МАТЕМАТИЧЕСКИЕ ОСНОВЫ ИНФОРМАТИКИ И ПРОГРАММИРОВАНИЯ

УДК 510.52

О ГЕНЕРИЧЕСКОЙ СЛОЖНОСТИ ПРОБЛЕМЫ ОБЩЕЗНАЧИМОСТИ БУЛЕВЫХ ФОРМУЛ[1]

А. Н. Рыбалов

Институт математики им. С. Л. Соболева СО РАН, г. Новосибирск, Россия

Генерический подход к алгоритмическим проблемам предложен А. Мясниковым, И. Каповичем, П. Шуппом и В. Шпильрайном в 2003 г. В рамках этого подхода рассматривается поведение алгоритмов на множествах почти всех входов. В данной работе изучается генерическая сложность проблемы общезначимости (тождественной истинности) булевых формул. Доказывается, что эта проблема неразрешима за полиномиальное время на любом полиномиальном строго генерическом множестве формул при условии её трудноразрешимости в худшем случае.

Ключевые слова: *генерическая сложность, проблема общезначимости булевых формул.*

DOI 10.17223/20710410/32/9

ON GENERIC COMPLEXITY OF THE VALIDITY PROBLEM FOR BOOLEAN FORMULAS

A. N. Rybalov

Sobolev Institute of Mathematics SB RAS, Novosibirsk, Russia

E-mail: alexander.rybalov@gmail.com

Generic-case approach to algorithmic problems was suggested by Miasnikov, Kapovich, Schupp and Shpilrain in 2003. This approach studies behavior of an algorithm on typical (almost all) inputs and ignores the rest of inputs. In this paper, we consider generic complexity of the validity problem for Boolean formulas and prove that this problem is generically hard if it is hard in the worst case.

Keywords: *generic complexity, validity problem for Boolean formulas.*

Введение

В работе [1] развита теория генерической сложности вычислений. В рамках этого подхода алгоритмическая проблема рассматривается не на всём 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,

[1]Работа поддержана грантом РФФИ № 16-01-00577.

быстро решающие 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. Более того, может так оказаться, что проблема трудноразрешима или вообще неразрешима в классическом смысле, но легкоразрешима на генерическом множестве. В [1, 2] доказано, что таким поведением обладают многие алгоритмические проблемы алгебры, а в [3] построено генерическое множество, на котором разрешима классическая проблема остановки для машин Тьюринга с лентой, бесконечной в одном направлении. Для многих классических NP-полных проблем существуют полиномиальные генерические алгоритмы [4].

Проблема общезначимости булевых формул состоит в следующем: для любой булевой формулы, записанной в стандартном базисе $\{\vee, \wedge, \neg\}$, определить, является ли она тождественно истинной. Из классического результата С. Кука [5] о NP-полноте проблемы выполнимости следует, что проблема общезначимости является полной относительно полиномиальной сводимости в классе co-NP (он состоит из множеств, являющихся дополнениями к множествам из класса NP). Это означает, что, при условии неравенства классов P и NP, для неё не существует полиномиального алгоритма, решающего её на всём множестве булевых формул. Поэтому возникает вопрос об изучении подпроблем проблемы общезначимости и построению для них эффективных разрешающих алгоритмов. Естественным желанием является то, чтобы в эти классы попадало как можно больше формул, а в идеале «почти все» формулы. В терминах теории генерической сложности речь идёт об алгоритмах, работающих быстро на генерических множествах формул.

В данной работе доказывается, что проблема общезначимости булевых формул неразрешима за полиномиальное время на любом полиномиальном строго генерическом множестве формул при условии несовпадения классов P и NP и совпадения классов P и BPP. Здесь класс BPP — это класс проблем, разрешимых за полиномиальное время на вероятностных машинах Тьюринга. Большинство исследователей сейчас считает, что имеет место равенство $\mathrm{P} = \mathrm{BPP}$. Это равенство означает, что любой полиномиальный вероятностный алгоритм можно эффективно дерандомизировать, т. е. построить полиномиальный детерминированный алгоритм, решающий ту же задачу. Хотя это равенство пока ещё не доказано, имеются серьёзные результаты в его пользу [6]. При доказательстве основного результата работы использованы методы, развитые в [7, 8].

1. Определения

Следуя [1], дадим основные определения теории генерической сложности вычисления. Пусть I — множество всех входов, а I_n — множество входов размера n. Для любого подмножества $S \subseteq I$ определим следующую последовательность

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

Величина $\rho_n(S)$ — это вероятность получить вход из множества S при случайной и равномерной генерации входов из I_n. *Асимптотической плотностью* S назовем следующий 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.

Следуя [1], назовём множество S *строго пренебрежимым*, если последовательность $\rho_n(S)$ экспоненциально быстро сходится к нулю, т. е. существуют константы $0 < \sigma < 1$ и $C > 0$, такие, что для любого n

$$\rho_n(S) < C\sigma^n.$$

Теперь S называется *строго генерическим*, если его дополнение $I \setminus S$ строго пренебрежимо.

Множество $S \subseteq I$ *генерически разрешимо за полиномиальное время*, если существует множество $G \subseteq I$, такое, что

1) G генерическое;
2) G разрешимое за полиномиальное время;
3) $S \cap G$ разрешимое за полиномиальное время.

Если G строго генерическое, то S называется *строго генерически разрешимым за полиномиальное время*. Генерический алгоритм $\mathcal{A}$ для S работает на входе $x \in I$ следующим образом. Сначала $\mathcal{A}$ решает, принадлежит ли x множеству G. Если $x \in G$, то $\mathcal{A}$ может решить S на G, иначе $\mathcal{A}$ отвечает «Я НЕ ЗНАЮ!» Таким образом, $\mathcal{A}$ корректно решает S на «почти всех» входах (входах из генерического множества).

Имеется существенное различие между генерически разрешимыми проблемами и строго генерически разрешимыми проблемами. Допустим, имеется проблема S, разрешимая на некотором разрешимом генерическом множестве G, для которого

$$\frac{|G \cap I_n|}{|I_n|} = \frac{n-1}{n}.$$

Таким образом G — генерическое, но не строго генерическое множество. Теперь хоть проблема S и разрешима для почти всех входов, тем не менее есть быстрый способ получить «плохой» вход, на котором генерический алгоритм не работает. Быстрый (полиномиальный) алгоритм для генерации плохих входов следующий:

1) сгенерировать равномерно случайный вход x размера n;
2) если $x \in G$, повторить шаг 1, иначе закончить.

Действительно, вероятность получить только хорошие входы за n^2 раундов

$$\left(\frac{n-1}{n}\right)^{n^2} = \left(\left(1 - \frac{1}{n}\right)^n\right)^n \to e^{-n},$$

поэтому с вероятностью, очень близкой к 1, будет получен плохой вход. С другой стороны, легко видеть, что если проблема разрешима на строго генерическом множестве, то такой простой алгоритм генерации потребует экспоненциального числа раундов и будет неэффективным. Для приложений к криптографии это означает, что просто генерическая легкоразрешимость проблемы не делает эту проблему бесполезной для создания на её основе криптосистемы, так как для неё существует эффективная процедура генерации трудных входов. В то же время строго генерически легкоразрешимые проблемы в этом смысле бесполезны для криптографии.

2. Представление булевых формул

Классическое представление булевых формул с помощью таблиц истинности с практической точки зрения является громоздким в том смысле, что размер таблицы истинности растет экспоненциально с ростом числа переменных. Гораздо более компактным и практичным является представление формул с помощью бинарных деревьев. Оно часто используется в программировании различных приложений, связанных с символьными вычислениями. Кроме того, оно удобно для различного рода подсчётов.

Пусть ϕ — булева формула в базисе $\{\vee, \wedge, \neg\}$. Без ограничения общности можно считать, что в ней отрицания находятся только над переменными. Любую булеву формулу можно легко привести к такому виду с помощью законов де Моргана, поэтому в дальнейшем будем рассматривать только такие формулы. Естественным образом формуле ϕ можно сопоставить бинарное дерево T_ϕ, которое представляет конструкцию ϕ из переменных и их отрицаний с помощью конъюнкций и дизъюнкций. Внутренние вершины T_ϕ помечены символами $\vee$ и $\wedge$, а листья T_ϕ — переменными или их отрицаниями. С другой стороны, по любому такому бинарному дереву можно восстановить булеву формулу. Это дает взаимно однозначное представление булевых формул размеченными бинарными деревьями. Если T_ϕ имеет n листьев, то не более n переменных могут встретиться в T_ϕ, поэтому в дальнейшем будем полагать, что все переменные T_ϕ лежат в множестве $\{x_1, \ldots, x_n\}$. *Представление* ϕ состоит из бинарного дерева T_ϕ. Заметим также, что число булевых операций в ϕ равно $n - 1$. Под размером формулы ϕ будем понимать число листьев n.

Например, на рис. 1 представлена формула $(\neg x_1) \wedge (x_2 \vee \neg x_3)$.

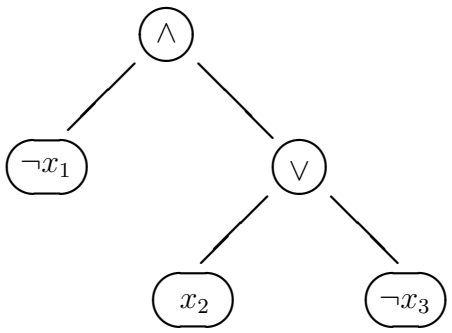

Рис. 1

В дальнейшем будем отождествлять булевы формулы с их представлениями. Обозначим через $\mathcal{F}$ множество всех булевых формул, а через $\mathcal{F}_n$ — множество всех формул в $\mathcal{F}$ размера n. Напомним, что числа Каталана C_n определяются следующим образом:

$$C_n = \frac{1}{n+1}\binom{2n}{n},$$

где $\binom{2n}{n}$ — соответствующий биномиальный коэффициент.

Лемма 1. $|\mathcal{F}_n| = 2^{n-1}(2n)^n C_{n-1}$.

Доказательство. Любая формула из $\mathcal{F}$ размера n есть размеченное бинарное дерево с n листьями и $n-1$ внутренними вершинами. Известно [9], что существует C_{n-1}

неразмеченных бинарных деревьев с n листьями. Каждая внутренняя вершина такого дерева может быть помечена символами $\vee$ или $\wedge$, поэтому есть всего 2^{n-1} таких разметок. Каждый лист может быть помечен одной из n переменных или её отрицанием, поэтому существует $(2n)^n$ таких разметок. Это показывает, что $|\mathcal{F}_n| = 2^{n-1}(2n)^n C_{n-1}$. Лемма доказана. ■

Для любой формулы ϕ определим множество

$$OR(\phi) = \{\phi \vee \psi,\ \psi \text{ — произвольная формула}\}.$$

Лемма 2. Для любой формулы ϕ множество $OR(\phi)$ не строго пренебрежимо. Более того,

$$\frac{|OR(\phi) \cap \mathcal{F}_n|}{|\mathcal{F}_n|} > \frac{1}{(16n)^k}$$

для любого $n > k$, где k — размер формулы ϕ.

Доказательство. Пусть формула ϕ имеет размер k. Тогда для любой формулы $\phi \vee \psi$ из множества $OR(\phi) \cap \mathcal{F}_n$ формула ψ должна иметь размер $n - k$. Кроме того, в этой формуле может участвовать любая из n переменных. Поэтому аналогично тому, как это делалось в доказательстве леммы 1, можно подсчитать

$$|OR(\phi) \cap \mathcal{F}_n| = 2^{n-k-1}(2n)^{n-k} C_{n-k-1}.$$

Отсюда

$$\frac{|OR(\phi) \cap \mathcal{F}_n|}{|\mathcal{F}_n|} = \frac{2^{n-k-1}(2n)^{n-k}C_{n-k-1}}{2^{n-1}(2n)^n C_{n-1}} = \frac{1}{(4n)^k}\frac{C_{n-k-1}}{C_{n-1}} = \frac{1}{(4n)^k}\frac{n}{n-k}\frac{\binom{2(n-k-1)}{n-k-1}}{\binom{2(n-1)}{n-1}} >$$

$$> \frac{1}{(4n)^k}\frac{(n-1)!}{(n-k-1)!}\frac{2(n-k-1)\dots(n-k)}{2(n-1)\dots n} = \frac{1}{(4n)^k}\frac{((n-1)\dots(n-k))^2}{2(n-1)\dots(2n-2k-1)} >$$

$$> \frac{1}{(4n)^k}\Big(\frac{(n-1)\dots(n-k)}{2(n-1)\dots(2n-2k-1)}\Big)^2 > \frac{1}{(4n)^k}\frac{1}{2^{2k}} = \frac{1}{(16n)^k}.$$

Таким образом, имеем $\dfrac{|OR(\Phi)_n|}{|\mathcal{F}_n|} > \dfrac{1}{(16n)^k}$, что и требовалось доказать. ■

3. Основной результат

Теорема 1. Если существует строго генерическое полиномиальное множество булевых формул, на котором проблема общезначимости булевых формул разрешима за полиномиальное время, то существует вероятностный полиномиальный алгоритм, разрешающий эту проблему на всём множестве формул.

Доказательство. Допустим, что существует строго генерическое разрешимое множество формул G, такое, что существует полиномиальный алгоритм $\mathfrak{A}$, определяющий для любой булевой формулы $\phi \in G$, является ли она тождественно истинной. Построим теперь алгоритм $\mathfrak{B}$, определяющий тождественную истинность любой формулы ϕ. На формуле ϕ размера n алгоритм $\mathfrak{B}$ будет работать следующим образом:

1) Проверяет, принадлежит ли ϕ множеству G. Если да, то с помощью алгоритма $\mathfrak{A}$ определяет выполнимость ϕ. Если нет, то переходит к шагу 2.
2) Генерирует случайную формулу ψ размера $n^2 - n$.

3) Проверяет, принадлежат ли формулы $\phi \vee \psi$ и $\phi \vee \neg\psi$ множеству G. Если обе формулы принадлежат G, то с помощью алгоритма $\mathfrak{A}$ определяет их общезначимость и переходит к шагу 4. Если не принадлежат, то выдаёт ответ «НЕ ОБЩЕЗНАЧИМА».
4) Так как обязательно хотя бы одна из формул ψ и $\neg\psi$ не является тождественно истинной, то возможны следующие варианты:
   — Если $\phi \vee \psi$ и $\phi \vee \neg\psi$ тождественно истинны, то ϕ тоже тождественно истинна и алгоритм выдаёт ответ «ОБЩЕЗНАЧИМА».
   — Если хотя бы одна из них не тождественно истинна, то и ϕ тоже не тождественно истинна и алгоритм выдаёт ответ «НЕ ОБЩЕЗНАЧИМА».

   В любом из этих двух случаев алгоритм выдаёт правильный ответ.

Заметим, что алгоритм выдаёт правильный ответ на шагах 1 и 4, а на шаге 3 может выдать неправильный ответ. Нужно доказать, что вероятность того, что ответ выдаётся на шаге 3, меньше $1/2$. Вероятность того, что случайная формула вида $\phi \wedge \psi$ из $OR(\phi)_{n^2}$ не попадёт в G, не больше

$$\frac{|(\mathcal{F} \setminus G)_{n^2}|}{|OR(\phi)_{n^2}|} = \frac{|(\mathcal{F} \setminus G)_{n^2}|}{|\mathcal{F}_{n^2}|} \frac{|\mathcal{F}_{n^2}|}{|OR(\Phi)_{n^2}|}$$

Так как G строго генерическое, то существует константа $\alpha > 0$, такая, что

$$\frac{|(\mathcal{F} \setminus G)_{n^2}|}{|\mathcal{F}_{n^2}|} < \frac{1}{2^{\alpha n^2}}$$

для любого n. С другой стороны, по лемме 2

$$\frac{|\mathcal{F}_{n^2}|}{|OR(\Phi)_{n^2}|} < (16n^2)^n.$$

Поэтому искомая вероятность не больше

$$\frac{(16n^2)^n}{2^{\alpha n^2}} = \frac{2^{4n+2n\log n}}{2^{\alpha n^2}}$$

и при больших n меньше $1/4$. Аналогично делается оценка для формул вида $\phi \vee \neg\psi$. Вероятность же непопадания в G хотя бы одной из формул $\phi \vee \psi$ или $\phi \vee \neg\psi$ не больше $1/4 + 1/4 = 1/2$. Это означает, что вероятность выдачи ответа на шаге 3 меньше $1/2$.

Осталось доказать полиномиальность алгоритма. Для этого нужно за полиномиальное время уметь генерировать случайно и равномерно формулу размера $N = n^2 - n$. Это делается следующим образом:

1) Генерируем некоторую последовательность (далее «слово») из N символов a и $N - 1$ символов p.
2) Делаем такой циклический сдвиг этого слова, чтобы оно начиналось на символ a и заканчивалось на p. Этому слову соответствует обратная польская запись для скобочного выражения от символов a.
3) По слову ищём скобочное выражение следующим образом: пробегаем по всем символам слова, если встречаем символ a, то помещаем его в стек. Если встречаем символ p, то извлекаем два элемента из стека, затем добавляем между ними символ p, заключаем их в скобки и помещаем в стек. Если по ходу процедуры стек окажется пуст, то переходим к шагу 2. Если все пройдёт нормально и мы дойдём до конца слова, и при этом в стеке останется всего один элемент, то искомым скобочным выражением и будет этот элемент. Иначе переходим к шагу 2.

4) Вместо букв p подставляем $\vee$ или $\wedge$ — равновероятно.
5) Каждую букву a в слове заменяем на переменную $x_1, \ldots, x_N$ или её отрицание (какую — выбираем равновероятно).

Корректность этого алгоритма и равномерность генерации формул следует из того, что существует взаимно-однозначное соответствие между обратными польскими записями из N символов a и $N-1$ символов p и бинарными деревьями с N листьями, которые помечены символом a [9].

Итак, в предположении существования полиномиального строго генерического множества, на котором проблема выполнимости булевых формул разрешима за полиномиальное время, построен вероятностный полиномиальный алгоритм, разрешающий эту проблему на всём множестве формул. ■

Непосредственно из теоремы 1 следует

Теорема 2. Если $\mathrm{P} \neq \mathrm{NP}$ и $\mathrm{P} = \mathrm{BPP}$, то не существует строго генерического полиномиального подмножества булевых формул, на котором проблема общезначимости булевых формул разрешима за полиномиальное время.

ЛИТЕРАТУРА

1. *Kapovich I., Miasnikov A., Schupp P., and Shpilrain V.* Generic-case complexity, decision problems in group theory and random walks // J. Algebra. 2003. V. 264. No. 2. P. 665–694.
2. *Kapovich I., Miasnikov A., Schupp P., and Shpilrain V.* Average-case complexity for the word and membership problems in group theory // Adv. Math. 2005. V. 190. P. 343–359.
3. *Hamkins J. D. and Miasnikov A. G.* The halting problem is decidable on a set of asymptotic probability one // Notre Dame J. Formal Logic. 2006. V. 47. No. 4. P. 515–524.
4. *Gilman R., Miasnikov A. G., Myasnikov A. D., and Ushakov A.* Report on generic case complexity // Herald of Omsk University. 2007. Special Issue. P. 103–110.
5. *Cook S. A.* The complexity of theorem proving procedures // Proc. 3d Annual ACM Symposium on Theory of Computing. N. Y., USA, 1971. P. 151–158.
6. *Impagliazzo R. and Wigderson A.* $\mathrm{P} = \mathrm{BPP}$ unless E has subexponential circuits: Derandomizing the XOR Lemma // Proc. 29th STOC. El Paso: ACM, 1997. P. 220–229.
7. *Myasnikov A. and Rybalov A.* Generic complexity of undecidable problems // J. Symbolic Logic. 2008. V. 73. No. 2. P. 656–673.
8. *Rybalov A.* Generic complexity of presburger arithmetic // Theory Comput. Systems. 2010. V. 46. No. 1. P. 2–8.
9. *Кнут Д.* Искусство программирования. М.: Вильямс, 2010. 720 с.

REFERENCES

1. *Kapovich I., Miasnikov A., Schupp P., and Shpilrain V.* Generic-case complexity, decision problems in group theory and random walks. J. Algebra, 2003, vol. 264, no. 2, pp. 665–694.
2. *Kapovich I., Miasnikov A., Schupp P., and Shpilrain V.* Average-case complexity for the word and membership problems in group theory. Adv. Math., 2005, vol. 190, pp. 343–359.
3. *Hamkins J. D. and Miasnikov A. G.* The halting problem is decidable on a set of asymptotic probability one. Notre Dame J. Formal Logic, 2006, vol. 47, no. 4, pp. 515–524.
4. *Gilman R., Miasnikov A. G., Myasnikov A. D., and Ushakov A.* Report on generic case complexity. Herald of Omsk University, 2007, Special Issue, pp. 103–110.
5. *Cook S. A.* The complexity of theorem proving procedures. Proc. 3d Annual ACM Symposium on Theory of Computing, N. Y., USA, 1971, pp. 151–158.

6. *Impagliazzo R. and Wigderson A.* P = BPP unless E has subexponential circuits: Derandomizing the XOR Lemma. Proc. 29th STOC, El Paso, ACM, 1997, pp. 220–229.
7. *Myasnikov A. and Rybalov A.* Generic complexity of undecidable problems. J. Symbolic Logic, 2008, vol. 73, no. 2, pp. 656–673.
8. *Rybalov A.* Generic complexity of presburger arithmetic. Theory Comput. Systems, 2010, vol. 46, no. 1, pp. 2–8.
9. *Knuth D. E.* The Art of Computer Programming. Reading, Massachusetts, Addison-Wesley, 1997.