

## 1 Введение

Булевы игры используются для моделирования многих комбинаторных задач, задач планирования и т. д. Цель «Верификатора» (V) — подобрать значения переменных, чтобы сделать заданную пропозициональную формулу истинной, а цель «Фальсификатора» (F) — сделать её ложной. В работе рассмотрены различные варианты последовательности ходов и порядка выбора переменных.

## 2 Основные понятия

Пусть дана булева функция с четным количеством аргументов:

$$f : \mathbb{B}^{2k} \rightarrow \mathbb{B}$$

Где  $k = 1, 2, \dots$ . Теперь рассмотрим игру для двух игроков, определенную на этой функции. Игроков будем называть "Фальсификатор" (F) и "Верификатор" (V). Пусть F ходит первым, и каждый из игроков знает, какие ходы были сделаны его оппонентом. В свой ход каждый игрок выбирает любую из переменных, от которых зависит  $f$ , и присваивает ей значение 0 или 1. Одну и ту же переменную нельзя выбирать дважды, поэтому в игре будет сделано всего  $2k$  ходов, после которых будет установлено значение всех переменных, его можно представить как вектор  $\vec{b} \in \mathbb{B}^{2k}$ . V побеждает, если  $f(\vec{b}) = 1$ , иначе побеждает F.

**Пример** Положим  $k = 2$  и пусть функция для игры задана как:

$$f_0(x_1, x_2, x_3, x_4) = \bar{x}_1 \bar{x}_2 \bar{x}_3 \bar{x}_4 \vee \bar{x}_1 x_2 \bar{x}_3 x_4 \vee x_1 \bar{x}_2 x_3 \bar{x}_4 \vee x_1 x_2 x_3 x_4$$

Нетрудно заметить, что  $f_0$  записана в СДНФ, а значит у нее всего 4 выполняющих набора:

$$\begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \\ 0 \\ 1 \end{bmatrix}, \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}$$

Пусть первым все еще ходит F. Проанализировав ситуацию, видим, что любое действие F приведет к исключению ровно двух выполняющих наборов.

Предположим, что ход F это  $x_1 = 1$ . Тогда мы можем представить, что мы вычеркиваем из таблицы истинности этой функции 8 строк, "в игре" остается ровно два выполняющих набора:

$$\begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}, \begin{bmatrix} 1 \\ 0 \\ 1 \\ 0 \end{bmatrix}$$

Заметим, что размер наборов формально не уменьшился, но первая переменная во всех наборах теперь равна, поэтому ее можно исключить без потери информации. Таким образом, можем сказать, что теперь игра идет на функции:

$$\begin{aligned} f_1(x_2, x_3, x_4) &= f_0(1, x_2, x_3, x_4) \\ &= \bar{1} \bar{x}_2 \bar{x}_3 \bar{x}_4 \vee \bar{1} x_2 \bar{x}_3 x_4 \vee \\ &\quad 1 \bar{x}_2 x_3 \bar{x}_4 \vee 1 x_2 x_3 x_4 \\ &= \bar{x}_2 \bar{x}_3 \bar{x}_4 \vee x_2 x_3 x_4 \end{aligned}$$

Теперь мы переходим к ходу игрока V, который побеждает, если итоговое значение функции окажется равно 1, поэтому логичной кажется "жадная" стратегия - найти ход, который приведет к тому, что количество выполняющих наборов максимально возможно.

Заметим, что у V есть идеальный ход -  $x_3 = 1$ , он сохраняет оба набора. Теперь имеем формулу:

$$f_2(x_2, x_4) = f_0(1, x_2, 1, x_4) = x_2 x_4 \vee \bar{x}_2 \bar{x}_4$$

Теперь снова наступил ход F. Заметим, что любой его ход приводит к поражению на следующем ходу. Положим, что  $x_2 = 0$ . Тогда ход V это  $x_4 = 0$ . Итог игры:

$$f_0(1, 0, 1, 0) = 1$$

Победил V. Интерпретацию этой игры на таблице истинности можно увидеть на Рис. 1.

### 2.1 Сценарии проведения игры

Ранее был рассмотрен наиболее очевидный случай - игроки совершают ходы по очереди, однако интересны и иные случаи. Игрок может иметь уже зафиксированный набор переменных на момент начала игры, например переменные  $x_1, x_2$  могут находиться под контролем V, а  $x_3, x_4$  под контролем F, каждый из них может придавать значения только своим переменным. Далее в работе такая ситуация будет представлена следующим порядком игры - пусть первым действием игрок F выберет себе  $k$  переменных, которым он сможет присваивать значения, а остальные  $k$  переменных будут под контролем V, дальше игра происходит по стандартному порядку.

Итак, теперь формально перечислим рассматриваемые варианты последовательности ходов:

1. Сначала F фиксирует набор подконтрольных переменных.
  - 1.1. F придает значение всем своим переменным сразу, потом тоже самое делает V
  - 1.2. V придает значение всем своим переменным сразу, потом тоже самое делает F

$x_4$	$x_3$	$x_2$	$x_1$	$f_0$
0	0	0	0	1
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	0
0	1	0	1	1
0	1	1	0	0
0	1	1	1	0
1	0	0	0	0
1	0	0	1	0
1	0	1	0	1
1	0	1	1	0
1	1	0	0	0
1	1	0	1	0
1	1	1	0	0
1	1	1	1	1

Рис. 1: Интерпретация примера игры на таблице. Значения в первых четырех столбиках раскрашены в цвет, если они были выбраны каким-то из игроков. Значения в последнем столбике раскрашены в цвет хода, на котором они были исключены из игры. Красный - первый ход (F), синий - второй ход (V), зеленый - третий ход (F), желтый - четвертый ход (V).

1.3. F и V придают значения своим переменным по очереди, первым ходит F

1.4. F и V придают значения своим переменным по очереди, первым ходит V

2. F и V не фиксируют переменные до начала игры, а каждый раз выбирают из общего, полного набора.

2.1. F и V ходят по очереди, начинает F

2.2. F и V ходят по очереди, начинает V

Далее в тексте будут ссылки на номера порядков ходов, они соответствуют их номерам в списке выше.

Можно придумать и другие варианты проведения игры, далее будет показано рассуждение, позволяющее анализировать и другие сценарии.

## 2.2 Оцениваемая величина и ее значимость

В данной работе мы не будем привязываться к конкретной функции, вместо этого мы будем говорить о стратегиях, позволяющих F побеждать при любой функции с заданным наперед количеством аргументов.

Такой подход требует разбиения всего множества булевых функций на классы, потому что если рассмат-

ривать его целиком, то никаких стратегий найти не получится ( $f(\dots) = 1$  - функция, на которой F никогда не сможет победить).

Выбор этого разбиения будет обусловлен простым рассуждением. Зафиксируем произвольный порядок игры. Пусть известно, что F имеет выигрышную стратегию (побеждает вне зависимости от решений противника) на некоторой функции  $f_1$ . Теперь рассмотрим функцию  $f_0$ , которая отличается от  $f_1$  значением только на одном наборе  $\vec{x}_0$ , причем  $f_1(\vec{x}_0) = 1$  и  $f_0(\vec{x}_0) = 0$ , иными словами мы определим  $f_0$  так:

$$f_0(\vec{x}) = \begin{cases} 0, & \vec{x} = \vec{x}_0 \\ f_1(\vec{x}), & \text{иначе} \end{cases}$$

Что будет если F будет играть на  $f_0$  также как и на  $f_1$ ? Набор  $\vec{x}_0$  никогда не получается в результате применения такой стратегии, потому что иначе нельзя было бы говорить о выигрышной стратегии на  $f_1$ , получается, что его значение не важно, и F сможет той же стратегией выигрывать и на  $f_0$ . Это рассуждение уместно выразить такой теоремой:

**Теорема** Если игрок F имеет выигрышную стратегию для каждой функции с  $m > 0$  выполняющими наборами, то он имеет стратегию и для каждой функции с  $m - 1$  и менее (вплоть до нуля) выполняющим набором вне зависимости от количества переменных и распорядка ходов.

**Доказательство.** Основное рассуждение уже было приведено выше. Остается только добавить, что всякая функция с  $m - 1$  выполняющим набором может быть получена из некоторой функции с  $m$  выполняющими наборами путем исключения одного из наборов. Окончательная версия доказываемого утверждения получается с помощью тривиальной индукции по  $m$ .

Это говорит о том, что все функции следует разделить на  $2k + 1$  группу, в каждой из которых находятся только функции с фиксированным количеством выполняющих наборов. Теперь можно поставить такую задачу: "При каком максимальном количестве выполняющих наборов F все еще имеет стратегию для победы?"

Будем обозначать искомое значение  $M$ . Оказывается, что найти его точно бывает проблематично, поэтому далее мы рассмотрим подход для построения его оценки снизу.

## 3 Базовый случай, игра на четырех переменных

Рассматриваемый далее в статье алгоритм для построения оценки является индуктивным относительно

Вариант	Оценка
1.1	4
1.2	10
1.3	5
1.4	9
2.1	3
2.2	5

Таблица 1: Результаты работы программы. Первый столбик соответствует вариантам, предложенным в 2.2. Второй столбик - максимальное количество выполняющих наборов, при котором у F все еще есть стратегия.

но количества аргументов функции, поэтому для начала следует рассмотреть базовый случай. Но если анализировать функции двух аргументов, то в игре будет всего два хода, поэтому многие из вариантов расписания ходов совпадут, что нежелательно. Поэтому рассмотрим случай функции от четырех аргументов при всех очередностях ходов, описанных в пункте 2.2. Детальное аналитическое обоснование точных оценок оказывается достаточно объемным, **читатель может ознакомиться с ним в отдельном документе**, в текущей работе приведем оценки, полученные с помощью программного перебора.

### 3.1 Краткое описание программы

При желании читатель может самостоятельно ознакомиться с написанным кодом, в нем не используются какие-либо сложные алгоритмы, задача решается полным перебором всех булевых функций от четырех аргументов.

Единственная оптимизация, которая используется в предложенной программе заключается в следующем: дерево для каждого сценария игры строится всего один раз, исходя из заданной последовательности ходов. Потом листы дерева игры (конечные состояния) заполняются значениями в соответствии с текущей функцией, после этого происходит стандартное восхождение от листьев к корню, значение корня равно 1, если на этой функции V имеет стратегию для победы и 0, если такую стратегию имеет F.

### 3.2 Результаты работы программы

К сожалению, из-за наивности реализации, программа работает порядка 10 минут для получения полного результата. Результат работы приведен в Таблице 1.

## 4 Индуктивное построение оценок

Рассмотрим показательную ситуацию. Представим, что игрок F на своем очередном ходу может придать значение только одной переменной  $x_j$ , то есть у него всего два хода  $x_j = 0$  или  $x_j = 1$ . При этом на данный момент из игры еще не исключены  $l$  выполняющих наборов.

Ключевым наблюдением заключается в том, что все выполняющие наборы делятся на две группы: те, у которых  $x_j = 0$  и те, у которых  $x_j = 1$ . Одну из этих групп F сможет исключить полностью, логично исключить ту, в которой больше наборов. При этом худшим случаем является равномерное распределение выполняющих наборов на две группы, тогда F сможет исключить  $l/2$ , если  $l$  - четное или  $(l-1)/2$ , если  $l$  - нечетное, эти два случая легко обобщить:

$$\text{Кол-во исключаемых наборов} \geq \lfloor l/2 \rfloor$$

Где  $\lfloor x \rfloor$  - целая часть числа  $x$ . Фактически, мы доказали очень важную теорему:

**Теорема** За любой свой ход, на котором он устанавливает значение хотя бы одной переменной, F сможет удалить хотя бы половину (с округлением вниз) выполняющих наборов из игры.

### 4.1 Применении теоремы для построения оценки для произвольного $k$

Напомним, что  $k$  - некоторое натуральное число,  $2k$  - количество аргументов для всех рассматриваемых функций.

Рассмотрим простой случай: ходы игроков чередуются, это случаи 1.3, 1.4, 2.1, 2.2. Пусть изначально у функции  $f$  было  $l_k$  выполняющих наборов, потом происходят два хода - ходы V и F. Будем предполагать, что V имеет идеальный ход, который сохранит все его наборы. Тогда, согласно доказанной теореме, после двух ходов количество выполняющих наборов равно:

$$l_{k-1} = \lfloor l_k/2 \rfloor$$

При этом игра свелась к  $2(k-1)$  переменным. Заметим, что для каждого  $l_{k-1}$  мы можем найти два подходящих  $l$ , при этом максимальный из них выражается формулой

$$l_k = 2l_{k-1} + 1$$

Формально, мы имеем рекуррентное соотношение, поскольку оценки для  $l_2$  уже получены. Введем замену индексов  $p = k - 2$  и получим задачу:

$$\begin{cases} l_p = 2l_{p-1} + 1 \\ l_0 = \text{const} = a \end{cases}$$

Докажем, что случаи 1.1 и 1.2 тоже можно свести к ней же.

Стратегия F предполагает какой-то выбор фиксируемых переменных из  $2(k-1)$  переменной, однако набор включает  $2k$ , пусть F отдаст одну случайную переменную и заберет себе одну случайную переменную. Свою случайную переменную он использует для реализации теоремы, сократив количество выполняющих наборов. Можно предположить, что мы отдали противнику идеальную переменную и он сможет сохранить все наборы, придав ей правильное значение. Тогда ее можно вовсе не рассматривать.

Таким образом, имеем ту же самую структуру, в которой F уменьшил количество выполняющих наборов в два раза, V играл идеально и потерял ноль, поэтому ее описывает та же рекуррентная последовательность.

## 4.2 Решение рекуррентной последовательности

Напомним задачу:

$$\begin{cases} l_{p+1} = 2l_p + 1 \\ l_0 = a \end{cases}$$

Пусть определена производящая функция:

$$L(x) = \sum_{p=0}^{\infty} l_p x^p$$

Проведем суммирование в исходном уравнении:

$$\sum_{p=0}^{\infty} l_{p+1} x^p = 2 \sum_{p=0}^{\infty} l_p x^p + \sum_{p=0}^{\infty} x^p$$

Решим его:

$$\begin{aligned} \frac{L(x) - a}{x} &= 2L(x) + \frac{1}{1-x} \\ L(x)(1-2x) &= \frac{x}{1-x} + a \\ L(x) &= \frac{1}{2} \frac{x+a-ax}{(x-1)(x-1/2)} \end{aligned}$$

Разложим на простейшие и перепишем в виде суммы:

$$\begin{aligned} L(x) &= \frac{1}{2} \left[ \frac{2}{x-1} - \frac{1+a}{x-1/2} \right] \\ &= \frac{1+a}{1-2x} - \frac{1}{1-x} \\ &= (1+a) \sum_{p=0}^{\infty} 2^p x^p - \sum_{p=0}^{\infty} x^p \\ &= \sum_{p=0}^{\infty} [(1+a)2^p - 1] x^p \end{aligned}$$

Отсюда окончательно заключаем:

$$l_p = (1+a)2^p - 1$$

## 4.3 Итоги

Полученная формула для оценок может быть использована для построения графиков, они представлены на Рис. 2.

График функции  $l_p = (a+1)2^p - 1$

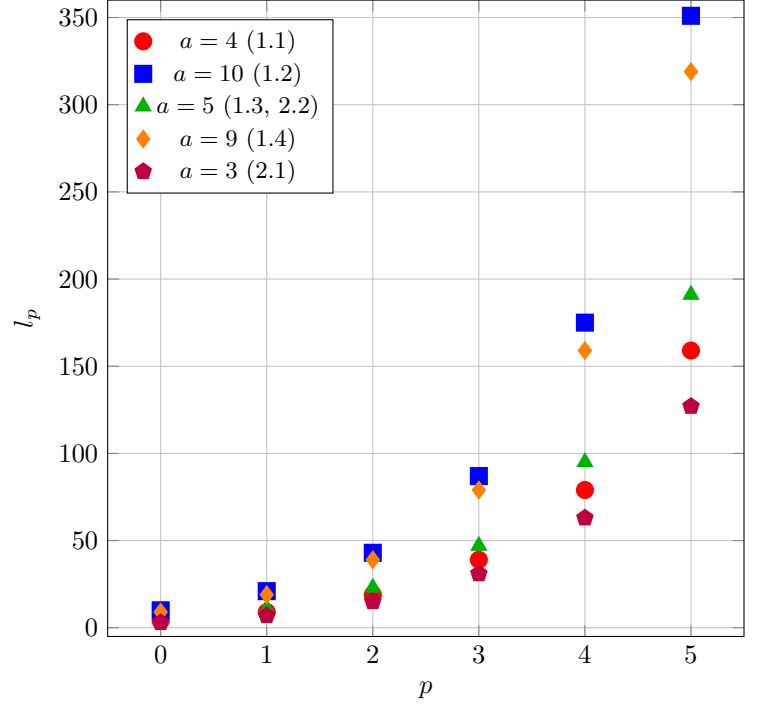


Рис. 2: График функции  $l_p = (a+1)2^p - 1$  для различных значений  $a$

## 5 Вывод

В работе представлен метод для построения оценок максимального количества выполняющих наборов функции, при которых игрок F все еще имеет стратегию для победы. Этот алгоритм был применен на нескольких конкретных примерах.

Для этого была написана программа, осуществляющая перебор, её результаты работы были подтверждены аналитически. На основании этих результатов построены соответствующие оценки.