Сложность пропозициональных доказательств (черновик краткого конспекта*лекций)

Д.М. Ицыксон †

28 сентября 2017 г.

Содержание

*Этот текст содержит массу опечаток, ошибок и неточностей. Просьба сообщать о найденных недостатках по электронной почте, в теме письма напишите слово КОНСПЕКТ. Все найденные недостатки будут исправляться в новых версиях конспекта.

†ПОМИ РАН. E-mail: dmitrits@pdmi.ras.ru.

1 Системы доказательств

Пусть Σ — конечное множество, которое мы будем называть алфавитом. Языком мы будем называть произвольное множество конечных слов над алфавитом Σ. Другими словами, язык — это подмножество Σ^*.

Мы сейчас определим понятие системы доказательств для языка L, доказывать мы будем утверждения вида $x \in L$.

Системой доказательств для языка L называется полиномиальный по времени алгоритм Π, который принимает два входа x (строка, для которой доказывается принадлежность к языку L) и w (доказательство) и выдает ответ из множества $\{0, 1\}$. Алгоритм должен обладать следующими свойствами:

- (Корректность) если $\Pi(x, w) = 1$ для некоторых строк $x, w \in \Sigma^*$, то $x \in L$;
- (Полнота) для любой строки $x \in L$ существует строка $w \in \Sigma^*$, что $\Pi(x, w) = 1$.

Система доказательств Π для языка L называется полиномиально ограниченной, если существует такой полином q, что для всех $x \in L$ существует строка w длины не больше, чем $q(|x|)$, что $\Pi(x, w) = 1$.

Класс сложности P состоит из языков, для которых существует полиномиальный по времени алгоритм, проверяющий принадлежность этому языку. Класс сложности NP состоит из языков, для которых существует полиномиально ограниченная система доказательств. Легко понять, что $\mathrm{P} \subseteq \mathrm{NP}$, поскольку для языка из P легко построить систему доказательств, которая не использует доказательство. Неизвестно, является ли это включение строгим.

Между языками определяются сведения. Будем говорить, что язык L сводится за полиномиальное время (или по Карпу) к языку L', если существует такая полиномиально вычислимая функция f, что для всех строк x выполняется $x \in L$ тогда и только тогда, когда $f(x) \in L'$. Обозначаются сведения так: $L \leq_p L'$.

Легко проверить, что сведения обладают свойством рефлексивности, транзитивности и замкнутости относительно принадлежности классу P.

В классе NP лежит специфический язык SAT, который состоит из всех выполнимых пропозициональных формул в конъюнктивной нормальной форме (КНФ). Действительно, доказательством пренадлежности этому языку будет набор значений переменных, которые выполняют эту формулу.

Теорема 1.1 (Кук, Левин)**.** Любой язык из класса NP сводится к SAT за полиномиальное время.

Доказательство этой теоремы мы не приводим. Вы можете его прочитать, например, тут: [AB09].

Класс coNP состоит из языков, дополнение которых лежит в классе NP.

Заметим, что если $\mathrm{P} = \mathrm{NP}$, то и $\mathrm{NP} = \mathrm{coNP} = \mathrm{P}$. Значит, из неравенства классов NP и coNP следует неравенство классов P и NP.

Например, в классе coNP лежит множество невыполнимых формул в КНФ UNSAT. Действительно, дополнение этого языка состоит из выполнимых формул в КНФ и строк, которые не являются формулами в КНФ.

Из теоремы Кука-Левина легко получается:

Следствие 1.1. Любой язык из класса coNP сводится к UNSAT за полиномиальное время.

Язык UNSAT является основным объектом изучения в теории сложности пропозициональных доказательств. Пропозициональной системой доказательств называется система доказательств для языка UNSAT.

Предложение 1.1. Классы NP и coNP замкнуты относительно полиномиальных по времени сведениям.

Доказательство. Пусть L сводится к L' с помощью функции f и $L' \in$ NP. Пусть Π' — это полиномиально ограниченная система доказательств для L'. Определим $\Pi(x, w) = \Pi'(f(x), w)$. Нетрудно проверить, что Π — это полиномиально ограниченная система доказательств для L. В случае coNP доказательство аналогичное, надо рассматривать полиномиально ограниченную систему доказательств для дополнения языка L'. □

Предложение 1.2 (Кук, Рекхау [CR74])**.** NP = coNP тогда и только тогда, когда для UNSAT есть полиномиально ограниченная система доказательств.

Доказательство. Если NP = coNP, то UNSAT $\in$ NP, следовательно по определению класса NP существует полиномиально ограниченная система доказательств для языка NUSAT.

Пусть UNSAT $\in$ NP, поскольку все языки из coNP сводятся к UNSAT, то по предложению 1.1 coNP $\subseteq$ NP. Нетрудно понять, что из UNSAT $\in$ NP следует, что SAT $\in$ coNP. Тогда по теореме Кука-Левина и предложению 1.1 выполняется NP $\subseteq$ coNP. □

Основная программа исследований в теории сложности пропозициональных доказательств состоит в доказательстве суперполиномиальных нижних оценок на длину доказательства во все более и более сильных системах доказательств. Тем самым мы медленно будем приближаться к доказательству неравенства классов NP и coNP.

2 Деревья решений

Мы начнем изучать системы доказательств с самых слабых. Первая система доказательств — это таблица истинности, мы просто выписываем значение формулы при всех значениях переменных. Понятно, что проверка такого доказательства занимает полиномиальное от длины доказательства время. Длина такого доказательства всегда 2^n, где n — это число переменных. Стоит отметить, что нам важна только длина доказательств, само оно не особенно содержательное, поскольку состоит из одних нулей.

Рассмотрим более интересную систему доказательств, деревья решений.

Деревом решений для невыполнимой КНФ формулы ϕ называется бинарное дерево, все вершины которого кроме листьев помечены переменными формулы. Одно из двух исходящих ребер, из вершины, помеченной переменной x помечено подстановкой $x := 0$, другое $x := 1$. Листья дерева помечены дизъюнктами формулы ϕ, причем дизъюнкт в любом листе опровергается подстановкой, которая читается на пути от корня до этого листа.

Такое дерево является доказательством невыполнимости формулы ϕ. Действительно, мы просто перебираем возможные значения переменных и убеждаемся, что во всех случаях мы получаем противоречие с дизъюнктами формулы ϕ. Проверить такое доказательство легко: достаточно проверить что дерево корректное, т.е. для всех листьев надо проверить, что дизъюнкт опровергается подстановкой, которая соответствует пути до этого листа.

Размером дерева удобно считать число листьев, поскольку число вершин не более, чем в два раза превосходит число листьев. Нетрудно видеть, что всегда существует дерево размером 2^n, где n число переменных, достаточно просто перебрать все значения всех переменных. Однако в реальности деревья могут быть значительно короче.

Кроме размера важной характеристикой является глубина дерева (максимальное расстояние от корня до листа).

Предложение 2.3. Размер дерева решений не превосходит 2^d, где d — это его глубина.

Наша ближайшая цель состоит в том, чтобы привести пример формулы (семейства формул), которая имеет полиномиальный размер, но требует экспоненциального дерева решений.

У нас будет несколько базовых примеров, которые мы будем доказывать в разных системах доказательств. Первый такой пример — это принцип Дирихле (pigeonhole principle). Формула $\mathrm{PHP}_{\mathrm{n}}^{\mathrm{m}}$ утверждает, что m кроликов сидят в n клетках так, что каждый кролик сидит в одной клетке, и в каждой клетке сидит не более одного кролика. Формула содержит переменные $p_{i,j}$, где $i \in [m], j \in [n]$, которая означает, что i-й кролик сидит в j-й клетке. Формула содержит два типа дизъюнктов. Дизъюнкты кроликов:

- $p_{i,1} \vee p_{i,2} \vee \cdots \vee p_{i,n}$, для всех $i \in [m]$;

и дизъюнкты для клеток:

- $\neg p_{i,j} \vee \neg p_{k,j}$, для всех $i \neq k \in [m]$ и всех $j \in [n]$;

Нетрудно видеть, что при $m > n$ формула $\mathrm{PHP}_{\mathrm{n}}^{\mathrm{m}}$ является невыполнимой.

Докажем для начала нижнюю оценку на глубину дерева решений для формулы $\mathrm{PHP}_{\mathrm{n}}^{\mathrm{m}}$. Это можно сделать с помощью следующей игры двух игроков. Дана невыполнимая формула ϕ и два игрока: Алиса и Боб, по очереди Алиса выбирает переменную формулы, Боб выбирает значение этой переменной. Игра заканчивается, как только сформированная подстановка противоречит дизъюнкту формулы ϕ. Боб зарабатывает очко за каждый свой ход. Цель Боба заработать как можно больше очков.

Лемма 2.1. Если в указанной игре для невыполнимой формулы ϕ у Боба есть стратегия, которая позволяет ему заработать как минимум t очков, то глубина дерева решений формулы ϕ как минимум t.

Доказательство. Пусть есть дерево решений для формулы ϕ глубины меньше t, тогда Алиса будет делать запросы согласно этому дереву и противоречие будет найдено за менее, чем t шагов, следовательно Боб заработает менее t очков, противоречие. □

С помощью этой игры очень легко доказать, что глубина дерева решений для формулы $\mathrm{PHP}_{\mathrm{n}}^{\mathrm{m}}$ не меньше, чем n. Действительно, Боб может на все вопросы отвечать 0. Заметим, что ни один из клеточных дизъюнктов не может опровергнуться, значит опровергнулся кроличий дизъюнкт, следовательно Боб давал как минимум n ответов и заработал как минимум n очков. Эту оценку можно улучшить до $\Omega(n^2)$, выбрав более хитрую стратегию, это остается в качестве упражнения.

Однако из нижней оценки на глубину не следует нижняя оценка на размер дерева решений. Чтобы доказать оценку на размер, мы модифицируем игру. Теперь первого игрока зовут

Прувер, а второго Делэйер. Прувер также выбирает значение переменной, а Делэер либо отвечает $*$, либо выбирает значение переменной из $\{0,1\}$. В случае ответа $*$ Прувер выбирает значение сам. Игра заканчивается, когда текущая подстановка опровергает дизъюнкт формулы ϕ. На этот раз Делэер получает очки не за все ответы, а только за ответ $*$. В этом виде игра была предложена в работе Пудлака и Иммальяццо 2000 года [PI00].

Лемма 2.2. Если в игре Прувера и Делэера для невыполнимой формулы ϕ у Делэера есть стратегия, которая позволяет ему заработать как минимум t очков, то размер любого дерева решений формулы ϕ как минимум 2^t.

Доказательство. Рассмотрим какое-нибудь дерево решений T и рассмотрим стратегию Прувера на этом дереве: он спрашивает значение переменной (сначала в корне дерева), если Делэер говорит значение, то Прувер переходит по соответствующему ребру дерева, если Делэер возвращает $*$, то Прувер выбирает значение случайным образом и перемещается в дереве согласно этому выбору. Игра с вероятностью 1 заканчивается в листе дерева. Поскольку на пути до каждого листа Делэер заработал как минимум t очков, то вероятность оказаться в этом листе не более 2^{-t}. Следовательно число листов как минимум 2^t. □

Теперь можно доказать оценку на размер дерева.

Предложение 2.4. Размер любого дерева решений формулы $\mathrm{PHP}_{\mathrm{n}}^{\mathrm{m}}$ при $m > n$ не меньше, чем 2^n.

Доказательство. Достаточно описать стратегию для Делэера, которая гарантирует ему заработать как минимум n очков. Стратегия очень простая: если Прувер спрашивает про переменную $p_{i,j}$ и при этом в клетке j уже кто-то сидит, то Делэер отвечает 0, в противном случае он отвечает $*$. Поскольку Делэер следит за тем, чтобы в одну клетку не посадили двух кроликов, то игра может закончиться только тем, что нарушился кроличий дизъюнкт $p_{k,1} \vee p_{k,2} \vee \cdots \vee p_{k,n}$. Покажем, что для каждой клетки Делэер заработал хотя бы одно очко. Действительно, кто подставил $p_{k,j} = 0$. Если это был Прувер, то это был ответ $*$ и очко для Делэера. Если это был Делэер, то он это мог сделать только потому, что в клетке j уже кто-то сидит. Но туда кого-то посадить мог только Прувер, поскольку Делэер не дает ответы 1 сам. Следовательно, за этот ответ Делэер заработал очко. Итого, Делэер заработал как минимум n очков. □

Хорошо, нижнюю оценку мы доказали. А насколько она точная? Тривиальная верхняя оценка для формулы $\mathrm{PHP}_{\mathrm{n}}^{\mathrm{n+1}}$ — это $2^{O(n^2)}$, поскольку число переменных $n(n+1)$. Следующая лемма показывает, что верхнюю оценку можно улучшить.

Лемма 2.3. Существует дерево решений для формулы $\mathrm{PHP}_{\mathrm{n}}^{\mathrm{m}}$ при $m > n$ размера $2^{O(n \log n)}$.

Доказательство. Поскольку формула $\mathrm{PHP}_{\mathrm{n}}^{\mathrm{m}}$ при $m > n$ содержит все дизъюнкты $\mathrm{PHP}_{\mathrm{n}}^{\mathrm{n+1}}$, то достаточно доказывать только для $m = n + 1$.

Пусть минимальное дерево для формулы $\mathrm{PHP}_{\mathrm{n}}^{\mathrm{n+1}}$ имеет размер T_n. Оценим T_n через T_{n-1}, для этого рассмотрим дерево, которое сначала перебирает значение переменной $p_{n+1,1}$, в ветви со значением 0 спрашивается значение $p_{n+1,2}$ и т.д. В этом дереве есть ветвь, которая соответствует подстановке $p_{n+1,1} = p_{n+1,2} = \cdots = p_{n+1,n} = 0$, эта ветвь опровергает дизъюнкт, который соответствует $(n+1)$-му кролику. У нас есть еще n ветвей, в которых мы $n+1$ кролика куда-то посадили. В каждой из этих ветвей надо расщепиться по всем другим кроликам сидящим в

той же клетке. Каждый раз, значению 1 будет соответствовать немедленное противоречию с клеточным дизъюнктом. Останутся n вершин, в каждой из которой $(n+1)$ кролик где-то сидит и больше никто в ней не сидит. Т.е. n раз надо опровергнуть формулу $\mathrm{PHP}^{\mathrm{n}}_{\mathrm{n}-1}$ (возможно только клетки в ней иначе переименованы). Таким образом мы можем оценить $T_n \leq nT_{n-1}+n^2+1$. При этом $T_1 \leq 2$. Докажем индукцией по n, что $T_n \leq 2^{2n\log n}$. База очевидна, индукционный переход: $T_n \leq nT_{n-1}+n^2+1 \leq 2^{2(n-1)\log(n-1)+\log n}+n^2+1 \leq 2^{(2n-1)\log n}+2^{3\log n} \leq 2^{2n\log n}$ при $n \geq 2$. □

Оказывается, что нижнюю оценку тоже можно улучшить до $2^{\Omega(n\log n)}$. Впервые это было доказано Данчевым и Риисом в 2001 году [DR01]. Мы приведем доказательство, использующее асимметриченые игры Прувера и Делэера, предложенные Бейесдорфом, Галези и Лауриа в 2010 году [BGL10]. Асимметричная игра Прувера и Делэера отличается от обычной тем, что вместе с ответом $*$ делэер сообщает два положительных вещественных числа (a_0, a_1), что $\frac{1}{a_0}+\frac{1}{a_1}=1$. При этом, если Прувер выбирает 0, то Делэер получает $\log a_0$ очков, а если Прувер выбирает 1, то $\log a_1$ очков. В случае обычной игры $a_0 = a_1 = 2$. Оказывается, что лемма 2.2 выполняется и для асимметричной игры.

Лемма 2.4. Если в асимметричной игре Прувера и Делэера для невыполнимой формулы ϕ у Делэера есть стратегия, которая позволяет ему заработать как минимум t очков, то размер любого дерева решений формулы ϕ как минимум 2^t.

Доказательство. Рассмотрим какое-нибудь дерево решений T и рассмотрим стратегию Прувера на этом дереве: он спрашивает значение переменной (сначала в корне дерева), если Делэер говорит значение, то Прувер переходит по соответствующему ребру дерева, если Делэер возвращает $*$ и два числа (a_0, a_1), то Прувер выбирает значение случайным образом: 0 с вероятностью $\frac{1}{a_0}$ и 1 с вероятностью $\frac{1}{a_1}$ и переходит по соответствующему ребру дерева. Игра с вероятностью 1 заканчивается в листе дерева. Рассмотрим какой-нибудь лист дерева, в котором закончилась игра, пусть на пути до этого листа Делэер k раз выдавал $*$ с числами $(a_0^{(1)}, a_1^{(1)}), (a_0^{(2)}, a_1^{(2)}), \ldots, (a_0^{(k)}, a_1^{(k)})$ и лист соответствует выборам $i_1, i_2, \ldots, i_k \in \{0,1\}$. Поскольку на пути до каждого листа Делэер заработал как минимум t очков, то вероятность оказаться в этом листе равна $\frac{1}{\prod_{j=1}^{k} a_{i_j}^{(j)}} = 2^{-\sum_{j=1}^{k}\log a_{i_j}^{(j)}} \leq 2^{-t}$. Следовательно число листов как минимум 2^t. □

Теперь мы готовы доказать асимптотически точную нижнюю оценку.

Теорема 2.1. Размер любого дерева решений формулы $\mathrm{PHP}^{\mathrm{m}}_{\mathrm{n}}$ при $m > n$ не меньше, чем $2^{\Omega(n\log n)}$.

Доказательство. Мы опишем стратегию для Делэера в асимметричной игре. Числа (a_0, a_1) будут одинаковыми для всех ходов, а чему они равны мы выберем позже.

Также как и раньше Делэер не будет позволять сажать кролика в клетку, в которой уже кто-то сидит, но теперь будут выполняться и другие ограничения. Пусть α — это текущая подстановка, которая соответствует ответам Прувера, ответы Делэера мы сюда не включаем. Обозначим через $J_i(\alpha)$ множество клеток, которые явно запрещены кролику i и при этом в этих клетках никто не сидит. Более формально $J_i(\alpha) = \{j \in [n] \mid \alpha(p_{i,j}) = 0 \text{ и } \alpha(p_{i',j}) \neq 1 \text{ при } i' \in [m]\}$.

Стратегия Делэера будет такой. Пусть Прувер спрашивает значение переменной $p_{i,j}$:

- Он отвечает 0, если в j-й клетке уже кто-то сидит или i-й кролик уже где-то сидит.
- Он отвечает 1, если $|J_i(\alpha)| \geq n/2$, j-я клетка свободна и i-й кролик еще нигде не сидит.
- Он отвечает $*$ в противном случае.

Ответ 1 быстрее приводит к противоречию, поэтому мы выберем $a_1 > 2$. Рассмотрим конец игры. Как и раньше, противоречие не может быть в дизъюнкте, который соответствует клетке, поскольку Делэер не дает посадить кролика в клетку, в которой уже кто-то сидит.

Значит, противоречие случилось в дизъюнкте, который соответствовал кролику: $p_{i,1} \vee p_{i,2} \vee \cdots \vee p_{i,n}$. Раз такое противоречие произошло, то оказалось так, что Делэер не смог ответить 1 в тот момент, когда случилось $|J_i(\alpha)| \geq n/2$. Это могло произойти только в том случае, если в клетке, про которую был запрос уже кто-то сидит. Значит, есть момент, в который есть как минимум $\frac{n}{2} - 1$ занятая клетка и, поскольку Делэер не дает пруверу не сажаем двух кроликов в одну клетку, то эти клетки заняты разными кроликами.

Мы рассмотрим эти $\frac{n}{2} - 1$ клеток и кроликов в них. И для каждой клетки посмотрим, кто сажал в нее кролика. В первом случае этого кролика сажал Прувер, это значит, что за это Делэер получил $\log a_1$ очков. Во втором случае этого кролика сажал Делэер. А это значит, что в тот момент $|J_i(\alpha)| \geq n/2$, и все эти нули мог ставить только Прувер, поскольку нули Делэер ставит только по причине того, что либо клетка уже кем-то занята, либо кролик где-то сидит. Значит, за эту клетку Делэер получил как минимум $\frac{n}{2} \log a_0$ очков. Итого, мы получили, что Делэер заработает как минимум $(\frac{n}{2} - 1) \min\{\log a_1, \frac{n}{2} \log a_0\}$ очков. Осталось подобрать нужные a_1 и a_0.

Мы хотим, чтобы $\log a_1 = \Omega(\log n)$ и при этом $\log a_0 = \Omega(\log n/n)$. $a_0 = a_1/(a_1 - 1) = 1 + 1/(a_1 - 1) \geq e^{1/(a_1-1)}$. Выберем $1/(a_1 - 1) = \log n/n$, т.е. $a_1 = n/logn + 1$, тогда $\log a_0 \geq \Omega(\log n/n)$, а $\log a_1 = \Omega(\log n)$. □

3 Метод резолюций

3.1 Ветвящияся программы

Если мы хотим усилить деревья решений, то естественной идеей является такая: в деревьях решений могут быть вершины с одинаковыми поддеревьями, такие вершины можно склеивать. Т.е. вместо деревьев решений у нас будут ветвящиеся программы.

Ветвящаяся программа для формулы ϕ в КНФ называется ориентированный граф без циклов с одним входом (вершиной входящей степени ноль), в которой все вершины кроме выходов (вершин исходящей степени ноль) помечены переменными формулы ϕ, при этом из вершины, помеченной переменной x ровно два ребра, одно из них помечено подстановкой $x = 0$, другое $x = 1$. Каждый выход помечен каким-нибудь дизъюнктом формулы ϕ и для каждого набора значений переменных, если выйти из входа программы и идти по ребрам, согласованным с данным набором, то такой путь закончится в выходе, в котором написан дизъюнкт, который ложен для данного набора значений переменных. Размером ветвящейся программы мы называем число вершин графа.

Отметим, что не каждому пути из входа ветвящейся программы соответствует подстановка, поскольку в определении не запрещается тестировать переменную несколько раз на одном пути. Но можно рассматривать только консистентные пути, в которой одной переменной не

подставляются разные значения. Каждому набору значений переменных соответствует некоторый консистентный путь от входа к выходу. Поэтому достаточно проверить, что подстановка вдоль любого консистентного пути от входа к выходу опровергает дизъюнкт, записанный в выходе.

Нетрудно понять, что если у формулы ϕ есть ветвящаяся программа, то она невыполнимая, поскольку каждый набор значений переменных приведет в дизъюнкт, который опровергается этим набором. Кроме того дерево решений тоже является ветвящейся программой, поэтому ветвящаяся программа существует для любой невыполнимой формулы. Однако ветвящиеся программы не являются системой доказательств, поскольку по помеченному графу трудно проверить, что он является корректной ветвящейся программой для формулы. Мы не можем проверять все наборы значений переменных, поскольку число путей может быть гораздо больше, чем размер ветвящейся программы. Чтобы убедиться в этом поймем, что для невыполнимой формулы ϕ всегда существует ветвящаяся программа размера $O(\phi)$. Эта программа устроена следующим образом: пусть формула $\phi = C_1 \wedge C_2 \wedge \cdots \wedge C_m$, Ветвящаяся программа будет иметь выделенные вершины $v_1, v_2, \ldots, v_m$ (это не все вершины, а некоторые), при этом вершина v_1 будет входом ветвящейся программы. В вершине v_i для $i \in [m-1]$ будет запрос к первой переменной дизъюнкта C_i, по значению, которое этот дизъюнкт выполняет, мы переходим в v_{i+1}, по противоположному значению мы переходим в вершину, помеченную второй переменной C_i, по значению, которое выполняет C_i мы переходим в v_{i+1}, а по значению, которое не выполняет C_i переходим в вершину, помеченную третьей переменной C_i и т.д., если все переменные закончились, то это очередная вершина будет выходом, помеченным дизъюнктом C_i. Вершина v_m будет выходом, помеченным дизъюнктом C_m, поскольку, раз формула невыполнимая и ни один из предыдущих дизъюнктов не опровергся, то обязательно опровергается последний дизъюнкт.

Чтобы ветвящиеся программы можно было бы эффективно проверять на корректность, мы потребуем дополнительное ограничений, одноразовость программы. Ветвящаяся программа называется одноразовой, если на каждом пути от входа до выхода каждая переменная как пометка вершины встречается не более одного раза. Нетрудно понять, что минимальное дерево решений обязательно является одноразовой ветвящейся программой.

Оказывается, корректность одноразовой программы для формулы ϕ можно проверить за полиномиальное от размера формулы ϕ время. Мы должны проверить выполнение трех свойств: 1) То, что граф является ориентированным графом без циклов. На самом деле это проверять необязательно, достаточно требовать, чтобы нам вместе с графом предоставили его топологическую сортировку, т.е. нумерацию вершин, при которой ребра идут от вершин с меньшими номерами к вершинам с большими номерами. 2) То, что на любом пути от входа до выхода каждая переменная встречается один раз. Для каждой переменной можно посчитать список переменных, которые можно увидеть, выходя из этой вершины. Легче всего это сделать, начав обход от вершины с максимальным номером, в обратном порядке. Такие списки для очередной вершины получаются, как объединение списков для потомков. Имея такие списки легко проверить свойство одноразовости. 3) Вместо того, чтобы проверять, что для каждого набора значений переменных мы найдем дизъюнкт, который этим набором опровергается, мы будем проверять, что каждый путь из входа до выхода противоречит дизъюнкту, который в этом выходе написан. Проверку этого свойства мы ненадолго отложим. Для него нам потребуется определить резолюционную систему доказательств.

3.2 Резолюции

Говорят, что из дизъюнктов $A \vee x$ и $B \vee \neg x$ по правилу резолюции можно вывести дизъюнкт $A \vee B$. Говорят, что дизъюнкты $A \vee x$ и $B \vee \neg x$ резольвируются по переменной x. Правило резолюции обладает таким свойством, если какой-то набор значений переменных выполняет обе посылки правила, то этот же набор выполняет и заключение правила. Или наоборот, если какой-то набор значений переменных опровергает заключение правила, то он же опровергает и одну из посылок.

Резолюционным доказательством невыполнимости формулы ϕ является вывод пустого дизъюнкта (тождественно ложного) из дизъюнктов формулы ϕ по правилам резолюции. Обычно доказательство представляется в виде ориентированного графа без циклов, выходами которого помечены дизъюнкты формулы ϕ, каждая из остальных вершин помечена каким-нибудь дизъюнктом, причем из каждой вершины исходит два ребра, и дизъюнкт в вершине — это результат применения правила резолюции к дизъюнктам, которыми помечены концы исходящих ребер. Один из входов графа помечен пустым дизъюнктом. Размером резолюционного доказательства мы называем число вершин в графе.

Если резолюционное доказательство формулы ϕ существует, то формула ϕ невыполнимая. Действительно, пустой дизъюнкт опровергается любым набором значений переменных, значит, для каждого набора хотя бы один из дизъюнктов, из которых этот пустой дизъюнкт получился, тоже опровергается, значит опровергается хотя бы одна из посылок для того дизъюнкта и т.д., получаем, что один из дизъюнктов формулы ϕ опровергается этим набором.

Резолюционное доказательство нетрудно проверить за полиномиальное от его размера время. Нам еще нужно убедиться в полноте этой системы доказательств, т.е. в том, что у любой невыполнимой формулы есть резолюционное доказательство. Это мы покажем, указав связь между деревьями решений и ветвящимися программами и резолюционными доказательствами.

Если граф доказательства является деревом, то такое доказательство называют *древовидным*. Если на любом пути от входа до выхода правила резолюции применяются по различным переменным, то такое доказательство называют *регулярным*.

3.2.1 Построение ветвящейся программы по резолюционному доказательству

По резолюционному доказательству формулы ϕ легко построить ветвящуюся программу для формулы ϕ.

1. Мы удалим все входы, кроме одного, помеченного пустым дизъюнктом. После этого мы повторим это, пока не останется граф только с одним входом.

2. Развернем стрелки в графе. Теперь у графа будет единственный вход, помеченный пустым дизъюнктом, это будет входом ветвящейся программы. Каждый выход помечен дизъюнктами формулы ϕ.

3. Каждую вершину кроме выходов пометим переменной, по которой происходит применение правила резолюции в данной вершине.

4. Пусть из вершины помеченной $A \vee B$ идут ребра в вершины, помеченные $A \vee x$ и $B \vee \neg x$. Пометим ребро из $A \vee B$ в $A \vee x$ подстановкой $x = 0$, а ребро из $A \vee B$ в $B \vee \neg x$ подстановкой $x = 1$.

Заметим, что мы строили ветвящуюся программу так, чтобы подстановка вдоль любого консистентного пути от входа до вершины опровергала дизъюнкт в этой вершине. Это верно для пустого дизъюнкта во входе, и это сохраняется при увеличении пути на одно ребро. Значит, в выходе подстановка тоже опровергает дизъюнкт (а там записан дизъюнкт формулы ϕ). Таким образом мы получили корректную ветвящуюся программу.

Важно отметить, что описанное преобразование сохраняет граф. Т.е. из древовидного резолюционного доказательства мы построим дерево решений, а из регулярного резолюционного доказательства мы построим одноразовую ветвящуюся программу.

3.2.2 Построение регулярного резолюционного доказательства по одноразовой ветвящейся программе

Теперь мы покажем, что для деревьев решений и одноразовых ветвящихся программ возможно и обратное построение. Тем самым мы докажем, что любая невыполнимая формула имеет резолюционное доказательство.

Мы будем доказывать для одноразовых ветвящихся программ. Дерево решений может не являться одноразовой программой (если мы два раза спрашиваем значение какой-то переменной), но в таком случае его можно сократить.

Будем строить резолюционное доказательство по одноразовой ветвящейся программе для формулы ϕ. Конструкцию удобно начинать с выходов, они уже помечены дизъюнктами формулы ϕ, мы пометим дизъюнктами все оставшиеся вершины программы так, чтобы дизъюнкт опровергался подстановкой от входа программы до текущей вершины. Если это свойство будет выполняться для входа, то во входе будет написан пустой дизъюнкт. Это построение удобно делать в обратном топологическом порядке, от выходов. Рассмотрим вершину v ветвящейся программы, пусть из нее выходят ребра в вершины u и w, которые уже помечены дизъюнктами C_u и C_w. Пусть переменная v помечена переменной x, ребро (v, u) помечено подстановкой $x = 0$, а ребро (v, w) помечено подстановкой $x = 1$. Мы знаем, что C_u не содержит $\neg x$, а C_v не содержит x, иначе бы подстановки выполняли бы дизъюнкт. Если C_u содержит x, а C_v содержит $\neg x$, то пометим вершину v результатом применения правила резолюции, примененному к C_v и C_w. Если же C_u не содержит x, то C_u опровергается подстановкой на пути от корня до v и мы можем пометить v дизъюнктом C_u. Аналогично, в случае, если C_u содержит x, а C_w не содержит $\neg x$, то мы пометим v дизъюнктом C_w. Заметим, что мы таким образом получили не совсем резолюционное доказательство, поскольку иногда мы не применяли правило резолюции, а просто копировали одну из посылок. Но из такого графа нетрудно получить резолюционное доказательство, выкидывая ненужные ребра и склеивая вершины.

Нетрудно проверить, что из дерева решений мы получим древовидное резолюционное доказательство, а из одноразовой ветвящейся программы мы получим регулярное резолюционное доказательство.

Также заметим, что построение резолюционного доказательства по одноразовой ветвящейся программе помогает нам проверить одноразовую ветвящуюся программу на корректность. Для этого достаточно восстановить по одноразовой ветвящейся программе регулярное резолюционное доказательство и проверить его. Если быть аккуратным, то возможно, что мы примем и некорректную одноразовую ветвящуюся программу, которая порождает корректное резолюционное доказательство, но это не страшно.

3.3 Нижняя оценка для принципа Дирихле

Для удобства иногда в метод резолюций добавляют еще одно правило вывода: правило ослабление, которое позволяет вывести дизъюнкт $A \vee B$ из дизъюнкта A. Это правило тоже обладает тем свойством, что если набор выполняет посылку, то он выполняет и заключение правила. Поэтому добавление этого правила не влияет на корректность метода резолюций. Полнота метода резолюций тоже не пострадает, поскольку новое правило можно не использовать.

От правила ослабления можно избавится:

Предложение 3.5. Если доказательство невыполнимости формулы ϕ использует правила ослабления и резолюции, то его можно переписать в доказательство ϕ, использующее только правило резолюции, при этом размер доказательства не увеличится.

Доказательство. Упражнение. □

Кроме того правило ослабления делает систему доказательств семантически полной.

Предложение 3.6. Пусть из множества дизъюнктов $C_1, C_2, \ldots, C_k$ семантически следует дизъюнкт D. Т.е. любой набор значений переменных, который выполняет $C_1, C_2, \ldots, C_k$ также выполняет и D. Тогда дизъюнкт D можно вывести из $C_1, C_2, \ldots, C_k$ с помощью правил ослабления и резолюции.

Нам же будет удобно считать, что при необходимости мы будем пользоваться правилом ослабления, так как имея правило ослабления мы можем делать подстановки в доказательства.

Пусть F — формула в КНФ. Формула $F|_{x=a}$, где $a \in \{0, 1\}$ получается из F в результате подстановки. Это значит, что все дизъюнкты, которые подстановка $x = a$ выполняет удаляются из формулы, а из остальных дизъюнктов удаляется литерал с переменной x. Если после такого удаления в формуле получится пустой дизъюнкт, то эта формула превратилась в константу 0, если же в формуле не осталось дизъюнктов, то она превратилась в константу 1.

Предложение 3.7. Пусть F — невыполнимая формула, $C_1, C_2, \ldots, C_k$ резолюционное доказательство F, использующая правила резолюции и ослабления. Тогда $C_1|_{x=a}, C_2|_{x=a}, \ldots, C_k|_{x=a}$ — это резолюционное доказательство $F|_{x=a}$.

Доказательство. Упражнение. □

Наша ближайшая цель доказать следующую теорему

Теорема 3.1 ([Hak85])**.** Любое резолюционное доказательство формулы $\mathrm{PHP}_{\mathrm{n}}^{\mathrm{n}+1}$ имеет размер как минимум $2^{\Omega(n)}$.

Известно, что эта оценка точная и существует верхняя оценка $n^3 2^n$.

Будем говорить, что набор значений переменных формулы $\mathrm{PHP}_{\mathrm{n}}^{\mathrm{n}+1}$ i-критический для $i \in [n+1]$, если он выполняет все условия, кроме условия для i-го кролика. Это значит, что все кролики, кроме i-го сидят ровно в одной клетке, а i-й кролик нигде не сидит. Отметим, что в критическом наборе все клетки заняты. Дальнейшее рассуждение мы будем проводить только на множестве критических наборов.

Мы введем преобразование позитивизация, которое позволит нам избавиться от отрицаний переменных. Вместо $\neg x_{i,j}$ мы напишем $\bigvee_{k \neq i \in [n+1]} x_{k,j}$. На множестве критических наборов

разницы не будет, поскольку если клетка с номером j не занята кроликом i, то в ней сидит один из остальных кроликов и наоборот.

Для дизъюнкта C будем обозначать C^+ дизъюнкт, в котором мы избавились от всех отрицаний.

Лемма 3.1 ([BP96])**.** Для любого дизъюнкта C и критического набора, значения на этом наборе C и C^+ совпадают.

Позитивизацию можно применить и к резолюционному доказательству. Отметим, что пустой дизъюнкт останется пустым дизъюнктом после позитивизации.

Лемма 3.2. Пусть $C_1, \ldots, C_s$ — резолюционное доказательство $\mathrm{PHP}^{\mathrm{n}+1}_{\mathrm{n}}$. Тогда есть некоторое $i \in [s]$, при котором $|C_i^+| \geq \frac{(n+1)^2}{9}$.

Доказательство. Определим для каждого дизъюнкта C, использующего переменные $p_{i,j}$, меру $\mu(C)$, которая равна числу кроликов $i \in [n+1]$ для которых существует i-критический набор, который опровергает C.

Заметим, что мера аксиом для клетки равна нулю, поскольку любой критический набор выполняет аксиомы клетки. А мера кроличьих аксиом равняется единице. Мера пустого дизъюнкта равняется $n+1$.

Нетрудно видеть, что выполняется свойство полуаддитивности μ, т.е., если дизъюнкт C получается по правилам из A и B, то $\mu(C) \leq \mu(A) + \mu(B)$, поскольку любой опровергающий набор для C является опровергающим для одного из A или B.

Из полуаддитивности следует, что в доказательстве есть дизъюнкт C с $\frac{n+1}{3}\mu(C)\frac{2}{3}(n+1)$. Мы докажем, что C^+ содержит много переменных. Пусть $\mu(C) = s$, мы знаем, что $\frac{n+1}{3}s\frac{2}{3}(n+1)$. Мы покажем, что C^+ содержит как минимум $s((n+1)-s)$ переменных.

Рассмотрим какого-нибудь кролика i для которого существует i-критический набор α, который опровергает клоз C. Рассмотрим другого кролика j, для которого любой j-критический набор выполняет клоз C. В наборе α все кролики где-то сидят, в частности кролик j сидит в клетке k_j. Рассмотрим набор α_j, который отличается от α тем, что кролик i сидит в клетке k_j, а кролик j нигде не сидит. Набор α_j является j-критическим. Следовательно α_j выполняет дизъюнкт C. Заметим, что α и α_j отличаются только значениями переменной x_{i,k_j} и x_{j,k_j}. Следовательно переменная x_{i,k_j} содержится в C^+. Таким образом мы для кролика i найдем $n+1-s$ переменных x_{i,k_j} в C^+ и им будут соответствовать разные k_j, так как это занятые клетки одного и того же набора α. Есть s разных кроликов i, следовательно C^+ содержит как минимум $s((n+1)-s) \geq \frac{(n+1)^2}{9}$ переменных. □

Теперь мы готовы доказать теорему 3.1.

Доказательство теоремы 3.1. Назовем позитивированный дизъюкнт толстым, если в нем как минимум $n(n+1)/100$ переменных. Поскольку всего переменных $n(n+1)$, то существует такая переменная $p_{i,j}$, которая входит в как минимум $\frac{1}{100}$ от всех толстых дизъюнктов доказательства. Сделаем подстановку в позитивизированное доказательство $p_{i,j} = 1$ и $p_{k,j} = 0$ для всех $k \neq i$ и $p_{i,t} = 0$ для $t \neq i$. Получим фактически доказательство $\mathrm{PHP}^{\mathrm{n}}_{\mathrm{n}}$. Повторим такие подстановки $n/100$ раз. В итоге получим доказательство $\mathrm{PHP}^{0.99\mathrm{n}+1}_{0.99\mathrm{n}}$, в котором толстых дизъюнктов в $(99/100)^{n/100}$ раз меньше, чем в исходном доказательстве. Но в получившемся позитивизированном доказательстве по лемме 3.2 есть толстый дизъюнкт. Следовательно в исходном

позитивизированном доказательстве как минимум $(100/99)^{n/100}$ толстых дизъюнктов, следовательно в исходном доказательстве как минимум $(100/99)^{n/100}$ дизъюнктов. □

3.4 Размер и ширина резолюционных доказательств

Пусть ϕ — невыполнимая формула в КНФ. Введем несколько обозначений:

- $S_T(\phi)$ — размер минимального древовидного резолюционного доказательства для формулы ϕ. Как мы знаем, с точностью до мультипликативной константы $S_T(\phi)$ совпадает с размером минимального дерева решений для ϕ.
- $S_R(\phi)$ — минимальный размер резолюционного доказательства формулы ϕ.

Поскольку древовидное доказательство является частным случаем обычного, то всегда выполняется неравенство $S_T(\phi) \geq S(\phi)$.

Мы введем еще одну меру сложности формулы — это минимальная ширина доказательства. Шириной дизъюнкта мы называем число литералов в нем (мы считаем, что дизъюнкт не содержит несколько литералов по одной переменной), шириной резолюционного доказательства мы называем максимальную ширину дизъюнкта, который в этом доказательстве использовался. Для невыполнимой формулы ϕ мы будем обозначать $w_R(\phi)$ минимальную возможную ширину резолюционного доказательства формулы ϕ.

Предложение 3.8. Для невыполнимой формулы ϕ выполняется неравенство $S_R(\phi) \leq (2n+1)^{w_r(\phi)}$, где n — число переменных формулы ϕ.

Доказательство. Число дизъюнктов ширины не более k можно оценить сверху так: есть k мест для литералов, на каждое место можно поставить один из $2n$ литералов или не поставить ничего, т.е. таких дизъюнктов не больше $(2n+1)^k$. □

Тем самым, если $w_R(\phi)$ маленькая (например, константная), то и размер доказательства маленький. Оказывается можно показать и наоборот, что если $S_R(\phi)$ маленькая, то и $w_R(\phi)$ маленькая.

Лемма 3.3. Пусть ϕ — невыполнимая формула в k-КНФ, x — переменная формулы ϕ, $a \in \{0,1\}$. Тогда если для некоторого натурального числа w выполняется $w_R(\phi|_{x=a}) \leq w-1$ и $w_R(\phi|_{x=1-a}) \leq w$, то $w_R(\phi) \leq \max\{k, w\}$.

Доказательство. Пусть x^1 обозначает x, а x^0 обозначает $\neg x$. Рассмотрим вывод $\phi|_{x=a}$ ширины не более $w-1$, этот вывод легко перестроить либо в вывод x^{1-a} либо в вывод пустого дизъюнкта из ϕ ширины w. Для этого достаточно вернуть литералы x^{1-a} во все дизъюнкты формулы $\phi|_{x=a}$, которые получились выкидыванием из дизъюнктов ϕ литерала x^{1-a}. Если получился вывод пустого дизъюнкта, то на этом можно закончить. Используя дизъюнкт x^{1-a}, можно вывести все дизъюнкты формулы $\phi|_{x=1-a}$ с помощью вывода ширины не более k. А после этого можно воспользоваться выводом пустого дизъюнкта из формулы $\phi|_{x=1-a}$ ширины w. Итого, мы получили вывод пустого дизъюнкта из ϕ ширины не более $\max\{k, w\}$. □

Теорема 3.2 (Бен-Сассон, Вигдерсон [BSW01])**.** Для каждой невыполнимой формулы ϕ в k-КНФ выполняются неравенства:

1. $S_T(\phi) \geq 2^{w_R(\phi)-k}$

2. $S_R(\phi) \geq 2^{\frac{(w_R(\phi)-k)^2}{8n}}$

Доказательство. 1. Докажем по индукции по числу переменных. Удобнее доказывать утверждение для минимального размера дерева решений. База индукции для формулы из 1 переменной, чтобы она была невыполнимой, формула должна содержать дизъюнкты x и $\neg x$. Размер минимального дерева решений равен 3, а ширина 1.

Пусть в корне минимального дерева решений для формулы ϕ стоит расщепление по переменной x, то в одной из ветви стоит минимальное дерево решений для формулы $\phi|_{x=0}$, а в другом минимальное дерево решений для формулы $\phi|_{x=1}$. Т.е. $S_T(\phi) = S_T(\phi|_{x=0}) + S_T(\phi_{x=1})$.

Поймем, что в случае, когда $w_R(\phi) \leq k$, утверждение очевидно, поэтому далее будем считать, что $w_R(\phi) \geq k+1$.

Нетрудно понять, что $w_R(\phi|_x = b) \leq w_R(\phi)$ для всех $b \in \{0,1\}$. Из леммы 3.3 следует, что не может такого быть, что $w_R(\phi|_x = b) \leq w_R(\phi) - 2$ для всех $b \in \{0,1\}$, так как из этого следовало бы, что $w_R(\phi) \leq \max k, w_R(\phi) - 1$. Аналогично не может быть, что $w_R(\phi|_x = b) \leq w_R(\phi) - 1$ и $w_R(\phi|_x = 1-b) \leq w_R(\phi) - 2$ для некоторого $b \in \{0,1\}$. Значит, есть два варианта, либо а) $w_R(\phi|x=0) = w_R(\phi|_{x=1}) = w_R(\phi) - 1$, либо б) $w_R(\phi|_{x=a}) = w_R(\phi)$ для некоторого $a \in \{0,1\}$. Разберем эти два случая отдельно.

а) $S_T(\phi) \geq S_T(\phi|_{x=0}) + S_T(\phi|_{x=1}) \geq 2^{w_R(\phi)-1-k} + 2^{w_R(\phi)-1-k} = 2^{w_R(\phi)-k}$. Во втором неравенстве мы дважды применили индукционное предположение.

б) $S_T(\phi) \geq S_T(\phi|_{x=a}) \geq 2^{w_R(\phi)-k}$. Во втором неравенстве мы применили индукционное предположение.

2. Пусть $W \geq 1$ — некоторый параметр, значение которого мы подберем позже. Мы будем говорить, что дизъюнкт толстый, если в нем как минимум W литералов.

Утверждение 1. Пусть F — множество толстых дизъюнктов от n переменных. Тогда существует такой литерал ℓ, который входит в как минимум $\frac{W}{2n}|S|$ дизъюнктов из множества S.

Доказательство. Всего есть $2n$ литералов, если бы каждый литерал входил в меньше, чем $\frac{W}{2n}|S|$ дизъюнктов, то общее число литералов можно было бы менее $W|S|$. Но в каждый толстый дизъюнкт входит как минимум W литералов, следовательно общее число литералов должно быть не менее $W|S|$, противоречие. □

Из утверждения freqlit следует, что существует литерал, выполнив который, выполнится как минимум доля $\frac{W}{2n}$ толстых дизъюнктов. Т.е. после подстановки, которая выполняет этот литерал, останутся невыполненными не более доли $\left(1 - \frac{W}{2n}\right)$ толстых дизъюнктов.

Обозначим $a = \left(1 - \frac{W}{2n}\right)^{-1} \geq e^{W/2n}$.

Утверждение 2. ?? Если невыполнимая k-КНФ формула содержит резолюционное опровержение, в котором менее a^b толстых дизъюнктов, то $w_R(\phi) \leq W + b + k$.

Доказательство. Доказательство по индукции по b и числу переменных n. База $n = 1$ очевидна. Проверим базу для $b = 0$, в этом случае доказательство либо не содержит толстых дизъюнктов совсем, тогда его ширина не больше, чем $\max\{W, k\} \leq W + k$, либо содержит всего один толстый дизъюнкт. Заметим, что его ширина не может быть хотя бы $W + 1$, так как если мы применим к этому дизъюнкту какое-то правило, то его ширина уменьшится не более, чем на 1, т.е. есть еще хотя бы один толстый дизъюнкт. Еще возможен вариант, что этот

дизъюнкт не используется в доказательстве, тогда его можно просто удалить. Значит, ширина доказательства не больше $\max\{W+1,k\} \leq W+k$, поскольку $k \geq 1$.

Теперь допустим, что утверждение выполняется для всех меньших значений n и b. По утверждению 1 существует литерал x, который входит в как минимум долю $\frac{1}{a}$ всех толстых дизъюнктов резолюционного доказательства. Если мы сделаем подстановку $x=1$ в доказательство, то получим доказательство формулы $\phi|_{x=1}$, в котором будет не более a^{b-1} толстых дизъюнктов. Тогда по индукционному предположению $w_R(\phi|_{x=1}) \leq W+b+k-1$. Применяя индукционное предположение по числу переменных, получаем $w_R(\phi|_{x=0}) \leq W+b+k$. Тогда по лемме 3.3 $w_R(\phi) \leq W+b+k$. □

Выберем b такое, что $a^b = S_R(\phi)$. Тогда
По утверждению **??** мы получим, что $w_R(\phi) \leq W+b+k$.

$$b = \frac{\log S_R(\phi)}{\log a} \leq \frac{2n \log S_R(\phi)}{W \log e} \leq \frac{2n \log S_R(\phi)}{W}$$

Итого, $w_R(F) \leq W+b+k \leq W + \frac{2n \log S_R(\phi)}{W} + k$. Сумма $W + \frac{2n \log S_R(\phi)}{W}$ минимальна, когда $W = \sqrt{2n \log S_R(\phi)}$. Получаем, что $w_R(\phi) \leq 2\sqrt{2n \log S_R(\phi)} + k$. Отсюда получаем требуемую оценку. □

Из доказанной теоремы следует, что для того, чтобы доказать нижнюю оценку на размер резолюционного доказательства формулы в k-КНФ для небольшого k достаточно доказать нижнюю оценку на ширину $w_R(\phi) = \Omega(n^{\frac{1}{2}+\epsilon})$ для некоторого $\epsilon > 0$.

3.5 Цейтинские формулы

Цейтинская формула $Ts_{G,f}$ строится по простому неориентированному графу $G(V,E)$. каждому ребру $e \in E$ соответствует переменная x_e, есть функция $f : V \to \{0,1\}$, для каждой вершины $v \in E$ записывается формула в КНФ, кодирующая $\sum\limits_{u \in V:(u,v)\in E} p_{(u,v)} \bmod 2 = f(v)$, где $\oplus$ используется для обозначения суммы по модулю 2. Конъюнкция этих формул называется цейтинской формулой. Заметим, что если степень вершин G ограничена константой d, то $Ts_{G,f}$ — это формула в d-КНФ, в которой число дизъюнктов не превосходит $2^d|V|$.

Если для какой-то компоненты связности $U \subseteq V$ выполняется $\sum\limits_{v \in U} f(v) \bmod 2 = 1$, то цейтинская формула невыполнима $Ts_{G,f}$. Действительно, достаточно просуммировать (по модулю два) все равенства, которые записаны в вершинах и получится $0=1$, поскольку каждая переменная будет встречаться ровно два раза.

Если же для каждой компоненты связности $\sum\limits_{v \in V} f(v) \bmod 2 = 0$, то цейтинская формула выполнима: присвоим всем переменным значения 0, равенства не будут выполняться в четном числе вершин для каждой компоненты связности. Рассмотрим две вершины из одной компоненты связности, в которых не выполняются равенства, и заменим значения переменных на противоположные вдоль пути между этими двумя вершинами. Заметим, что после такой операции число вершин, в которых не выполняется равенство, уменьшится на 2. Осталось повторить эту операцию, пока равенство не начнет выполняться во всех вершинах.

Для неориентированного графа $G(V,E)$ и для двух дизъюнктных множеств $A, B \subseteq V$ обозначим через $E(A,B)$ множество ребер, в котором один конец лежит в A, другой в B. Рас-

ширительной способностью графа G называется число $e(G)$, которое равняется минимальному значению $|E(U, V \setminus V)|$ по всем $U \subseteq V$, что $\frac{|V|}{3} \leq |U| \leq \frac{2|V|}{3}$.

Теорема 3.3. Пусть $Ts_{G,f}$ — невыполнимая цейтинская формула, построенная по связному графу G. Тогда $w_R(Ts_{G,f}) \geq e(G)$.

Доказательство. Введем меру μ на множестве дизъюнктов. $\mu(C)$ равняется минимальному множеству вершин графа G, что из условий четности в этих вершинах семантически следует дизъюнкт C. Для дизъюнктов формулы $Ts_{G,f}$ мера равняется 1. Мера пустого дизъюнкта равняется n, поскольку все условия цейтинской формулы кроме одного в связном графе можно выполнить, так как у нас есть ребра, которые участвуют только один раз, значит в каждой компоненте связности, которые получаются при удалении вершины, можно сделать четную сумму пометок.

Нетрудно убедиться, что мера μ полуаддитивна, т.е., что если дизъюнкт C получился по правилу резолюции из дизъюнктов A и B, то $\mu(C) \leq \mu(A) + \mu(B)$. Действительно, дизъюнкт C всегда следует из условий для объединения множества вершин, из которых следует A и B.

Рассмотрим некоторое доказательство формулы $Ts_{G,f}$. В этом доказательстве рассмотрим дизъюнкт C с мерой $n/3 \leq \mu(C) \leq 2n/3$. Такой есть из того, что мера дизъюнктов формулы равны 1, мера пустого дизъюнкта равна 1 и свойства полуаддитивности. Пусть дизъюнкт C семантически следует из условий для вершин множества $U \subseteq V$ и $|U| = \mu(C)$. Мы покажем, что для любого ребра $e \in E(U, V \setminus U)$ переменная x_e содержится в дизъюнкте C, из этого будет следовать, что $|C| \geq |E(U, V \setminus U)| \geq e(G)$, что и требуется доказать.

Рассмотрим $e \in E(U, V \setminus U)$, пусть $v \in U$ — конец ребра e. Мы знаем, что из множества вершин $U \setminus \{v\}$ семантически не следует дизъюнкт C, следовательно существует набор значений переменных σ, который выполняет все условия $U \setminus \{v\}$, но не выполняет дизъюнкт C. Набор σ не выполняет условие в вершине v, так как иначе бы он выполнял бы все условия для вершин из U, а следовательно выполнял бы и C. Если x_e не входит в C, то можно поменять значение переменной x_e в σ, чтобы получившийся набор σ_e выполнил бы условие в вершине v (поскольку условие — это просто условие о четности). Поскольку ребро e не имеет больше концов в U, то σ_e выполняет все U, но все еще не выполняет C. Противоречие, поэтому x_e должно входить в C. □

Список литературы

[AB09] Sanjeev Arora and Boaz Barak. *Computational Complexity: A Modern Approach.* Cambridge University Press, New York, NY, USA, 1st edition, 2009.

[BGL10] Olaf Beyersdorff, Nicola Galesi, and Massimo Lauria. A lower bound for the pigeonhole principle in tree-like resolution by asymmetric prover-delayer games. *Inf. Process. Lett.*, 110(23):1074–1077, 2010.

[BP96] Paul Beame and Toniann Pitassi. Simplified and improved resolution lower bounds. In *37th Annual Symposium on Foundations of Computer Science, FOCS '96, Burlington, Vermont, USA, 14-16 October, 1996*, pages 274–282, 1996.

[BSW01] E. Ben-Sasson and A. Wigderson. Short proofs are narrow — resolution made simple. *Journal of ACM*, 48(2):149–169, 2001.

[CR74] Stephen A. Cook and Robert A. Reckhow. On the lengths of proofs in the propositional calculus (preliminary version). In *Proceedings of the 6th Annual ACM Symposium on Theory of Computing, April 30 - May 2, 1974, Seattle, Washington, USA*, pages 135–148, 1974.

[DR01] Stefan S. Dantchev and Søren Riis. Tree Resolution Proofs of the Weak Pigeon-Hole Principle. In *Proceedings of the 16th Annual IEEE Conference on Computational Complexity, Chicago, Illinois, USA, June 18-21, 2001*, pages 69–75. IEEE Computer Society, 2001.

[Hak85] Armin Haken. The intractability of resolution. *Theoretical Computer Science*, 39:297–308, 1985.

[PI00] Pavel Pudlák and Russell Impagliazzo. A lower bound for DLL algorithms for k-SAT (preliminary version). In *Proceedings of the Eleventh Annual ACM-SIAM Symposium on Discrete Algorithms, January 9-11, 2000, San Francisco, CA, USA.*, pages 128–136, 2000.