}$$

Таким образом, $A_{TM} \leqslant_m \overline{EQ_{TM}}$ и $\overline{A_{TM}} \leqslant_m EQ_{TM}$. Откуда согласно следствиям 5 и 7 следует, что EQ_{TM} нераспознаваем.

Построим редукцию от A_{TM} к EQ_{TM}:
$G =$ *на входе* $\langle M, w\rangle$, *где* M – *МТ*:
1. *Построить МТ* $M' =$ *на входе* x:
    1. *Допустить.*
2. *Построить МТ* $M'' =$ *на входе* x:
    1. *Запустить* M *на* w *и ответить то же самое.*
3. *Вывести на ленту* $\langle M', M''\rangle$.

По построению в описании G верно, что

$$L(M') = \Sigma^*,\ L(M'') = \begin{cases} \varnothing, & w \notin L(M), \\ \Sigma^*, & w \in L(M). \end{cases}$$

Таким образом, $A_{TM} \leqslant_m EQ_{TM}$ и $\overline{A_{TM}} \leqslant_m \overline{EQ_{TM}}$. Откуда согласно следствиям 5 и 7 следует, что $\overline{EQ_{TM}}$ нераспознаваем. □

9 Теорема о рекурсии

Рассмотрим вопросы построения машины Тьюринга, которая печатает своё описание на ленту и останавливается. Назовём её $SELF$.

Лемма 6. *Пусть $w \in \Sigma^*$. Тогда существует вычислимая функция $q\colon \Sigma^* \to \Sigma^*$ такая, что верно равенство $q(w) = \langle P_w\rangle$, где P_w – машина Тьюринга, которая печатает w на ленту и останавливается.*

Доказательство. Опишем вычислимую функцию q в виде машины Тьюринга Q:
$Q =$ *на входе* w:
1. *Построить машину Тьюринга* $P_w =$ *на входе* x:
    1. *Удалить вход.*
    2. *Напечатать* w.
    3. *Остановиться.*

2. *Вывести P_w на ленту.*

□

Машина Тьюринга $SELF$ будет состоять из двух частей:

$$\langle SELF\rangle = \langle AB\rangle,$$

где A, B – два независимых алгоритма, которые работают по очереди.

Опишем сначала B:

$B =$ *на входе* $\langle M\rangle$, *где* M *– часть машины Тьюринга* :

1. *В согласно лемме 6 вычислить* $q(\langle M\rangle) = P_{\langle M\rangle}$.
2. *Соединить* $P_{\langle M\rangle}$ *с* $\langle M\rangle$ *так, чтобы получилась машина Тьюринга.*
3. *Вывести полученную машину Тьюринга на ленту и остановиться.*

Возможность построения B следует из леммы 6. Тогда, если положить равенство $A = P_{\langle B\rangle}$, можно определить $SELF$ в виде конкатенации A и B. При запуске $SELF$ сначала выполнит A, т.е. напечатает описание $\langle B\rangle$ на ленту, затем передаст управление B. В свою очередь B напечатает $P_{\langle B\rangle}$, т.е. A, на ленту и объединит его с $\langle B\rangle$ в единую машину Тьюринга:

$$\langle P_{\langle B\rangle}B\rangle = \langle AB\rangle = \langle SELF\rangle.$$

Теорема 33 (о рекурсии)**.** *Пусть* $t\colon \Sigma^* \times \Sigma^* \to \Sigma^*$ *– вычислимая функция, которая описывается машиной Тьюринга* T. *Тогда существует вычислимая функция* $r\colon \Sigma^* \to \Sigma^*$, *которая описывается машиной Тьюринга* R, *при этом для любой* $w \in \Sigma^*$

$$r(w) = t(\langle R\rangle, w).$$

Доказательство. Построим МТ R, реализующую вычислимую функцию r в три части:

$$\langle R\rangle = \langle ABT\rangle.$$

МТ B определим аналогично построению $SELF$. В качестве A рассмотрим $\tilde{P}_{\langle BT\rangle}$, где $\tilde{P}_w$ определена аналогично Q из леммы 6, кроме того, что она не удаляет входную строку с ленты.

При вычислении $r(w)$ после работы подпрограммы A на ленте будет записано $w\langle BT\rangle$. Затем B построит $\langle A\rangle = \langle \tilde{P}_{\langle BT\rangle}\rangle$ и объединит вместе с содержимым ленты в единую машину Тьюринга:

$$w\langle ABT\rangle = w\langle R\rangle.$$

После чего вычисления продолжит T на входе, на входе которой будет $w\langle R\rangle$. Т.е. $t(\langle R\rangle, w) = r(w)$. □

Замечание. Теорема о рекурсии утверждает, что допустимо построение такой МТ, которая может напечатать на ленту своё собственное описание и продолжить на нём вычисления.

Пример 33. Опишем МТ $SELF$ в терминах теоремы о рекурсии:

$SELF =$ *на входе* w :

1. *Получить своё описание в силу теоремы о рекурсии.*
2. *Вывести* $\langle SELF\rangle$ *на ленту.*

Тогда часть T имеет следующий вид:

$T =$ *на входе* $\langle M, w\rangle$, *где* M *– МТ* :

1. *Напечатать* $\langle M\rangle$ *и остановится.*

Представим ещё одно доказательство теоремы 22 на основе теоремы о рекурсии.

Теорема 34. *Язык A_{TM} неразрешим по Тьюрингу.*

Доказательство. Предположим, что A_{TM} разрешим, и M' решает A_{TM}. Построим следующую машину Тьюринга M_1:
$M_1 =$ *на входе* w :

1. *Получить своё описание* $\langle M_1 \rangle$ *в силу теоремы о рекурсии.*
2. *Запустить* M' *на* $\langle M_1, w \rangle$.
3. *Ответить противоположное* M'.

Тогда по построению M_1 допускает w тогда и только тогда, когда M_1 отвергает w. Получили противоречие. Следовательно, A_{TM} неразрешим по Тьюрингу. □

Определение. Говорят, что машина Тьюринга M *минимальна*, если для любой другой машины Тьюринга M', такой что $L(M) = L(M')$, выполняется условие

$$|\langle M \rangle| \leqslant |\langle M' \rangle|.$$

Через MIN_{TM} обозначим язык всех минимальных машин Тьюринга:

$$MIN_{TM} = \{\langle M \rangle : \ M \text{ – минимальная машина Тьюринга}\}.$$

Теорема 35. *Язык MIN_{TM} нераспознаваем по Тьюрингу.*

Доказательство. Предположим, что MIN_{TM} распознаваем. Тогда в силу теоремы 13 MIN_{TM} перечислим, и существует E – перечислитель MIN_{TM}. Построим машину Тьюринга M_2:
$M_2 =$ *на входе* w :

1. *Получить своё описание* $\langle M_2 \rangle$ *в силу теоремы о рекурсии.*
2. *Симулировать E до тех пор, пока он не выведет машину Тьюринга M' такую, что* $|\langle M_2 \rangle| < |\langle M' \rangle|$.
3. *Симулировать M' на w и ответить то же самое.*

По построению $L(M_2) = L(M')$ и M_2 имеет более краткое описание, чем M', что невозможно, т.к. $M' \in MIN_{TM}$. Получили противоречие. Таким образом, MIN_{TM} не распознаваем. □

Теорема 36 (о неподвижной точке)**.** *Пусть $t\colon \Sigma^* \to \Sigma^*$ – вычислимая функция. Предполагается, что все $w \in \Sigma^*$, которые не являются описаниями машин Тьюринга, рассматриваются в качестве машин Тьюринга, которые всегда отвергают.*

Тогда существует машина Тьюринга M, для которой $t(\langle M \rangle)$ – описание машины Тьюринга, которая эквивалентна M.

Доказательство. Построим машину Тьюринга M:
$M =$ *на входе* w :

1. *Получить своё описание* $\langle M \rangle$ *в силу теоремы о рекурсии.*
2. *Вычислить* $t(\langle M \rangle) = \langle M' \rangle$.
3. *Симулировать M' на w и ответить также.*

Таким образом, $\langle M \rangle$ и $t(\langle M \rangle) = \langle M' \rangle$ по построению эквивалентны, поскольку M симулирует M'. □

10 Понятие информации

Рассмотрим применение концепции машины Тьюринга для определения понятия информации. Количество информации, содержащейся в одной строчке не обязательно прямопропорционально её длине. Например, рассмотрим две строки

$$u = 0101010101, \quad v = 10010111.$$

Хотя строка u длиннее строки v, интуитивно она содержит меньше информации, так как является повтором подстроки 01 пять раз. Формализовать данное интуитивное предположение можно, если вместо самих строк использовать их описание.

Определение. *Описанием строки* $x \in \Sigma^*$ называется пара $\langle M, w\rangle$, где M – МТ, которая, получая на вход w, завершает вычисления с x на ленте.

Замечание. Поскольку далее в этом разделе будет предполагаться, что $\Sigma = \{0, 1\}$, то следует объяснить, каким образ могут быть разделены описания M и w в формате входа $\langle M, w\rangle$. Простейшим способом является удвоить все символы в описании M, а разделитель обозначить, например, подстрокой 01:

$$\underbrace{110000111111\ldots0011}_{\langle M\rangle}01\underbrace{10001101}_{w}.$$

Также в дальнейшем будем полагать, что разделитель включен в описание M.

Определение. *Минимальным описанием* $d(x)$ *строки* $x \in \Sigma^*$ называется кратчайшее по длине описание x. Если таких описаний несколько, то в качестве минимального выбирается лексикографически первое.

Определение. *Описательной сложностью строки* $x \in \Sigma^*$ называется длина минимального описания:

$$K(x) = |d(x)|.$$

Теорема 37. *Существует* $C \in \mathbb{N}$ *такая, что для всех* $x \in \Sigma^*$ *справедливо неравенство*

$$K(x) \leqslant |x| + C.$$

Доказательство. Пусть M_1 – МТ, которая сразу останавливается на любом входе. Тогда по определению $\langle M_1, x\rangle$ – описание x. При этом справедливо неравенство:

$$K(x) \leqslant |\langle M_1, x\rangle| = |x| + |\langle M_1\rangle|.$$

□

Теорема 38. *Существует* $C \in \mathbb{N}$ *такая, что для всех* $x \in \Sigma^*$ *справедливо неравенство*

$$K(xx) \leqslant K(x) + C.$$

Доказательство. Построим следующую МТ:

$M_2 =$ на входе $\langle M, w\rangle$, где M – МТ :

1) Запустить M на w и оставить на ленте только s – результат вычислений M.

2) Продублировать s.

Тогда по построению $\langle\langle M_2\rangle d(x)\rangle$ – описание xx, что приводит к соотношениям

$$K(xx) \leqslant |\langle M_2\rangle| + |d(x)| = K(x) + |\langle M_2\rangle|.$$

□

Теорема 39. *Существует* $C \in \mathbb{N}$ *такая, что для любых* $x, y \in \Sigma^*$ *справедливо неравенство*

$$K(xy) \leqslant 2K(x) + K(y) + C.$$

Доказательство. Построим следующую МТ:

$M_3 =$ на входе $\langle\langle M', w'\rangle, \langle M'', w''\rangle\rangle$ где M', M'' – МТ :

1) Разбить ленту на две области.

2) В первой области запустить M' на w' и оставить на этой части ленты только s' – результат вычислений M'.

3) В второй области запустить M'' на w'' и оставить на этой части ленты только s'' – результат вычислений M''.

4) Объединить s' и s'' в одну строку.

Тогда по построению $\langle\langle M_2\rangle d(x), d(y)\rangle$ – описание xy. Если предположить, что описание $d(x)$ отделено от $d(y)$ за счёт удваивания символов, то можно получить соотношения

$$K(xy) \leqslant |\langle\langle M_2\rangle d(x), d(y)\rangle| = 2|d(x)| + |d(y)| + |\langle M_2\rangle| = 2K(x) + K(y) + |\langle M_2\rangle|.$$

□

Замечание. Теорему 39 можно усилить, если для разделения $d(x)$ и $d(y)$ записать перед $d(x)$ его длину целым бинарным числом с удвоенными символами. Тогда будет справедлива оценка

$$K(xy) \leqslant 2\log_2(K(x)) + K(x) + K(y) + C.$$

Определение. Строка $x \in \Sigma^*$ называется *c-сжимаемой*, если $K(x) \leqslant |x| - c$. Если строка не сжимаема на 1, то такая строка называется *несжимаемой.*

Теорема 40. *Существуют несжимаемые строки любой длины.*

Доказательство. Для любого $n \in \mathbb{N}$ существует ровно 2^n различных строк длины n. Тогда строк длины не более 2^{n-1} существует

$$\sum_{i=0}^{n-1} 2^i = 2^n - 1.$$

Таким образом, даже если предположить, что каждая строка меньшей длины является описанием какой-либо строки большей длины, то по крайней мере для одной строки длины n не найдётся более краткого описания. Т.е. эта строка будет несжимаемой. □

Следствие 8. *Для любого $n > c$ существует по крайней мере $2^n - 2^{n-c+1} + 1$ несжимаемых на c строк.*

Определение. *Вычислимым свойством строк* называется вычислимая функция $f\colon \Sigma^* \to \{TRUE, FLASE\}$. Говорят, что *свойство f выполняется почти для всех строк*, если доля строк w длины n, для которых $f(w) = TRUE$ стремится к 0 при $n \to \infty$.

Теорема 41. *Пусть вычислимое свойство f выполняется почти для всех строк. Тогда для любого $c > 0$ свойство f ложно только для конечного числа несжимаемых на c строк.*

Доказательство. Если число строк, для которых f ложно, конечно, то теорема выполняется автоматическ. Будем далее полагать, что число таких строк бесконечно. В этом случае следующая МТ будет завершать свои вычисления за конечное время:

$M_4 =$ на входе i, где i – двоичное натуральное число :

1) Последовательным перебором вычислить $s \in \Sigma^*$,

где s – лексикографически i-я строка, для которой $f(s) = FALSE$.

2) Вывести s.

Тогда для произвольной строки $x \in \Sigma^*$, такой что $f(x) = FALSE$, существует $i_x \in \mathbb{N}$, для которого $\langle M, i_x \rangle$ – описание x.

$$|\langle M, i_x \rangle| = |i_x| + k, \text{ где } k = |\langle M \rangle|.$$

Поскольку f выполняется почти для всех строк, то для каждого фиксированного $c > 0$ можно подобрать $n \in \mathbb{N}$ так, что не более $\frac{1}{2^{c+k+1}}$ доли всех строк длины не более n не обладают свойством f. Если $x \in \Sigma^*$ такая, что $f(x) = FALSE$, $|x| = n$, то

$$i_x \leqslant \frac{2^{n+1}}{2^{c+k+1}} \leqslant 2^{n-c-k}.$$

Т.к. i_x – бинарное число, то $|i_x| \leqslant n - c - k$. Т.е.

$$|\langle M, i_x \rangle| \leqslant (n - c - k) + k = n - b,$$

$$K(x) \leqslant n - b.$$

Следовательно, любая достаточно длинная строка, не обладающая свойством f, является c-сжимаемой. Или же только конечное число строк, не обладающих свойством f, являются несжимаемыми на c. □

Теорема 42. *Существует $c \in \mathbb{N}$ такое, что для любой $x \in \Sigma^*$ минимальное описание $d(x)$ несжимаемо на c.*

Доказательство. Построим следующую МТ:

$M_5 =$ на входе $\langle M, w \rangle$, где M – МТ :
1) Запустить M на w.
2) Отвергнуть, если вывод не имеет вид $\langle M', w' \rangle$, где M' – МТ.
2) Запустить M' на w' и вывести на ленту результат вычислений M'.

Пусть $c = |\langle M \rangle| + 1$. Предположим, что для некоторой строки $x \in \Sigma^*$ минимальное описание $d(x)$ c-сжимаемо. Тогда

$$|d(d(x))| \leqslant |d(x)| - c.$$

Но при этом $\langle \langle M \rangle, d(d(x)) \rangle$ – описание x, и при этом

$$|\langle \langle M \rangle, d(d(x)) \rangle| = (c - 1) + |d(d(x))| \leqslant c - 1 + |d(x)| - c = |d(x)| - 1.$$

Т.е. описание $\langle \langle M \rangle, d(d(x)) \rangle$ короче минималного описания x. Получили противоречие. Тогда $d(x)$ несжимаема на c. □

Замечание. Приведём без доказательства некоторые свойства описательной сложности:

1. описательная сложность $K \colon \Sigma^* \to \mathbb{N}$ не является вычислимой функцией;
2. задача проверки, является ли некоторая строка несжимаемой, неразрешима по Тьюрингу;
3. не существует бесконечного подмножества множества всех несжимаемых строк, которое было бы распознаваемо по Тьюрингу.