S©MR ISSN 1813-3304

СИБИРСКИЕ ЭЛЕКТРОННЫЕ
МАТЕМАТИЧЕСКИЕ ИЗВЕСТИЯ

Siberian Electronic Mathematical Reports
http://semr.math.nsc.ru

*Том 14, стр. 732–736 (2017)* УДК 510.57
DOI 10.17377/semi.2017.14.062 MSC 11U99

# ГЕНЕРИЧЕСКАЯ ТЕОРЕМА КЛИНИ О НЕПОДВИЖНОЙ ТОЧКЕ

А.Н. РЫБАЛОВ

Abstract. Kleene's fixed point theorem states that any algorithmic mapping $\mathcal{A}$ of the set of Turing machines to the set of Turing machines has a fixed point: there is a Turing machine $M$ such that machine $\mathcal{A}(M)$ computes the same function as $M$. In this paper we prove a generic analog of this theorem: any algorithmic mapping $\mathcal{A}$ of a set of „almost all“ Turing machines to the set of Turing machines has a fixed point.

**Keywords:** Kleene fixed point theorem, generic computability.

## 1. Введение

Генерический подход к теории вычислимости и вычислительной сложности был предложен в работе [2]. В рамках этого подхода алгоритмическая проблема рассматривается не на всем множестве входов, а на некотором подмножестве "почти всех"входов. Такие входы образуют так называемое генерическое множество. Понятие "почти все"формализуется введением естественной меры на множестве входных данных. С точки зрения практики, алгоритмы, решающие быстро проблему на генерическом множестве, так же хороши, как и быстрые алгоритмы для всех входов. Классическим примером такого алгоритма является симплекс-метод – он за полиномиальное время решает задачу линейного программирования для большинства входных данных, но имеет экспоненциальную сложность в худшем случае.

Одним из направлений исследований в теории генерической вычислимости является изучение генерических аналогов классических результатов теории вычислимости и математической логики. Например, в работе [8] был доказан

Rybalov, A.N., Generic Kleene fixed point theorem.



Работа поддержана грантом РФФИ (проект №16-01-00577).
*Поступила 26 апреля 2017 г., опубликована 1 августа 2017 г.*

генерический аналог теоремы Геделя о неполноте, а в работе [6] были получены генерические аналоги классических результатов о неразрешимых алгоритмических проблемах. Классическая теорема Клини о неподвижной точке (см. [1, 3, 4, 5, 7, 9]), известная также как теорема Клини о рекурсии, утверждает, что для любого алгоритмически вычислимого отображения множества программ машин Тьюринга на множество программ машин Тьюринга существует неподвижная точка: найдется такая машина, что и она и ее образ под действием этого отображения вычисляют одну и ту же функцию. В данной работе доказывается генерический аналог этой теоремы: для любого алгоритмически вычислимого отображения генерического рекурсивного подмножества программ машин Тьюринга на множество программ машин Тьюринга существует неподвижная точка.

## 2. Генерические алгоритмы

Пусть $I$ – некоторое множество. Функция $size : I \to \mathbb{N}$ называется *функцией размера*, если для любого $n \in \mathbb{N}$ множество $I_n = \{x \in I : size(x) = n\}$ конечно. Например, если $I = \Sigma^*$ – множество слов над конечным алфавитом $\Sigma$, то функцией размера будет функция, определенная для любого слова $w$ как его длина $|w|$. Также для множества натуральных чисел $\mathbb{N}$ функция размера сопоставляет любому натуральному числу длину его двоичной записи. Как обычно делается в теории вычислимости, мы будем под алгоритмическими проблемами понимать проблемы распознавания подмножеств из некоторого множества входов с определенной на нем функцией размера.

Для подмножества $S \subseteq I$ определим последовательность

$$\rho_n(S) = \frac{|S_n|}{|I_n|},\ n = 1, 2, 3, \ldots,$$

где $S_n = S \cap I_n$ – множество входов из $S$ размера $n$. Заметим, что $\rho_n(S)$ это вероятность попасть в $S$ при случайной и равновероятной генерации входов из $I_n$. *Асимптотической плотностью* $S$ назовем предел (если он существует)

$$\rho(S) = \lim_{n\to\infty} \rho_n(S).$$

Множество $S$ называется *генерическим*, если $\rho(S) = 1$ и *пренебрежимым*, если $\rho(S) = 0$. Очевидно, что $S$ генерическое тогда и только тогда, когда его дополнение $I \setminus S$ пренебрежимо.

Алгоритм $\mathcal{A}$ с множеством входов $I$ и множеством выходов $J \cup \{?\}$ $(? \notin J)$ называется *генерическим*, если

(1) $\mathcal{A}$ останавливается на всех входах из $I$,
(2) множество $\{x \in I : \mathcal{A}(x) =?\}$ пренебрежимо.

Генерический алгоритм $\mathcal{A}$ вычисляет функцию $f : I \to J$, если для всех $x \in I$ $\mathcal{A}(x) = y \in J \Rightarrow f(x) = y$. Ситуация $\mathcal{A}(x) =?$ означает, что $\mathcal{A}$ не может вычислить функцию $f$ на аргументе $x$. Но условие 2 гарантирует то, что $\mathcal{A}$ корректно вычисляет $f$ на почти всех входах (входах из генерического множества).

## 3. Нормализованные машины Тьюринга

Будем рассматривать машины Тьюринга с рабочим алфавитом $\{0, 1\}$. Программа машины Тьюринга с внутренними состояниями $\{q_0, q_1, \ldots, q_n\}$ над алфавитом $\{0, 1\}$ состоит из правил вида $(q_i, a) \to (q_j, b, S)$ – по одному правилу

для всех ненулевых (нефинальных) состояний $q_i$ и символов алфавита $a$. Всего получается $2n$ правил. Такое правило предписывает, что машина, находясь в нефинальном состоянии $q_i$ и видя кареткой на ленте символ $a$, должна перейти в состояние $q_j$ (оно может быть и финальным $q_0$), записать на ленте символ $b$ и сдвинуть каретку на сдвиг $S \in \{R, L\}$ (вправо, влево). Под размером машины будем понимать число нефинальных внутренних состояний $n$. Будем отождествлять машины и их программы.

Машина Тьюринга называется *нормализованной*, если из любого нефинального состояния $q_i$ в соответствующих ему правилах в программе возможен переход в состояния $q_j$, где $j \leq 2i + 1$. Смысл этого ограничения соответствует процессу разумного написания программы: начиная с первого правила, в каждом очередном новом правиле можно либо попасть в одно из старых состояний, либо в одно новое. Следующая лемма говорит о том, что класс вычислимых функций при этом не сужается.

**Лемма 1.** *Для любой машины Тьюринга $M$ существует нормализованная машина Тьюринга $M'$, вычисляющая ту же функцию.*

*Доказательство.* Заметим, что если в машине переименовать нефинальные состояния $q_i$ в $q_j$ и наоборот, а также в программе поменять местами соответствующие им правила, то работа машины на любом входе не изменится. Поэтому начиная с первого состояния можно нормализовывать все правила, переименовывая если нужно состояния, в которые происходят переходы. □

В дальнейшем будем рассматривать только нормализованные машины Тьюринга, для краткости опуская слово „нормализованная“. Через $\mathcal{P}$ обозначим множество всевозможных программ таких машин.

**Лемма 2.** *Число машин Тьюринга размера $n$ равно*

$$|\mathcal{P}_n| = 4^{2n}(n+1)^{2(n-\lfloor (n-1)/2 \rfloor)} \prod_{i=1}^{\lfloor (n-1)/2 \rfloor} (2i+2)^2.$$

*Доказательство.* Для каждого из 2 правил, соответствующих состоянию $q_i$, если $2i + 1 < n$, существует $2i + 2$ варианта выбора состояния для перехода (учитываем также финальное состояние). Для состояния $q_i$, где $2i + 1 \geq n$, существует $n+1$ вариант выбора состояния для перехода. Также есть 2 варианта для символа записи и 2 варианта для сдвига. Перемножая эти количества для всех $n$ состояний (по 2 раза каждое), получаем нужный результат. □

Пусть $M$ – любая машина Тьюринга, обозначим через $c(M)$ множество машин, которые получаются из $M$ добавлением любого количества новых состояний и приписыванием к программе $M$ произвольных правил для новых состояний так, чтобы машина оставалась нормализованной. Все получающиеся таким образом машины будут вычислять ту же функцию, что и $M$, потому что на любом входе новые состояния будут недостижимы и работа будет происходить фактически по программе машины $M$.

**Лемма 3.** *Для любой машины $M$ множество $c(M)$ не пренебрежимо.*

*Доказательство.* Пусть $M$ имеет $k$ состояний. Рассмотрим множество машин $c(M)_{k+n}$ размера $k + n$ с $n$ новыми состояниями. Посчитаем число таких машин. Оно равно числу фрагментов программ для новых состояний. Аналогично

тому, как это делалось в доказательстве леммы 2, получаем

$$|c(M)_{k+n}| = 4^{2n}(k+n+1)^{2(k+n-\lfloor (k+n-1)/2 \rfloor)} \prod_{i=k+1}^{\lfloor (k+n-1)/2 \rfloor} (2i+2)^2.$$

Поэтому

$$\rho(c(M)) = \lim_{n\to\infty} \frac{|c(M)_{k+n}|}{|\mathcal{P}_{k+n}|} =$$

$$= \lim_{n\to\infty} \frac{4^{2n}(k+n+1)^{2(k+n-\lfloor (k+n-1)/2 \rfloor)} \prod_{i=k+1}^{\lfloor (k+n-1)/2 \rfloor} (2i+2)^2}{4^{2(k+n)}(n+k+1)^{2(k+n-\lfloor (k+n-1)/2 \rfloor)} \prod_{i=1}^{\lfloor (k+n-1)/2 \rfloor} (2i+2)^2} =$$

$$= \lim_{n\to\infty} \frac{1}{4^{2k} \prod_{i=1}^{k} (2i+2)^2} = \lim_{n\to\infty} const = const > 0.$$

□

Теперь все готово, чтобы доказать основной результат статьи.

**Теорема 1.** *Для любого генерического алгоритма $\mathcal{A} : \mathcal{P} \to \mathcal{P} \cup \{?\}$ существует такая машина Тьюринга $M$, что $\mathcal{A}(M) \neq ?$ и машина $\mathcal{A}(M)$ вычисляет ту же функцию, что и $M$.*

*Доказательство.* Допустим противное, то есть, что существует такой генерический алгоритм $\mathcal{A} : \mathcal{P} \to \mathcal{P} \cup \{?\}$, что для любой машины Тьюринга $M$ если $\mathcal{A}(M) \neq ?$, то $\mathcal{A}(M)$ вычисляет функцию, отличную от той, что вычисляет $M$. Построим теперь обычный алгоритм $\mathcal{B} : \mathcal{P} \to \mathcal{P}$, определенный для любой машины Тьюринга, но не имеющий неподвижной точки. Это будет противоречить классической теореме о неподвижной точке. Алгоритм $\mathcal{B}$ будет работать на машине $M$ следующим образом. Перебирает множество $c(M)$ в порядке возрастания размеров машин и на каждой машине запускает алгоритм $\mathcal{A}$ до тех пор пока не найдет машину $M^*$ такую, что $\mathcal{A}(M^*) \neq ?$. Это всегда произойдет, так как, по лемме 3, множество $c(M)$ не является пренебрежимым, а множество $\{M \in \mathcal{P} : \mathcal{A}(M) = ?\}$ пренебрежимо. По предположению, машина $\mathcal{A}(M^*)$ вычисляет функцию, отличную от функции, вычисляемой машиной $M^*$, которая вычисляет ту же функцию, что и $M$. Поэтому алгоритм $\mathcal{B}$ не имеет неподвижной точки. Противоречие. □

## References


[1] N.J. Cutland, *Computability: An introduction to recursive function theory*, Cambridge University Press, 1980. MR0572788

[2] I. Kapovich, A. Myasnikov, P. Schupp, V. Shpilrain, *Generic-case complexity, decision problems in group theory and random walks*, Journal of Algebra, **264**:2 (2003), 665–694. MR1981427

[3] S.C. Kleene, *On Notation for Ordinal Numbers*, Journal of Symbolic Logic, **3** (1938), 150—155. Zbl 0020.33803

[4] S.C. Kleene, *Introduction to Metamathematics*, North-Holland, 1952. MR0051790

[5] A.I. Maltsev, *Algorithms and recursive functions*, Groningen : Wolters-Noordhoff, 1970. MR0263632

[6] A. Myasnikov, A. Rybalov, *Generic complexity of undecidable problems*, Journal of Symbolic Logic, **73**:2 (2008), 656–673. MR2414470

[7] H. Rogers, *Theory of Recursive Functions and Effective Computability*, MIT Press, 1967. MR0224462

[8] A. Rybalov, *Generic incompletness of Formal Arithmetic*, Siberian electronic mathematical reports, **12** (2015), 185–189. MR3493725
[9] Vereshchagin N., Shen A, *Computable functions*, American Mathematical Society. Student mathematical library, **19** (2003). MR1946348

Alexander Nikolaevich Rybalov
Omsk State University,
prospekt Mira 55A,
Omsk 644077, Russia,
Sobolev Institute of Mathematics,
Pevtsova str. 13,
Omsk 644043, Russia
*E-mail address*: alexander.rybalov@gmail.com