Э. А. Гирш

ПРИНЦИП РАЗДЕЛЕНИЯ ЗНАКОВ ДЛЯ ЗАДАЧИ ПРОПОЗИЦИОНАЛЬНОЙ ВЫПОЛНИМОСТИ

§1. Введение

Общеизвестно, что задача выполнимости формул в КНФ (в дальнейшем обозначаемая ВЫП) NP-полна и полиномиальный алгоритм для ее разрешения неизвестен (см., например, [1]). При введении ограничений на какие-либо параметры формулы возникают новые задачи – подзадачи общей задачи ВЫП, которые также широко изучались. Например, в результате ограничения длины дизъюнкции возникают классы k–КНФ (формулы, каждая дизъюнкция которых содержит не более k литералов). В работе [10] были определены классы формул в КНФ с ограничением на число вхождений каждого литерала, обозначаемые КНФ–(p,m) (каждый положительный литерал входит в такую формулу не более p раз, каждый отрицательный – не более m раз). Для некоторых из изучавшихся подклассов КНФ задача выполнимости оказалась разрешима за полиномиальное время. Это задача выполнимости формул в 2–КНФ, КНФ–(1,1), хорновских КНФ, формул, реализующих системы линейных уравнений в $\mathbb{Z}_2$, и некоторых других классов формул в КНФ (обзор таких результатов имеется в [3]). Что же касается задачи выполнимости для формул таких подклассов, как 3–КНФ, КНФ–$(1,\infty)$, КНФ–(1,2) и многих других, она оказалась также NP-полна (эти задачи будут в дальнейшем обозначаться соответственно 3-ВЫП, ВЫП-$(1,\infty)$, ВЫП-(1,2) и т. д.). Тем не менее, для этих подзадач, как и для задачи выполнимости КНФ вообще, велись исследования по поводу улучшения верхних оценок, начиная с [11] (см. также [12]) и [2] (см. также [4]), где была впервые понижена очевидная верхняя оценка $p(L)2^N$ для общей задачи ВЫП (здесь и далее p – подходящий полином, L – длина исходной формулы, N – число

При частичной поддержке гранта INTAS–РФФИ 95–0095 и фонда Pro Mathematica (Франция)

переменных в ней, K – число дизъюнкций). Табл. 1 содержит известные результаты на эту тему, включая результаты, полученные в данной статье.

	число дизъюнкций K	длина длина L	число переменных N
КНФ	1.2389^K	1.0801^L	2^N
k–КНФ			$\psi(k)^N$
3–КНФ			1.4963^N
КНФ–(1,∞)	1.1939^K	1.0644^L	1.4423^N
k–КНФ–(1,∞)			$(\psi(k)+1)^{N/3}$
3–КНФ–(1,∞)			1.3566^N
КНФ–(1,2)		1.0416^L	1.1299^N
3–КНФ–(1,2)	1.1299^K		

Табл. 1. Верхние оценки для задачи выполнимости формул различных классов с точностью до полиномиального сомножителя $p(L)$. Величина $\psi(k) \in (1;2)$ определяется в §9.

Верхняя оценка $p(L)2^N$ для общей задачи выполнимости формулы в КНФ является тривиальной, оценка $p(L)1.2389^K$ получена в [6]. Оценка $p(L)\psi(k)^N$ для формул в k–КНФ была опубликована Б. Мониеном и Е. Шпекенмейером в [12] (величина $\psi(k) \in (1;2)$ определяется в §9), оценка $p(L)1.4963^N$ опубликована И. Шиермейером в [13], а также О. Кулльманном в [8]. О. Кулльманн и Г. Люкхардт доказали в [9], что выполнимость формулы в КНФ можно выяснить за время $p(L)1.0801^L$. Там же доказывается, что задача ВЫП-(1,2) решается за время $p(L)1.0416^L \approx p(L)1.1299^N$, задача 3-ВЫП-(1,2) – за время $p(L)1.1299^K$, а задача ВЫП-(1,∞) – за время $p(L)1.4423^N$ (последний результат был ранее опубликован Г. Люкхардтом в [10], там же этот класс формул и был введен). Данная статья также посвящена верхним оценкам для задачи выполнимости формул в КНФ–(1,∞) и k–КНФ–(1,∞). Верхняя оценка $p(L)1.1939^K$ для задачи ВЫП-(1,∞) доказывается в §7, оценка $p(L)1.0644^L$ доказывается в §8, оценки $p(L)1.3566^N$ для задачи 3-ВЫП-$(1,\infty)$ и $p(L)\psi(k)^{N/3}$ для задачи k-ВЫП-$(1,\infty)$ доказываются в §9.

При сведéнии задачи выполнимости формулы в КНФ к задаче выполнимости формулы из какого-то подкласса КНФ размер формулы значительно возрастает. Поэтому неудивительно, что до сих пор неизвестны примеры, когда алгоритм для какого-либо подкласса, имеющий хорошую верхнюю оценку, мог быть применен для улучшения верхней оценки общей задачи ВЫП. В данной статье, однако, приводится пример такого применения: с помощью одного из приводимых алгоритмов можно выяснить выполнимость формулы в КНФ, у которой $L > 8.49N$, быстрее, чем это сделает наилучший известный на текущий момент алгоритм из [9] (1.0801^L). С другой стороны, возрастание размера формулы при сведéнии позволяет надеяться на наличие алгоритмов, которые осуществляли бы обратное сведéние задачи выполнимости для какого-нибудь подкласса КНФ к общей задаче ВЫП со значительным уменьшением размера формулы. До сих пор таких алгоритмов известно не было. В этой статье приводится пример подобного сведения для формул в КНФ–$(1,\infty)$–$(+/-)$; определение и значимость этого класса формул для задачи ВЫП-$(1,\infty)$ будут подробнее обсуждаться чуть ниже.

Все приводимые алгоритмы используют *метод расщепления*, берущий начало от процедуры Девиса-Патнема ([5]). Коротко, суть его заключается в том, что формула F выполнима тогда и только тогда, когда выполнима хотя бы одна из формул $F[a]$ и $F[\overline{a}]$. В более широком смысле, метод расщепления заключается в том, что алгоритм строит некоторое дерево, сводя выполнимость формулы к выполнимости нескольких формул, полученных из нее в результате присваиваний. При этом над формулами производятся также некоторые не изменяющие выполнимости формулы *полиномиальные преобразования*, которые занимают в каждом случае полиномиальное время (удаление чистых литералов, 1-дизъюнкций и другие). Процедура, выполняющая эти преобразования, является общей для всех алгоритмов этой статьи и приводится в §4 (функция $REDUCE$). Общий метод для оценки размера таких деревьев (и тем самым времени работы алгоритма) предложили О. Кулльманн и Г. Люкхардт в [9]. Каждой вершине такого дерева приписывается некоторый вектор положительных чисел и связанный с ним многочлен. Число листьев в дереве оценивается через наибольший из положительных корней многочленов, сопоставленных вершинам дерева. Этот метод, а также его обобщение на случай, когда листьями дерева являются формулы, обрабатываемые каким-либо другим алгоритмом, излагается в §3.

Основным методом, применяемым в этой статье для решения задачи ВЫП-(1,∞), является сведéние этой задачи к задаче выполнимости формул в КНФ–(1,∞), не содержащих дизъюнкций, в которые входят одновременно и положительные, и отрицательные литералы (формулы в КНФ–(1,∞)–(+/–)). Основная идея заключается в том, что задача выполнимости формул, которые относительно "трудно" разобрать обычным методом расщепления, оказывается легко сводима к задаче выполнимости для формул указанного вида. Этот вид является весьма специальным и формулы такого вида поддаются более быстрому расщеплению. Класс формул, не содержащих дизъюнкций, в которые входят одновременно и положительные, и отрицательные литералы, хорош еще и тем, что к задаче выполнимости именно таких формул сводится ВЫП при сведéнии ВЫП$\to$ВЫП-(1,∞). Поэтому оказывается проще организовать эффективное обратное сведéние.

Расщепления, в результате которых исчезают дизъюнкции, содержащие литералы, встречающиеся в формуле только один раз (1-литералы), являются очень "выгодными", так как после этого можно удалить дополнительно все дизъюнкции, содержащие литерал, противоположный удаленному 1-литералу. Общеизвестно, что невыполнимая формула должна содержать хотя бы одну дизъюнкцию, состоящую только из положительных литералов. В случае КНФ–(1,∞) этим гарантируется наличие дизъюнкции, состоящей только из 1-литералов, эта идея является ключевой в алгоритме Г. Люкхардта из [10], решающем задачу ВЫП-(1,∞) за время $p(L)1.4423^N$. Можно заметить дополнительно, что если переменная, по которой делается расщепление, встречается еще в какой-нибудь дизъюнкции вместе с 1-литералом, это делает расщепление еще более "выгодным". Оказывается, что наличие такой дизъюнкции можно гарантировать во всех случаях кроме тех, когда вопрос о выполнимости формулы может быть сведен за полиномиальное время к вопросу о выполнимости формулы класса КНФ–(1,∞)–(+/–). Несколько обобщив эту идею, можно ее сформулировать в качестве следующего принципа.

Принцип разделения знаков. Если для каждой переменной v формулы F ровно один из литералов v и $\overline{v}$ обладает свойством P, то существует ровно две альтернативы:

1) в формуле F имеются дизъюнкции C и D и переменная v такие, что литерал v входит в C, литерал $\overline{v}$ входит в D и хотя бы один из литералов дизъюнкции D и все литералы дизъюнкции C обладают свойством P;
2) выполнимость формулы F эквивалентна выполнимости ее части, состоящей из дизъюнкций, не включающих в себя одновременно литералов, обладающих свойством P и не обладающих им.

Принцип разделения знаков является частным случаем расширенного принципа знака, который был введен в [6].

Принцип черных и белых литералов. Если для каждой переменной v формулы F не более одного из литералов v и $\overline{v}$ обладает свойством P, то существует ровно две альтернативы:

1) в формуле F имеются дизъюнкция, содержащая хотя бы один литерал, обладающий свойством P, и не содержащая литералов, противоположных литералам, обладающим свойством P;
2) если исключить из формулы F все дизъюнкции, содержащие литералы, противоположные литералам, обладающим свойством P, то ее выполнимость от этого не изменится.

Этот принцип представляет собой развитие обобщенного принципа знака, предложенного в [9]. Его справедливость вытекает из того, что если в любом выполняющем наборе для формулы F, не удовлетворяющей 1), значения литералов, обладающих свойством P, заменить на $False$, этот набор останется выполняющим.

Для одного и того же класса формул КНФ–(1,∞) в этой статье приводится несколько алгоритмов (получаемых подстановкой в основной Алгоритм 5.1 различных процедур, выясняющих выполнимость формулы в КНФ-$(1,\infty)-(+/-)$). Чтобы решить задачу ВЫП-(1,∞) за наименьшее время, следует запустить их параллельно. С другой стороны ясно, что гарантированная верхняя оценка сложности алгоритма, время работы которого оценивается как функция L, меньше таких оценок для других алгоритмов в случае, когда входная формула достаточна короткая. Например, алгоритм, решающий задачу ВЫП-(1,∞) за время $p(L)1.0644^L \approx p(L)2^{0.08992L}$ лучше алгоритма, решающего задачу ВЫП-(1,∞) за время $p(L)1.4423^N \approx p(L)2^{0.5284N}$ в случае, когда $L < 0.5284/0.08992 \approx 5.88N$. То же можно сказать и о других параметрах входной формулы. Соответственно, алгоритм, решающий задачу ВЫП-(1,∞) за время $p(L)1.1939^K$, приведен в §7 (Алгоритм для формул с малым числом дизъюнкций), алгоритм, решающий задачу ВЫП-(1,∞) за время $p(L)1.0644^L$, приведен в §8 (Алгоритм для коротких формул), алгоритм, решающий задачу 3-ВЫП-(1,∞) за время $p(L)1.3566^N$, приведен в §9 (Алгоритм для формул с малым числом переменных).

Статья организована следующим образом:

§1. Введение.
§2. Основные определения.
§3. Оценка количества листьев в произвольном дереве расщепления.
§4. Полиномиальные преобразования.
§5. Приведение формул класса КНФ-(1,∞) к КНФ-(1,∞)-(+/−).
§6. Оценка числа расщепления для алгоритма приведения формул класса КНФ–(1,∞) к КНФ–(1,∞)–(+/−).
§7. Алгоритм для формул с малым числом дизъюнкций.
§8. Алгоритм для коротких формул.
§9. Алгоритм для формул с малым числом переменных.
§10. Следствие для общей задачи ВЫП.

§2. Основные определения.

Пусть V – множество булевых переменных. Отрицание переменной v обозначается $\overline{v}$. Введем для любого множества U обозначение $\overline{U} = \{\overline{u} \mid u \in U\}$. *Литералами* называются элементы множества $W = V \cup \overline{V}$. *Положительными* будем называть литералы, являющиеся переменными. *Отрицательными* будем называть литералы, являющиеся отрицаниями переменных. Если w обозначает отрицательный литерал $\overline{v}$, то $\overline{w}$ обозначает переменную v. Дизъюнкцию $w_1 \vee w_2 \vee \cdots \vee w_s$ будем отождествлять с множеством литералов $\{w_1, w_2, \ldots, w_s\} \subset W$. Никакая дизъюнкция не содержит какой-либо переменной вместе с отрицанием, т.е. $w_i \neq \overline{w}_j$ при $i \neq j$. Пустая дизъюнкция отождествляется с $False$. *Положительными* будем называть дизъюнкции, содержащие только положительные литералы. *Отрицательными* будем называть дизъюнкции содержащие только отрицательные литералы. Под *формулой в КНФ* (или *КНФ-формулой*) будем понимать конечное множество дизъюнкций, интерпретируемое как их конъюнкция. Пустая формула отождествляется с $True$. Под *длиной дизъюнкции* будем понимать ее мощность как множества, под *длиной формулы* – сумму мощностей всех входящих в нее дизъюнкций. Дизъюнкцию длины ровно k будем называть *k-дизъюнкцией*. Если мы говорим, что *переменная* $v \in V$ входит в дизъюнкцию или в формулу, это означает, что туда входит литерал v либо литерал $\overline{v}$. Напротив, если речь идет о вхождении *литерала* v, имеется в виду только этот литерал, а не противоположный ему. Далее на протяжении всей статьи F – формула в КНФ с переменными из V.

КНФ–(1,s) – класс формул в КНФ, определяемый условием: формула F принадлежит этому классу, если каждая переменная $v \in V$ встречается в F не более одного раза как v и не более s раз как $\overline{v}$. (Здесь $s \in \mathbb{N} \cup \{+\infty\}$.)

k–КНФ–(1,s) – класс формул в КНФ, определяемый условием: формула F принадлежит этому классу, если она является формулой класса КНФ–$(1,s)$ и каждая ее дизъюнкция содержит не более k литералов. (Здесь $k, s \in \mathbb{N} \cup \{+\infty\}$. При $k = +\infty$ класс k–КНФ–$(1,s)$ совпадает с классом КНФ–$(1,s)$.)

КНФ–(1,s)–(+/–) – класс формул в КНФ, определяемый условием: формула F принадлежит этому классу, если она является формулой класса КНФ–$(1,s)$ и содержит только положительные и отрицательные дизъюнкции.

k–КНФ–(1,s)–(+/–) – класс формул в КНФ, определяемый условием: формула F принадлежит этому классу, если она является формулой класса k–КНФ–$(1,s)$ и содержит только положительные и отрицательные дизъюнкции.

Переменную v будем называть *(i,j)-переменной*, если литерал v входит в формулу F ровно i раз, а литерал $\overline{v}$ – ровно j раз. Переменную v будем называть *(i,j+)-переменной*, если литерал v входит в формулу F ровно i раз, а литерал $\overline{v}$ – не менее j раз.

Пусть имеются литерал l и дизъюнкции C_1 и C_2 такие, что

$l \in C_1$, $\overline{l} \in C_2$ и никакой другой литерал $l' \in C_1$ не удовлетворяет $\overline{l}' \in C_2$. В этом случае дизъюнкция $C_1 \cup C_2 \setminus \{l, \overline{l}\}$ называется *l-резольвентой* дизъюнкций C_1 и C_2.

Пусть $l \in W$. Формула $DP_l(F)$ образована путем

- добавления в F всех l-резольвент и
- удаления из F всех дизъюнкций, содержащих l или $\overline{l}$.

Набором будем называть произвольное конечное подмножество W, не содержащее одновременно никакую переменную вместе с ее отрицанием.

Определим формулу $F[S]$, полученную из КНФ-формулы F с помощью набора $S = \{v_1, \ldots, v_s\}$ как формулу, получаемую из F удалением всех дизъюнкций, содержащих литералы v_i, и удалением литералов $\overline{v_i}$ из оставшихся дизъюнкций. Для краткости будем писать $F[v_1, \ldots, v_s]$ вместо $F[\{v_1, \ldots, v_s\}]$.

Под *деревом расщепления* будем понимать дерево, вершины которого помечены КНФ-формулами таким образом, что если некоторая вершина помечена КНФ-формулой F, то ее сыновья помечены формулами $F[I_1]$, $F[I_2]$, ... , $F[I_s]$ для некоторых наборов I_1, I_2, ... , I_s.

Пример дерева расщепления приведен на Рис. 1.

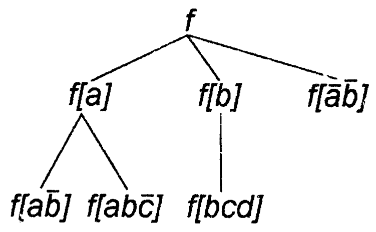


Рис. 1.

При оценке количества листьев в таком дереве (и, следовательно, сложности алгоритма), будут использованы вспомогательные результаты, касающиеся *векторов расщепления*, введенных в [9]. Эти результаты представлены в §3. Ниже вводятся некоторые необходимые для этого понятия.

Пусть имеется дерево расщепления T, вершины которого помечены объектами, обладающими некоторой характеристикой их сложности: каждому объекту F сопоставлено некоторое неотрицательное число $\mu(F)$ (*сложность* F, или *мера сложности* F), причем каждая вершина этого дерева обладает свойством, что сложность объекта, которым она помечена, больше сложности каждого из объектов, которыми помечены ее сыновья. В нашем случае объектами будут формулы, а в качестве μ мы будем выбирать одну из трех мер

1) объекты – КНФ-формулы, $\mu_1(F)$ – число дизъюнкций в формуле F;
2) объекты – КНФ-формулы, $\mu_2(F)$ – длина формулы F;
3) объекты – КНФ-формулы, $\mu_3(F)$ – число переменных в формуле F.

Эти меры сложности в дальнейшем будут обозначаться $\mathfrak{K}(F)$, $\mathfrak{L}(F)$, $\mathfrak{N}(F)$ соответственно. Для простоты будем предполагать, что каждая внутренняя вершина нашего дерева расщепления имеет степень не менее двух (деревья, не удовлетворяющие этому условию, нам не понадобятся).

Понятия вектора расщепления и числа расщепления, определяемые ниже, предполагают выбор некоторой меры сложности. Поэтому когда будет неясно заранее, какая мера сложности имеется в виду, эти понятия будут использоваться со словами "...относительно меры сложности $\mathfrak{K}$ ($\mathfrak{L}$, $\mathfrak{N}$)". Тем не менее, при доказательстве общих результатов о числах расщепления мы не будем фиксировать конкретную меру сложности. Для этих целей не важно даже, что объектами являются формулы.

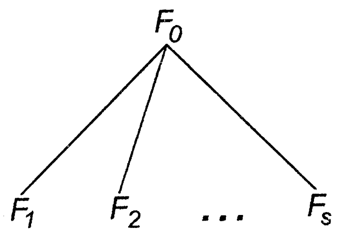


Рис. 2.

Рассмотрим произвольную внутреннюю вершину нашего дерева, помеченную объектом F_0. Пусть ее сыновья помечены объектами $F_1, F_2, \ldots, F_s$ (см. Рис. 2). *Вектором расщепления в вершине* называется вектор из s чисел $(t_1, \dots, t_s)$, где $s \geqslant 2$ и каждое из t_i – положительное число, не превосходящее $\mu(F_0) - \mu(F_i)$.

Числом расщепления $\tau(\vec{t})$ *в вершине* с вектором расщепления $\vec{t} = (t_1, \dots, t_s)$ называется единственный положительный корень *характеристического многочлена вектора расщепления*

$$\mathfrak{h}_{\vec{t}}(x) = 1 - \sum_{i=1}^{s} x^{-t_i}. \qquad (\star)$$

Положим $\tau(\emptyset) = 1$ в случае, когда вершина является листом.

Замечание. Характеристический многочлен $\mathfrak{h}_{\vec{t}}(x)$ является монотонной функцией x на интервале $(0, +\infty)$. Кроме того, $\mathfrak{h}_{\vec{t}}(0) = -\infty$, $\mathfrak{h}_{\vec{t}}(+\infty) = +1$. Следовательно, уравнение $\mathfrak{h}_{\vec{t}}(x) = 0$ имеет единственное решение на интервале $(0, +\infty)$.

Наибольшее из чисел расщепления $\tau(\vec{t})$ во всех вершинах дерева T называется *числом расщепления дерева* T. Обозначим его $\tau_{max,T}$ (или просто τ_{max}).

Ниже приводятся простые свойства чисел расщепления. В дальнейшем при работе с числами расщепления мы будем часто использовать их и не будем каждый раз явно на них ссылаться.

Свойства чисел расщепления.

1) *Числа расщепления строго больше единицы.*
2) *От перестановки компонент вектора расщепления число расщепления не изменяется.*
3) *Если вектор расщепления* $\vec{t}_1$ *покоординатно не превосходит другого вектора* $\vec{t}_2$ *той же размерности, то и число расщепления, соответствующее вектору* t_1*, не превосходит числа расщепления вектора* t_2*.*
4) *Пусть* a, b, c, d *– натуральные числа. Если* $|a-b| < |c-d|$ *и* $a+b = c+d$*, то* $\tau((a,b)) < \tau((c,d))$*.*

§3. Оценка количества листьев в произвольном дереве расщепления.

Лемма 3.1 (О. Кулльманн, Г. Люкхардт, [9]). *Количество листьев в дереве расщепления с объектом* F_0 *в корне не превосходит* $(\tau_{max})^{\mu(F_0)}$*.*

Доказательство Леммы 3.1 приводится в [9]. В дальнейшем нам понадобится также следующий обобщенный вариант этой леммы (Лемма 3.1 является следствием этого обобщения).

Лемма 3.2. *Пусть* $f(m) = \alpha\lambda^m$, *где* $\lambda > 1$, $\alpha \geqslant 0$. *Пусть* G_l *обозначает объект, которым помечен лист* l *дерева* T, F_0 *обозначает объект, которым помечен корень дерева* T. *Тогда*

$$\sum_{l-\text{лист } T} f(\mu(G_l)) \leqslant \alpha(\max(\tau_{max}, \lambda))^{\mu(F_0)}.$$

Доказательство – индукция по построению дерева.

База. Дерево, состоящее из единственной вершины. Для него $\tau_{max} = 1$, $\alpha(\max(\lambda, \tau_{max}))^{\mu(F_0)} = \alpha\lambda^{\mu(F_0)} = f(\mu(F_0))$.

Переход. Рассмотрим дерево T, представленное на Рис. 3.

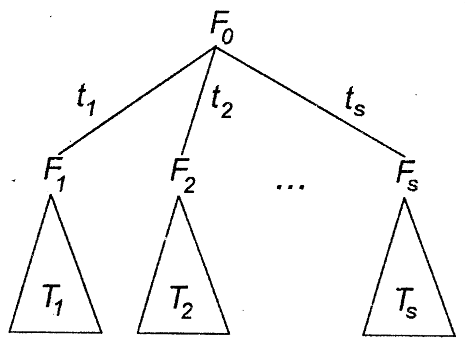


Рис. 3.

Пусть $\vec{t} = (t_1, t_2, \ldots, t_s)$ – вектор расщепления в его корне.

$$\sum_{l-\text{лист } T} f(\mu(G_l)) =$$

$$= \sum_{j=1}^{s} \left(\sum_{l-\text{лист } T_j} f(\mu(G_l)) \right)$$

$$\leqslant \alpha \sum_{j=1}^{s} (\max(\lambda, \tau_{max,T_j}))^{\mu(F_j)} \quad \text{(по предположению индукции)}$$

$$\leqslant \alpha \sum_{j=1}^{s} (\max(\lambda, \tau_{max,T_j}))^{\mu(F_0)-t_j} \quad \text{(по определению вектора } \vec{t}\text{)}$$

$$\leqslant \alpha \sum_{j=1}^{s} (\max(\lambda, \tau_{max,T}))^{\mu(F_0)-t_j} \quad \text{(так как } \tau_{max,T_j} \leqslant \tau_{max,T}\text{)}$$

$$= \alpha (\max(\lambda, \tau_{max,T}))^{\mu(F_0)} \cdot \sum_{j=1}^{s} (\max(\lambda, \tau_{max,T}))^{-t_j}$$

$$= \alpha (\max(\lambda, \tau_{max,T}))^{\mu(F_0)} (1 - \mathfrak{h}_{\vec{t}}(\max(\lambda, \tau_{max,T})))$$

$$\text{(по определению } \mathfrak{h}_{\vec{t}}\text{)}$$

$$\leqslant \alpha (\max(\lambda, \tau_{max,T}))^{\mu(F_0)} \qquad \text{(по монотонности } \mathfrak{h}_{\vec{t}}\text{)}$$

□

§4. Полиномиальные преобразования.

Все алгоритмы, приводимые в этой статье, после расщепления формулы производят над ней полиномиальные преобразования с помощью функции $REDUCE$, которая приводится в этом параграфе. В результате из формулы исключаются чистые литералы (литералы, отрицания которых не входят в формулу), 1-дизъюнкции и т. д. В том числе, на шаге (R4) из формулы удаляются некоторые заблокированные дизъюнкции. Понятие заблокированной дизъюнкции было введено Кулльманном в [7]. Заблокированной дизъюнкцией называется дизъюнкция D, отрицание одного из литералов которой не встречается в дизъюнкциях, не содержащих отрицания другого литерала дизъюнкции D. В нашем случае будут удаляться заблокированные дизъюнкции специального вида, а именно, удаляется каждая дизъюнкция D такая, что в формуле F есть дизъюнкция C и переменные a и b, удовлетворяющие условиям

- $\{a, b\} \subset C$,
- $\{\overline{a}, \overline{b}\} \subset D$.

По аналогии с заблокированной, удаляется и каждая дизъюнкция E, такая, что в формуле F есть дизъюнкции C и D и переменные

a, b и c, удовлетворяющие условиям

- $\{a, b\} \subset C$,
- $\{\overline{a}, c\} \subset D$,
- $\{\overline{b}, \overline{c}\} \subset E$.

Как будет показано ниже (Лемма 4.1), все эти преобразования не изменяют выполнимости формулы. В результате этих преобразований "простые" участки формулы удаляются, а относительно оставшейся части формулы можно сказать, что каждая переменная входит в нее хотя бы три раза, каждая дизъюнкция содержит хотя бы два литерала и т. д., т.е. формула не является "особенной".

Функция $REDUCE$**.**

Вход:

- число $k \in \mathbb{N} \cup \{+\infty\}$,
- формула F, принадлежащая классу k–КНФ–(1,∞).

Выход: формула, принадлежащая классу k–КНФ–(1,∞), эквивалентная исходной формуле F.

Метод:

(R1) Произвести удаление 1-дизъюнкций: если в F есть дизъюнкция, содержащая единственный литерал l, заменить F на $F[l]$. Повторять этот шаг, пока в F есть такие дизъюнкции.

(R2) Произвести удаление чистых литералов: если литерал $\overline{l}$ встречается, а l не встречается в F, заменить F на $F[\overline{l}]$. Повторять этот шаг, пока в F есть такие литералы.

(R3) Если имеется переменная $d \in V$ такая, что
- формула $DP_d(F)$ принадлежит классу k–КНФ–(1,∞),
- $DP_d(F)$ содержит дизъюнкций не больше, чем F,
- длина $DP_d(F)$ не превосходит длины F,

положить $F = DP_d(F)$. Если имеется несколько таких переменных, выбрать в качестве d входящую в F наименьшее число раз.

(R4) Произвести удаление заблокированных дизъюнкций: если в F есть дизъюнкции C и D и переменные a и b такие, что $\{a, b\} \subset C$ и $\{\overline{a}, \overline{b}\} \subset D$, то удалить из F дизъюнкцию D.

(R5) Если в F есть дизъюнкции C, D и E и переменные a, b и c такие, что $\{a, b\} \subset C$, $\{\overline{a}, c\} \subset D$ и $\{\overline{b}, \overline{c}\} \subset E$, то удалить из F дизъюнкцию E.

(R6) Если имеются дизъюнкции $C, D \in F$ и литерал $l \in C \cap \overline{D}$ такие, что $C \backslash \{l\} \subset D \backslash \{\bar{l}\}$, положить $F = (F \backslash \{D\}) \cup (\{C \backslash \{l\}\})$.

(R7) Если на шагах (R1)–(R6) формула F была изменена – перейти снова к шагу (R1). В противном случае выдать F.

$\square$

Лемма 4.1. *Функция REDUCE выдает по формуле F в k–КНФ–$(1,\infty)$ эквивалентную ей в смысле выполнимости формулу в k–КНФ–$(1,\infty)$ и вычисляет ответ за время, полиномиальное от длины входной формулы.*

Доказательство. Шаги (R1)–(R6) повторяются не более $L + N$ раз, так как ни один из них не увеличивает ни числа переменных в формуле, ни ее длины, и при каждой итерации хотя бы на одном из них одна из этих величин уменьшается. Поскольку каждый из них отнимает полиномиальное время, то же можно сказать и о всей функции $REDUCE$. То, что полученная формула принадлежит классу k–КНФ–$(1,\infty)$, очевидно.

Докажем, что на каждом из шагов выполнимость формулы F не изменяется.

(R1) Удаление 1-дизъюнкций: если дизъюнкция содержит только один литерал, то его содержит и любой выполняющий набор.

(R2) Удаление чистых литералов: если литерал l не встречается в F, то в любом наборе, выполняющем F, можно заменить l на $\bar{l}$, получив снова выполняющий набор.

(R3) Покажем, что формулы F и $DP_d(F)$ эквивалентны. Рассмотрим какой-нибудь набор, выполняющий F. Поскольку из каждых двух дизъюнкций, резольвента которых была добавлена, по крайней мере одна выполняется даже если из нее исключить вхождение переменной d, выполняется и резольвента. Обратно, рассмотрим какой-нибудь набор I, выполняющий $DP_d(F)$. Покажем, что к нему можно добавить d либо $\overline{d}$ так, чтобы он стал выполняющим для F. Допустим, в F имеется две дизъюнкции D_1 и D_2, которые I не выполняет, такие что D_1 содержит литерал d, а D содержит литерал $\overline{d}$. Но тогда литерал l не выполняет и резольвенту D_1 и D_2, которая содержится в $DP_d(F)$, что противоречит тому, что он выполняет $DP_d(F)$.

(R4) Удаление заблокированных дизъюнкций: предположим, что формула F была невыполнима, а стала выполнима. Рассмотрим выполняющий набор. Он обязательно должен не выполнять дизъюнкцию D, т.е. в нем содержатся литералы a и b. Заменим в нем a на $\overline{a}$. Дизъюнкции C и D теперь выполняются, а выполнимость других дизъюнкций не изменилась (поскольку F является формулой в КНФ–$(1,\infty)$, литерал a входит только в дизъюнкцию C). Мы получили выполняющий набор для невыполнимой формулы. Противоречие. Обратное очевидно.

(R5) Предположим, что формула F была невыполнима, а стала выполнима. Рассмотрим выполняющий набор. Он обязательно должен не выполнять дизъюнкцию E, т.е. в нем содержатся литералы b и c. Если он содержит также и литерал a, заменим в нем b на $\overline{b}$, в противном случае заменим c на $\overline{c}$. Дизъюнкции C, D и E теперь выполняются, а выполнимость других дизъюнкций не изменилась (литералы a, b и c не входят ни в какие дизъюнкции формулы F, кроме C, D и E). Мы получили выполняющий набор для невыполнимой формулы. Противоречие. Обратное очевидно.

(R6) Пусть F_1 – формула F до применения этого шага, $F_2 = (F_1 \backslash \{D\}) \cup (\{C \backslash \{l\}\})$. Заметим, что формула $DP_l(F_1)$ эквивалентна в смысле выполнимости формуле $DP_l(F_2)$. Следовательно, F_1 и F_2 также эквивалентны.

□

Ниже приводятся некоторые полезные свойства формулы $REDUCE(k,F)$.

Лемма 4.2.

1) *Любая переменная v в $REDUCE(k,F)$ встречается по крайней мере один раз как литерал v и по крайней мере два раза как литерал $\overline{v}$.*
2) *Формула $REDUCE(k,F)$ не содержит таких дизъюнкций C и D, что $\exists a_1, a_2 \in V\ ((\{a_1, a_2\} \subset C) \,\&\, (\{\overline{a}_1, \overline{a}_2\} \subset D))$.*
3) *Формула $REDUCE(k,F)$ не содержит таких дизъюнкций C, D и E, что $\exists a_1, a_2, b \in V\ ((\{a_1, a_2\} \subset C) \,\&\, (\{\overline{a}_1, b\} \subset D) \,\&\, (\{\overline{a}_2, \overline{b}\} \subset E))$.*

4) *Если в формуле $REDUCE(k,F)$ имеется положительная дизъюнкция $C = \{a_1, a_2, \dots, a_s\}$, то в ней имеется и по две дизъюнкции, содержащие $\overline{a}_i$, для каждого $i \in \overline{1,s}$, причем все эти дизъюнкции различны.*
5) *Если в формуле $REDUCE(k,F)$ имеется положительная дизъюнкция $C = \{a_1, a_2, \dots, a_s\}$ и дизъюнкция D, содержащая литерал $\overline{a}_1$ и хотя бы один положительный литерал a_{s+1}, то формула $REDUCE(k,F)$ содержит по крайней мере две дизъюнкции, содержащие $\overline{a}_i$, для каждого $i \in \overline{2, s+1}$, причем все эти дизъюнкции различны.*

Доказательство.

1) Все остальные переменные удаляются на шагах (R2)–(R3) функции $REDUCE$.
2) Такие дизъюнкции удаляются на шаге (R4) функции $REDUCE$.
3) Такие дизъюнкции удаляются на шаге (R5) функции $REDUCE$.
4) Следует из пунктов 1) и 2).
5) Следует из пунктов 1), 2) и 3).

□

§5. Приведение формул класса КНФ–(1,∞) к КНФ–(1,∞)–(+/–).

Три основных алгоритма этой статьи очень похожи друг на друга и получаются из Алгоритма 5.1, описанного в этом параграфе. Суть метода заключается в следующем. Если во входной формуле в КНФ–(1,∞) имеются дизъюнкция $C = \{a_1, a_2, \dots, a_s\}$ и дизъюнкция D, удовлетворяющие условиям

- C – положительная,
- D содержит литерал $\overline{a}_1$,
- D содержит хотя бы один положительный литерал,

то расщепление по переменной a_1 оказывается очень выгодным, так как при обоих присваиваниях $\{a_1\}$ и $\{\overline{a}_1\}$ исчезает единственное вхождение по крайней мере одного литерала. Если же такой пары дизъюнкций во входной формуле нет, то, согласно принципу разделения знаков, в формуле можно оставить только положительные и отрицательные дизъюнкции, не изменяя ее выполнимости. Вопрос о выполнимости полученной формулы в

КНФ–(1,∞)–(+/–) решается уже разными алгоритмами, вызываемыми Алгоритмом 5.1. Чтобы получить алгоритм, хорошо работающий для формул с малым числом дизъюнкций, мы обрабатываем формулы в КНФ–(1,∞)–(+/–) Алгоритмом 7.1; для коротких формул мы используем Алгоритм 8.1; для формул с малым числом переменных – Алгоритм 9.1. Некоторые из этих алгоритмов работают самостоятельно; в процессе работы других при расщеплении формулы в КНФ–(1,∞)–(+/–) вновь образуются дизъюнкции, содержащие как положительные, так и отрицательные литералы, и для их обработки вновь вызывается функция $SPLIT$, являющаяся основной частью Алгоритма 5.1.

Функция $SPLIT$**.**

Вход:

- число $k \in \mathbb{N} \cup \{+\infty\}$,
- формула F, принадлежащая классу k–КНФ–(1,∞), обработанная функцией $REDUCE$,
- алгоритм $\mathcal{M}$, решающий задачу выполнимости для формул класса k–КНФ–(1,∞)–(+/–).

Выход: “Выполнима”, если F выполнима; в противном случае “Невыполнима”.

Метод:

(S1) Если в F нет ни одной положительной дизъюнкции, положить $F = True$.

(S2) Если для каждой положительной дизъюнкции C формулы F и каждой дизъюнкции D формулы F, содержащей положительный литерал, выполняется $\overline{C} \cap D = \emptyset$, то положить $G = F[\{\overline{v} \in \overline{V} \mid \exists C \in F\ \exists u \in V\ (\{\overline{v}, u\} \subset C)\}]$ и выдать результат вызова $\mathcal{M}(k, G)$.

(S3) Выбрать в F дизъюнкцию $C = \{a_1, a_2, \dots, a_s\}$ и дизъюнкцию D, удовлетворяющие следующим условиям:
- C – положительная,
- D содержит литерал $\overline{a}_1$,
- в D входит хотя бы один положительный литерал b.

Построить формулы

$$F_1 = REDUCE(k, F[a_1]),$$
$$F_2 = REDUCE(k, F[\overline{a}_1]).$$

Если функция $SPLIT$ выдает ответ "Выполнима" хотя бы для одной из них, выдать ответ "Выполнима"; иначе "Невыполнима".

□

Лемма 5.1. *Функция $SPLIT$ правильно решает задачу о выполнимости входной формулы F в k–КНФ–$(1,\infty)$, обработанной функцией $REDUCE$, и выдает ответ за конечное время.*

Доказательство. Функция $REDUCE$ не изменяет выполнимости формулы по Лемме 4.1. Если формула не содержит положительных дизъюнкций, то она выполнима набором $\{\overline{v} \mid v \in V\}$, состоящим только из отрицательных литералов. Так что шаг (S1) также корректен. Заметим, что формула, удовлетворяющая условию этого шага, превращается в $True$ и немедленно передается на шаге (S2) алгоритму $\mathcal{M}$.

Пусть P_+ – множество всех положительных дизъюнкций формулы F, P_- – множество всех ее отрицательных дизъюнкций, $M = F \backslash (P_+ \cup P_-)$. Если выполняется условие, проверяемое на шаге (S2), то среди переменных, отрицания которых входят в M, нет ни одной из переменных, входящих в P_+. Поэтому, если некоторый набор I выполняет F, то и набор I', полученный из I заменой всех отрицательных литералов, входящих в M, на соответствующие им положительные, также выполняет F. Действительно, поскольку эти переменные без отрицания не входят ни в P_+, ни в P_-, ложное значение в результате такой замены могут получить лишь дизъюнкции из M. Но каждая из этих дизъюнкций содержит отрицательный литерал, который мы только что включили в набор I'. Значит, если F выполнима, то и $G = F[\{\overline{v} \in \overline{V} \mid \exists C \in F\ \exists u \in V\ (\{\overline{v}, u\} \subset C)\}] = F[\{\overline{v} \in \overline{V} \mid \exists C \in M (\overline{v} \in C))\}]$ выполнима. Обратное тривиально. К тому же, G является формулой класса k–КНФ–$(1,\infty)$–$(+/-)$. Следовательно, шаг (S2) тоже корректен.

Если не выполнилось условие шага (S2), то условие шага (S3) обязано выполниться. На шаге (S3) выполнимость F сводится к выполнимости формул

$$F_1 = REDUCE(k, F[a_1]),$$
$$F_2 = REDUCE(k, F[\overline{a}_1]),$$

каждая из которых имеет меньшее число переменных, чем F. Поскольку ни один из шагов этого алгоритма не увеличивает числа

переменных в формуле F, эта рекурсия должна завершиться за конечное время.
□

Алгоритм 5.1

Вход:

- число $k \in \mathbb{N} \cup \{+\infty\}$,
- формула F, принадлежащая классу k–КНФ–$(1,\infty)$,
- алгоритм $\mathcal{M}$, решающий задачу выполнимости для формул класса k–КНФ–$(1,\infty)$–$(+/-)$.

Выход: "Выполнима", если F выполнима; в противном случае "Невыполнима".

Метод:

(M1) Положить $F = REDUCE(k, F)$.
(M2) Выдать результат вызова $SPLIT(k, F, \mathcal{M})$.

□

Корректность Алгоритма 5.1 следует из Леммы 4.1 и Леммы 5.1. □

Время работы Алгоритма 5.1

Построим естественным образом дерево $T_{5.1}$ формул, расщепляемых этим алгоритмом. Внутренние вершины этого дерева будут помечены формулами, расщепляемыми на шаге (S3), а листьями будут формулы, которые Алгоритм 5.1 обрабатывает не расщепляя, а передавая их алгоритму $\mathcal{M}$.

Чтобы получить из этого алгоритма алгоритм, который реально решает задачу выполнимости, нужно предъявить алгоритм $\mathcal{M}$, решающий задачу выполнимости для формул класса k–КНФ–$(1,\infty)$–$(+/-)$. Такие алгоритмы $\mathcal{M}_K$, $\mathcal{M}_L$, $\mathcal{M}_N$, будут предъявлены в §§7–9. Чтобы минимизировать время работы Алгоритма 5.1, нужно выбирать алгоритм $\mathcal{M}$ в зависимости от вида исходной формулы. Соответственно мы получим верхние оценки по разных параметрам: при применении в качестве $\mathcal{M}$ алгоритма $\mathcal{M}_K$, алгоритм $\mathcal{M}$ будет работать $p(L)1.1939^K$ для формул из КНФ–$(1,\infty)$; при применении алгоритма $\mathcal{M}_L$, алгоритм $\mathcal{M}$ будет работать $p(L)1.0644^L$ для формул из КНФ–$(1,\infty)$, при применении алгоритма $\mathcal{M}_N$, алгоритм $\mathcal{M}$ будет работать $p(L)1.3566^N$ для формул из КНФ–(1,∞). Более точные формулировки будут даны в §§7–9.

Доказательство каждой из этих оценок также будет разбито на два этапа. На первом (Леммы 6.1–6.3) будет доказана верхняя оценка для числа расщепления для дерева $T_{5.1}$, т.е. в некотором смысле верхняя оценка времени работы Алгоритма 5.1. На втором этапе после приведения алгоритмов $\mathcal{M}_K$, $\mathcal{M}_L$, $\mathcal{M}_N$ будет показано, что каждый из них обрабатывает формулы в 3–КНФ–(1,∞)–(+/–) относительно быстро. С помощью Леммы 3.2 эти оценки будут объединены в Теоремах 7.1–9.1 для доказательства общей оценки времени работы всего Алгоритма 5.1 при использовании различных алгоритмов $\mathcal{M}$.

□

§6. Оценка числа расщепления для алгоритма приведения формул класса КНФ–(1,∞) к КНФ–(1,∞)–(+/–).

При оценке общего времени работы Алгоритма 5.1 нам понадобятся оценки числа расщепления дерева, которое он строит, т.е. дерева $T_{5.1}$, определенного в §5. Ниже приводятся такие оценки для различных мер сложности ($\mathfrak{K}$ – число дизъюнкций в формуле, $\mathfrak{L}$ – длина формулы, $\mathfrak{N}$ – число переменных в формуле). В этом параграфе C, D, c, a_1, a_2, …, a_s, b, F_1, F_2 обозначают те же объекты, что и в Алгоритме 5.1.

Замечание. В процессе оценки чисел расщепления мы будем часто использовать такие выражения как “переменная a будет удалена на шаге (R2)”. На самом деле, эта переменная может быть удалена и раньше, например, при удалении другой переменной на шаге (R1). Когда мы употребляем такие выражения, имеется в виду, что переменная будет удалена не позже этого шага; желаемое число переменных (дизъюнкций, вхождений) исчезнет из формулы в любом случае.

Лемма 6.1. *Число расщепления дерева $T_{5.1}$ относительно меры сложности $\mathfrak{K}$ не превосходит* 1.1939.

Доказательство. Рассмотрим два случая.

Случай 1. $s = 2$.

Заметим, что число дизъюнкций в формуле F_1 меньше числа дизъюнкций в формуле F по крайней мере на три. А именно, кроме дизъюнкции C, удаляемой сразу, в функции $REDUCE$ на шаге (R2) удаляются все дизъюнкции, содержащие $\overline{a}_2$ (так как после удаления дизъюнкции C из формулы F исчезает единственное вхождение литерала a_2). По Лемме 4.2 таких дизъюнкций по крайней мере две.

Число дизъюнкций в формуле F_2 меньше числа дизъюнкций в формуле F по крайней мере на пять. Во-первых, сразу удаляются дизъюнкции, содержащие $\overline{a}_1$. По Лемме 4.2 таких дизъюнкций по крайней мере две. Во-вторых, на шаге (R1) будет удалена дизъюнкция, получившаяся из дизъюнкции C после удаления a_1. Кроме того, поскольку единственное вхождение литерала b исчезает, на шаге (R2) будут удалены все дизъюнкции, содержащие $\overline{b}$. Поскольку их не менее двух по Лемме 4.2, а C и D таковыми не являются, получаем, что будет удалено не менее пяти дизъюнкций.

Итого, в этом случае вершине соответствует вектор расщепления $(3, 5)$ и соответственно число расщепления $\tau((3, 5)) < 1.1939$.

Случай 2. $s \geqslant 3$.

Заметим, что число дизъюнкций в формуле F_1 меньше числа дизъюнкций в формуле F по крайней мере на пять. А именно, кроме дизъюнкции C, удаляемой сразу, в функции $REDUCE$ на шаге (R2) удаляются все дизъюнкции, содержащие $\overline{a}_2$ и $\overline{a}_3$ (так как после удаления дизъюнкции C из формулы F исчезают единственные вхождения литералов a_2 и a_3). По Лемме 4.2 таких дизъюнкций по крайней мере четыре.

Число дизъюнкций в формуле F_2 меньше числа дизъюнкций в формуле F по крайней мере на три. Во-первых, сразу удаляются дизъюнкции, содержащие $\overline{a}_1$. По Лемме 4.2 таких дизъюнкций по крайней мере две. Во-вторых, поскольку единственное вхождение литерала b исчезает, на шаге (R2) будут удалены все дизъюнкции, содержащие $\overline{b}$. Поскольку их не менее двух по Лемме 4.2, а D таковой не является, получаем, что будет удалено не менее трех дизъюнкций.

Итого, в этом случае вершине также соответствует вектор расщепления $(3, 5)$ и соответственно число расщепления $\tau((3, 5)) < 1.1939$.

□

Лемма 6.2. *Число расщепления дерева* $T_{5.1}$ *относительно меры сложности* $\mathfrak{L}$ *не превосходит* 1.0644.

Доказательство.

Введем обозначения $\Delta_1 = \mathfrak{L}(F) - \mathfrak{L}(F_1)$, $\Delta_2 = \mathfrak{L}(F) - \mathfrak{L}(F_2)$, $\Delta = \Delta_1 + \Delta_2$. Рассмотрим следующие случаи.

Случай 1. Литерал $\overline{a}_1$ входит в формулу F не менее трех раз.

Случай 1.1. $s = 2$.

Во-первых, и в F_1, и в F_2 относительно F исчезают все вхождения переменных a_1 и a_2. Это происходит в результате удаления переменной a_1 и последующего преобразования на шаге (R1) функции $REDUCE$.

Поэтому $\Delta_1, \Delta_2 \geqslant 7$, $\Delta \geqslant 14$.

Каждая из дизъюнкций, содержащих $\overline{a}_1$ либо $\overline{a}_2$ (а таких дизъюнкций имеется не менее пяти по Лемме 4.2 и поскольку число вхождений $\overline{a}_1$ в формулу F больше двух), кроме самих уже учтенных литералов $\overline{a}_1$ и $\overline{a}_2$, дает в Δ вклад еще не менее двух. А именно, 2-дизъюнкции дают вклад по 1 в Δ_1 и Δ_2 из-за преобразований (R1) и (R2); 3-дизъюнкции дают вклад 2 в Δ, т.к. при исчезновении 3-дизъюнкции удаляются сразу два вхождения (не считая $\overline{a}_1$ либо $\overline{a}_2$). Итого, в Δ добавляется по крайней мере десять, а в каждое из Δ_1 и Δ_2 – не менее двух.

Из всех векторов расщепления вида (Δ_1, Δ_2) с суммой $\Delta_1 + \Delta_2 = \Delta \geqslant 24$ и $\Delta_1, \Delta_2 \geqslant 9$ наибольшее число расщепления соответствует вектору (9,15). Это число $\tau((9, 15)) < 1.0609$.

Случай 1.2. $s \geqslant 3$.

Во-первых, в F_1 относительно F исчезают все вхождения переменных a_1, a_2 и a_3. Это происходит в результате удаления переменной a_1 и последующих преобразований на шаге (R2) функции $REDUCE$ – всего не менее десяти вхождений. В F_2 относительно F исчезают все вхождения переменной a_1, т.е. по крайней мере четыре вхождения.

Поэтому $\Delta_1 \geqslant 10, \Delta_2 \geqslant 4$, $\Delta \geqslant 14$.

Дизъюнкции, содержащие $\overline{a}_2$ и $\overline{a}_3$, удаляются при переходе от F к F_1 на шаге (R2) функции $REDUCE$. По Лемме 4.2 их по крайней мере четыре. Каждая из них содержит по крайней мере один литерал не считая самих литералов $\overline{a}_2$ и $\overline{a}_3$. Это дает добавку не менее четырех в Δ_1 (и соответственно в Δ).

Каждая из дизъюнкций, содержащих $\overline{a}_1$ (а их не менее трех), кроме самого литерала $\overline{a}_1$, дает в Δ вклад еще не менее двух. А именно, 2-дизъюнкции дают вклад по 1 в Δ_1 и Δ_2 из-за преобразований (R1) и (R2); 3-дизъюнкции дают вклад 2 в Δ, т.к. при исчезновении 3-дизъюнкции удаляются сразу два вхождения (не считая $\overline{a}_1$). Итого, в Δ добавляется еще по крайней мере шесть, в том числе в Δ_2 – не менее трех.

Итого, $\Delta \geqslant 24$, $\Delta_1 \geqslant 14$, $\Delta_2 \geqslant 7$.

Из всех векторов расщепления вида (Δ_1, Δ_2) с суммой $\Delta_1 + \Delta_2 = \Delta \geqslant 24$ и $\Delta_1, \Delta_2 \geqslant 7$ наибольшее число расщепления соответствует вектору (7,17). Это число $\tau((7,17)) < 1.0637$.

Заметим, что в Случае 1 мы не использовали факт, что в одну из дизъюнкций, содержащих $\overline{a}_1$, входит положительный литерал b. Это нам понадобится, чтобы использовать эти рассуждения еще раз, при доказательстве Теоремы 8.1.

Случай 2. Литерал $\overline{a}_1$ входит в формулу F ровно два раза.
Случай 2.1. $s = 2$.

Во-первых, в F_1 относительно F исчезают все вхождения переменных a_1 и a_2 (всего не менее шести вхождений), а в F_2 – все вхождения переменных a_1, a_2 и b (всего не менее девяти вхождений). Это происходит в результате удаления переменной a_1 и последующих преобразований на шагах (R1) и (R2) функции $REDUCE$.

Поэтому $\Delta_1 \geqslant 6$, $\Delta_2 \geqslant 9$, $\Delta \geqslant 15$.

Каждая из дизъюнкций, содержащих $\overline{a}_2$ (а их не менее трех), кроме самого литерала $\overline{a}_2$, дает в Δ вклад еще не менее двух. А именно, 2-дизъюнкции дают вклад по 1 в Δ_1 и Δ_2 из-за преобразований (R1) и (R2); 3-дизъюнкции дают вклад 2 в Δ, т.к. при исчезновении 3-дизъюнкции удаляются сразу два вхождения (не считая $\overline{a}_2$). Итого, в Δ добавляется еще по крайней мере четыре, в том числе в Δ_2 – не менее двух.

Заметим, что среди дизъюнкций, в которые входит литерал $\overline{b}$, есть по крайней мере одна, не содержащая ни переменной a_1 (дизъюнкций, содержащих переменную a_1, всего три и одна из них положительна), ни переменной a_2 (по Лемме 4.2). В нее входит еще по крайней мере один литерал, и он будет удален вместе со всей дизъюнкцией на шаге (R2) при переходе от F к F_2.

Итак, к этому моменту мы имеем, что $\Delta_1 \geqslant 8$, $\Delta_2 \geqslant 10$, $\Delta \geqslant 20$.

Литералы, входящие в дизъюнкции, в которые входит литерал $\overline{a}_1$, также удаляются при переходе от F к F_2. По Лемме 4.2 в них мы уже подсчитали только литералы, соответствующие переменным a_1 и b.

Случай 2.1.1. Среди дизъюнкций, содержащих $\overline{a}_1$, есть 2- дизъюнкция.

Заметим, что в этом случае ни одна из дизъюнкций, содержащих $\overline{a}_1$, не может содержать $\overline{b}$ (иначе обе они были бы удалены ранее на шаге (R6)), а значит удаляется еще один дополнительный литерал (или два, если все-таки одна из них содержит более двух литералов). Кроме того, это означает, что кроме той дизъюнкции, содержащей $\overline{b}$, которую мы уже учли, надо учесть еще одну, а она содержит кроме $\overline{b}$ по крайней мере еще один литерал. Итак, если одна из этих дизъюнкций содержит более двух литералов, то $\Delta \geqslant 23$. В противном случае дополнительно можно учесть, что при переходе от F к F_1 дизъюнкция $\{\overline{a}_1, b\}$ становится 1-дизъюнкцией и, следовательно, удаляются все вхождения переменной b, т.е. в этом случае мы получаем, что $\Delta \geqslant 23$.

Случай 2.1.2. Каждая из дизъюнкций, содержащих $\overline{a}_1$, содержит по крайней мере три литерала.

В таком случае эти дизъюнкции добавляют к Δ еще по крайней мере два вхождения, т.е. $\Delta \geqslant 22$.

Случай 2.1.2.1. Ни одна из дизъюнкций, содержащих $\overline{a}_1$, не содержит $\overline{b}$, либо одна из этих дизъюнкций содержит по крайней мере четыре литерала.

В этом случае мы можем учесть еще одно вхождение, т.е. $\Delta \geqslant 23$.

Случай 2.1.2.2. Дизъюнкция, содержащая $\overline{b}$, но не содержащая $\overline{a}_1$, содержит более двух литералов, или таких дизъюнкций имеется более одной.

В этом случае мы также можем учесть еще одно вхождение, т.е. $\Delta \geqslant 23$.

Случай 2.1.2.3. Формула F имеет вид

$$(\overline{a}_1 \vee b \vee l_1) \,\&\, (\overline{a}_1 \vee \overline{b} \vee l_2) \,\&\, (\overline{b} \vee l_3) \,\&\, F',$$

где F' не содержит вхождений b.

В этом случае, если l_1 – отрицательный литерал, то при переходе к F_2 все эти дизъюнкции (точнее, те, в которые они превратятся после удаления a_1) будут удалены на шаге (R3) функции $REDUCE$ (с помощью операции DP_b), т.е. $\Delta \geqslant 25$. Если же l_1 – положительный литерал, то при переходе к F_2 будут удалены все вхождения этой переменной; по Лемме 4.2 она не входит в дизъюнкции, в которые входит b. Предположим, что все эти вхождения мы уже учли; это значит, что все они – в 2-дизъюнкции, содержащие $\overline{a}_2$. Но все такие дизъюнкции одинаковы (а формула – это множество)! Значит, по крайней мере одно вхождение $\overline{l}_1$ мы не учли, т.е. $\Delta \geqslant 23$.

Из всех векторов расщепления вида (Δ_1, Δ_2) с суммой $\Delta_1 + \Delta_2 \geqslant 23$ и $\Delta_1, \Delta_2 \geqslant 8$ наибольшее число расщепления соответствует вектору (8,15). Это число $\tau((8,15)) < 1.0644$.

Случай 2.2. $s \geqslant 3$.

Так же, как и в Случае 2.1, будем доказывать, что $\Delta \geqslant 23$, $\Delta_1, \Delta_2 \geqslant 8$.

Во-первых, в F_1 относительно F исчезают все вхождения переменных a_1, a_2 и a_3 (всего не менее девяти вхождений), а в F_2 – все вхождения переменных a_1 и b (всего не менее шести вхождений). Это происходит в результате удаления переменной a_1 и последующих преобразований на шаге (R2) функции $REDUCE$.

Поэтому $\Delta_1 \geqslant 9$, $\Delta_2 \geqslant 6$, $\Delta \geqslant 15$.

Кроме того, при переходе от F к F_1 удаляются все дизъюнкции, содержащие $\overline{a}_2$ или $\overline{a}_3$, т.е. по крайней мере четыре вхождения литералов, отличных от самих $\overline{a}_2$ и $\overline{a}_3$.

Заметим, что среди дизъюнкций, в которые входит литерал $\overline{b}$, есть по крайней мере одна, не содержащая ни переменной a_1 (дизъюнкций, содержащих переменную a_1, всего три и одна из них положительна), ни переменных a_2 и a_3 (по Лемме 4.2). В нее входит еще по крайней мере один литерал кроме $\overline{b}$, и он буден удален вместе со всей дизъюнкцией на шаге (R2) при при переходе от F к F_2.

Заметим, что $\Delta_2 \geqslant 8$, так как $\Delta_2 = 7$ только если обе дизъюнкции, содержащие $\overline{a}_1$, являются 2-дизъюнкциями, но в этом случае имеется по крайней мере две дизъюнкции, в которые входит $\overline{b}$, но не входит $\overline{a}_1$, т.е. $\Delta_2 \geqslant 8$.

Итак, к этому моменту мы имеем, что $\Delta_1 \geqslant 13$, $\Delta_2 \geqslant 8$, $\Delta \geqslant 20$. Дальнейшее рассмотрение (учет литералов, которые входят в дизъюнкции, в которые входит $\overline{a}_1$) происходит в точности как в Случае 2.1.

□

Лемма 6.3. *Число расщепления дерева $T_{5.1}$ относительно меры сложности $\mathfrak{N}$ не превосходит* 1.3248.

Доказательство. Рассмотрим два случая.

Случай 1. $s = 2$.

Заметим, что число переменных в формуле F_1 меньше числа переменных в формуле F по крайней мере на два. А именно, кроме удаляемой сразу переменной a_1, в функции $REDUCE$ на шаге (R2) удаляется переменная a_2 (так как после удаления дизъюнкции C из формулы F исчезает единственное вхождение литерала a_2).

Число переменных в формуле F_2 меньше числа дизъюнкций в формуле F по крайней мере на три. Во-первых, сразу удаляется переменная a_1. Во-вторых, на шаге (R1) будет удалена переменная a_2, оказавшаяся в 1-дизъюнкции, получившейся из дизъюнкции C после удаления a_1. Кроме того, поскольку единственное вхождение литерала b исчезает, на шаге (R2) будет удалена и переменная b.

Итого, в этом случае вершине соответствует вектор расщепления $(2, 3)$ и число расщепления $\tau((2, 3)) < 1.3248$.

Случай 2. $s \geqslant 3$.

Заметим, что число переменных в формуле F_1 меньше числа переменных в формуле F по крайней мере на три. А именно, кроме удаляемой сразу переменной a_1, в функции $REDUCE$ на шаге (R2) удаляются переменные a_2 и a_3 (так как после удаления дизъюнкции C из формулы F исчезают единственные вхождения литералов a_2 и a_3).

Число переменных в формуле F_2 меньше числа дизъюнкций в формуле F по крайней мере на два. Во-первых, сразу удаляется переменная a_1. Во-вторых, поскольку единственное вхождение литерала b исчезает, на шаге (R2) будет удалена переменная b.

Итого, в этом случае вершине также соответствует вектор расщепления $(2, 3)$ и число расщепления $\tau((2, 3)) < 1.3248$.
□

§7. Алгоритм для формул с малым числом дизъюнкций.

Чтобы получить из Алгоритма 5.1 алгоритм, хорошо работающий для формул с малым числом дизъюнкций, применим в нем в качестве $\mathcal{M}$ следующий алгоритм.

Алгоритм 7.1 (Алгоритм $\mathcal{M}_K$)

Вход:

- число $k \in \mathbb{N} \cup \{+\infty\}$,
- формула F, принадлежащая классу k–КНФ–(1,∞)–(+/–), обработанная функцией $REDUCE$.

Выход: "Выполнима", если F выполнима; в противном случае "Невыполнима".

Метод:

(K1) Проверить, имеются ли в F положительные дизъюнкции, содержащие более двух литералов. Если их нет, проверить, является ли F формулой в КНФ–(1,3). Если является, произвести следующие операции:
- Положить $H = F$.
- Положить $P_+ = \{$множество всех положительных дизъюнкций $H\}$.
- Для каждой $C = \{u_1, u_2\} \in P_+$ проделать следующую операцию: удалить C из H, а затем заменить в H все вхождения $\overline{u}_2$ на u_1.
- Выдать результат применения к формуле H какого-либо алгоритма $\mathcal{A}_K$, выясняющего выполнимость k– КНФ за время, не превосходящее $p(L)1.3289^{\mathfrak{K}(H)}$, где p – некоторый полином (например, такие алгоритмы приводятся в [6] и [12]).

(K2) Пусть в F имеется положительная дизъюнкция, содержащая не менее трех литералов, а в ней переменная v_1, входящая в F по крайней мере четыре раза. В этом случае построить формулы

$$G_1 = REDUCE(k, F[v_1]),$$
$$G_2 = REDUCE(k, F[\overline{v}_1]).$$

Если функция $SPLIT$ выдает ответ "Выполнима" хотя бы для одной из них, выдать ответ "Выполнима"; иначе "Невыполнима".

(K3) Если F не является формулой в КНФ–(1,3), выбрать переменную v_1, которая входит в формулу F наибольшее число раз. Построить формулы

$$G_1 = REDUCE(k, F[v_1]),$$
$$G_2 = REDUCE(k, F[\overline{v}_1]).$$

Если функция $SPLIT$ выдает ответ "Выполнима" хотя бы для одной из них, выдать ответ "Выполнима"; иначе "Невыполнима".

(K4) Пусть в формуле F имеются положительная дизъюнкция $C = \{v_1, v_2, \ldots, v_s\}$ и отрицательная дизъюнкция D такие, что

- $s \geqslant 3$;
- $\overline{v}_1 \in D$;
- дизъюнкция D содержит (1,2)-переменную u_1.

Построить в этом случае формулы

$$G_1 = REDUCE(k, F[v_1]),$$
$$G_2 = REDUCE(k, F[\overline{v}_1]).$$

Если функция $SPLIT$ выдает ответ "Выполнима" хотя бы для одной из них, выдать ответ "Выполнима"; иначе "Невыполнима".

(K5) Найти в формуле F дизъюнкции $C = \{v_1, v_2\}$ и D такие, что

- v_1 – (1,3)-переменная;
- $\overline{v}_1 \in D$;
- дизъюнкция D содержит (1,2)-переменную u_1.

Построить в этом случае формулы

$$G_1 = REDUCE(k, F[v_1]),$$
$$G_2 = REDUCE(k, F[\overline{v}_1]).$$

Если функция $SPLIT$ выдает ответ "Выполнима" хотя бы для одной из них, выдать ответ "Выполнима"; иначе "Невыполнима".

□

Лемма 7.1. *Алгоритм* 7.1 *работает корректно (т.е., по крайней мере одно из условий* (K1)–(K5) *всегда выполняется) и правильно решает задачу о выполнимости входной формулы.*

Доказательство. Заметим, что дизъюнкции, необходимые на шаге (K5), будут найдены. Дело в том, что если не выполнились условия шагов (K1) и (K3), то в F имеются положительные дизъюнкции длины не менее трех. Если к тому же не выполнилось условие шага (K2), все они состоят исключительно из

(1,2)-переменных; рассмотрим одну из этих дизъюнкций. Если не выполнилось и условие шага (K4), то одна из переменных этой дизъюнкции входит в отрицательную дизъюнкцию, в которую кроме нее входят только (1,3)-переменные (формула F не содержит (1,4+)-переменных, так как не выполнилось условие шага (K3)). Каждая из этих (1,3)-переменных входит в положительную 2-дизъюнкцию (т.к. не выполнилось условие шага (K2)). Последние упомянутые отрицательная и положительная дизъюнкции подходят на роль дизъюнкций D и C на шаге (K5). Итак, условие хотя бы одного из шагов (K1)–(K5) выполняется для любой формулы, прошедшей обработку функцией $REDUCE$. Поскольку каждый шаг сокращает число дизъюнкций в формуле, алгоритм должен закончить свою работу.

Единственное место алгоритма, которое может вызвать сомнения в том, что он выдает правильный ответ, это шаг (K1) – обратное приведение формулы в k–КНФ–$(1,\infty)$ к k–КНФ. Заметим, что любой выполняющий набор для исходной формулы порождает выполняющий набор для формулы, передаваемой алгоритму $\mathcal{A}_K$, если для каждой положительной дизъюнкции $\{v_1, v_2\}$ оставить в нем ровно одну из переменных v_1 или v_2 (одна из них в нем обязательно есть, т.к. он выполняющий, а другая без отрицания входит единственный раз именно в эту дизъюнкцию), а затем произвести в нем такие же замены литералов, как производятся в самой формуле. В обратную же сторону, по набору, выполняющему формулу, передаваемую алгоритму $\mathcal{A}_K$, можно построить набор, выполняющий формулу F, произведя все эти замены в противоположном направлении: все появившиеся обратно положительные 2-дизъюнкции будут иметь ровно по одному положительному и отрицательному литералу.
□

Рассмотрим дерево $T_{7.1}$, образованное естественным образом формулами, расщепляемыми на шагах (K2)–(K5) Алгоритма 7.1 и на шаге (S3) функции $SPLIT$, вызываемой из Алгоритма 7.1. Из каждой его внутренней вершины выходит два ребра, соответствующих формулам G_1 и G_2 либо F_1 и F_2 соответственно. Листьями дерева $T_{7.1}$ будут формулы, которые, получив на входе, Алгоритм 7.1 не расщепляет, а передает алгоритму $\mathcal{A}_K$ либо выдает ответ сразу.

Лемма 7.2. *Число расщепления дерева $T_{7.1}$ относительно меры* *сложности* $\mathfrak{K}$ *не превосходит* 1.1939.

Доказательство.

По Лемме 6.1 достаточно показать, что числа расщепления, соответствующие шагам (K2)–(K5), не превосходят 1.1939.

Случай 1. Расщепление произошло на шаге (K2).
В этом случае число дизъюнкций в формуле G_1 меньше числа дизъюнкций в формуле F по крайней мере на пять за счет исчезновения положительной дизъюнкции C, содержащей v_1, и четырех дизъюнкций, содержащих отрицания других переменных, входящих в C (их существование гарантируется Леммой 4.2).

Что касается формулы G_2, число дизъюнкций в ней меньше числа дизъюнкций в формуле F по крайней мере на три за счет того, что при ее построении исчезают дизъюнкции, содержащие $\overline{v}_1$.

Этому случаю соответствует вектор расщепления $(3,5)$ и число расщепления $\tau((3,5)) < 1.1939$.

Случай 2. Расщепление произошло на шаге (K3).
В этом случае формула F не является формулой в КНФ–(1,3). Значит, на шаге (K3) в качестве переменной v_1 будет выбрана (1,4+)-переменная. Пусть она входит в дизъюнкцию

$$\{v_1, v_2, \ldots, v_s\}. \tag{7.1}$$

Случай 2.1. $s = 2$.
В этом случае $\mathfrak{K}(G) - \mathfrak{K}(G_1) \geqslant 3$, так как кроме самой дизъюнкции (7.1), удаляемой сразу, на шаге (R2) удаляются дизъюнкции, содержащие $\overline{v}_2$ (по Лемме 4.2 их не менее двух). При переходе от G к G_2 будут удалены как сама дизъюнкция (7.1) (на шаге (R1)), так и дизъюнкции, содержащие $\overline{v}_1$. Поскольку v_1 – (1,4+)-переменная, их не меньше четырех. Итого, $\mathfrak{K}(G) - \mathfrak{K}(G_2) \geqslant 5$. Этому случаю соответствует вектор расщепления $(3,5)$ и число расщепления $\tau((3,5)) < 1.1939$.

Случай 2.2. $s \geqslant 3$.
В этом случае $\mathfrak{K}(G) - \mathfrak{K}(G_1) \geqslant 5$, так как кроме самой дизъюнкции (7.1), удаляемой сразу, на шаге (R2) удаляются дизъюнкции, содержащие $\overline{v}_2$ и $\overline{v}_3$ (по Лемме 4.2 их не менее четырех). С другой стороны, $\mathfrak{K}(G) - \mathfrak{K}(G_2) \geqslant 4$. При переходе от G к G_2 будут удалены дизъюнкции, содержащие $\overline{v}_1$. Поскольку v_1 – (1,4+)-переменная,

их не меньше четырех. Этому случаю соответствует вектор расщепления $(4, 5)$ и число расщепления $\tau((4, 5)) < 1.1674$.

Случай 3. Расщепление произошло на шаге (K4).
В этом случае $\mathfrak{K}(G) - \mathfrak{K}(G_1) \geqslant 5$, так как кроме самой дизъюнкции C, удаляемой сразу, на шаге (R2) удаляются дизъюнкции, содержащие $\overline{v}_2$ и $\overline{v}_3$ (по Лемме 4.2 их не менее четырех). С другой стороны, $\mathfrak{K}(G) - \mathfrak{K}(G_2) \geqslant 3$. При переходе от G к G_2 будут удалены две дизъюнкции, содержащие $\overline{v}_1$. Кроме того, будет удалено вхождение (1,2)-переменной u_1. Это значит, что на шаге (R3) (или на шаге (R2)) будет удалена еще по крайней мере одна дизъюнкция. Этому случаю соответствует вектор расщепления $(3, 5)$ и число расщепления $\tau((3, 5)) < 1.1939$.

Случай 4. Расщепление произошло на шаге (K5).
В этом случае $\mathfrak{K}(G) - \mathfrak{K}(G_1) \geqslant 3$, так как кроме самой дизъюнкции C, удаляемой сразу, на шаге (R2) удаляются дизъюнкции, содержащие $\overline{v}_1$ (по Лемме 4.2 их не менее двух). С другой стороны, $\mathfrak{K}(G) - \mathfrak{K}(G_2) \geqslant 5$. При переходе от G к G_2 будут удалены три дизъюнкции, содержащие $\overline{v}_1$. Кроме того, будет удалено вхождение (1,2)-переменной u_1. Это значит, что на шаге (R3) (или на шаге (R2)) будет удалена еще по крайней мере одна дизъюнкция. Этому случаю соответствует вектор расщепления $(3, 5)$ и число расщепления $\tau((3, 5)) < 1.1939$.

$\square$

Теорема 7.1. *Пусть в Алгоритме* 5.1 *в качестве алгоритма* $\mathcal{M}$ *используется Алгоритм* 7.1. *В таком случае Алгоритм* 5.1 *решает задачу о выполнимости формулы в* k*–КНФ–*$(1,\infty)$ *за время, не превосходящее* $p(L)1.1939^K$*, где* K *– число дизъюнкций во входной формуле,* L *– ее длина,* p *– некоторый полином.*

Доказательство. То, что алгоритм правильно решает задачу о выполнимости входной формулы, следует из Леммы 4.1, Леммы 5.1 и Леммы 7.1. Число расщепления дерева $T_{5.1}$ относительно меры сложности $\mathfrak{K}$ не превосходит 1.1939 по Лемме 6.1. Поэтому по Лемме 3.2 для доказательства теоремы достаточно доказать, что Алгоритм 7.1 вычисляет ответ за время не более $q(L)1.1939^K$, где q – некоторый полином, K – число дизъюнкций в формуле, подаваемой на вход Алгоритму 7.1, L – ее длина.

При передаче формулы алгоритму $\mathcal{A}_K$ число дизъюнкций в ней значительно уменьшается. А именно, если H – формула, которую алгоритм $\mathcal{A}_K$ получил на входе, то $\mathfrak{K}(H) = \mathfrak{K}(F) - \mathfrak{N}(F)/2$. Алгоритм $\mathcal{A}_K$ заканчивает свою работу за время, не превосходящее $p_1(L)1.2664^{\mathfrak{K}(H)}$, где p_1 – некоторый полином. Поскольку F является формулой в КНФ–(1,3), а каждая ее дизъюнкция содержит не менее двух литералов, $\mathfrak{N}(F) \geqslant \mathfrak{K}(F)/2$. Поэтому

$$1.2664^{\mathfrak{K}(H)} = 1.2664^{(\mathfrak{K}(F)-\mathfrak{N}(F)/2)} \leqslant 1.2664^{\frac{3}{4}\mathfrak{K}(F)} < 1.1938^{\mathfrak{K}(F)}.$$

Итак, на обработку каждого из листьев дерева $T_{7.1}$ затрачивается время не более $p_2(L)1.1938^m$. Пусть $f(m) = p_2(L)1.1938^m$. Алгоритм 7.1 затрачивает на обработку входной формулы время не более

$$p_3(L) \sum_{l-\text{лист } T_{7.1}} f(\mathfrak{K}(l)). \tag{7.2}$$

По Лемме 7.2 число расщепления дерева $T_{7.1}$ не превосходит 1.1939. Поэтому по Лемме 3.2 сумма (7.2) не превосходит $p_3(L)p_2(L)\max(1.1939, 1.1938)^K = p(L)1.1939^K$.
□

§8. Алгоритм для коротких формул.

Чтобы получить из Алгоритма 5.1 алгоритм, хорошо работающий для коротких формул, применим в нем в качестве $\mathcal{M}$ следующий алгоритм.

Алгоритм 8.1 (Алгоритм $\mathcal{M}_L$)

Вход:

- число $k \in \mathbb{N} \cup \{+\infty\}$,
- формула F, принадлежащая классу k–КНФ–(1,∞)–(+/–), обработанная функцией $REDUCE$.

Выход: “Выполнима”, если F выполнима; в противном случае “Невыполнима”.

Метод:

(L1) Проверить, является ли F формулой в КНФ–(1,2). Если является, то выдать ответ, выдаваемый для этой формулы алгоритмом Кулльманна и Люкхардта из [9] для формул КНФ–(1,2), работающим время $p_1(L)1.0416^L$.

(L2) Выбрать переменную v_1, которая входит в формулу F наибольшее число раз. Построить формулы

$$G_1 = REDUCE(k, F[v_1]),$$
$$G_2 = REDUCE(k, F[\overline{v}_1]).$$

Если функция $SPLIT$ выдает ответ "Выполнима" хотя бы для одной из них, выдать ответ "Выполнима"; иначе "Невыполнима".

□

Теорема 8.1. *Пусть в Алгоритме* 5.1 *в качестве алгоритма* $\mathcal{M}$ *используется Алгоритм* 8.1. *В таком случае Алгоритм* 5.1 *решает задачу о выполнимости формулы в* k*–КНФ–*$(1,\infty)$ *за время, не превосходящее* $q(L)1.0644^L$, *где* L *– длина входной формулы,* q *– некоторый полином.*

Доказательство. То, что алгоритм правильно решает задачу о выполнимости входной формулы, следует из Леммы 4.1, Леммы 5.1 и корректности используемого внутри Алгоритма 8.1 алгоритма из [9]. Число расщепления дерева $T_{5.1}$ относительно меры сложности $\mathfrak{L}$ не превосходит 1.0644 по Лемме 6.2. Поэтому по Лемме 3.2 для доказательства теоремы достаточно доказать, что Алгоритм 8.1 вычисляет ответ за время не более $p(L)1.0644^L$, где p – некоторый полином, L – длина формулы, подаваемой на вход Алгоритму 7.1.

Рассмотрим дерево $T_{8.1}$, образованное естественным образом формулами, расщепляемыми на шаге (L2) Алгоритма 8.1 и на шаге (S3) функции $SPLIT$, вызываемой из Алгоритма 8.1. Из каждой его внутренней вершины выходит два ребра, соответствующих формулам G_1 и G_2 либо F_1 и F_2 соответственно. Листьями дерева $T_{8.1}$ будут формулы, которые, получив на входе, Алгоритм 8.1 не расщепляет, а передает алгоритму Кулльманна и Люкхардта из [9] либо выдает ответ сразу.

Поскольку время работы алгоритма из [9] не превосходит $p_1(L)1.0416^L$, для доказательства искомой оценки по Лемме 3.2 достаточно показать, что число расщепления дерева $T_{8.1}$ не превосходит 1.0644. Числа расщепления, соответствующие шагу (S3), удовлетворяют этому условию по Лемме 6.2. Осталось доказать то же самое про числа расщепления, соответствующие шагу (L2). Но как уже было отмечено, разбор Случая 1 в Лемме 6.2 не использует факта, что в одну из дизъюнкций, содержащих $\overline{a}_1$, входит положительный литерал b. Условие Случая 1 Леммы 6.2 на шаге (L2) также выполняется, поэтому эти рассуждения можно использовать и для шага (L2).

Теорема доказана.

□

§9. Алгоритм для формул с малым числом переменных.

Чтобы получить из Алгоритма 5.1 алгоритм, хорошо работающий для формул с малым числом переменных, применим в нем в качестве $\mathcal{M}$ алгоритм, приведенный ниже. В нем употребляется в качестве подпрограммы какой-нибудь алгоритм, хорошо работающий для формул класса k–КНФ. Мы не будем фиксировать здесь конкретный алгоритм, так как случай его применения – критическая точка оценки, чем лучший алгоритм мы возьмем, тем лучшую оценку получим.

Алгоритм 9.1 (Алгоритм $\mathcal{M}_N$)

Вход:

- число $k \in \mathbb{N} \cup \{+\infty\}$,
- формула F, принадлежащая классу k–КНФ–(1,∞)–(+/–).

Выход: "Выполнима", если F выполнима; в противном случае "Невыполнима".

Метод:

(N1) Пусть в F имеется положительная дизъюнкция, состоящая по крайней мере из четырех литералов $C = \{v_1, v_2, \ldots, v_s\}$. В этом случае построить формулы

$$G_1 = REDUCE(k, F[v_1]),$$
$$G_2 = REDUCE(k, F[\overline{v}_1]).$$

Если хотя бы для одной из них функция $SPLIT$ выдает "Выполнима", выдать "Выполнима"; иначе "Невыполнима".

(N2) Пусть в F имеется положительная дизъюнкция из трех литералов $C = \{v_1, v_2, v_3\}$. В этом случае построить формулы

$$G_1 = F[v_1],$$
$$G_2 = F[\overline{v}_1].$$

Если хотя бы для одной из них алгоритм $\mathcal{M}_N$ выдает "Выполнима", выдать "Выполнима"; иначе "Невыполнима".

(N3) Положить $H_1 = F$.

(N4) Положить $\mathcal{P} = \{$множество всех положительных дизъюнкций $F\}$.

(N5) Для каждой $C = \{u_1, u_2\} \in \mathcal{P}$ проделать следующую операцию: удалить C из H_1, а затем заменить в H_1 все вхождения $\overline{u}_2$ на u_1.

(N6) Положить $H = H_1$. Выдать результат применения какого-либо алгоритма $\mathcal{A}_N$, выясняющего выполнимость k–КНФ, к формуле $REDUCE(k, H)$.

□

Лемма 9.1. *Алгоритм* 9.1 *правильно решает задачу о выполнимости входной формулы.*

Доказательство. По Лемме 4.1 и Лемме 5.1, единственное место Алгоритма 9.1, которое может вызвать сомнения в том, что он выдает правильный ответ, это шаги (N3)–(N6) – обратное приведение формулы в k–КНФ–$(1,\infty)$ к k–КНФ. Заметим, что любой выполняющий набор для исходной формулы F порождает выполняющий набор для формулы H (формула эквивалентная которой передается алгоритму $\mathcal{A}_N$): для каждой положительной дизъюнкции $\{v_1, v_2\}$ оставим в этом выполняющем наборе ровно одну из переменных v_1 или v_2 (одна из них в нем обязательно есть, т.к. он выполняющий, а другая без отрицания входит единственный раз именно в эту дизъюнкцию), а затем произведем в нем такие же замены литералов, какие производятся в формуле H_1. В обратную же сторону, по набору, выполняющему формулу H, можно построить набор, выполняющий формулу F, произведя все эти замены в противоположном направлении: все появившиеся обратно положительные 2-дизъюнкции будут иметь ровно по одному положительному и отрицательному литералу.

□

Лемма 9.2. *Пусть F – формула в k–КНФ–(1,∞)–(+/–), обработанная функцией $REDUCE$, L – ее длина, N – число встречающихся в ней переменных, $\mathcal{T}(n) = p_1(L)\gamma^n$ – верхняя оценка времени работы алгоритма $\mathcal{A}_N$ для формулы, в которую входит n переменных, $1 < \gamma \leqslant 2$, p_1 – некоторый полином. Тогда*

(1) *если $k = 3$, то Алгоритм 9.1 вычислит ответ для формулы F за время $p_2(L)\,(\gamma+1)^{N/3}$ для подходящего полинома p_2;*

(2) *если $k > 3$, то Алгоритм 9.1 вычислит ответ для формулы F за время $q_2(L)\max\left(1.3803, (\gamma+1)^{1/3}\right)^N$ для подходящего полинома q_2.*

Доказательство. (1) Заметим, что в этом случае Алгоритм 9.1 никаких действий на шаге (N1) не производит. Пусть формула F содержит $t \leqslant N/3$ положительных дизъюнкций, состоящих из трех литералов. Рассмотрим дерево $T_{9.1}$, образованное естественным образом формулами, расщепляемыми на шаге (N2). Из каждой его внутренней вершины выходит два ребра, соответствующих формулам G_1 и G_2. Будем называть эти ребра соответственно ребрами типа (1) и типа (2). Листьями дерева $T_{9.1}$ будут формулы, которые Алгоритм 9.1 не расщепляет, а передает алгоритму $\mathcal{A}_N$.

При проходе по ребру типа (2) удаляется одна переменная, при проходе по ребру типа (1) удаляется одна переменная и удаляются единственные положительные вхождения еще двух переменных, отрицательные вхождения которых удаляются затем на шаге (R2) функции $REDUCE$, вызываемой на шаге (N6). Количество остальных переменных (входящих в положительные 2-дизъюнкции) уменьшается в два раза на шаге (N5). Поэтому если путь из корня (исходной формулы, переданной из Алгоритма 5.1) до некоторого листа проходит ровно через i_1 ребер типа (1) и ровно через i_2 ребер типа (2), то в формуле, находящейся в этом листе, имеется не более $(N - 3i_1 - i_2)/2$ переменных. Заметим, что $i_1 + i_2 = t$, так как в результате расщепления не могут исчезать положительные 3-дизъюнкции, отличные от непосредственно расщепляемой, и не могут появиться новые. Для данных i_1 и i_2 в дереве $T_{9.1}$ имеется не более $\binom{t}{i_2}$ различных листьев с подобным путем из корня. Поскольку процессы расщепления формулы, вызова функции и шаги (N3)–(N6) отнимают лишь полиномиальное время, Алгоритм 9.1 должен выдать ответ, затратив время не более чем

$$\begin{aligned}
&p_3(L)\sum_{i_2=0}^{t}\binom{t}{i_2}\mathcal{T}((N-3i_1-i_2)/2)=\\
&=p_3(L)\sum_{i_2=0}^{t}\binom{t}{i_2}\gamma^{(N-3i_1-i_2)/2}p_1(L)=\\
&=p_1(L)p_3(L)\sum_{i_2=0}^{t}\binom{t}{i_2}\gamma^{(N+2i_2-3t)/2}=\\
&=\gamma^{(N-3t)/2}p_2(L)\sum_{i_2=0}^{t}\binom{t}{i_2}\gamma^{i_2}=\\
&=\gamma^{(N-3t)/2}p_2(L)(\gamma+1)^t=\\
&=\gamma^{N/2}p_2(L)\left(\frac{\gamma+1}{\gamma^{3/2}}\right)^t\leqslant\\
&\leqslant\gamma^{N/2}p_2(L)\left(\frac{\gamma+1}{\gamma^{3/2}}\right)^{N/3}=\\
&=p_2(L)(\gamma+1)^{N/3},
\end{aligned}$$

где $p_3(L)$ – некоторый подходящий полином, $p_2 = p_1p_3$. Справедливость неравенства вытекает из того, что для $1 < \gamma \leqslant 2$ верно, что $\gamma + 1 > \gamma^{3/2}$, а $t \leqslant N/3$.

(2) Построим дерево $T'_{9.1}$, внутренними вершинами которого являются формулы, расщепляемые на шаге (N1) Алгоритма 9.1 и на шаге (S3) функции $SPLIT$, вызываемой из Алгоритма 9.1. Из каждой его внутренней вершины выходит два ребра, соответствующих формулам G_1 и G_2 либо F_1 и F_2 соответственно. Листьями дерева $T'_{9.1}$ будут формулы, которые, получив на входе, Алгоритм 9.1 расщепляет не на шаге (N1), а на шаге (N2), либо передает алгоритму $\mathcal{A}_N$. Каждый из этих листьев представляет собой некоторую формулу G в k–КНФ–(1,∞)–(+/–), каждая из положительных дизъюнкций которой содержит не более трех литералов. Такая формула обрабатывается по (1) за время не более чем $p_2(L)(\gamma+1)^{\mathfrak{N}(G)/3}$ (нетрудно заметить, что в доказательстве

пункта (1) ограничение $k = 3$ используется только применительно к положительным дизъюнкциям). Шагу (N1) соответствует вектор расщепления (1,4) и число расщепления $\tau(1,4) < 1.3803$. По Лемме 6.3 числа расщепления дерева $T'_{9.1}$, соответствующие остальным вершинам, также не превосходят 1.3803. Отсюда по Лемме 3.2 получаем искомое утверждение.

□

Теорема 9.1. *Пусть F – формула в k–КНФ–$(1,\infty)$, L – ее длина, N – число встречающихся в ней переменных, $\mathcal{T}(n) = \gamma^N p_1(L)$ – верхняя оценка времени работы алгоритма $\mathcal{A}_N$ для формулы, в которую входит n переменных, $1 < \gamma \leqslant 2$, p, p_1 – подходящие полиномы. Пусть также в Алгоритме 5.1 в качестве алгоритма $\mathcal{M}$ используется Алгоритм 9.1. В таком случае Алгоритм 5.1 решает задачу о выполнимости формулы в k–КНФ–$(1,\infty)$ и выдает ответ за время, не превосходящее*

(1) $p(L)(\max(1.3248, (\gamma+1)^{1/3}))^N$, *если* $k = 3$;
(2) $p(L)(\max(1.3803, (\gamma+1)^{1/3}))^N$, *если* $k > 3$.

Доказательство. То, что алгоритм правильно решает задачу о выполнимости входной формулы, следует из Леммы 4.1, Леммы 5.1 и Леммы 9.1.

(1) Заметим, что время работы Алгоритма 5.1 полиномиально зависит от времени, затрачиваемого Алгоритмом 9.1 на обработку *всех* передаваемых в него формул. Это время равно $\sum\limits_{l\text{–лист } T_{5.1}} f(\mathfrak{N}(G_l))$, где G_l – формула, которой помечен лист l дерева $T_{5.1}$, $f(\nu)$ – время, затрачиваемое Алгоритмом 9.1 на обработку формулы, в которой встречаются ν переменных. По Лемме 9.2 $f(\nu) = p_2(L)(\gamma+1)^{\nu/3}$. По Лемме 6.3 имеем $\tau_{max} \leqslant 1.3248$. Таким образом, по Лемме 3.2 искомая сумма равна $p(L)(\max(1.3248, (\gamma+1)^{1/3}))^N$ для подходящего полинома p.

(2) Аналогично (1).

□

Следствие 9.1. *Пусть число k, подаваемое на вход Алгоритма 5.1, равно трем. Если в качестве алгоритма $\mathcal{A}_N$ взять алгоритм из* [8], *то время работы Алгоритма 5.1 составит* $p(L)1.3566^N$.

Доказательство. В [8] Куллманном показано, что его алгоритм выясняет выполнимость формулы в 3–КНФ[1] за время $p(L)1.49625^{\mathfrak{N}(F)}$. Из этого по Теореме 9.1 вытекает искомая оценка $p(L)(\max(1.3248, 2.49625^{1/3}))^N \leqslant p(L)(\max(1.3248, 1.3566))^N = p(L)1.3566^N$.
□

Следствие 9.2. *Пусть число k, подаваемое на вход Алгоритма 5.1, больше трех. Тогда если в качестве алгоритма $\mathcal{A}_N$ взять алгоритм из* [12], *то время работы Алгоритма* 5.1 *составит* $p(L)(\psi(k)+1)^{N/3}$, *где* $\psi(k) = \tau((1,2,\ldots,k-1))$.

Доказательство. В [12] Б. Мониеном и Е. Шпекенмейером показано, что их алгоритм выясняет выполнимость формулы в k– КНФ за время $p(L)\psi(k)^N$, где $\psi(k) = \tau((1,\ldots,k-1))$. Несложные вычисления показывают, что $\tau((1,4)) < \tau((1,2,3)) \leqslant \ldots \leqslant \tau((1,2,\ldots,k-1))$. Из этого по Теореме 9.1 вытекает искомая оценка $p(L)(\max(1.3803, (\psi(k)+1)^{1/3}))^N = p(L)(\psi(k)+1)^{N/3}$.
□

Ниже приводится таблица приближенных значений $\psi(k)$ и $(\psi(k)+1)^{1/3}$.

k	$\psi(k)$	$(\psi(k)+1)^{1/3}$
4	1.8393	1.4161
5	1.9276	1.4306
6	1.9660	1.4368
7	1.9836	1.4397

Табл. 2. Приближенные значения $\psi(k)$ и $(\psi(k)+1)^{1/3}$.

§10. Следствие для общей задачи ВЫП.

Для задачи выполнимости формулы в КНФ, равно как и для ее подзадач, подобных рассмотренной в этой статье, известны различные верхние оценки в зависимости от параметров входной формулы (длины, числа дизъюнкций, числа переменных). Эти

[1] В статье [8] алгоритм сформулирован в терминах двойственной задачи, задачи тавтологичности 3-ДНФ.

верхние оценки достигаются при использовании разных алгоритмов. Чтобы получить ответ на вопрос о выполнимости данной формулы как можно быстрее, следует запустить эти алгоритмы на исполнение параллельно либо запустить тот из них, оценка времени работы которого для данной формулы является наилучшей. Таким образом возникают оценки, зависящие от дополнительных параметров – $\frac{\mathfrak{K}}{\mathfrak{N}}$, $\frac{\mathfrak{L}}{\mathfrak{N}}$ и т. д., "объединяющие" оценки различных алгоритмов. Некоторые оценки такого рода для различных классов булевых формул рассмотрены в [9]. Ниже мы рассмотрим одну верхнюю оценку для формул в КНФ, возникающую при применении результатов данной статьи. Кроме очевидных оценок для формул в КНФ–(1,∞), возникающих при объединении оценок §§7–9, оказывается, что из этих результатов можно вывести и одну оценку для произвольных формул в КНФ.

Рассмотрим формулу F в КНФ. Мы можем преобразовать ее в формулу G в КНФ–(1,∞) за полиномиальное время, причем ее параметры изменятся следующим образом:

$$\begin{aligned}\mathfrak{K}(G) &= \mathfrak{K}(F) + \mathfrak{N}(F),\\ \mathfrak{L}(G) &= \mathfrak{L}(F) + 2\cdot\mathfrak{N}(F),\\ \mathfrak{N}(G) &= 2\cdot\mathfrak{N}(F).\end{aligned}$$

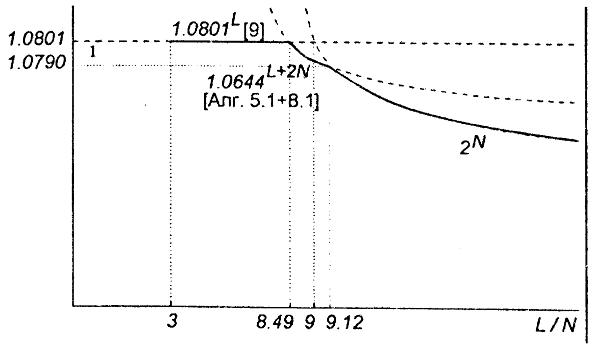
Это можно сделать, заменив все вхождения каждого положительного литерала $v \in V$ на отрицание новой переменной x_v и добавив дизъюнкцию $v \vee x_v$. Эквивалентность выполнимости формул F и G очевидна.

Если теперь применить оценку Теоремы 8.1 к нашей формуле, мы получим, что вопрос о выполнимости формулы F в КНФ можно выяснить за время

$$p(\mathfrak{L}(G))1.0644^{\mathfrak{L}(G)} \leqslant p'(\mathfrak{L}(F))1.0644^{(\mathfrak{L}(F)+2\cdot\mathfrak{N}(F))},$$

где p, p' – некоторые полиномы. Заметим, что при определенных значениях $\frac{\mathfrak{L}(F)}{\mathfrak{N}(F)}$ это выражение дает оценку лучшую, чем $1.0801^{\mathfrak{L}(F)}$-алгоритм из [9] и обычный $2^{\mathfrak{N}(F)}$-алгоритм. Наиболее оптимальным будет алгоритм, который обрабатывает формулы, у которых $\mathfrak{L}(F) < 8.49\mathfrak{N}(F)$, с помощью $1.0801^{\mathfrak{L}(F)}$-алгоритма из [9], формулы, у которых $\mathfrak{L}(F) > 9.12\mathfrak{N}(F)$, с помощью обычного $2^{\mathfrak{N}(F)}$-алгоритма, а остальные – с помощью Алгоритма 5.1, в котором в качестве алгоритма $\mathcal{M}$ используется Алгоритм 8.1.

Ниже представлен график, отражающий зависимость основания α экспоненты $\alpha^{\mathfrak{L}(F)}$ от $\frac{L}{N} = \frac{\mathfrak{L}(F)}{\mathfrak{N}(F)}$. Данные для формул, у которых $\frac{L}{N} < 3$, не приводятся, так как такие формулы содержат переменные, входящие не более двух раз, и преобразуются за полиномиальное время в формулы, у которых $\frac{L}{N} \geqslant 3$. По-видимому, для выяснения выполнимости таких формул существует еще более эффективный алгоритм.



Литература

1. М. Гэри, Д. Джонсон, *Вычислительные машины и труднорешаемые задачи.* Москва, Мир, 1982.
2. Е. Я. Данцин, *Системы доказательства таетологичности, основанные на методе расщеплений.* Диссертация на соискание степени канд. ф.-м. н.. Ленинград, ЛОМИ, 1983.
3. Е. Я. Данцин, *Алгоритмика задачи выполнимости.* — в сб.: Проблемы сокращения перебора, в серии : Вопросы кибернетики, N 131 (1987), Москва, АН СССР.
4. Е. Я. Данцин, В. Я. Крейнович, *Экспоненциальные верхние оценки для задачи выполнимости.* — в сб.: Девятая всесоюзная конференция по математической логике. Тезисы докладов. Ленинград, 1988, "Наука".
5. M. Davis, H. Putnam, *A computing procedure for quantification theory.* — J. Assoc. Comp. Mach. **7** (1960), 201–215.
6. E. A. Hirsch, *Deciding satisfiability of formulas with K clauses in less than $2^{0.30897K}$ steps.* Steklov Institute of Mathematics at St.Petersburg, PDMI preprint 4/1997, 1997.
7. E. A. Hirsch, *Two new upper bounds for SAT.* Proceedings of the Ninth Annual ACM-SIAM Symposium on Discrete Algorithms, 1998. To appear.
8. O. Kullmann, *A systematical approach to 3-SAT-decision, yielding 3-SAT-decision in less than 1.5045^n steps.* — Theoretical Computer Science. To appear.

9. O. Kullmann, *Worst-case analysis, 3-SAT decision and lower bounds: approaches for improved SAT algorithms.* — DIMACS Proceedings SAT Workshop 1996 (Eds.: Gu, Du, Pardalos, eds.), American Mathematical Society, 1996.
10. O. Kullmann and H. Luckhardt, *Deciding propositional tautologies: Algorithms and their complexity.* Frankfurt a. M., manuscript, December 1996.
11. H. Luckhardt, *Obere Komplexitätsschranken für TAUT-Entscheidungen.* — Proc. Frege Conference 1984. 331–337, Schwerine, Akademie-Verlag Berline.
12. B. Monien, E. Speckenmeyer, *3-satisfiability is testable in $O(1.62^r)$ steps.* — Bericht Nr. 3/1979. Universität-Gesamthochschule-Paderborn, Reihe Theoretische Informatik.
13. B. Monien, E. Speckenmeyer, *Solving satisfiability in less then 2^n steps.* — Discrete Applied Mathematics **10** (1985), 287–295.
14. I. Schiermeyer, *Pure literal look ahead: An $O(1.497^n)$ 3-Satisfiability algorithm.* — Workshop on the Satisfiability Problem Technical Report (J. Franco, G. Gallo, H. Kleine Büning, E. Speckenmeyer, C. Spera University Köln, Report No. **96-230**, eds.), Siena, April, 29 – May, 3, **1996**.

Hirsch E. A. Separating sings in the propositional satisfiability problem.

In 1980 B. Monien and E. Speckenmeyer, and (independently) Dantsin, proved that satisfiability of a propositional formula in CNF can be checked in less than 2^N steps (N is the number of variables). Later many other upper bounds for SAT and its subproblems we proved. A formula in CNF is in CNF-$(1,\infty)$, if each positive literal occurs in it at most once. H. Luckhardt in 1984 studied formulas in CNF-$(1,\infty)$. In this paper we prove several new upper bounds for formulas in CNF-$(1,\infty)$ by introducing new *signs separation principle*. Namely, we present algorithms working the time of the order 1.1939^K and 1.0644^L for a formula consisting of K clauses containing L literals occurences. We also present an algorithm for formulas in CNF-$(1,\infty)$ whose clauses are bounded in length.

С.-Петербургское отделение
Математического института
им. В. А. Стеклова РАН

Поступило 19 апреля 1997 г.