ИПМ им. М.В. Келдыша РАН • Электронная библиотека
Математические вопросы кибернетики • Выпуск 21

И. В. Савицкий,
С. С. Марченков

Порождение малых сложностных классов с помощью логических формул, схем и операций ограниченной конкатенации

Рекомендуемая форма библиографической ссылки: Савицкий И. В., Марченков С. С. Порождение малых сложностных классов с помощью логических формул, схем и операций ограниченной конкатенации // Математические вопросы кибернетики. Вып. 21. — М.: ФИЗМАТЛИТ, 2023. — С. 52–110. URL: https://library.keldysh.ru/mvk.asp?id=2023-52 DOI: 10.20948/mvk-2023-52

ПОРОЖДЕНИЕ МАЛЫХ СЛОЖНОСТНЫХ КЛАССОВ С ПОМОЩЬЮ ЛОГИЧЕСКИХ ФОРМУЛ, СХЕМ И ОПЕРАЦИЙ ОГРАНИЧЕННОЙ КОНКАТЕНАЦИИ

И. В. САВИЦКИЙ, С. С. МАРЧЕНКОВ

(МОСКВА)

Оглавление

DOI: 10.20948/mvk-2023-52

Введение

В течение последних 70 лет в теории рекурсивных функций наблюдается процесс поиска и исследования «малых» классов рекурсивных функций (МКРФ). Внимание специалистов привлекают прежде всего классы всюду определенных рекурсивных функций, которые достаточно широко представлены в математической практике и обладают некоторыми дополнительными свойствами. Как правило, к ним относятся индуктивная определимость класса с помощью легко формулируемых функциональных, логических либо словарных операций, рекурсивная перечислимость класса и возможность использования функций класса в известных алгоритмических конструкциях. В последние годы к этим свойствам стали добавлять конечную порождаемость класса относительно операции суперпозиции. Следует отметить, что для ряда классов наличие или отсутствие последнего свойства представляет довольно серьезную проблему; ее решению для различных классов посвящены десятки статей многих авторов.

Если рассматривать порождающие операции, задающие МКРФ, то они (за исключением операции суперпозиции) либо представляют собой «ограниченные» варианты примитивной рекурсии (ограниченная рекурсия, ограниченное суммирование, ограниченное мультиплицирование, ограниченная конкатенация), либо используют логические средства с ограниченными кванторами, либо определяются на основе (равномерно) ограниченных схем из функциональных элементов. Еще один способ задания МКРФ — определение с помощью различных типов абстрактных вычислительных устройств, работающих с ограничениями на время вычислений либо объем используемой памяти.

Исторически первым МКРФ следует считать класс K функций, элементарных по Кальмару [48]. Он определяется из некоторых простейших функций с использованием операций суперпозиции, ограниченного суммирования и ограниченного мультиплицирования. Стоит отметить, что класс K имеет целый ряд эквивалентных определений [1,6,17,19,41,57], в частности, его можно определить с помощью операций суперпозиции и ограниченной конкатенации [9]. Среди первых МКРФ стоит также отметить начальные классы иерархии Гжегорчика [41], состоящей из счетной возрастающей последовательности классов $\{\mathcal{E}^n\}$ примитивно-рекурсивных функций, в объединении дающей весь класс примитивно-рекурсивных функций. Все они первоначально определены на основе операций суперпозиции и ограниченной рекурсии, но наибольший интерес представляют первые четыре класса иерархии. Отметим, что класс $\mathcal{E}^3$ совпадает с классом K.

Для практических приложений класс K (и даже класс $\mathcal{E}^2$) обычно оказывается слишком широким. Среди более узких классов наиболее известен класс P функций, вычислимых за полиномиальное время на машинах Тьюринга.

В 1970-х годах активно изучалась связь между сложностью схем из функциональных элементов и сложностью вычислений на машинах Тьюринга [30, 54]. В [54] Н. Пиппенджер выделил один из подклассов P — класс NC (название дал С. Кук от «Nick's Class» [37]), каждую функцию которого, говоря содержательно, можно параллельно вычислить на полиномиальном числе процессоров за полилогарифмическое время. Оказалось,

что удобным способом формализации параллельных вычислений являются схемы из функциональных элементов (число процессоров соответствует сложности схемы, а время вычисления — ее глубине), поэтому «схемное» определение NC стало классическим.

Класс NC естественным образом представляется в виде иерархии классов NC^k, которые определяются схемами глубины $O((\log_2 n)^k)$ со стандартными элементами $\&, \vee, \neg$. Выделяют также «промежуточные» классы AC^k (определяются «альтернирующими схемами», в которых элементы $\&, \vee$ могут иметь произвольное число входов) и TC^k (определяются «пороговыми схемами», состоящими из пороговых элементов с произвольным количеством входов).

Нас в первую очередь будут интересовать нижние классы этой иерархии AC^0 и TC^0 (класс NC^0 является вырожденным). Функции из этих классов вычислимы на полиномиальном числе процессоров за константное время; можно сказать, что их вычисление распараллеливается идеально. Эти классы действительно являются малыми и находятся близко к нижней границе по «объему» включаемых функций среди широко изучаемых сложностных классов. Однако они не являются вырожденными, содержат многие важные для практического применения функции и имеют по несколько эквивалентных определений (они будут рассмотрены в обзоре). Отметим, что пороговые схемы (и соответственно класс TC^0) вводились из соображений моделирования работы нейронных сетей [55].

Одной из важных причин интереса к классам AC^0 и TC^0 является наличие нетривиальной нижней оценки сложности $\mathrm{AC}^0 \subset \mathrm{TC}^0$, которую независимо получили М. Аджтай [24] и М. Ферст, Дж. Б. Сакс и М. Сипсер [40]. Это одно из немногих известных строгих включений сложностных классов (не считая тех, что следуют из иерархий по времени и памяти). На текущий момент неизвестно, строго ли класс TC^0 включен в класс NP, но он является одним из первых кандидатов на «отделение» от NP. Этот вопрос в последнее время активно исследуется [53].

Тонким моментом при задании функций схемами является вопрос унифицированности. Поскольку каждая функция задается целым семейством схем, требуется налагать определенные ограничения «равномерности» на их структуру. Понятие унифицированности было введено А. Бородиным в [30]. В. Л. Руццо [58] исследовал разные виды унифицированности для подклассов NC (в числе прочего, подклассы NC были охарактеризованы в терминах альтернирующих машин Тьюринга — еще одного способа формализации параллельных вычислений).

Наиболее острым вопрос выбора вида унифицированности оказывается для нижних классов иерархии AC^0 и TC^0. Для крупных классов разные виды унифицированности часто дают один и тот же результат, а малые классы при слишком мягком требовании унифицированности «пополняются» функциями, которые не должны им принадлежать при «естественном» определении. Как оказалось, предложенные в [58] виды унифицированности для классов AC^0 и TC^0 не вполне удовлетворительны.

В [44] Н. Иммерман с помощью формул логики первого порядка определил класс FO, установил его связь с некоторыми моделями параллельных вычислений и предложил использовать его в качестве унифицированной версии класса AC^0. В [27] Д. А. Баррингтон, Н. Иммерман и Г. Страубинг

обосновали этот выбор путем введения жесткого (LOGTIME) варианта унифицированности, который в применении к классу AC^0 дает класс FO. Они также ввели класс FOM для характеризации унифицированного класса TC^0 и доказали принадлежность ряда часто используемых функций классам FO и FOM.

В настоящее время LOGTIME-унифицированность считается «стандартным» видом унифицированности как для AC^0 и TC^0, так и для других «схемных» классов [25].

Описанные выше способы определения малых классов функций (в частности, классов AC^0 и TC^0) не являются индуктивными и содержат много технических деталей, которые могут порождать сомнения в «устойчивости» этих классов. Чтобы получить индуктивные определения, оказалось полезным использовать операции ограниченной конкатенации, которые позволяют описывать параллельные вычисления.

Впервые на операцию конкатенации как на «базис арифметики» обратил внимание В. Куайн [56]. Впоследствии операция конкатенации использовалась в словарных примитивных рекурсиях при определении класса примитивно рекурсивных функций и некоторых его подклассов в словарных терминах. Операция ограниченной конкатенации появилась в качестве основной порождающей операции при введении некоторых МКРФ, когда требовалось проводить рекурсию, опираясь на позиционное представление натуральных чисел и используя конкатенацию в качестве основной действующей функции.

В [47] Дж. Линд и А.Р. Мейер ввели один из слабых видов операции ограниченной конкатенации, который совместно с другими видами рекурсии позволил охарактеризовать класс функций, вычислимых на машинах Тьюринга с использованием логарифмического объема памяти. В [34] П. Клот на основе ослабления этой операции ввел операцию конкатенации по двоичному представлению (CRN) и предложил индуктивный способ определения класса AC^0, а в [35] П. Клот и Г. Такети дали индуктивное определение класса TC^0 на основе той же операции. Несмотря на использование определенной формы операции конкатенации, эти определения опираются на арифметические функции и работу с двоичными записями чисел, поэтому они не являются чисто словарными.

В [9] С.С. Марченков предложил использовать операцию ограниченной конкатенации для получения словарной характеризации класса K, что было проделано Е.А. Орловым в [19]. Позже в [12] С.С. Марченков ввел более слабую операцию ограниченной префиксной конкатенации и основанный на ней словарный класс BPC (эта операция оказалась близка к введенной в [47], но рассматривалась уже без использования других видов рекурсии). В [21] И.В. Савицкий доказал, что класс BPC совпадает с классом TC^0, тем самым показав, что класс TC^0 допускает чисто словарное описание.

Отметим, что существуют и другие способы использования конкатенации в схемах рекурсии, позволяющие, например, охарактеризовать класс P [11, 17, 29].

В работе [43] В. Гессе, Э. Аллендер и Д.А. Баррингтон доказали ряд нетривиальных результатов о принадлежности арифметических функций (в частности, функции $\lfloor x/y \rfloor$) классу TC^0, при этом использовалось его логическое определение FOM.

Немного позже С. А. Волков [3–5], используя этот результат, получил простой конечный базис по суперпозиции в классе FOM. Этот базис, наряду с базисом в классе K [10], является одним из «лучших» базисов по суперпозиции, известных для широко исследуемых классов функций.

Результаты С. А. Волкова развил С. Маззанти, построив базисы по суперпозиции в классе AC^0 [49, 51, 52]. Другие базисы в TC^0 были построены С. С. Марченковым в [13, 15, 16] на основе определения BPC и некоторых его модификаций (на тот момент еще не было известно, что $BPC = TC^0$).

§ 1. Задание классов с помощью логических систем первого порядка

В этом параграфе даны определения классов FO и FOM, приведены технические приемы работы с ними и результаты о принадлежности некоторых функций. Предикаты класса FO задаются с помощью формул логики предикатов первого порядка, в семантике которых переменные пробегают номера разрядов в двоичной записи входа, а в качестве исходных предикатов присутствуют сравнение переменных и получение бита двоичной записи значения переменной. Класс FOM получается добавлением в логическую систему дополнительного мажоритарного квантора.

Классы FO и FOM имеют ряд эквивалентных определений (например, «схемные» определения AC^0 и TC^0). Определения FO и FOM можно назвать центральными: они являются «связующим звеном» между другими определениями и именно через формализм FO и FOM доказана принадлежность многих функций данным классам.

Класс FO был введен в [44], а основная техника работы с ним — в [27]. Класс FOM введен в [27]. Ряд результатов, требующих наиболее сложной техники, получен в [43].

1.1. Классы FO и FOM. Через $|X|$ будем обозначать длину слова X в произвольном алфавите. Будем рассматривать предикаты $\rho(X_1, \ldots, X_n;\ y_1, \ldots, y_m)$, определенные на множестве

$$\bigcup_{\substack{X_1,\ldots,X_n\in\{0,1\}^*\\ |X_1|=\ldots=|X_n|\geqslant 2}} \{(X_1, \ldots, X_n)\} \times \{0, \ldots, q-1\}^m,$$

где $\{0,1\}^*$ — множество всех слов в алфавите $\{0,1\}$ и $q = |X_1|$. Эти предикаты зависят от переменных двух типов: «длинных» словарных переменных $X_1, \ldots, X_n$, принимающих в качестве значений произвольные слова в алфавите $\{0,1\}$ одинаковой длины не менее 2, и «коротких» числовых переменных $y_1, \ldots, y_m$, которые принимают значения от 0 до $q-1$.

Далее будем использовать $\bar{X}$ в качестве сокращения для $(X_1, \ldots, X_n)$. Через $|\bar{X}|$ будем обозначать длину переменных X_i (которая одинакова у всех переменных).

Элементарными FO*-формулами* будем называть следующие выражения:

1. $X_i(y)$, где X_i — «длинная» переменная, а y — «короткая» переменная.

   Эта формула задает элементарный предикат $\rho(\bar{X};\ y)$, истинный в том и только том случае, если y-й символ слова X_i является единицей (нумерация слева направо, самый левый символ имеет номер 0).

2. $\mathrm{BIT}(x, y)$, где x, y — «короткие» переменные.

   Эта формула задает элементарный предикат $\rho(\bar{X};\ x, y)$, истинный в том и только том случае, если y-й разряд двоичной записи числа x является единицей (нумерация разрядов справа налево, младший разряд имеет номер 0).

3. $(x \leqslant y)$, где x, y — «короткие» переменные.

   Эта формула задает элементарный предикат $\rho(\bar{X};\ x, y)$, истинный в том и только том случае, если число x не превосходит числа y.

FO-*формула* получается из элементарных FO-формул с помощью операций логики высказываний $\&, \vee, \neg$, а также с помощью навешивания кванторов существования $(\exists x)$ и общности $(\forall x)$ по «коротким» переменным.

FOM-*формула* получается из элементарных FO-формул с помощью тех же операций и кванторов, а также дополнительно навешивания мажоритарного квантора (Mx) по «короткой» переменной.

Каждая FO- и FOM-формула со свободными «короткими» переменными $y_1, \ldots, y_m$ задает предикат $\rho(\bar{X};\ y_1, \ldots, y_m)$. Этот предикат получается из элементарных предикатов в соответствии с общепринятой семантикой операций логики высказываний и навешивания кванторов существования и общности. При этом считаем, что кванторы существования и общности перебирают числа из множества $\{0, \ldots, |\bar{X}| - 1\}$.

Мажоритарный квантор M применяется к предикату следующим образом. Предикат $(Mx)\rho(\bar{X};\ y_1, \ldots, y_m, x)$ принимает значение истина в том и только том случае, если при данных значениях $\bar{X}, y_1, \ldots, y_m$ предикат $\rho(\bar{X};\ y_1, \ldots, y_m, x)$ принимает значение истина на более чем половине всех значений x из $\{0, \ldots, |\bar{X}| - 1\}$.

FO-*определимыми* называем предикаты, которые можно задать FO-формулой. Аналогично вводятся FOM-*определимые* предикаты.

Пусть $\mathbb{N} = \{1, 2, \ldots\}$, $\mathbb{N}_0 = \mathbb{N} \cup \{0\}$, а $\mathrm{len}(x) = \lfloor \log_2 x \rfloor + 1$ обозначает длину двоичного представления числа x (длина числа 0 равна 0).

С помощью FO-формул без свободных «коротких» переменных можно задавать предикаты на множестве $\mathbb{N}_0^n$, если подавать входы в двоичной записи на места «длинных» переменных. Более подробно, при $q \geqslant \mathrm{len}(x)$ обозначим через $\mathrm{bin}^q(x)$ слово в алфавите $\{0, 1\}$ длины q, являющееся двоичной записью числа x (с необходимым количеством незначащих нулей).

Класс FO_* определим как множество всех предикатов $\rho(x_1, \ldots, x_n)$: $\mathbb{N}_0^n \to \{\mathrm{true}, \mathrm{false}\}$, таких, что существует FO-определимый предикат $R(X_1, \ldots, X_n)$, зависящий только от «длинных» переменных, для которого

$$\rho(x_1, \ldots, x_n) \equiv R(\mathrm{bin}^q(x_1), \ldots, \mathrm{bin}^q(x_n))$$

при всех $q \geqslant \max(\mathrm{len}(x_1), \ldots, \mathrm{len}(x_n), 2)$. Класс FOM_* определяется аналогично.

Функции натурального аргумента будем задавать побитово с наложением условия полиномиального роста. А именно, класс FO определяем как

множество функций натурального аргумента $f(x_1, \ldots, x_n)$: $\mathbb{N}_0^n \to \mathbb{N}_0$, для которых выполнены два условия:

1. Существует константа $k \geqslant 1$ такая, что $\mathrm{len}(f(x_1, \ldots, x_n)) \leqslant s^k$, где $s = \max(\mathrm{len}(x_1), \ldots, \mathrm{len}(x_n), 2)$.
2. Предикат $\mathrm{BIT}(f(x_1, \ldots, x_n), y)$ принадлежит FO_*.

Класс FOM определяется аналогично.

1.2. Полиномиальные переменные. При работе с классами FO и FOM нередко требуется пользоваться «короткими» переменными, значения которых могут превосходить длину входа $|\bar{X}|$. При $k \geqslant 1$ будем называть k-переменными (полиномиальными переменными) переменные, принимающие значения из множества $\{0, \ldots, |\bar{X}|^k - 1\}$.

Обобщим понятия FO- и FOM-определимости. Предикат $\rho(\bar{X};\ y_1, \ldots, y_m)$ с k-переменными $y_1, \ldots, y_m$, т.е. имеющий область определения

$$\bigcup_{\substack{X_1, \ldots, X_n \in \{0,1\}^* \\ |X_1| = \ldots = |X_n| \geqslant 2}} \{(X_1, \ldots, X_n)\} \times \{0, \ldots, l-1\}^m,$$

где $l = |\bar{X}|^k$, будем называть k-FO-*определимым*, если он задается FO-формулой, в семантике которой все кванторы перебирают числа из множества $\{0, \ldots, |\bar{X}|^k - 1\}$. Аналогично вводятся k-FOM-*определимые* предикаты.

Легко видеть, что при $k = 1$ указанные определения дают FO- и FOM-определимые предикаты. Позже мы покажем, что классы FO и FOM не изменятся, если вместо FO- и FOM-определимых предикатов использовать k-FO и k-FOM-определимые предикаты с произвольным $k \geqslant 1$ (утверждение 2). Предварительно введем необходимую технику.

Пусть предикат $\rho(\bar{X};\ y_1, \ldots, y_m)$ не зависит от k и имеет область определения

$$\bigcup_{\substack{X_1, \ldots, X_n \in \{0,1\}^* \\ |X_1| = \ldots = |X_n| \geqslant 2}} \{(X_1, \ldots, X_n)\} \times \mathbb{N}_0^m.$$

Будем называть этот предикат k-FO-определимым при любом k, если при каждом k существует k-FO-определимый предикат, совпадающий с ρ на своей области определения. Аналогичную фразу будем использовать и в связи с k-FOM-определимостью.

1.3. Некоторые FO- и FOM-определимые предикаты. Введем функцию усеченной разности $x \dot{-} y = \max(x - y, 0)$ и функцию $\mathrm{sg}(x) = \min(x, 1)$. Функцию $\lfloor x/y \rfloor$ доопределяем нулем при $y = 0$. Через $\mathrm{rm}(x, y)$ обозначаем остаток от деления x на y (0 при $y = 0$). При использовании функции возведения в степень x^y считаем, что $0^0 = 1$.

Будем использовать в формулах операции логики высказываний $\to, \sim, \oplus$. Они выразимы с помощью исходных операций $\&, \vee, \neg$.

Ниже показана k-FO-определимость ряда предикатов при любом k. Отметим, что эти построения не являются доказательствами принадлежности каких-либо предикатов классу FO_*: для этого нужно было бы выражать предикаты от «длинных» переменных.

Сравнения k-переменных $=, \neq, <, >, \geqslant$ выразимы через сравнение $\leqslant$ с помощью операций логики высказываний.

График сложения k-переменных $(z = x + y)$ реализуется с помощью моделирования сложения двоичных записей чисел «в столбик» [27]. Пусть $\mathrm{CARRY}(x, y, i)$ истинно в том и только том случае, если при сложении x и y есть перенос в разряд i. Тогда

$$\mathrm{CARRY}(x, y, i) \equiv (\exists j)[(j < i) \,\&\, \mathrm{BIT}(x, j) \,\&\, \mathrm{BIT}(y, j) \,\& \\ \& \,(\forall t)[(j < t < i) \to (\mathrm{BIT}(x, j) \vee \mathrm{BIT}(y, j))]],$$

$$(z = x + y) \equiv (\forall i)[\mathrm{BIT}(z, i) \sim (\mathrm{BIT}(x, i) \oplus \mathrm{BIT}(y, i) \oplus \mathrm{CARRY}(x, y, i))].$$

Здесь и далее предполагается, что для кванторов по k-переменным соответствующие интервалы значений переменных всегда определяются подходящими значениями $|\bar{X}|^k$.

Графики констант:

$$(x = 0) \equiv (\forall i)\neg\mathrm{BIT}(x, i),$$
$$(x = 1) \equiv (\forall i)[\mathrm{BIT}(x, i) \sim (i = 0)],$$
$$(x = a) \equiv (\exists z)[(z = a - 1) \,\&\, (x = z + 1)],$$

где $(x = z + 1) \equiv (\exists v)[(v = 1) \,\&\, (x = z + v)]$.

График логарифма вычисляется с помощью проверки старшего бита:

$$(y = \lfloor \log_2 x \rfloor) \equiv (\mathrm{BIT}(x, y) \,\&\, (\forall i)[(i > y) \to \neg\mathrm{BIT}(x, i)]) \vee ((x = 0) \,\&\, (y = 0)).$$

Тогда можно вычислить *график длины двоичной записи числа*:

$$(y = \mathrm{len}(x)) \equiv ((x \neq 0) \,\&\, (\exists z)[(z = \lfloor \log_2 x \rfloor) \,\&\, (y = z + 1)]) \vee ((x = 0) \,\&\, (y = 0)).$$

Графики усеченной разности и умножений на константы определяются с помощью сложения:

$$(z = x \dot{-} y) \equiv (x = z + y) \vee ((x < y) \,\&\, (z = 0)),$$
$$(y = 2x) \equiv (y = x + x),$$
$$(y = ax) \equiv (\exists z)[(z = (a - 1)x) \,\&\, (y = z + x)].$$

Можно сравнить значение k-переменной с $|\bar{X}|^k - 1$ следующим образом:

$$(x = |\bar{X}|^k - 1) \equiv (\forall y)(y \leqslant x).$$

Если для некоторой функции f уже получен предикат $(y = f(\bar{X};\ x_1, \ldots, x_n))$, то

$$(y < f(\bar{X};\ x_1, \ldots, x_n)) \equiv (\forall z)[(z = f(\bar{X};\ x_1, \ldots, x_n)) \to (y < z)].$$

Остальные сравнения y и $f(\bar{X};\ x_1, \ldots, x_n)$ выражаются с помощью логических операций.

1.4. Суммирование в классах FO и FOM. Через $\mathrm{bitsum}(x)$ обозначим количество единиц в двоичной записи числа x.

Лемма 1 [27]. *Предикат* $(y = \mathrm{bitsum}(x))$ k-FO*-определим при любом* k.

Доказательство. Обозначим $L = \lfloor \log_2(|\bar{X}|^k) \rfloor$, $L_2 = \lfloor \log_2 L \rfloor$. Будем вычислять $\mathrm{bitsum}(x)$ путем генерации последовательности частичных сумм. Последовательность кодируется набором битов: на каждое число выделяется L_2 битов и соединение двоичных записей чисел дает код последовательности. Длина кода последовательности должна не превосходить L, чтобы код можно было хранить в k-переменной.

Все возможные последовательности такого вида можно перебирать кванторами. При этом требуется проверять, что каждый следующий элемент последовательности является суммой предыдущего элемента и количества единиц в очередном блоке записи числа x.

Для организации процесса проверки нужно генерировать еще две последовательности: одна последовательность указывает биты начала двоичной записи каждого числа и позволяет выделять отдельные числа из основной последовательности. Вторая последовательность хранит номера первых разрядов блоков x и позволяет получать доступ к содержимому этих блоков при проверке корректности основной последовательности.

Процесс вычисления $\mathrm{bitsum}(x)$ проходит в два этапа. Сначала реализуется подсчет единиц в L_2 подряд идущих битах x. Для этого генерируется последовательность из L_2 частичных сумм, а биты x суммируются по одному (при $(L_2)^2 > L$, т. е. $L < 9$, предикат реализуется таблично). На втором этапе подсчитываются единицы во всем числе x. Для этого генерируется последовательность из $\lfloor L/L_2 \rfloor$ частичных сумм, а биты x суммируются блоками по L_2 бита (подсчет числа единиц блока реализован на первом этапе).

Следующая лемма позволяет производить в FO суммирование полилогарифмического от длины входа количества чисел. Набросок доказательства (для $m = 1$) дан в [27] (см. также [46]). Используемый алгоритм более подробно описан в [32], но без адаптации к классу FO.

Лемма 2 [27]. *Пусть* $m \geqslant 1$, *значение функции* $h(\bar{X};\ i) < 2^{|\bar{X}|^k}$ *при* $0 \leqslant i \leqslant \lfloor \log_2(|\bar{X}|^k) \rfloor^m$, *а предикат* $\mathrm{BIT}(h(\bar{X};\ i),\ j)$ k-FO*-определим. Тогда предикат*

$$S(\bar{X};\ j) \equiv \mathrm{BIT}\left(\sum_{i=0}^{\lfloor \log_2(|\bar{X}|^k) \rfloor^m} h(\bar{X},\ i),\quad j \right)$$

k-FO*-определим.*

Доказательство. Сначала рассмотрим случай $m = 1$. Основной техникой является преобразование набора чисел с помощью применения операции $\mathrm{bitsum}(x)$, в результате чего количество суммируемых чисел уменьшается до логарифма от исходного количества.

Пусть $n = |X|^k$, $l = \lfloor \log_2 n \rfloor$ и суммируются числа $x_i = (x_{in} \ldots x_{i1})_2$, $i = \overline{1, l}$. С помощью $\mathrm{bitsum}(x)$ вычислим значения $y_j = \sum\limits_{i=1}^{l} x_{ij}$, $j = \overline{1, n}$. Каждое число y_j имеет двоичную запись $(y_{jl_2} \ldots y_{j1})_2$, где $l_2 = \mathrm{len}(l)$. Перегруппируем биты y_j и сформируем числа

$$y'_1 = (y_{n1} \ldots y_{11})_2, \quad y'_2 = (y_{n2} \ldots y_{12} 0)_2, \quad \ldots, \quad y'_{l_2} = (y_{nl_2} \ldots y_{1l_2} \underbrace{0 \ldots 0}_{l_2 - 1})_2.$$

Сумма чисел $y'_1, \ldots y'_{l_2}$ равна искомой сумме, количество этих чисел $l_2 = \lfloor \log_2 l \rfloor + 1$, а их длины не превосходят $n + l$. Кроме того, для вычисления каждого бита каждого из этих чисел требуется доступ только к l битам исходных чисел.

Повторяя эту процедуру некоторое число l^* раз, можно уменьшить количество суммируемых чисел до двух. Для вычисления каждого бита этих двух чисел требуется знать не более $l_2^{l^*}$ битов чисел $y'_1, \ldots, y'_{l_2}$, полученных на первом шаге (на каждом следующем шаге требуется хранить еще меньше информации).

Число l^* можно оценить как $O(\log_2^* n)$, где функция сверхлогарифма $\log_2^* n$ имеет очень медленный рост и определяется соотношением $\lfloor \underbrace{\log_2 \ldots \log_2 n}_{\log_2^* n} \rfloor = 1$.

Для реализации в классе FO итеративного процесса генерируется последовательность промежуточных результатов этого процесса с помощью перебора кванторами. Последовательность имеет $O(\log_2^* n)$ элементов двоичной длины $l_2^{O(\log_2^* n)}$, поэтому для хранения требует $o(\log_2 n)$ битов и умещается в k-переменной (при малых n предикат можно доопределить таблично).

Полученные в итоге два числа складываются «в столбик» с помощью техники, которая демонстрировалась ранее. Отметим, что превышение промежуточными числами длины n не имеет значения, т.к. для вычисления разрядов $0, \ldots, n-1$ результата не требуется доступа к старшим битам промежуточных чисел.

В общем случае лемма доказывается индукцией по m. Если уже определено суммирование l^{m-1} чисел, то для подсчета суммы l^m чисел нужно разбить их на l групп по l^{m-1}, подсчитать сумму в каждой группе, а затем просуммировать получившиеся l чисел.

Обозначим через $\sim i$ число, которое равно $(|\bar{X}| - 1) - i$, если $i < |\bar{X}|$, и $|\bar{X}|^k - 1$ в ином случае (здесь i — k-переменная). Нетрудно видеть, что $X(\sim i) \equiv \mathrm{BIT}(x, i)$, где X является двоичной записью x (возможно, с незначащими нулями). В этом случае также выполняется $X(i) \equiv \mathrm{BIT}(x, \sim i)$.

Через $(x = X)$ обозначим предикат, проверяющий на равенство значение k-переменной x и числа, двоичной записью которого является значение «длинной» переменной X.

Следствие 1 [27]. *Предикаты* $(z = xy)$, $(z = \lfloor x/y \rfloor)$, $(z = \mathrm{rm}(x, y))$, $(y = x^a)$, $(y = \lfloor \sqrt[a]{x} \rfloor)$, *где* a — *натуральное число*, $(z = \sim i)$ *и* $(x = X)$ k-FO-*определимы при любом* k.

Доказательство. Умножение чисел сводится к суммированию, причем количество суммируемых чисел есть длина k-переменной $\lfloor \log_2(|\bar{X}|^k) \rfloor$. Возведение в степень a получается многократным умножением. Деление и вычисление корня получаются как обратные функции к умножению и возведению в степень с помощью перебора значений кванторами. Остаток от деления получается при помощи деления и умножения. Значение $\sim i$ можно получить с помощью извлечения корня из $|\bar{X}|^k - 1$. Наконец,

$$(x = X) \equiv (\forall i)[\mathrm{BIT}(x, i) \sim X(\sim i)].$$

В классе FOM можно производить суммирование чисел, количество которых полиномиально от длины входа.

Утверждение 1 [27]. *Пусть значение функции* $h(\bar{X};\ i) < 2^{|\bar{X}|^k}$ *при* $0 \leqslant i < |\bar{X}|^k$, *а предикат* $\mathrm{BIT}(h(\bar{X};\ i),\ j)$ k-FOM-*определим. Тогда предикат*

$$S(\bar{X};\ j) \equiv \mathrm{BIT}\left(\sum_{i=0}^{|\bar{X}|^k-1} h(\bar{X},\ i),\quad j\right)$$

k-FOM-*определим.*

Доказательство. Пусть $n = |\bar{X}|^k$. Вначале покажем, как подсчитать число BSUM_R истинных значений произвольного k-FOM-определимого предиката $R(z)$. С помощью квантора M нетрудно определить квантор H, который проверяет, что количество истинных значений R равно $\lfloor n/2 \rfloor$: $(Hx)R(x) \equiv \neg(Mz)R(z)\,\&\,(\exists y)(Mz)[R(z) \vee (z = y)]$. Далее подсчитываем число истинных значений предиката при $x < \lfloor n/2 \rfloor$:

$$(x = \mathrm{BSUM}_{R,\,<\lfloor n/2 \rfloor}) \equiv (Hz)[((z < \lfloor n/2 \rfloor)\,\&\,R(z)) \vee (\lceil n/2 \rceil \leqslant z < n - x)].$$

Аналогично подсчитывается число истинных значений при $x > \lfloor n/2 \rfloor$, и с помощью сложения этих значений и проверки $R(\lfloor n/2 \rfloor)$ выражается предикат $(x = \mathrm{BSUM}_R)$.

Производим суммирование по тому же алгоритму, что и в лемме 2. На первом шаге для подсчета битовых сумм используем только что построенный предикат $(x = \mathrm{BSUM}_R)$. Ко второму шагу количество суммируемых чисел становится равно $\lfloor \log_2(|\bar{X}|^k - 1) \rfloor + 1$, и каждый бит их суммы можно выразить в FO по лемме 2.

1.5. Альтернативный способ задания функций. По определению функции из классов FO и FOM задаются через FO- (FOM)-определимые предикаты вида $\mathrm{BIT}(f(\bar{X}),\ Y)$. Переменная Y, содержащая номер бита, является «длинной», и работа с ней затруднена. Мы покажем, что можно изменить способ задания функций, заменив переменную Y на «короткую» переменную y. Но, поскольку значение функции может превышать длину входа, для этого потребуется использовать k-переменные и k-FO- (k-FOM)-определимые предикаты.

Следующее утверждение показывает, что классы FO и FOM не меняются, если использовать k-FO- (k-FOM)-определимые предикаты вместо FO- (FOM)-определимых.

Утверждение 2. *Пусть* $k, k' \in \mathbb{N}$, *и предикат* $\rho'(\bar{X};\ y_1, \ldots, y_m)$ k'-FO-*определим. Тогда существует* k-FO-*определимый предикат* $\rho(\bar{X};\ y_1, \ldots, y_m)$, *который совпадает с* ρ' *на области, где оба предиката определены. Аналогичное утверждение верно для* k-FOM-*определимости.*

Доказательство. Пусть Φ' — исходная FO-формула, k'-определяющая предикат ρ'. Будем строить формулу Φ для k-определения предиката ρ.

В случае $k \geqslant k'$ достаточно минимальной перестройки формулы: при навешивании каждого квантора (Qx), $Q \in \{\exists, \forall, M\}$, нужно добавить ограничение $(x < |\bar{X}|^{k'})$. Это ограничение выразимо с помощью предикатов $(x = |\bar{X}|^k - 1)$, $(y = \sqrt[k]{x})$, $(y = x^{k'})$.

Предположим теперь, что $k < k'$. Пусть $l = \lfloor \log_2(|\bar{X}|^k - 1) \rfloor$ и $N = 2^l$ — максимальная степень двойки, меньшая $|\bar{X}|^k$. Эти числа нетрудно вы-

числить:

$$(x = l) \equiv (\exists y)[(y = |\bar{X}|^k - 1)\ \&\ (x = \lfloor \log_2 y \rfloor)],$$
$$(x = N) \equiv (\exists y)[(y = l)\ \&\ \mathrm{BIT}(x, y)\ \&\ (\forall j)[(j \neq y) \to \neg \mathrm{BIT}(x, j)]].$$

Выберем натуральное число t (зависящее только от k, k') такое, что $N^t \geqslant |\bar{X}|^{k'}$. Заменим в формуле Φ' свободные k'-переменные $y_1, \ldots, y_m$ на одноименные k-переменные. Каждую из остальных k'-переменных u заменим на набор k-переменных $(u_1 \ldots u_t)$. Будем считать, что в переменных $u_1, \ldots, u_t$ находятся цифры значения u в позиционной системе счисления с основанием N (в порядке возрастания старшинства).

В общих чертах опишем преобразование формулы при указанной замене. Отметим, что в качестве аргументов у элементарных предикатов могут присутствовать как k'-переменные вида $(u_1 \ldots u_t)$ (связанные кванторами в Φ'), так и k-переменные $y_1, \ldots, y_m$.

Предикат $\mathrm{BIT}((u_1 \ldots u_t), (v_1 \ldots v_t))$ реализуется вычислением $\mathrm{BIT}(u_i, v_1 - (i-1)l)$ при нужном i. Поиск i реализуется сравнением v_1 с числами вида il при $i \in \{1, \ldots, t-1\}$. Длина двоичной записи числа $(u_1 \ldots u_t)$ не превосходит lt, поэтому при $lt < N$ номера любых битов k'-переменной $(u_1 \ldots u_t)$ «помещаются» в v_1. При малых $|\bar{X}|$, когда возможен случай $lt \geqslant N$, предикат реализуется таблично.

Предикат $\mathrm{BIT}((u_1 \ldots u_t), x)$ реализуется аналогично, только вместо v_1 используется k-переменная x. Предикат $\mathrm{BIT}(x, (u_1 \ldots u_t))$ реализуется формулой $\mathrm{BIT}(x, u_1)$. Предикат $((u_1 \ldots u_t) \leqslant (v_1 \ldots v_t))$ реализуется через последовательную проверку $u_i \leqslant v_i$, $i = \overline{t, 1}$.

С использованием построенного выше предиката $\mathrm{BIT}((u_1 \ldots u_t), y)$ для k'-переменной и имеющегося изначально предиката $\mathrm{BIT}(x, y)$ для k-переменной можно побитово сравнить значения k- и k'-переменной и реализовать предикаты $(x = (u_1 \ldots u_t))$ и $(x \leqslant (u_1 \ldots u_t))$ (при $lt > N$ предикаты реализуются таблично).

Предикат $X_i((u_1 \ldots u_t))$ реализуется как $(\exists z)[(z = (u_1 \ldots u_t))\ \&\ X_i(z)]$. Структура логических связок при перестроении формулы Φ' не меняется. Если подформула $\Psi'(u)$ перестроена в $\Psi(u_1, \ldots, u_t)$, то $(\exists u)\Psi'(u)$ перестраивается в

$$(\exists u_1) \ldots (\exists u_t)[(u_1, \ldots, u_t < N)\ \&\ \Psi(u_1, \ldots, u_t)].$$

Квантор общности рассматривается аналогично.

Осталось рассмотреть навешивание мажоритарного квантора. Пусть подформула $\Psi'(u)$ была перестроена в $\Psi(u_1, \ldots, u_t)$. Добавим ограничение $(u_1, \ldots, u_t < N)$ и вычислим количество истинных значений полученной формулы путем t-кратного применения операции суммирования из утверждения 1 (на первом шаге суммируются нули и единицы).

Полученную сумму сравниваем с $\lfloor |\bar{X}|^{k'}/2 \rfloor$ (сравниваемые числа не помещаются в k-переменные, и доступ к ним производится по отдельным разрядам двоичной записи).

Теперь можно сформулировать эквивалентные определения классов FO и FOM. Напомним, что $\mathrm{bin}^q(x)$ — это слово в алфавите $\{0, 1\}$ длины q, являющееся двоичной записью числа x (с нужным количеством незначащих нулей).

Утверждение 3. *Пусть функция* $f(x_1, \ldots, x_n)$ *и константа* k *таковы, что*

$$\mathrm{len}(f(x_1, \ldots, x_n)) \leqslant s^k, \quad s = \max(\mathrm{len}(x_1), \ldots, \mathrm{len}(x_n), 2).$$

Тогда $f \in$ FO *в том и только том случае, если существует* k-FO*-определимый предикат* $F(\bar{X};\ y)$ *такой, что при всех* $q \geqslant s$

$$\mathrm{BIT}(f(x_1, \ldots, x_n),\ y) \equiv F(\mathrm{bin}^q(x_1), \ldots, \mathrm{bin}^q(x_m);\ y).$$

Аналогичное утверждение верно для класса FOM.

Доказательство. Через $F'(\bar{X}, Y)$ обозначим FO-определимый предикат, задающий предикат $\mathrm{BIT}(f(x_1, \ldots, x_n), y)$ в соответствии с определением FO.

Пусть существует предикат F. Покажем, что с помощью него можно построить предикат F'. Благодаря утверждению 2 достаточно построить k-FO-определимый предикат

$$F'(\bar{X},\ Y) \equiv (\exists y)[(y = Y)\ \&\ F(\bar{X};\ y)].$$

В силу выбора константы k, если значение Y не умещается в k-переменную y, то предикат F' заведомо ложен (построенная формула этому отвечает).

Обратно, пусть существует FO-определимый предикат F'. Чтобы получить предикат F, нужно в формуле, выражающей F', заменить все элементарные формулы $Y(i)$ на выражения $\mathrm{BIT}(y, \sim i)$. Для случая FOM построения аналогичны.

1.6. Функции из классов FO и FOM. *Характеристической функцией* произвольного предиката $\rho(x_1, \ldots, x_n)$ называем функцию

$$\chi_\rho(x_1, \ldots, x_n) = \begin{cases} 0, & \neg\rho(x_1, \ldots, x_n), \\ 1, & \rho(x_1, \ldots, x_n). \end{cases}$$

Нетрудно видеть что FO_* есть класс предикатов, характеристические функции которых лежат в FO. Аналогичное верно и для FOM. Кроме того, верно включение $\mathrm{FO} \subseteq \mathrm{FOM}$.

Операция суперпозиции включает в себя подстановку функций вместо переменных, перестановку и отождествление переменных, введение и удаление фиктивных переменных.

Утверждение 4. *Классы* FO *и* FOM *замкнуты относительно операции суперпозиции.*

Доказательство. Замкнутость классов относительно перестановки и отождествления переменных, а также введения и удаления фиктивных переменных очевидна.

Пусть $\bar{x} = (x_1, \ldots, x_n)$ и $f(\bar{x}) = g_0(g_1(\bar{x}), \ldots, g_m(\bar{x}))$, где $g_0, \ldots, g_m \in \mathrm{FO}$. В силу ограниченности роста функций $g_0, \ldots, g_m$ существует общая константа k такая, что

$$\mathrm{len}((g_1(\bar{x})),\ \ldots,\ \mathrm{len}((g_m(\bar{x})) \leqslant s^k, \qquad s = \max(\mathrm{len}(x_1), \ldots, \mathrm{len}(x_n), 2),$$
$$\mathrm{len}(g_0(y_1, \ldots, y_m)) \leqslant t^k, \qquad t = \max(\mathrm{len}(y_1), \ldots, \mathrm{len}(y_m), 2).$$

Пусть $G_0(Y_1, \ldots, Y_m;\ z)$, $G_1(\bar{X};\ y)$, $\ldots$, $G_m(\bar{X};\ y)$ — k-FO-определимые предикаты, задающие по утверждению 3 функции $g_0, g_1, \ldots, g_m$ соответственно. Для построения k^2-FO-определимого предиката $F(\bar{X};\ z)$, задающего функцию f, достаточно в формуле предиката G_0 заменить все вхождения $Y_i(t)$ на $G_i(\bar{X};\ \sim t)$, $i = \overline{1, m}$. Для класса FOM построения аналогичны.

Пользуясь утверждением 3 и предикатом $(x = X)$, можно получить функциональные аналоги леммы 2 и утверждения 1.

Утверждение 5 [27].

1. *Класс* FO *замкнут относительно операции ограниченного суммирования вида*
$$\sum_{z=0}^{\lfloor \log_2 \log_2 y \rfloor^m} g(x_1, \ldots, x_n, z), \quad m \in \mathbb{N}.$$
2. *Класс* FOM *замкнут относительно операции ограниченного суммирования вида*
$$\sum_{z=0}^{\lfloor \log_2 y \rfloor} g(x_1, \ldots, x_n, z).$$

Пусть $x \wedge y$ обозначает число, двоичная запись которого является побитовой конъюнкцией двоичных записей x и y (выравненных незначащими нулями до одинаковой длины), а $\mathrm{bit}(x, y)$ есть характеристическая функция предиката $\mathrm{BIT}(x, y)$.

Селекторными функциями назовем функции $I_i^n(x_1, \ldots, x_n) = x_i$, $i = \overline{1, n}$, $n \in \mathbb{N}$.

Утверждение 6. *Константы, селекторные функции и функции* $x + y$, $x \dot{-} y$, $\mathrm{bit}(x, y)$, $x \wedge y$, $\lfloor \log_2 x \rfloor$, $2^{\lfloor \log_2 x \rfloor \cdot \lfloor \log_2 y \rfloor}$ *входят в* FO. *Функции* xy *и* $\mathrm{bitsum}(x)$ *входят в* FOM.

Доказательство. Функции $x + y$ и $x \dot{-} y$ получаются с помощью сложения (вычитания) «в столбик» аналогично тому, как определялся предикат $(z = x + y)$ от «коротких» переменных. Константа 0 определяется тождественно ложным предикатом, а константа 1 — предикатом $R(X;\ y) \equiv (y = 0)$. Остальные константы получаются с помощью последовательного прибавления единицы.

Селекторная функция $I_i^n(x_1, \ldots, x_n) = x_i$ определяется предикатом $I_i(\bar{X};\ y) \equiv X_i(\sim y)$. Функция $x \wedge y$ задается предикатом $A(X, Y;\ z) \equiv X(z)\ \&\ Y(z)$. Предикат $\mathrm{BIT}(x, y)$ определяется формулой $B(X, Y) \equiv (\exists y)[(y = Y)\ \&\ X(\sim y)]$. Пусть

$$(x = \lfloor \log_2 X \rfloor) \equiv X(\sim x)\ \&\ (\forall z)[(z > x) \to \neg X(\sim z)].$$

Функция $\lfloor \log_2 x \rfloor$ задается предикатом

$$L(X;\ y) \equiv (\exists x)[(x = \lfloor \log_2 X \rfloor)\ \&\ \mathrm{BIT}(x, y)].$$

Функция $2^{\lfloor \log_2 x \rfloor \cdot \lfloor \log_2 y \rfloor}$ (с учетом следствия 1) 2-FO-определяется предикатом

$$P(X, Y;\ z) \equiv (\exists x)(\exists y)[(x = \lfloor \log_2 X \rfloor)\ \&\ (y = \lfloor \log_2 Y \rfloor)\ \&\ (z = xy)].$$

С помощью аналогичной техники работы с двоичными записями чисел можно определить функции $x \cdot 2^{\min(y,\ \lfloor \log_2 z \rfloor)}$, $x \cdot \mathrm{sg}(y) \in \mathrm{FO}$. Тогда с учетом равенства $\lfloor \log_2 x \rfloor + 1 = \lfloor \log_2 (x + x) \rfloor$ умножение и битовая сумма выражаются

в FOM с помощью суммирования:

$$xy = \sum_{z=0}^{\lfloor \log_2 y \rfloor + 1} x \cdot 2^{\min(z,\, \lfloor \log_2 y \rfloor + 1)} \cdot \mathrm{sg}(\mathrm{bit}(y, z)), \quad \mathrm{bitsum}(x) = \sum_{z=0}^{\lfloor \log_2 x \rfloor + 1} \mathrm{bit}(x, z).$$

Используя суперпозиции полученных выше функций, легко доказать принадлежность классу FO многих других функций: $\mathrm{sg}(x)$, $\min(x, y)$, $\max(x, y)$, $2^{\lfloor \log_2 x \rfloor}$, $2^{\lfloor \log_2 x \rfloor^2}$. Также нетрудно показать принадлежность функций $\mathrm{len}(x)$, $2^{\mathrm{len}(x)}$ и $2^{\mathrm{len}(x) \cdot \mathrm{len}(y)}$ классу FO и принадлежность предикатов $(x = y)$, $(x \leqslant y)$ и других сравнений $(\neq, <, >, \geqslant)$ классу FO_*.

Ряд более сложных результатов сформулирован в следующей теореме.

Т е о р е м а 1 [43].

1. *Класс* FO *замкнут относительно операции ограниченного мультиплицирования*

$$\prod_{z=0}^{\lfloor \log_2 \log_2 y \rfloor^m} \min(g(x_1, \ldots, x_n, z),\ 2^{\lfloor \log_2 \log_2 y \rfloor^m}), \quad m \in \mathbb{N}.$$

2. *Предикат* $(x \equiv y^z \pmod u)$ k-FO-*определим при любом* k.

3. *Класс* FOM *замкнут относительно операции ограниченного мультиплицирования*

$$\prod_{z=0}^{\lfloor \log_2 y \rfloor} g(x_1, \ldots, x_n, z).$$

4. *Функции* $x^{\lfloor \log_2 y \rfloor}$ *и* $\lfloor x/y \rfloor$ *входят в класс* FOM.

Доказательство этих результатов использует сложную технику, которая развивалась в ряде работ (см., например, [28, 33]) и была нетривиальным образом адаптирована к классу FOM в [43]. Отметим лишь некоторые основные пункты.

1. Целая часть от деления сводится к возведению в степень через приближение значения $1/\alpha$ степенным рядом (с помощью формулы суммы геометрической прогрессии):

$$1/\alpha = \sum_{i=0}^{n} (1 - \alpha)^i + O(2^{-n}), \quad 1/2 < \alpha \leqslant 1.$$

2. Возведение в степень элементарно сводится к перемножению $|\bar{X}|^k - 1$ чисел.

   Для перемножения числа задаются своими остатками от деления на малые простые числа (в соответствии с китайской теоремой об остатках).

   В этом представлении вычисление произведения сводится к перемножению чисел по малому простому модулю p, что, благодаря возможности дискретного логарифмирования (предикат $(x \equiv y^z \pmod u)$), делается с использованием суммирования.

3. Выражение предиката ($x \equiv y^z \pmod{u}$) реализовано через возведение в степень в группе с малым (логарифмическим от степени) числом применений групповой операции. Известный метод последовательного возведения в квадрат не годится для вычислений в FO, поэтому используется более изощренный «параллельный» метод.

С помощью приближения степенным рядом можно доказать принадлежность FOM и других арифметических функций: например функций $\lfloor \sqrt[k]{x} \rfloor$, где k — натуральное число. Кроме того, из теоремы 1 следует, что функция $\mathrm{rm}(x, y)$ принадлежит FOM.

§ 2. Задание классов с помощью абстрактных машин и унифицированных схем

В этом параграфе мы приведем эквивалентные определения классов FO и FOM, основанных на машинных и схемных описаниях. Все эти определения так или иначе связаны с параллельными вычислениями: рассматриваемые классы содержат функции, вычислимые на полиномиальном количестве процессоров за константное время.

Основное внимание будет уделено схемным определениям AC^0 и TC^0; будет приведено доказательство их эквивалентности классам FO и FOM соответственно.

Интерес к схемным определениям AC^0 и TC^0 связан с тем, что благодаря им выяснено положение этих классов среди известных сложностных классов, а также доказано нетривиальное строгое включение $\mathrm{AC}^0 \subset \mathrm{TC}^0$ и некоторые другие нижние оценки.

2.1. Класс FO и параллельные вычисления. Первые результаты о связи класса FO с другими моделями вычислений касаются параллельных вычислений с общей памятью. Мы охарактеризуем их очень кратко.

Будем называть CRAM-машиной машину с произвольным доступом к памяти, которая имеет несколько процессоров, работающих с общей памятью без ограничений на одновременное чтение и запись (в случае одновременной записи приоритет имеет процессор с наименьшим номером).

Внешняя память машины представляет собой бесконечную последовательность регистров, которые могут хранить числа из $\mathbb{N}_0$. Каждый процессор также имеет конечное число локальных регистров. Процессоры синхронно работают в дискретном времени и выполняют общую программу. В программе могут присутствовать команды чтения и записи во внешнюю память по адресу, операции с локальными регистрами: присваивание, сложение, вычитание, битовый сдвиг ($x := \lfloor x/2^y \rfloor$), сравнение; а также команды условного перехода к другой позиции в программе. Процессор имеет доступ к своему номеру. Вход машины в начальный момент времени записан побитово в первых n регистрах памяти, выход записывается аналогично.

Следующий результат сформулирован в [44] и доказан в [45] для предикатов. На функции он распространяется естественным образом.

Теорема 2 [44, 45]. *Класс* FO *совпадает с классом функций, вычислимых на* CRAM*-машинах с полиномиальным от длины двоичного входа числом процессоров и константным временем работы.*

2.2. Машины Тьюринга с произвольным доступом ко входу. Чтобы с помощью машин Тьюринга задавать классы функций, вычислимых за малое (меньшее длины входа) время, нужно дать им возможность считывать произвольные символы входа без необходимости обходить все содержимое ленты.

Машина Тьюринга с произвольным доступом ко входу (далее машина Тьюринга) состоит из входной ленты, адресной ленты, k рабочих лент, считывающе-записывающих головок на адресной и рабочих лентах, а также конечного множества состояний Q и программы. Каждая лента бесконечна в обе стороны и разбита на клетки, которые могут содержать символы $0, 1$ и «пустой» символ Λ.

Машина работает в дискретном времени. Входная лента содержит двоичную запись входа (незанятые клетки содержат Λ) и не меняется в процессе вычисления. Содержимое адресной ленты в каждый момент времени t представляет собой двоичную запись некоторого числа $i(t)$ (пустая лента соответствует числу 0). В начальный момент адресная и рабочие ленты пусты. В каждый момент времени машина находится в некотором состоянии из Q, в начальный момент это специально выделенное начальное состояние.

Программа машины представляет собой набор команд вида

$$xa_0a_1\ldots a_kq_i \to b_0b_1\ldots b_kD_0D_1\ldots D_kq_j,$$

где $x, a_0, \ldots, a_k, b_0, \ldots, b_k \in \{0, 1, \Lambda\}$, $D_0, \ldots D_k \in \{L, R, S\}$, $q_i, q_j \in Q$. Левые части разных команд попарно различны.

На каждом такте машина считывает символ x входной ленты с номером $i = i(t)$ (число $i(t)$, напомним, берется с адресной ленты) и текущие обозреваемые головками символы $a_0, \ldots, a_k$ адресной и рабочих лент. Затем машина ищет в программе команду с левой частью, соответствующей считанным символам и текущему состоянию q_i. В соответствии с командой машина заменяет обозреваемые головками символы адресной и рабочих лент на $b_0, \ldots, b_k$, независимо перемещает каждую головку на одну клетку влево или вправо (либо оставляет на месте) и заменяет текущее состояние на q_j.

Машина останавливается в случае перехода в одно из заключительных состояний q_0, q_1. Результат ее работы («нет» или «да») определяется по выбранному состоянию. Таким образом машина задает предикаты натурального аргумента. В случае если предикат зависит от нескольких переменных, считаем, что на вход подается конкатенация их двоичных записей, выравненных незначащими нулями до одинаковой длины n. В технических целях считаем, что если число переменных m не является степенью двойки, то в конец входа дописывается $2^l - m$ блоков из нулей длины n, где $2^{l-1} < m < 2^l$ (это увеличивает длину входа менее чем в 2 раза и делает ее равной $n \cdot 2^l$).

Через LOGTIME будем обозначать класс предикатов, вычислимых на машинах Тьюринга за время $O(\log_2 n)$, где n — длина двоичного входа. Более крупные сложностные классы тоже можно определить в терминах этой разновидности машин Тьюринга. Так, L — это класс предикатов, вычислимых на машине Тьюринга с использованием $O(\log_2 n)$ клеток адресной и рабочих лент, P — класс предикатов, вычислимых на машинах Тьюринга

за полиномиальное от n время, а PSPACE — класс предикатов, вычислимых на машине Тьюринга с использованием полиномиального от n числа клеток адресной и рабочих лент.

Утверждение 7 [27].

$$\mathrm{LOGTIME} \subseteq \mathrm{FO}_*.$$

Доказательство. Приведем лишь основные моменты построений. Будем считать, что все входные аргументы предиката записаны в одной «длинной» переменной X по тому же принципу, что и вход машины Тьюринга (этого нетрудно добиться с помощью введенной в § 1 техники работы с FO).

Обозначим $n = |X|$. Будем представлять вычисление машины в виде последовательности выполняемых команд (не конфигураций). Каждая команда кодируется константным числом битов, а всего команд в вычислении $O(\log_2 n)$. Значит, при некотором k последовательность команд помещается в k-переменную.

Строим FO-формулу: с помощью кванторов перебираем все возможные последовательности команд и проверяем, что они представляют корректное вычисление. Для этого нужно иметь возможность вычислить позиции всех головок и символы с лент на каждом такте. Чтобы вычислить позицию головки в момент времени t, нужно подсчитать количество символов движения R и L в предшествовавшем вычислении. Это реализуемо средствами FO с помощью суммирования $O(\log_2 n)$ чисел.

Для определения символа ленты в заданной клетке нужно найти последнюю команду предшествовавшего вычисления, при выполнении которой головка ленты обозревала эту клетку. Таким образом нетрудно получить все обозреваемые головками символы в момент времени t. Для вычисления считываемого символа входной ленты нужно вычислить и записать в k-переменную число, представленное в двоичном виде на адресной ленте (это не вызывает затруднений, так как длина этого числа $O(\log_2 n)$), после чего применить элементарный предикат $X(i)$.

С помощью указанных выше действий нетрудно проверить, что очередная команда применима к результату предшествовавшего вычисления. Поэтому средствами класса FO можно определить существование вычисления машины Тьюринга, выдающего «да» на данном входе.

Приведем еще один результат о моделировании вычислений на машинах Тьюринга с произвольным доступом ко входу средствами FO.

Лемма 3 [36]. *Пусть* $0 < \varepsilon < 1$ *и* $k \geqslant 1$. *Класс предикатов, вычислимых на машинах Тьюринга за время* $O((\log_2 n)^k)$ *с использованием* $O((\log_2 n)^{1-\varepsilon})$ *клеток адресной и рабочих лент, где* n *— длина двоичного входа, вложен в* FO_*.

Следствие 2 [36]. *Класс предикатов, вычислимых на машинах Тьюринга с использованием* $O(\log_2 \log_2 n)$ *клеток адресной и рабочих лент, где* n *— длина двоичного входа, вложен в* FO_*.

Лемма 3 в частном случае для времени $O((\log_2 n)^\varepsilon)$ доказывается путем генерации последовательности конфигураций. После этого можно сгенерировать последовательность конфигураций, отстоящих друг от друга

на $O((\log_2 n)^\varepsilon)$ тактов, и достичь времени $O((\log_2 n)^{2\varepsilon})$. Продолжая аналогично, за конечное число шагов можно достичь произвольного времени $O((\log_2 n)^k)$.

Доказательство леммы 3 является переложением на класс FO известного способа моделирования вычислений на машинах Тьюринга рудиментарными предикатами [14, 18]. В [36] оно проводится в рамках индуктивного описания класса FO (см. § 3), но его нетрудно провести и в логическом формализме.

2.3. Альтернирующие машины Тьюринга. Один из способов моделировать параллельные вычисления с помощью машин Тьюринга — это техника альтернирования. Будем рассматривать разновидность машин Тьюринга, в программе которых может быть несколько команд с одинаковой левой частью. Во время вычисления такая машина может выбрать любую из подходящих команд.

Все возможные вычисления машины Тьюринга такого вида на каждом фиксированном входе можно представить в виде дерева: в корне находится начальная конфигурация машины, в дочерних вершинах каждой конфигурации находятся конфигурации, в которые машина может перейти из данной. В листьях находятся конфигурации с заключительными состояниями. Временем вычисления считаем максимальную длину пути от корня к листу в этом дереве (считаем, что во всех ветвях машина должна останавливаться).

Альтернирующая машина Тьюринга — это машина Тьюринга указанного вида, каждое состояние которой помечено символом $\vee$ или $\&$. Результат работы альтернирующей машины Тьюринга на входе X определяется следующим образом. Строится дерево вычислений машины на входе X. Каждой нелистовой вершине дерева присваивается символ $\vee$ или $\&$, соответствующий состоянию в конфигурации этой вершины. Каждой листовой вершине дерева присваивается значение 0 или 1 в зависимости от результата работы машины в этой ветви вычисления. Далее полученное дерево с пометками рассматривается как схема из функциональных элементов $\vee$, $\&$ с заданными значениями 0, 1 на входах. В соответствии со стандартными принципами работы схем вычисляется значение (0 или 1) в корне дерева, которое и является результатом работы машины.

Числом альтернирований в таком вычислении называется максимальное количество изменений типа состояния ($\vee$, $\&$) в путях от корня дерева к его листьям.

Частным случаем альтернирующей машины Тьюринга является недетерминированная машина Тьюринга: у этой машины все состояния имеют тип $\vee$ и, соответственно, в любом ее вычислении нет альтернирований. С помощью таких машин определяется, например, класс NP функций, вычислимых на недетерминированных машинах Тьюринга за время, полиномиальное от длины двоичного входа.

Через ALOGTIME будем обозначать класс предикатов, вычислимых на альтернирующих машинах Тьюринга за время $O(\log_2 n)$, где n — длина двоичного входа. Через LH (логарифмическая иерархия) обозначаем класс предикатов, вычислимых на альтернирующих машинах Тьюринга за время $O(\log_2 n)$ с константным (зависящим только от машины) числом альтернирований.

Утверждение 8 [27].

$$\mathrm{FO}_* = \mathrm{LH}.$$

Мы не будем приводить доказательство этого утверждения. Отметим лишь, что оно следует из утверждения 7 и что применение квантора $\exists$ в FO-формуле соответствует серии последовательных ветвлений в $\vee$-состояниях вычисления альтернирующей машины; аналогичная связь имеется между квантором $\forall$ и $\&$-состояниями.

2.4. Унифицированные схемы. Будем использовать стандартное понятие схемы из функциональных элементов, реализующей булеву функцию (см. [23] или [25]). Функциональные элементы схем будут иметь типы $\&, \vee, \neg$ и maj ($\&, \vee$, maj могут иметь произвольное число входов). Элемент maj выдает 1, если более половины его входов единицы. Отметим, что для всех этих элементов не важен порядок входов. Сложностью схемы называется число функциональных элементов в ней, а глубиной — максимальное число функциональных элементов в путях от входов к выходу схемы.

Пусть $\Sigma = \{\Sigma(1), \Sigma(2), \ldots\}$ — семейство схем из функциональных элементов, в котором каждая схема $\Sigma(n)$ имеет n входов и один выход.

Предикат натурального аргумента $\rho(x)$ реализуется семейством схем Σ, если для любого x и каждого $q \geqslant \mathrm{len}(x)$ схема $\Sigma(q)$ при получении на входы двоичной записи x (с нужным число незначащих нулей) выдаст на выходе $\chi_\rho(x)$. Предикат от нескольких переменных реализуется аналогично, при этом на вход схеме подается конкатенация двоичных записей аргументов, выравненных незначащими нулями до одинаковой длины.

Чтобы реализуемые схемами классы предикатов можно было соотносить со стандартными сложностными классами, вводится требование унифицированности. Будем считать, что все вершины каждой схемы (входы, функциональные элементы и выходы) имеют уникальные номера, не превосходящие полинома от сложности схемы. Введем нумерацию также для типов элементов (при этом считаем, что все входы схемы имеют разные типы).

С каждым семейством («пронумерованных») схем Σ свяжем предикат $\mathrm{dc}_\Sigma(t, a, b, y)$, истинный только в тех случаях, когда при некотором n верно $y = 2^n - 1$, а в схеме $\Sigma(n)$ вершина с номером a имеет тип t и в нее ведет дуга из вершины с номером b.

Пусть A — сложностной класс предикатов. Будем называть семейство схем Σ A-унифицированным, если $\mathrm{dc}_\Sigma(t, a, b, y) \in A$.

Дадим теперь определения интересующих нас классов предикатов. Пусть $k \in \mathbb{N}_0$, а n — число входов схемы.

NC^k_* — это класс предикатов, реализуемых LOGTIME-унифицированными семействами схем полиномиальной от n сложности и глубины $O((\log_2 n)^k)$, использующих функциональные элементы $\neg$ и 2-входовые $\&$, $\vee$.

AC^k_* — это класс предикатов, реализуемых LOGTIME-унифицированными семействами схем полиномиальной от n сложности и глубины $O((\log_2 n)^k)$, использующих функциональные элементы $\neg$ и $\&$, $\vee$ без ограничений на число входов.

TC^k_* — это класс предикатов, реализуемых LOGTIME-унифицированными семействами схем полиномиальной от n сложности и глубины $O((\log_2 n)^k)$, использующих функциональные элементы $\neg$ и maj без ограничений на число входов.

Через NC_* обозначаем объединение всех классов NC^k_* $(k \geqslant 0)$. В этом классе функции реализуются схемами, имеющими полиномиальную от $\log_2 n$ глубину.

Определим NC^k как класс функций $f(x_1, \ldots, x_n)$, удовлетворяющих условиям:

1. Существует полином $p(l_1, \ldots, l_n)$ такой, что

$$\mathrm{len}(f(x_1, \ldots, x_n)) \leqslant p(\mathrm{len}(x_1), \ldots, \mathrm{len}(x_n)).$$

2. Предикат $\mathrm{BIT}(f(x_1, \ldots, x_n), y)$ принадлежит классу NC^k_*.

Классы AC^k, TC^k, NC определяются аналогично.

Поскольку подстановкой констант в функцию maj (с 3 или более входами) можно получить функции & и $\vee$ с произвольным количеством входов, а функцию maj можно реализовать схемами из 2-входовых элементов &, $\vee$ логарифмической глубины (см., например, [22] или [25]), определенные выше классы выстраиваются в иерархию

$$\mathrm{AC}^0 \subseteq \mathrm{TC}^0 \subseteq \mathrm{NC}^1 \subseteq \ldots \subseteq \mathrm{NC}^k \subseteq \mathrm{AC}^k \subseteq \mathrm{TC}^k \subseteq \mathrm{NC}^{k+1} \subseteq \ldots \subseteq \mathrm{NC}.$$

Нас будут интересовать нижние классы этой иерархии: AC^0 и TC^0 (класс NC^0 является вырожденным). Функции из этих классов реализуются схемами константной глубины.

Класс NC и его подклассы обычно рассматривают как классы функций, вычисление которых можно эффективно распараллелить. Классы AC^0 и TC^0 содержат функции, для которых распараллеливанием достигается константное время работы.

В литературе изучаются классы NC^k, AC^k, TC^k с разными видами унифицированности [38, 55, 58], а также их неунифицированные варианты [32, 40]. Мы, следуя [25], «по умолчанию» используем LOGTIME-вариант как наиболее жесткое требование к унифицированности.

Для классов NC^1 и старше LOGTIME-унифицированность эквивалентна ряду менее жестких вариантов унифицированности [27, 58]. При использовании этих вариантов унифицированности имеет место соотношение $\mathrm{NC}^1_* = \mathrm{ALOGTIME}$ [58].

2.5. Совпадение классов AC^0, FO и классов TC^0, FOM. Обоснование разумности использования LOGTIME-унифицированности для классов AC^0 и TC^0 дает следующая теорема.

Т е о р е м а 3 [27].

$$\mathrm{AC}^0 = \mathrm{FO}, \quad \mathrm{TC}^0 = \mathrm{FOM}.$$

Д о к а з а т е л ь с т в о. Будем доказывать равенства для классов предикатов (на классы функций теорема расширяется естественным образом). Покажем, что $\mathrm{FO}_* \subseteq \mathrm{AC}^0_*$. Считаем, что вход подается в виде одной «длинной» переменной X длины n.

Пусть предикат ρ задается FO-формулой. Можно считать, что эта формула имеет вид $(Q_1 y_1) \ldots (Q_d y_d)\Phi(X;\ y_1, \ldots, y_d)$, где $Q_1, \ldots, Q_d$ — кванторы, а подформула Φ не содержит кванторов.

Строим n^d схем $\Sigma_{y_1 \ldots y_d}$, структура которых соответствует формуле Φ с заменой элементарных формул $(x \leqslant y)$ и $\mathrm{BIT}(x, y)$ константами 0, 1 (значениями соответствующих элементарных предикатов при данных $y_1, \ldots, y_d$), а формул $X(y_i)$ — входными переменными x_{y_i}. Эти схемы имеют константную сложность.

Далее строим схему D в виде дерева, имеющего d ярусов: каждый ярус соответствует квантору Q_i (индекс возрастает от корня к листьям) и моделирует «выбор» значения переменной y_i для каждого выбора уже рассмотренных $y_1, \ldots, y_{i-1}$. Все функциональные элементы ярусов имеют по n входов; если $Q_i = \exists$, то в i-м ярусе все функциональные элементы имеют тип $\vee$, а если $Q_i = \forall$, — то тип $\&$. При такой структуре всякий путь в дереве от корня к листу фиксирует некоторый выбор значений всех переменных $y_1, \ldots, y_d$.

К n^d листьям дерева присоединяются схемы $\Sigma_{y_1 \ldots y_d}$ со всеми возможными значениями $y_1, \ldots, y_d$ (в соответствии с их выбором по пути от корня до данного листа). Нетрудно видеть, что полученная схема Σ имеет полиномиальную от n сложность, константную глубину и что она реализует предикат ρ.

Осталось пронумеровать все вершины схемы Σ таким образом, чтобы предикат dc_Σ для семейства этих схем при различных n был вычислим за логарифмическое время на машине Тьюринга. Номер элемента a будет кодировать набор чисел $(y_1, \ldots, y_d, z)$, где каждое число y_i кодируется $\mathrm{len}(n) + 1$ битами, а число z кодируется константным количеством k битов. Каждое число y_i указывает, какой «выбор» совершается в пути от корня до элемента a на i-м ярусе (0, если если элемент сам находится в дереве D на ярусе i или менее). Число k указывает номер элемента внутри схемы $\Sigma_{y_1 \ldots y_d}$ (0, если это элемент из дерева D или корень схемы $\Sigma_{y_1 \ldots y_d}$).

Чтобы при такой нумерации вычислить значение предиката $\mathrm{dc}_\Sigma(t, a, b, y)$, требуется:

1. Определить число n. Общая длина входа $N = 4 \cdot n$ вычисляется последовательным удвоением числа на адресной ленте до значения $2^l < n$, после чего в полученном числе некоторые нули заменяются на единицы (от старших к младшим) так, чтобы на каждом шаге число не превосходило N.

   Число на адресной ленте превосходит N, если считываемый символ входной ленты содержит Λ. Поделить число N на 4 не вызывает затруднений.

2. Выполнять операции сложения, вычитания и сравнения чисел длины $O(\log_2 n)$ за время $O(\log_2 n)$. Это не вызывает затруднений.

   Для вычисления функций $\mathrm{len}(x)$ и $\mathrm{bit}(x, y)$ от чисел длины $O(\log_2 n)$ нужно использовать двоичные счетчики: l-кратное увеличение/уменьшение числа в двоичной записи на единицу (преобразующее число 0 в число l или, наоборот, число l в число 0) требует линейного от l времени.

3. Определить наличие дуги: по кодам a и b нетрудно выяснить их положение в дереве D. Если оба элемента находятся в одной листовой схеме, то наличие дуги определяется таблично.

4. Тип функционального элемента из дерева D определяется таблично по номеру яруса. Чтобы проверить тип функционального элемента в схеме $\Sigma_{y_1,\ldots,y_d}$, нужно таблично определить, какой логической связке или элементарной подформуле формулы Φ он соответствует. В случае элементарной формулы нужно вычислить значение $(y_i \leqslant y_j)$ или $\mathrm{BIT}(y_i, y_j)$ (y_i и y_j извлекаются из кода a) либо (для $X(y_i)$) указать, что тип элемента соответствует входу x_{y_i}.

Доказательство включения $\mathrm{FOM}_* \subseteq \mathrm{TC}^0_*$ проводится аналогично: квантору M соответствует элемент maj, а &- и $\vee$-элементы получаются подстановкой констант в maj, поэтому их тоже можно использовать в схемах.

Теперь докажем, что $\mathrm{AC}^0_* \subseteq \mathrm{FO}_*$. Будем строить по «пронумерованной» схеме Σ FO-формулу. Поскольку схема имеет полиномиальный размер, номера ее элементов помещаются в k-переменные и их можно перебирать кванторами. Поскольку $\mathrm{LOGTIME} \subseteq \mathrm{FO}_*$, имеется возможность вычислять предикат $\mathrm{dc}_\Sigma(t, a, b, y)$ и проверять типы элементов и наличие дуг между элементами (а значит, и перебирать все возможные пути в схеме с учетом того, что ее глубина есть константа, не зависящая от схемы).

Индуктивно определяем предикаты $\mathrm{Acc}_k(X;\ a)$, которые для входа X вычисляют значение на выходе элемента a на расстоянии не больше k от входов схемы. $\mathrm{Acc}_0(X;\ a)$ для входа схемы x_i выдает $X(i)$. Предикат $\mathrm{Acc}_k(X;\ a)$ для $\vee$-элемента a перебирает квантором существования все функциональные элементы схемы, и для тех из них, из которых ведут дуги в a, вычисляет Acc_{k-1}. Для &-элемента используется квантор общности, а для $\neg$ ищется единственный предшествующий элемент.

Поскольку схема имеет константную глубину d, предикат $\mathrm{Acc}_d(X;\ a)$ корректно определяет выходы всех элементов схемы. Осталось найти элемент, являющийся выходом схемы, и подставить его номер в этот предикат. Доказательство включения $\mathrm{TC}^0_* \subseteq \mathrm{FOM}_*$ проводится аналогично, при этом элемент maj моделируются квантором M.

Для логически заданных классов тоже можно ввести понятие унифицированности: ее уровень определяется набором элементарных предикатов, зависящих только от «коротких» переменных, из которых строятся формулы. Унифицированный класс FO получается при использовании лишь элементарных предикатов $x \leqslant y$, $\mathrm{BIT}(x, y)$.

Можно рассматривать неунифицированный класс FO, в котором в качестве элементарных предикатов допускается использовать любые (в том числе невычислимые) предикаты от «коротких» переменных. В [44] указано, что неунифицированный класс FO совпадет с неунифицированным AC^0. Это нетрудно доказать аналогично теореме 3.

2.6. Несовпадение классов AC^0 и TC^0. Одна из причин, по которым классы AC^0 и TC^0 представляют особый интерес, — это наличие нетривиальной нижней оценки сложности: функции $\mathrm{bitsum}(x)$ и xy не входят в класс AC^0, что влечет строгое включение класса AC^0 в класс TC^0. Это одно из немногих известных строгих включений сложностных классов (не считая тех, что следуют из иерархий по времени и памяти).

Пусть $\bar{x} = (x_1, \ldots, x_n)$. Обозначим $l(\bar{x})$ булеву функцию $x_1 \oplus \ldots \oplus x_n$. Семейство функций $l(\bar{x})$ при всех n соответствует функции натурального аргумента $\mathrm{parity}(x)$, которая выдает 1, если двоичная запись числа x имеет нечетное число единиц.

В работах [24] и [40] независимо было доказано, что parity$(x) \notin \mathrm{AC}^0$. Точнее, в работе [24] этот факт был доказан для неунифицированного класса FO, а в работе [40] — для неунифицированного AC^0. Справедливость утверждения для унифицированных версий этих классов является очевидным следствием. Мы приведем версию теоремы из [40].

Теорема 4 [40]. *Функция $x_1 \oplus \ldots \oplus x_n$ не может быть реализована (неунифицированными) семействами схем полиномиальной от n сложности и константной глубины, использующих функциональные элементы $\neg$ и $\vee$, & без ограничений на число входов.*

Доказательство. Приведем общую схему доказательства, опуская детали получения числовых оценок.

Будем называть d-схемой схему из функциональных элементов полиномиальной сложности, представляющую собой d-ярусное дерево, в котором функциональные элементы каждого яруса имеют типы & и $\vee$ с произвольным числом входов (общий тип для всего яруса; элементы соседних ярусов имеют разные типы), а в листьях находятся переменные и их отрицания. Любую AC^0-схему можно преобразовать к такому виду. Сложностью d-схемы можно считать количество листьев дерева.

Двухъярусные схемы представляют собой ДНФ и КНФ. Известно, что линейная функция $l(\bar{x}) = x_1 \oplus \ldots \oplus x_n$ может быть реализована только совершенными ДНФ и КНФ сложности $n \cdot 2^{n-1}$. Поэтому функция $l(\bar{x})$ не реализуема 2-схемами.

При любой подстановке констант в $l(\bar{x})$ вместо части переменных все остальные переменные остаются существенными и получается функция l от меньшего числа переменных (или ее отрицание).

Пусть $d \geqslant 3$. Предположим, что функция $l(\bar{x})$ реализуема d-схемой. Покажем, что в этом случае схему можно перестроить так, что число ярусов уменьшится на 1.

На первом этапе покажем, что при реализации $l(\bar{x})$ d-схемами допустимо ограничить сложность всех 1-подсхем константой. Выберем произвольную d-схему, реализующую $l(\bar{x})$, и докажем, что существует такая подстановка констант вместо части переменных, что существенных переменных остается не менее $\sqrt{n}/2$, а сложность всех 1-подсхем (после применения преобразований $x \cdot 0 = 0$, $x \cdot 1 = x$, $x \vee 0 = x$, $x \vee 1 = 1$) станет константной.

Рассмотрим случайную подстановку со следующим распределением: для каждой переменной x_i вероятность подставить на ее место каждую из констант равна $(1 - 1/\sqrt{n})/2$. Тогда вероятность, что переменная останется существенной, равна $1/\sqrt{n}$, а вероятность, что существенных переменных останется менее $\sqrt{n}/2$, есть $O(n^{-1/2})$.

Можно показать, что для каждой 1-подсхемы вида $x_{i_1}^{\sigma_1} \vee \ldots \vee x_{i_k}^{\sigma_k}$ вероятность того, что ее сложность больше константы c, не превосходит $O(n^{-c/4})$. Для схем с большим числом входов это следует из высокой вероятности обратиться в константу. В случае малого числа входов подстановка с большой вероятностью уменьшит их число до константного количества. Для &-1-подсхемы рассуждения аналогичны.

Тогда выбором константы c (в силу полиномиальности количества 1-подсхем) можно добиться ненулевой вероятности того, что подстановка удовлетворяет требуемым условиям. Если подстановка дает схему для $\bar{l}(\bar{x})$,

то ее легко перестроить в схему для $l(\bar{x})$. Переход от $\sqrt{n}/2$ входов к n входам означает лишь полиномиальное увеличение сложности.

На втором этапе покажем, что при реализации $l(\bar{x})$ d-схемами можно ограничить сложность всех 2-подсхем константой. Этот шаг проделывается аналогично предыдущему: доказывается существование такой подстановки констант, что существенных переменных остается не менее $\sqrt{n}/2$, а сложность всех 2-подсхем станет константной. Для этого рассматривается та же случайная подстановка, что и на первом этапе.

Вероятность 2-схемы иметь малую сложность оценивается индукцией по максимальной сложности 1-подсхем. При этом 2-схема с большим числом 1-подсхем с высокой вероятностью обращается в константу, а для схем с малым числом 1-подсхем оценка сводится к индуктивному предположению с помощью дополнительной подстановки, уменьшающей сложность 1-подсхем.

На третьем этапе перестроим в схеме ярусы d и $d-1$. Каждая 2-подсхема представляет собой КНФ или ДНФ с некоторым количеством c переменных (число c не зависит от n). Перестроим все КНФ в ДНФ (или ДНФ в КНФ) сложности не более 2^c. После этого можно объединить ярусы $d-1$ и $d-2$ и получить $(d-1)$-схему, реализующую $l(\bar{x})$. Повторяя уменьшение числа ярусов, получим 2-схему для $l(\bar{x})$, что невозможно.

Очевидно, что $\mathrm{parity}(x) = \mathrm{bit}(\mathrm{bitsum}(x),\ 0)$. Кроме того, в [40] показано, что вычисление функции $\mathrm{bitsum}(x)$ можно свести к вычислению умножения чисел (см. аналогичные построения в теореме 6 из § 3). Вычисление функции xy можно свести к вычислению $\lfloor x/y \rfloor$ (см. утверждение 10 из § 4) и к вычислению x^2 (а значит, и к $x^{\lfloor \log_2 y \rfloor}$). Таким образом, с учетом результатов § 1 имеем

$$\mathrm{bitsum}(x),\ x^2,\ xy,\ \lfloor x/y \rfloor,\ x^{\lfloor \log_2 y \rfloor} \in \mathrm{TC}^0 \setminus \mathrm{AC}^0.$$

В [31] также показано, что график умножения не входит в AC^0, т. е. $(z = xy) \in \mathrm{TC}^0_* \setminus \mathrm{AC}^0_*$.

Приведем одно усиление результата теоремы. Из нижней оценки на глубину схемы, реализующей $x_1 \oplus \ldots \oplus x_n$ ([42], см. также [46]), следует, что битовая сумма $(\log_2 n)^{f(n)}$ битов, где $f(n) \to \infty$ — сколь угодно медленно растущая функция, не может быть вычислена средствами AC^0. Это значит, что не может быть вычислена также сумма $(\log_2 n)^{f(n)}$ слагаемых и произведение двух чисел двоичной длины $(\log_2 n)^{f(n)}$.

В то же время результаты из § 1 свидетельствуют, что сумма $(\log_2 n)^{O(1)}$ чисел и произведение $(\log_2 n)^{O(1)}$ чисел длины $(\log_2 n)^{O(1)}$ могут быть вычислены в AC^0. Таким образом, граница вычислительных возможностей AC^0 по части суммирования и умножения известна достаточно точно.

2.7. Положение классов AC^0 и TC^0 в иерархии сложностных классов. Сначала кратко охарактеризуем классы, лежащие между AC^0 и TC^0.

При $k \geqslant 2$ обозначим через $\mathrm{mod}_k(\bar{x})$ булеву функцию, выдающую 1, если количество единиц $\bar{x}$ делится на k. С помощью $\mathrm{AC}^0(k)$ (встречается также обозначение $\mathrm{ACC}(k)$) обозначим класс функций, реализуемых LOGTIME-унифицированными семействами схем полиномиальной от n сложности и константной глубины, использующих функциональные

элементы $\neg$ и $\&$, $\vee$, mod_k без ограничений на число входов. Класс ACC представляет собой объединение всех классов $\mathrm{AC}^0(k)$ ($k \geqslant 2$).

Поскольку $\mathrm{bitsum}(x) \in \mathrm{TC}^0$, нетрудно видеть, что $\mathrm{ACC} \subseteq \mathrm{TC}^0$. Из теоремы 4 следует, что $\mathrm{AC}^0 \subset \mathrm{AC}^0(2)$. Из ее усиления [39] следует, что $\mathrm{AC}^0 \subset \mathrm{AC}^0(k)$ при $k \geqslant 2$. В [20, 59] доказано, что $\mathrm{AC}^0(p) \subset \mathrm{TC}^0$ при простых p.

Нетрудно видеть, что если число a делит b, то $\mathrm{AC}^0(a) \subseteq \mathrm{AC}^0(b)$. В [59] доказано, что для любого простого числа p и числа r, не являющегося степенью p, верно $\mathrm{AC}^0(r) \not\subseteq \mathrm{AC}^0(p)$. Если, кроме того, p делит r, то верно $\mathrm{AC}^0(p) \subset \mathrm{AC}^0(r)$. Отсюда ясно, что при простом p верно $\mathrm{AC}^0(p) \subset \mathrm{ACC}$. Объединяя эти соотношения, при различных простых p и q имеем

$$\mathrm{AC}^0 \subset \mathrm{AC}^0(p) \not\subseteq \mathrm{AC}^0(q) \subset \mathrm{AC}^0(pq) \subseteq \mathrm{ACC} \subseteq \mathrm{TC}^0.$$

Отметим, что при простом p функции $\mathrm{bitsum}(x)$, x^2, xy, $\lfloor x/y \rfloor$, $x^{\lfloor \log_2 y \rfloor}$ не принадлежат классу $\mathrm{AC}^0(p)$, а предикат $(z = xy)$ не принадлежит $\mathrm{AC}^0_*(p)$.

Теперь, учитывая $\mathrm{NC}^1_* \subseteq \mathrm{L} \subseteq \mathrm{AC}^1_*$ [38], мы можем описать положение AC^0 и TC^0 в иерархии известных сложностных классов:

$$\mathrm{LOGTIME} \subseteq \mathrm{LH} = \mathrm{AC}^0_* \subset \mathrm{TC}^0_* \subseteq \mathrm{NC}^1_* = \mathrm{ALOGTIME} \subseteq \mathrm{L} \subseteq \mathrm{AC}^1_* \subseteq \mathrm{NC}_* \subseteq \mathrm{P} \subseteq \mathrm{NP}.$$

Является ли включение $\mathrm{TC}^0_* \subseteq \mathrm{NP}$ строгим, неизвестно. Неизвестно даже, является ли строгим включение $\mathrm{ACC}_* \subseteq \mathrm{NP}$. Однако в относительно недавней работе [53] доказано, что класс ACC_* строго вложен в класс $\mathrm{NTIME}(n^{(\log_2 n)^{O(1)}})$ предикатов, вычислимых на недетерминированных машинах Тьюринга за время $n^{p(\log_2 n)}$, где p — произвольный (не фиксированный) полином, а n — длина двоичного входа.

§ 3. Задание классов с использованием операций ограниченной конкатенации

В этом параграфе мы приведем индуктивные описания классов AC^0 и TC^0: они будут описаны как классы всех функций, которые можно получить из простых исходных функций при помощи некоторых операций. Помимо операции суперпозиции (которая будет присутствовать во всех описаниях), мы будем использовать два вида ограниченной конкатенации: операцию CRN и более сильную операцию $\mathrm{CON}_{\sqsubseteq}$.

Операция CRN была введена в [34], а операция $\mathrm{CON}_{\sqsubseteq}$ — в [12] (похожая операция была введена ранее в [47] для характеризации класса L, но она рассматривалась только в связке с другими вариантами рекурсии). Для сравнения мы также приведем одну существенно более сильную версию ограниченной конкатенации $\mathrm{CON}_{\leqslant}$, которая задает класс функций, элементарных по Кальмару [9, 19].

3.1. Конкатенация по двоичному представлению. Пусть A — некоторое множество функций, а $\mathscr{O}_1, \ldots, \mathscr{O}_k$ — операции над функциями. С помощью $[A]_{\mathscr{O}_1,\ldots,\mathscr{O}_k}$ (замыкание A относительно операций $\mathscr{O}_1, \ldots, \mathscr{O}_k$) будем обозначать класс всех функций, которые можно получить из функций множества A с помощью операций $\mathscr{O}_1, \ldots, \mathscr{O}_k$.

Через S будем обозначать операцию суперпозиции (в рассматриваемых нами описаниях классов она всегда будет присутствовать в качестве одной из операций). Запись $[A]_{\mathrm{S}}$ будем сокращать до $[A]$.

Далее используем сокращение $\bar{x} = (x_1, \ldots, x_n)$. Пусть $s_0(x) = 2x$, $s_1(x) = 2x+1$ и функции h_0, h_1 принимают лишь значения 0 и 1.

Говорят, что функция $f(\bar{x}, y)$ получена из функций $g(\bar{x})$, $h_0(\bar{x}, z)$, $h_1(\bar{x}, z)$ с помощью операции *конкатенации по двоичному представлению* ($f = \mathrm{CRN}_{z \sqsubset y}(g, h_0, h_1)$), если

$$\begin{cases} f(\bar{x},\ 0) = g(\bar{x}), \\ f(\bar{x},\ s_0(z)) = s_{h_0(\bar{x},\ z)}(f(\bar{x},\ z)),\ z \neq 0, \\ f(\bar{x},\ s_1(z)) = s_{h_1(\bar{x},\ z)}(f(\bar{x},\ z)). \end{cases}$$

Напомним, что $I_i^n(\bar{x}) = x_i$ — селекторные функции. Обозначим

$$A_0 = [\,0,\quad I_1^1(x),\quad s_0(x),\quad s_1(x),\quad \mathrm{len}(x),\quad \mathrm{bit}(x, y),\quad 2^{\mathrm{len}(x)\cdot\mathrm{len}(y)}\,]_{\mathrm{S,\,CRN}}.$$

Указанные далее результаты о принадлежности функций A_0 в основном получены в [34] (см. также [21, 36]).

С помощью суперпозиции функций 0, $s_0(x)$ и $s_1(x)$ нетрудно получить все константы. Все селекторные функции получаются из $I_1^1(x)$.

Обозначим $x *_b y$ конкатенацию двоичных записей чисел x и y: $x *_b y = x2^{\mathrm{len}(y)} + y$ (отметим, что $x *_b 0 = x$). Пусть $\mathrm{ones}_b(x) = (2^{\mathrm{len}(x)} - 1)$ — число длины x, составленное из единиц, и $\mathrm{shl}(x, y) = x \cdot 2^y$ — битовый сдвиг числа x влево. Нетрудно видеть, что

$$\begin{aligned} x *_b y &= \mathrm{CRN}_{z \sqsubset y}(I_1^1(x),\ 0,\ 1), \\ \mathrm{ones}_b(x) &= \mathrm{CRN}_{z \sqsubset x}(0,\ 1,\ 1), \\ \mathrm{shl}(x,\ \mathrm{len}(y)) &= \mathrm{CRN}_{z \sqsubset y}(I_1^1(x),\ 0,\ 0). \end{aligned}$$

Обозначим $\overline{\mathrm{sg}}(x) = 1 \dot{-} \mathrm{sg}(x)$. Имеем

$$\mathrm{sg}(x) = \mathrm{bit}(\mathrm{ones}_b(x),\ 0), \qquad \overline{\mathrm{sg}}(x) = \mathrm{bit}(\mathrm{shl}(1,\ \mathrm{len}(x)),\ 0).$$

Далее имеем

$$x \cdot \mathrm{sg}(y) = \mathrm{CRN}_{z \sqsubset x}(0,\ 0,\ \mathrm{sg}(y)), \qquad x \cdot \overline{\mathrm{sg}}(y) = x \cdot \mathrm{sg}(\overline{\mathrm{sg}}(y)).$$

Через $z \sqsubseteq_b y$ будем обозначать предикат, истинный в тех и только в тех случаях, когда двоичное представление z является префиксом двоичного представления y (считаем, что в двоичных представлениях нет незначащих нулей; для любого x считаем верным $0 \sqsubseteq_b x$).

Далее мы будем пользоваться другим способом задания операции CRN. Пусть функция h принимает только значения 0 и 1. Обозначим

$$\mathop{\mathrm{Con}_b}_{z \sqsubseteq_b y} h(\bar{x},\ z) = h(\bar{x},\ z_0) \circ \ldots \circ h(\bar{x},\ z_k),$$

где $z_0, \ldots, z_k$ — все префиксы двоичной записи числа y без незначащих нулей (включая «пустое слово» $z_0 = 0$ и $z_k = y$) в порядке возрастания длины,

а $h_0 \circ \ldots \circ h_k$ означает составление двоичной записи числа путем соединения символов $h_0, \ldots, h_k$ в единое слово.

Нетрудно видеть, что применение операции Con_b к функции $h(\bar{x},\, z)$ эквивалентно применению операции CRN к функциям $h(\bar{x},\, 0)$, $h(\bar{x},\, s_0(z))$, $h(\bar{x},\, s_1(z))$. Поэтому операция Con_b не выводит за пределы A_0.

Для любых предикатов $\rho_1(\bar{x}), \rho_2(\bar{x})$ верно

$$\chi_{\neg\rho_1}(\bar{x}) = \overline{\mathrm{sg}}(\chi_{\rho_1}(\bar{x})), \qquad \chi_{\rho_1\vee\rho_2}(\bar{x}) = \mathrm{sg}(\chi_{\rho_1}(\bar{x}) *_b \chi_{\rho_2}(\bar{x})).$$

Поэтому множество предикатов с характеристическими функциями из A_0 замкнуто относительно операций логики высказываний. Обозначим

$$(\exists z)_{z\sqsubseteq_b y}\rho(\bar{x},\, z) \equiv \rho(\bar{x},\, z_0) \vee \ldots \vee \rho(\bar{x},\, z_k),$$

где $z_0, \ldots, z_k$ — префиксы двоичной записи числа y без незначащих нулей (включая «пустое слово» $z_0 = 0$ и $z_k = y$). Предикат $(\forall z)_{z\sqsubseteq_b y}\rho(\bar{x},\, z)$ определяется аналогично. Нетрудно видеть, что для $\tau(\bar{x},\, y) \equiv (\exists z)_{z\sqsubseteq_b y}\rho(\bar{x},\, z)$ верно

$$\chi_\tau(\bar{x},\, y) = \mathrm{sg}\left(\operatorname*{Con}_{z\sqsubseteq_b y} \chi_\rho(\bar{x},\, z)\right),$$

а $(\forall z)_{z\sqsubseteq_b y}\rho(\bar{x},\, z) \equiv \neg(\exists z)_{z\sqsubseteq_b y}\neg\rho(\bar{x},\, z)$. Поэтому класс предикатов с характеристическими функциями из A_0 замкнут относительно навешивания кванторов $(\exists z)_{z\sqsubseteq_b y}$ и $(\forall z)_{z\sqsubseteq_b y}$.

Пусть $\mathrm{shr}(x, y) = \lfloor x/2^y \rfloor$ — число, которое получается из x удалением y последних символов двоичной записи (битовый сдвиг вправо). Имеем

$$\mathrm{shr}(x, y) = \operatorname*{Con}_{z\sqsubseteq_b x} \mathrm{bit}(z, y).$$

Далее

$$\mathrm{len}(x) \dot{-} y = \mathrm{len}(\mathrm{shr}(x,\, y)).$$

Теперь можно задать характеристическую функцию предиката $(\mathrm{len}(x) \leqslant y)$:

$$\chi_{\mathrm{len}\leqslant}(x, y) \equiv \overline{\mathrm{sg}}(\mathrm{len}(x) \dot{-} y).$$

Пусть $\mathrm{rev}(x)$ есть число, получающееся при записывании цифр двоичной записи x (без незначащих нулей) в обратном порядке. Имеем

$$\mathrm{rev}(x) = \mathrm{shr}\left(\operatorname*{Con}_{z\sqsubseteq_b x} \mathrm{bit}(x,\, \mathrm{len}(z)),\ \ 1\right).$$

При помощи обращения rev и отсечения младших битов shr построим функцию отсечения старших битов (чтобы младшие нули не стали незначащими после обращения, нужно дописать в конец двоичной записи числа единицу):

$$\mathrm{rm}(x,\, 2^y) = \mathrm{shr}(\mathrm{rev}(\mathrm{shr}(\mathrm{rev}(s_1(x)),\, \mathrm{len}(x) \dot{-} y)),\ \ 1).$$

3.2. Индуктивное описание класса $\mathbf{AC^0}$. Напомним, что $\mathrm{FO} = \mathrm{AC}^0 = \mathrm{LH}$ и

$$A_0 = [\,0,\quad I_1^1(x),\quad s_0(x),\quad s_1(x),\quad \mathrm{len}(x),\quad \mathrm{bit}(x, y),\quad 2^{\mathrm{len}(x)\cdot\mathrm{len}(y)}\,]_{\mathrm{S,\,CRN}}.$$

Т е о р е м а 5 [34].

$$\mathrm{FO} = A_0.$$

Д о к а з а т е л ь с т в о. Докажем включение $A_0 \subseteq \mathrm{FO}$. Используя результаты и технику из § 1, нетрудно установить, что класс FO содержит функцию $\mathrm{shr}(x, y)$ и все исходные функции A_0, а также замкнут относительно суперпозиции.

Пусть $g, h_0, h_1 \in \mathrm{FO}$ и $f = \mathrm{CRN}(g, h_0, h_1)$. Классу FO_* принадлежат предикаты

$$\rho_0(\bar{x}, y, z) \equiv (h_0(\bar{x}, \mathrm{shr}(y, z)) = 1), \qquad \rho_1(\bar{x}, y, z) \equiv (h_1(\bar{x}, \mathrm{shr}(y, z)) = 1).$$

Пусть натуральное число k таково, что $\mathrm{len}(g(\bar{x})) \leqslant (\max(\mathrm{len}(x_1), \ldots, \mathrm{len}(x_m), 2))^{k-1}$ и функция g k-FO-определяется предикатом $G(\bar{X};\ i)$ (см. утверждение 3 из § 1), а предикаты ρ_0 и ρ_1 k-FO-определяются предикатами $R_0(\bar{X}, Y, Z)$ и $R_1(\bar{X}, Y, Z)$.

Пусть теперь $H_0(\bar{X}, Y;\ i)$ получается из $R_0(\bar{X}, Y, Z)$ путем замены всех элементарных формул $Z(j)$ на $\mathrm{BIT}(i, \sim j)$. Предикат H_1 определяется аналогично. Наконец, определим

$$\begin{aligned} F(\bar{X}, Y;\ i) \equiv (\exists i')[(i' = (\mathrm{len}(Y) \dot{-} 1) \dot{-} i)\ \&\ \\ \&\ ((i < \mathrm{len}(Y))\ \&\ (\neg Y(i')\ \&\ H_0(\bar{X}, Y;\ i+1) \vee Y(i')\ \&\ H_1(\bar{X}, Y;\ i+1)) \vee \\ \vee\ (i \geqslant \mathrm{len}(Y))\ \&\ (\exists j)[(j = i \dot{-} \mathrm{len}(Y))\ \&\ G(\bar{X};\ j)])]. \end{aligned}$$

Нетрудно видеть, что по утверждению 3 из § 1 предикат $F(\bar{X}, Y;\ i)$ k-FO-определяет функцию $f(\bar{x}, y)$. Используемые при его задании предикаты, оперирующие с $\mathrm{len}(Y)$, k-FO-определимы при любом k.

Перейдем к доказательству $\mathrm{FO} \subseteq A_0$. Пусть предикат $\rho(\bar{x}) \in \mathrm{FO}_*$ задается FO-формулой $\Phi(\bar{X})$. Преобразуем формулу Φ в формулу, задающую предикат с помощью выразимых в A_0 операций.

Обозначим $q(\bar{x}) = x_1 *_b \ldots *_b x_n *_b 2$ и заменим вхождения $(\exists y_i)$ на $(\exists y_i)_{y_i \sqsubseteq_b q(\bar{x})}$, а $(\forall y_i)$ на $(\forall y_i)_{y_i \sqsubseteq_b q(\bar{x})}$. Заменим все вхождения «коротких» переменных y_i на выражения $\mathrm{len}(y_i)$, а вхождения «длинных» переменных $X_i(y_j)$ на $\mathrm{BIT}(x_i, \mathrm{len}(q(\bar{x})) \dot{-} \mathrm{len}(y_j))$.

Полученная формула реализует предикат ρ и содержит выражения $(\mathrm{len}(y_i) \leqslant \mathrm{len}(y_j))$, $\mathrm{BIT}(\mathrm{len}(y_i), \mathrm{len}(y_j))$ и $\mathrm{BIT}(x_i, \mathrm{len}(q(\bar{x})) \dot{-} \mathrm{len}(y_j))$, которые соединяются логическими связками и навешиванием кванторов вида $(\exists y_i)_{y_i \sqsubseteq_b q(\bar{x})}$ и $(\forall y_i)_{y_i \sqsubseteq_b q(\bar{x})}$.

Характеристические функции предикатов $\mathrm{BIT}(x, y)$ и $(\mathrm{len}(y_i) \leqslant \mathrm{len}(y_j))$, а также функции $\mathrm{len}(x)$ и $\mathrm{len}(x) \dot{-} \mathrm{len}(y)$ принадлежат A_0. Функция $q(\bar{x})$ получается с помощью $x *_b y$. Класс предикатов, характеристические функции которых выходят в A_0, замкнут относительно применения логических связок и навешивания кванторов вида $(\exists y)_{y \sqsubseteq_b x}$ и $(\forall y)_{y \sqsubseteq_b x}$, поэтому характеристическая функция предиката $\rho(\bar{x})$ принадлежит A_0.

Осталось средствами A_0 построить произвольную функцию $f(\bar{x})$ из FO. Из доказанного выше следует, что функция $\mathrm{bit}(f(\bar{x}), y)$ принадлежит A_0. Ясно, что $f(\bar{x}) \leqslant p(\bar{x})$, где $p(\bar{x})$ — некоторая функция, полученная суперпозициями констант и функций $2^{\mathrm{len}(x) \cdot \mathrm{len}(y)}$. Тогда

$$f(\bar{x}) = \operatorname*{Con}_{y \sqsubseteq_b p(\bar{x})} \mathrm{bit}(f(\bar{x}), \mathrm{len}(p(\bar{x})) \dot{-} \mathrm{len}(y)).$$

В [36] также имеется доказательство $A_0 = \mathrm{LH}$ без использования формализма FO.

3.3. Индуктивное описание класса TC^0. Напомним, что $\mathrm{FOM} = \mathrm{TC}^0$, и обозначим

$$T_0 = [\, xy, \quad 0, \quad I_1^1(x), \quad s_0(x), \quad s_1(x), \quad \mathrm{len}(x), \quad \mathrm{bit}(x,y), \quad 2^{\mathrm{len}(x)\cdot\mathrm{len}(y)} \,]_{\mathrm{S,\,CRN}}.$$

Класс функций T_0 получается добавлением функции xy во множество исходных функций класса A_0.

Теорема 6 [35].

$$\mathrm{FOM} = T_0.$$

Доказательство. Из результатов § 1 следует, что $xy \in \mathrm{FOM}$. Остальные исходные функции принадлежат $\mathrm{FO} \subseteq \mathrm{FOM}$, и класс FOM замкнут относительно суперпозиции. Доказательство замкнутости относительно операции CRN в точности повторяет аналогичное доказательство для класса FO из теоремы 5. Таким образом, $T_0 \subseteq \mathrm{FOM}$.

Для доказательства обратного включения сначала покажем, что

$$\mathrm{bitsum}(x) \in T_0.$$

Воспользуемся техникой подсчета количества единиц в двоичной записи числа при помощи умножения [40], адаптировав ее для унифицированного случая. Обозначим через $\mathrm{incr}(x,l)$ функцию, которая при $x = (x_{n-1}\ldots x_0)_2$ и $l \geqslant 1$ удовлетворяет соотношению

$$\mathrm{incr}(x,l) = \sum_{i=0}^{n-1} x_i 2^{li} = (x_{n-1} \underbrace{0\ldots0}_{l-1} x_{n-2} \ldots x_2 \underbrace{0\ldots0}_{l-1} x_1 \underbrace{0\ldots0}_{l-1} x_0)_2.$$

Напомним, что $\lfloor x/2^y \rfloor = \mathrm{shr}(x,y)$. Имеем $2^{\mathrm{len}(x)} = 2^{\mathrm{len}(x)\cdot\mathrm{len}(1)}$ и

$$\mathrm{incr}(x,\ 2^{\mathrm{len}(\mathrm{len}(y))}) = \\ = \mathrm{rev}\left(\operatorname*{Con}_b \limits_{z \sqsubseteq \exp(x,y)} \mathrm{bit}(x,\ \lfloor \mathrm{len}(z)/2^{\mathrm{len}(\mathrm{len}(y))} \rfloor) \cdot \overline{\mathrm{sg}}(\mathrm{rm}(\mathrm{len}(z),\ 2^{\mathrm{len}(\mathrm{len}(y))}))\right),$$

где $\exp(x,y) = \mathrm{shl}(2^{\mathrm{len}(x)\cdot\mathrm{len}(y)},\ \mathrm{len}(2))$. Такой выбор сделан, поскольку $2^{\mathrm{len}(\mathrm{len}(y))} \leqslant 2 \cdot \mathrm{len}(y)$.

Положим далее $l = 2^{\mathrm{len}(\mathrm{len}(x))}$, $n = \mathrm{len}(x)$ и $A = \mathrm{incr}(x,\ l) \cdot \mathrm{incr}(\mathrm{ones}_b(x),\ l)$. Преобразуем A следующим образом:

$$\left(\sum_{i=0}^{n-1} x_i 2^{il}\right) \cdot \left(\sum_{j=0}^{n-1} 2^{jl}\right) = \sum_{0 \leqslant i,j < n} x_i 2^{(i+j)l} = \sum_{p=0}^{2n-2} \left(\sum_{\substack{0 \leqslant i,j < n \\ i+j=p}} x_i\right) 2^{pl} = \sum_{p=0}^{2n-2} a_p 2^{pl}.$$

Нетрудно видеть, что коэффициенты a_p не превосходят $n < l < 2^l$. При этом

$$a_{n-1} = \sum_{\substack{0 \leqslant i,j < n \\ i+j=n-1}} x_i = \sum_{i=0}^{n-1} x_i = \mathrm{bitsum}(x).$$

Для выделения этого числа воспользуемся делением на степени двойки. При $x > 0$

$$\mathrm{bitsum}(x) = \mathrm{rm}(\lfloor A/2^{l(n-1)} \rfloor,\ 2^l),$$

где

$$l(n-1) = (\mathrm{len}(x) - 1) \cdot 2^{\mathrm{len}(\mathrm{len}(x))} = \mathrm{shl}(\mathrm{len}(\mathrm{shr}(x,\ 1)),\ \mathrm{len}(\mathrm{len}(x))).$$

При $x = 0$ полученная формула для $\mathrm{bitsum}(x)$ дает 0.

Обозначим $\mathrm{MAJ}(x)$ предикат, истинный только в том в случае, если более половины двоичных разрядов x (без учета незначащих нулей) содержат единицы. Пусть $\mathrm{maj}(x)$ — характеристическая функция предиката $\mathrm{MAJ}(x)$. Имеем

$$\mathrm{MAJ}(x) \equiv (\mathrm{bitsum}(x) > \lfloor \mathrm{len}(x)/2 \rfloor) \equiv (\mathrm{len}(\mathrm{shl}(1, \mathrm{shr}(\mathrm{len}(x), 1))) \leqslant \mathrm{bitsum}(x)).$$

Таким образом, $\mathrm{maj}(x) \in T_0$. Введем теперь префиксный мажоритарный квантор:

$$(Mz)_{z \sqsubseteq_b y} \rho(\bar{x},\ z) \equiv \mathrm{MAJ}(\underset{z \sqsubseteq_b y}{\mathrm{Con}_b}\ \chi_\rho(\bar{x},\ z)).$$

Из определения видно, что класс предикатов с характеристическими функциями из T_0 замкнут относительно навешивания этого квантора.

Дальнейшее доказательство включения $\mathrm{FOM} \subseteq T_0$ повторяет доказательство включения $\mathrm{FO} \subseteq A_0$, только при перестроении FOM-формулы, задающей предикат из FOM, вхождения (My_i) заменяются на $(My_i)_{y_i \sqsubseteq_b q(\bar{x})}$.

Имеем $x+y$, $x \dot{-} y \in \mathrm{FO} = A_0$ и $xy = \mathrm{shr}((x+y)^2 \dot{-} (x^2+y^2),\ 1)$. Кроме того, вычисление функции xy можно свести к вычислению $\lfloor x/y \rfloor$ (см. утверждение 10 из § 4). Отсюда и из хода доказательства теоремы 6 получаем следствие.

Следствие 3. *Пусть*

$$A = \{0,\ \ I_1^1(x),\ \ s_0(x),\ \ s_1(x),\ \ \mathrm{len}(x),\ \ \mathrm{bit}(x, y),\ \ 2^{\mathrm{len}(x) \cdot \mathrm{len}(y)}\}.$$

Класс TC^0 *совпадает со следующими классами функций*:

$$[\ A,\ \mathrm{maj}(x)\ ]_{\mathrm{S,\,CRN}}, \qquad [\ A,\ \mathrm{bitsum}(x)\ ]_{\mathrm{S,\,CRN}}, \qquad [\ A,\ xy\ ]_{\mathrm{S,\,CRN}},$$
$$[\ A,\ x^2\ ]_{\mathrm{S,\,CRN}}, \qquad [\ A,\ \lfloor x/y \rfloor\ ]_{\mathrm{S,\,CRN}}, \qquad [\ A,\ x^{\lfloor \log_2 y \rfloor}\ ]_{\mathrm{S,\,CRN}}.$$

Для других классов (например, $\mathrm{AC}^0(2)$, $\mathrm{AC}^0(6)$, NC^1) тоже существуют индуктивные описания, использующие операцию CRN и другие виды рекурсии. Некоторые из них получены в [35], а исчерпывающий обзор результатов имеется в [36]. В [50] получены описания классов AC^0, TC^0 и NC^1 с помощью варианта операции CRN, применяющегося только к одноместным функциям, а также получены индуктивные описания множеств одноместных функций из этих классов.

3.4. Операция ограниченной префиксной конкатенации и класс BPC. Будем рассматривать слова в алфавите $\{1, 2\}$, включая пустое слово Λ. Произвольному непустому слову $a_n a_{n-1} \ldots a_1$ $(a_i \in \{1, 2\})$ сопоставим число

$$\nu(a_n a_{n-1} \ldots a_1) = \sum_{i=1}^{n} a_i 2^{i-1},$$

слову Λ сопоставим число $\nu(\Lambda) = 0$. Нетрудно видеть, что отображение ν устанавливает взаимно-однозначное соответствие между множеством $\{1,2\}^*$ всех слов в алфавите $\{1,2\}$ и множеством $\mathbb{N}_0$. Слово $a_n a_{n-1} \ldots a_1$ будем называть диадическим представлением числа $\nu(a_n a_{n-1} \ldots a_1)$. Отображение ν позволяет говорить о словарных функциях на множестве $\{1,2\}^*$ как о числовых функциях на множестве $\mathbb{N}_0$. Отметим при этом, что словарные константы $1, 2$ совпадают с числовыми константами $1, 2$.

Будем использовать обозначение $|x|$ для длины слова x (с учетом описанного выше соглашения имеем $|x| = \lfloor \log_2(x+1) \rfloor$). На множестве $\{1,2\}^*$ рассматриваем бинарную операцию конкатенации $*$: выражение $x*y$ обозначает слово, которое получается из слова x приписыванием справа слова y. Через $x \sqsubseteq y$ обозначаем предикат, истинный только в тех случаях, когда x является префиксом y (в т. ч. при $x = \Lambda$ и $x = y$).

Пусть $g(\bar{x}, z)$ — функция, заданная на $\{1,2\}^*$. Будем говорить, что функция $f(\bar{x}, y)$ получена из функции $g(\bar{x}, z)$ с помощью операции *ограниченной префиксной конкатенации* ($\mathrm{CON}_{\sqsubseteq}$), если

$$f(\bar{x}, y) = \operatorname*{Con}_{z \sqsubseteq y} g(\bar{x}, z),$$

где правая часть равенства есть сокращение для выражения

$$g(\bar{x}, z_0) * g(\bar{x}, z_1) * \ldots * g(\bar{x}, z_k),$$

а $z_0, \ldots, z_k$ — все префиксы слова y (включая пустое слово $z_0 = \Lambda$ и слово $z_k = y$) в порядке возрастания длины.

Отметим, что операция $\mathrm{CON}_{\sqsubseteq}$ является более сильной, чем операция CRN: она допускает случай, когда некоторые слова $g(\bar{x}, z_i)$ являются пустыми и полностью исключаются из записи результата; операция CRN же составляет результат посимвольно без пропусков. Как будет видно далее, это различие является существенным.

Введем словарный вариант усеченной разности:

$$x \ominus y = \begin{cases} z, & \text{если } x = z * y, \\ \Lambda & \text{в противном случае} \end{cases}$$

и определим класс функций

$$\mathrm{BPC} = [\, 1, \quad 2, \quad x * y, \quad x \ominus y \,]_{\mathrm{S},\, \mathrm{CON}_{\sqsubseteq}}.$$

Через BPC_* обозначаем класс предикатов с характеристическими функциями из BPC.

Указанные далее результаты о принадлежности функций классу BPC в основном получены в [12] (см. также [14, 21]).

Имеем $\Lambda = x \ominus y$, остальные константы получаются суперпозициями функций 1, 2, $x*y$. Селекторные функции получаются с помощью $x*\Lambda$. Далее

$$\overline{\mathrm{sg}}(x) = 1 \ominus x, \qquad \mathrm{sg}(x) = \overline{\mathrm{sg}}(\overline{\mathrm{sg}}(x)),$$

$$x \cdot \mathrm{sg}(y) = ((x * 2) \ominus \overline{\mathrm{sg}}(y)) \ominus 2, \qquad x \cdot \overline{\mathrm{sg}}(y) = ((x * 2) \ominus \mathrm{sg}(y)) \ominus 2.$$

Пусть $\mathrm{ones}(x) = 2^{|x|} - 1$ выдает слово длины $|x|$, состоящее только из единиц, а $\mathrm{last}(x)$ выдает последний символ слова x (Λ при $x = \Lambda$). Имеем

$$\mathrm{ones}(x) = \left(\operatorname*{Con}_{z \sqsubseteq x} 1 \right) \ominus 1, \qquad \mathrm{last}(x) = (1 \cdot \mathrm{sg}(x \ominus 1)) * (2 \cdot \mathrm{sg}(x \ominus 2)).$$

Класс BPC$_*$ замкнут относительно операций логики высказываний:

$$\chi_{\neg\rho_1}(\bar{x}) = \overline{\mathrm{sg}}(\chi_{\rho_1}(\bar{x})), \qquad \chi_{\rho_1\vee\rho_2}(\bar{x}) = \mathrm{sg}(\chi_{\rho_1}(\bar{x}) * \chi_{\rho_2}(\bar{x})).$$

Обозначим

$$(\exists z)_{z\sqsubseteq y}\rho(\bar{x},\ z) \equiv \rho(\bar{x},\ z_0) \vee \ldots \vee \rho(\bar{x},\ z_k),$$

где $z_0, \ldots, z_k$ — все префиксы слова y (включая пустое слово $z_0 = \Lambda$ и слово $z_k = y$). Предикат $(\forall z)_{z\sqsubseteq y}\rho(\bar{x},\ z)$ определяется аналогично. Для $\tau(\bar{x},\ y) \equiv (\exists z)_{z\sqsubseteq y}\rho(\bar{x},\ z)$ верно

$$\chi_\tau(\bar{x},\ y) = \mathrm{sg}\left(\operatorname*{Con}_{z\sqsubseteq y}\chi_\rho(\bar{x},\ z)\right),$$

а $(\forall z)_{z\sqsubseteq y}\rho(\bar{x},\ z) \equiv \neg(\exists z)_{z\sqsubseteq y}\neg\rho(\bar{x},\ z)$. Поэтому класс BPC$_*$ замкнут относительно навешивания кванторов $(\exists z)_{z\sqsubseteq y}$ и $(\forall z)_{z\sqsubseteq y}$.

Для предикатов $(x = y)$ и $(|x| = |y|)$ имеем

$$\chi_=(x, y) = \mathrm{sg}((1 * x) \ominus y) \cdot \mathrm{sg}((1 * y) \ominus x), \qquad (|x| = |y|) \equiv (\mathrm{ones}(x) = \mathrm{ones}(y)).$$

Далее выражаем предикат $(|x| \leqslant |y|)$:

$$\chi_{||\leqslant||}(x, y) = \overline{\mathrm{sg}}(\mathrm{ones}(x) \ominus \mathrm{ones}(y)).$$

Через указанные предикаты нетрудно выразить и другие отношения $\neq, <, >, \geqslant$ длин слов.

Обозначим $\mathrm{symb}(x, y)$ y-й слева символ слова x (нумерация с единицы; число y имеет диадическое представление) или Λ, если символа с таким номером в слове x нет. Пусть $\chi_{||=||}$ — характеристическая функция предиката $(|x| = |y|)$. Имеем

$$\mathrm{symb}(x,\ |y|) = \operatorname*{Con}_{z\sqsubseteq x}(\mathrm{last}(z) \cdot \mathrm{sg}(\chi_{||=||}(z, y))).$$

Обозначим через $\mathrm{inv}(x)$ функцию, которая дает инверсию слова x (т. е. слово, составленное из символов слова x, записанных в обратном порядке). Имеем

$$\mathrm{inv}(x) = \operatorname*{Con}_{z\sqsubseteq \mathrm{ones}(x)}(\mathrm{symb}(x,\ |\mathrm{ones}(x) \ominus z|)).$$

Пусть $\rho(\bar{x},\ z)$ — предикат натурального аргумента. Обозначим

$$(\exists z)_{z\leqslant y}\rho(\bar{x},\ z) \equiv (\exists z)[(z \leqslant y)\ \&\ \rho(\bar{x},\ z)],$$
$$(\forall z)_{z\leqslant y}\rho(\bar{x},\ z) \equiv (\forall z)[(z \leqslant y) \to \rho(\bar{x},\ z)].$$

Класс BA ограниченно арифметических предикатов определяется как класс предикатов натурального аргумента, содержащий предикаты $(z = x + y)$ и $(z = x \cdot y)$ и замкнутый относительно перестановки и отождествления переменных, введения фиктивных переменных, подстановки констант, применения операций логики высказываний и навешивания кванторов $(\exists z)_{z\leqslant y}$ и $(\forall z)_{z\leqslant y}$.

Класс BA хорошо изучен, он имеет эквивалентное словарное определение (класс рудиментарных предикатов), и ему принадлежит большое количество используемых на практике отношений [14, 36]. Этот класс также тесно связан к классом FO-определимых предикатов, не использующих обращений к «длинным» переменным [46] (технические различия определений не позволяют говорить о совпадении).

У т в е р ж д е н и е 9. *Пусть* $\rho(x_1, \ldots, x_n) \in \mathrm{BA}$. *Тогда выполнено* $\rho(|x_1|, \ldots, |x_n|) \in \mathrm{BPC}_*$.

Д о к а з а т е л ь с т в о. Имеем

$$(|z| = |x| + |y|) \equiv (|z| = |x * y|), \qquad (|z| = |x| \cdot |y|) \equiv (|z| = |\operatorname*{Con}_{z \sqsubseteq y}(x \cdot \mathrm{sg}(z))|).$$

Кроме того,

$$(\exists z)_{z \leqslant |y|} \rho(|x_1|, \ldots, |x_n|,\ z) \equiv (\exists z)_{z \sqsubseteq y} \rho(|x_1|, \ldots, |x_n|,\ |z|).$$

Аналогичное равенство верно и для $(\forall z)_{z \leqslant |y|}$.

Пусть Φ — формула, выражающая предикат $\rho(x_1, \ldots, x_n)$ из BA. Заменяя все вхождения свободных переменных x_i на $|x_i|$, получим формулу, выражающую $\rho(|x_1|, \ldots, |x_n|)$. Последовательно заменяя кванторы, согласно указанному выше равенству, получим формулу, состоящую из предикатов вида $(|z| = |x| + |y|)$ и $(|z| = |x| \cdot |y|)$ (возможно, с константами на местах переменных), которые соединены операциями логики высказываний и навешиванием кванторов $(\exists z)_{z \sqsubseteq y}$ и $(\forall z)_{z \sqsubseteq y}$. Эта формула задает предикат из BPC_*.

Л е м м а 4 [21]. *Функция* $|x|$ *принадлежит классу* BPC.

Д о к а з а т е л ь с т в о. Известно, что предикаты $(y = 2^x)$ и $(z = x * y)$ принадлежат классу BA (см. [14] или [36]). Используя стандартную для класса BA технику (см. [14]), нетрудно получить, что $(y = |x|) \equiv (y = \lfloor \log_2(x+1) \rfloor) \in \mathrm{BA}$. Далее при $a = 1, 2$ имеем

$$(\mathrm{symb}(x, y) = a) \equiv (\exists u)_{u \leqslant x} (\exists u')_{u' \leqslant x} (\exists v)_{v \leqslant x} [(u' = u * a) \& (|u'| = y) \& (x = u' * v)] \in \mathrm{BA}.$$

Тогда предикат $(\mathrm{symb}(|x|,\ |y|) = a)$ принадлежит BPC_*. Обозначим его характеристическую функцию χ_a. Имеем

$$\mathrm{symb}(|x|,\ |y|) = (1 \cdot \mathrm{sg}(\chi_1(x, y))) * (2 \cdot \mathrm{sg}(\chi_2(x, y))).$$

Наконец, получаем

$$|x| = \operatorname*{Con}_{z \sqsubseteq x}(\mathrm{symb}(|x|,\ |z|)).$$

Докажем теперь, что функция $\mathrm{symb}(x, y)$ принадлежит BPC. Введем вспомогательную функцию

$$\mathrm{expand}(x, y) = \begin{cases} \underbrace{1 \ldots 1}_{x}, & \text{если } x \leqslant |y|, \\ \Lambda & \text{в ином случае.} \end{cases}$$

Из доказанного выше следует, что предикат $(x = |y|)$ принадлежит BPC_*. Пусть χ_l — его характеристическая функция. Имеем

$$\mathrm{expand}(x, y) = \operatorname*{Con}_{z \sqsubseteq y}(\mathrm{ones}(z) \cdot \mathrm{sg}(\chi_l(x, z))).$$

Тогда

$$\mathrm{symb}(x, y) = \mathrm{symb}(x,\ |\mathrm{expand}(y, x)|).$$

3.5. Совпадение классов BPC и TC^0.

Теорема 7 [21]. *Классы* BPC *и* TC^0 *совпадают.*

Доказательство. Сначала покажем, что $TC^0 \subseteq BPC$. Нам потребуется показать, что функции $x+1$ и $x \dot{-} 1$ принадлежат классу BPC. Это можно сделать путем выполнения арифметических действий в столбик:

$$\mathrm{CARRY}_+(x,\ |y|) \equiv (\forall z)_{z \sqsubseteq x}[(|z| > |y|) \to (\mathrm{symb}(x,\ |z|) = 2)],$$
$$\mathrm{CARRY}_-(x,\ |y|) \equiv (\forall z)_{z \sqsubseteq x}[(|z| > |y|) \to (\mathrm{symb}(x,\ |z|) = 1)].$$

Пусть $\mathrm{carry}_+(x,\ |y|)$, $\mathrm{carry}_-(x,\ |y|)$ — характеристические функции введенных предикатов. Таблично определим функцию, которая производит «прибавление» переноса к символу:

$$\mathrm{addcarry}(x,y) = 1 \cdot \mathrm{sg}(\chi_=(x,1)) \cdot \mathrm{sg}(\chi_=(y,\Lambda)) * 2 \cdot \mathrm{sg}(\chi_=(x,1)) \cdot \mathrm{sg}(\chi_=(y,1)) *$$
$$* 2 \cdot \mathrm{sg}(\chi_=(x,2)) \cdot \mathrm{sg}(\chi_=(y,\Lambda)) * 1 \cdot \mathrm{sg}(\chi_=(x,2)) \cdot \mathrm{sg}(\chi_=(y,1)).$$

Тогда

$$x+1 = \mathrm{carry}_+(x,\ |\Lambda|) * \operatorname*{Con}_{z \sqsubseteq x}\big((\mathrm{addcarry}(\mathrm{symb}(x,\ |z|),\ \mathrm{carry}_+(x,\ |z|))\big),$$
$$x \dot{-} 1 = \operatorname*{Con}_{z \sqsubseteq x}\big(\mathrm{addcarry}(\mathrm{symb}(x,\ |z|),$$
$$\mathrm{carry}_-(x,\ |z|)) \cdot \mathrm{sg}(\chi_{||\neq||}(z,1) * \overline{\mathrm{sg}}(\mathrm{carry}_-(x,\ |\Lambda|)))\big),$$

где $\chi_{||\neq||}$ — характеристическая функция предиката $|x| \neq |y|$.

Напомним, что функции класса BPC работают с диадическими представлениями чисел, а функции класса TC^0 — с двоичными. Будем кодировать двоичное представление числа x словом из $\{1,2\}^*$, получающимся заменой в этом представлении символов 0 на символы 2. Обозначим через $\mathrm{bin}(x)$ код двоичного представления числа x (без незначащих нулей). Эта функция переводит диадическое представление числа x в двоичное. Через $\mathrm{diad}(x)$ обозначим обратную функцию к bin, т. е. функцию, которая при любом $n \geqslant 0$ удовлетворяет соотношению

$$\mathrm{diad}(\underbrace{2 * \ldots * 2}_{n} * \mathrm{bin}(x)) = x.$$

Обозначим $\mathrm{norm}(x)$ функцию, отсекающую символы 2, стоящие в начале слова x (до первой единицы или конца слова). Имеем

$$\mathrm{norm}(x) = \operatorname*{Con}_{z \sqsubseteq x}(\mathrm{symb}(x,\ |z|) \cdot \mathrm{sg}(\chi_\rho(x,z))),$$

где

$$\rho(x,z) \equiv (\exists u)_{u \sqsubseteq z}(\mathrm{symb}(x,|u|) = 1).$$

Заметим, что для получения двоичного кода числа по диадическому достаточно вычесть 1 из числа, сделать одновременную замену символов $2 \leftrightarrow 1$ и приписать слева 1 (за исключением числа 0, для которого в обоих случаях код равен Λ). Эта процедура автоматически исключает незначащие нули. Пусть

$$\mathrm{change}(x) = 2 \cdot \mathrm{sg}(\chi_=(x,1)) * 1 \cdot \mathrm{sg}(\chi_=(x,2)).$$

Тогда

$$\mathrm{bin}(x) = \mathrm{sg}(x) * \underset{z \sqsubseteq x \dot{-} 1}{\mathrm{Con}} (\mathrm{change}(\mathrm{symb}(x \dot{-} 1,\ |z|))).$$

Функция diad определяется аналогично с использованием функции $x+1$:

$$\mathrm{diad}(x) = \left(\underset{z \sqsubseteq n(x)}{\mathrm{Con}} \left(\mathrm{change}(\mathrm{symb}(n(x),\ |z|)) \cdot \mathrm{sg}(\chi_{||\neq||}(z, 1))\right) + 1 \right) \cdot \mathrm{sg}(n(x)),$$

где $n(x)$ — сокращение для $\mathrm{norm}(x)$.

Покажем принадлежность классу BPC всех исходных функций класса TC^0 (используем множество исходных функций из следствия 3, содержащее функцию $\mathrm{bitsum}(x)$).

Ясно, что 0, $I_1^1(x) \in \mathrm{BPC}$. Затем имеем $s_0(x) = \mathrm{diad}(\mathrm{bin}(x) * 2)$, $s_1(x) = \mathrm{diad}(\mathrm{bin}(x) * 1)$, $\mathrm{len}(x) = |\mathrm{bin}(x)|$. Далее, $\mathrm{bit}(x, y) = (1 * \mathrm{symb}(\mathrm{inv}(\mathrm{bin}(x)),\ y+1)) \ominus 1$. Наконец, имеем

$$2^{\mathrm{len}(x) \cdot \mathrm{len}(y)} = \left(\underset{z \sqsubseteq \mathrm{bin}(x)}{\mathrm{Con}} \mathrm{ones}(\mathrm{bin}(y)) \cdot \mathrm{sg}(\chi_{||\neq||}(z, \Lambda)) \right) + 1,$$

$$\mathrm{bitsum}(x) = \left| \underset{z \sqsubseteq \mathrm{bin}(x)}{\mathrm{Con}} \chi_{=}(\mathrm{symb}(x,\ |z|),\ 1) \right|.$$

Осталось показать замкнутость класса BPC относительно операции CRN. Пусть функции $g(\bar{x})$, $h_0(\bar{x}, z)$ и $h_1(\bar{x}, z)$ принадлежат классу BPC, причем функции h_0 и h_1 принимают только значения $\Lambda, 1$, а $f(\bar{x}, y) = \mathrm{CRN}_{z \sqsubset y}(g, h_0, h_1)$. Обозначим

$$h(\bar{x}, z) = (h_0(\bar{x}, \mathrm{diad}(z \ominus 2)) \cdot \mathrm{sg}(\chi_{=}(\mathrm{last}(z), 2))) * \\ * (h_1(\bar{x}, \mathrm{diad}(z \ominus 1)) \cdot \mathrm{sg}(\chi_{=}(\mathrm{last}(z), 1))),$$

$$h'(\bar{x}, z) = (1 \ominus h(\bar{x}, z)) + 1.$$

Тогда

$$f(\bar{x}, y) = \mathrm{diad}\left(\mathrm{bin}(g(\bar{x})) * \underset{z \sqsubseteq \mathrm{bin}(y)}{\mathrm{Con}} (h'(\bar{x}, z) \cdot \mathrm{sg}(\chi_{||\neq||}(z, \Lambda))) \right).$$

Докажем теперь, что $\mathrm{BPC} \subseteq \mathrm{TC}^0$. В этом случае нам нужно кодировать диадическую запись при помощи двоичного преставления чисел. Будем кодировать символ 1 словом 11, а символ 2 словом 10. Обозначим $\mathrm{diad}_b(x)$ число, двоичная запись которого получается путем конкатенации кодов символов диадического представления x. Эта функция переводит двоичное представление числа в код диадического представления. Через $\mathrm{bin}_b(x)$ обозначим обратную функцию к diad_b, т.е. функцию, которая удовлетворяет соотношению

$$\mathrm{bin}_b(\mathrm{diad}_b(x)) = x.$$

Обозначим $x[y] = \mathrm{bit}(s_0(x),\ \mathrm{len}(x)+1 \dot{-} y)$. Эта функция выдает y-й слева символ двоичной записи числа x (без незначащих нулей; нумерация символов с единицы). Если y-го символа в двоичной записи x не существует, функция выдает 0.

Для получения диадического представления по двоичному нужно прибавить к числу 1, сделать одновременную замену символов $0 \to 1$ и $1 \to 2$ и удалить первый символ. Учитывая соответствие $2 \leftrightarrow 10$ и $1 \leftrightarrow 11$, это можно выразить формулами

$$\mathrm{diad}_b(x) = \operatorname*{Con}_{z \sqsubseteq_b (x+1) *_b (x+1)} (\mathrm{len}(z) > 2) \,\&\, ((\mathrm{rm}(\mathrm{len}(z),\, 2) = 1) \vee \vee ((x+1)[\lfloor \mathrm{len}(z)/2 \rfloor] = 0)),$$

$$\mathrm{bin}_b(x) = \left(\operatorname*{Con}_{z \sqsubseteq_b 2 *_b x} (x \neq 0) \,\&\, (\mathrm{len}(z) > \lfloor \mathrm{len}(2 *_b x)/2 \rfloor) \,\&\, (\mathrm{get}(x, z) = 0) \right) \dot{-}\, 1,$$

где $\mathrm{get}(x, z) = (2 *_b x)[2 \cdot (\mathrm{len}(z) \dot{-} \lfloor \mathrm{len}(2 *_b x)/2 \rfloor)]$. Здесь $2 *_b x = (10)_2 *_b x$. Для сокращения записи в этих выражениях мы использовали предикаты на местах их характеристических функций.

Имеем $(x \sqsubseteq_b y) \equiv (\exists z)_{z \sqsubseteq_b y} (z = x)$ и $\mathrm{TAIL}(x, y) \equiv (\mathrm{rev}(s_1(x)) \sqsubseteq_b \mathrm{rev}(s_1(y)))$. Покажем, что все исходные функции класса BPC принадлежат классу AC^0. Для функций $1, 2$ это очевидно. Имеем $x * y = \mathrm{bin}_b(\mathrm{diad}_b(x) *_b \mathrm{diad}_b(y))$ и

$$x \ominus y = \mathrm{bin}_b(\mathrm{shr}(\mathrm{diad}_b(x),\, \mathrm{len}(\mathrm{diad}_b(y)))) \cdot \mathrm{sg}(\chi_{\mathrm{TAIL}}(\mathrm{diad}_b(y),\, \mathrm{diad}_b(x))).$$

Операцию $\mathrm{CON}_{\sqsubseteq}$ будем выражать в классе TC^0 с помощью суммирования и мультиплицирования по утверждению 5 и теореме 1 из § 1, используя технику из [15].

Выражаем $|x| = \lfloor \log_2(x+1) \rfloor$ и $2^{|x|} = 2^{\lfloor \log_2(x+1) \rfloor}$. Далее,

$$\mathrm{pref}(x, y) = \mathrm{bin}_b(\mathrm{shr}(\mathrm{diad}_b(y),\, 2(|x| \dot{-} y))) \cdot \mathrm{sg}(\chi_{\leqslant}(|x|,\, y))$$

есть префикс длины y диадического представления числа x (если такой существует).

Пусть $g(\bar{x},\, z)$ принадлежит классу TC^0 и

$$f(\bar{x},\, y) = \operatorname*{Con}_{z \sqsubseteq y} g(\bar{x},\, z).$$

Пусть $l(\bar{x},\, y, z)$ выражает экспоненту суммы значений $|g(\bar{x},\, u)|$, когда u — префикс y длины больше $|z|$. Имеем

$$l(\bar{x},\, y, z) = \prod_{i=0}^{|y|} 2^{|g(\bar{x},\, \mathrm{pref}(y, i)) \cdot \chi_{>}(i,\, |z|)|}.$$

Тогда, учитывая равенство $\mathrm{len}(\mathrm{diad}_b(x)) = 2|x|$, получаем

$$f(\bar{x},\, y) = \mathrm{bin}_b \left(\sum_{z=0}^{|y|} \mathrm{diad}_b(g(\bar{x},\, \mathrm{pref}(y, z))) \cdot (l(\bar{x},\, y, z))^2 \right).$$

В [21] включение $\mathrm{BPC} \subseteq \mathrm{TC}^0$ доказано иным способом, без обращения к нетривиальным результатам теоремы 1 из § 1. Приведенное там доказательство существенно использует только индуктивное описание класса TC^0 через операцию CRN и функцию $\mathrm{bitsum}(x)$, а использование в нем функций $\mathrm{rm}(x, y)$ и xy легко устранимо.

Класс BPC дает компактное и чисто словарное определение класса TC^0. Отметим, что все исходные функции класса BPC лежат в классе AC^0,

но $\mathrm{AC}^0 \subset \mathrm{BPC}$ (в силу теоремы 4 из § 2). Это значит, что операция $\mathrm{CON}_{\sqsubseteq}$ является более сильной, чем операция CRN, которая не выводит из класса AC^0.

В [12] (см. также [17]) доказано, что функции класса BPC вычислимы на нестирающих многоголовочных машинах Тьюринга с выходной лентой. В силу совпадения классов BPC и TC^0 эти машины задают еще один способ вычислять функции из класса TC^0 (хотя точные границы класса функций, вычисляемых этими машинами, на текущий момент неизвестны).

3.6. Операция ограниченной конкатенации и класс K^d. В этом разделе мы снова рассматриваем функции на множестве $\{1, 2\}^*$ и отождествляем их с функциями на множестве $\mathbb{N}_0$ с помощью диадического представления чисел.

Говорим, что функция $f(\bar{x},\ y)$ получена из функции $g(\bar{x},\ z)$ с помощью *операции ограниченной конкатенации* ($\mathrm{CON}_{\leqslant}$), если

$$f(\bar{x},\ y) = \operatorname*{Con}_{z \leqslant y} g(\bar{x},\ z),$$

где правая часть равенства есть сокращение для выражения

$$g(\bar{x},\ \Lambda) * g(\bar{x},\ 1) * \ldots * g(\bar{x},\ y),$$

а значения функции g выписаны в порядке возрастания параметра z (имеется в виду диадическое представление чисел). Определим класс функций

$$K^d = [\ 1,\quad 2,\quad x * y,\quad x \ominus y\ ]_{\mathrm{S},\ \mathrm{CON}_{\leqslant}}.$$

Нетрудно видеть, что класс K^d намного шире класса BPC. Покажем, что он совпадает с классом K функций, элементарных по Кальмару. Обозначим через $\mathrm{SUM}_{\leqslant}$ и $\mathrm{PROD}_{\leqslant}$ *операции ограниченного суммирования* и *ограниченного мультиплицирования*

$$\sum_{z \leqslant y} g(\bar{x},\ z), \qquad \prod_{z \leqslant y} g(\bar{x},\ z)$$

и определим класс K следующим образом:

$$K = [\ x + 1,\quad x \dot{-} y\ ]_{\mathrm{S},\ \mathrm{SUM}_{\leqslant},\ \mathrm{PROD}_{\leqslant}}.$$

Т е о р е м а 8 [9, 19].

$$K^d = K.$$

Д о к а з а т е л ь с т в о. Известно [57], что класс K совпадает с классом функций, вычислимых на машинах Тьюринга с объемом памяти $2^{2^{\cdot^{\cdot^{\cdot^{2^n}}}}}$, где n — длина двоичного входа, а высота степенной башни не зависит от n. При этих ограничениях на машине Тьюринга нетрудно промоделировать применение операции ограниченной конкатенации, поэтому $K^d \subseteq K$.

Докажем включение $K \subseteq K^d$. Будем пользоваться иным определением класса K:

$$K = [\ x + 1,\quad x \dot{-} y,\quad 2^x\ ]_{\mathrm{S},\ \mathrm{SUM}_{\leqslant}}.$$

Переход к данному определению хорошо известен и технически не очень сложен [14].

Принадлежность классу K^d функций $\mathrm{sg}(x)$, $\overline{\mathrm{sg}}(x)$, $x \cdot \mathrm{sg}(y)$ и $x \cdot \overline{\mathrm{sg}}(y)$ устанавливается так же, как и для класса BPC (они получаются суперпозициями исходных функций).

Изложим основную идею доказательства. Каждой функции $f(\bar{x})$ из класса K сопоставим словарную функцию $\widehat{f}(\bar{x})$, где диадическое представление значения $\widehat{f}(\bar{x})$ состоит из $f(\bar{x})$ единиц (при $f(\bar{x}) = 0$ полагаем $\widehat{f}(\bar{x}) = \Lambda$). Исходные функции $x+1$, $x \dot{-} y$, 2^x класса K переведем в соответствующие функции вида $\widehat{f}$, а далее будем «моделировать» действия операций суперпозиции и ограниченного суммирования на функции класса K действиями операций суперпозиции и ограниченной конкатенации на «образы» этих функций.

Приступим к реализации этой идеи. Для функций $x+1$, x, $x \dot{-} y$ соответствующими «образами» будут функции

$$\operatorname*{Con}_{z \leqslant x} 1, \qquad (\operatorname*{Con}_{z \leqslant x} 1) \ominus 1, \qquad \widehat{x} \ominus \widehat{y}.$$

Заметим, что диадическое представление числа 2^x есть 1 при $x = 0$ и $1 \ldots 12$ ($x-1$ единиц) при $x > 0$. При $x > 0$ имеем $2^x = g(x)$, где

$$g(x) = ((\operatorname*{Con}_{z \leqslant x} 1) \ominus (1 * 1)) * 2.$$

В общем случае получаем

$$2^x = (g(x) \cdot \mathrm{sg}(x)) * \overline{\mathrm{sg}}(x).$$

Следовательно, результат применения операции «крышка» к функции 2^x имеет вид

$$\left(\operatorname*{Con}_{z \leqslant 2^x} 1\right) \ominus 1.$$

Пусть функция $f(\bar{x}, y)$ получается из функции $g(\bar{x}, z)$ с помощью операции ограниченного суммирования:

$$f(\bar{x}, y) = \sum_{z \leqslant y} g(\bar{x}, z)$$

и функция $\widehat{g}(\bar{x}, z)$ уже определена в классе K^d. Тогда функция $\widehat{f}(\bar{x}, y)$ будет равна

$$\operatorname*{Con}_{z \leqslant y} \widehat{g}(\bar{x}, z).$$

Остается рассмотреть операцию суперпозиции. Случаи перестановки и отождествления переменных, добавления и удаления фиктивных переменных очевидны. Пусть

$$f(\bar{x}) = g_0(g_1(\bar{x}), \ldots, g_m(\bar{x})).$$

Очевидно, что

$$\widehat{f}(\bar{x}) = \widehat{g}_0(|\widehat{g}_1(\bar{x})|, \ldots, |\widehat{g}_m(\bar{x})|).$$

Установим принадлежность функции $|x|$ классу K^d. Имеем

$$|x| = \operatorname*{Con}_{z \leqslant x} g(x, z),$$

где

$$g(x,z) = \begin{cases} z, & \text{если } \widehat{z} \leqslant x \text{ и } \widehat{z} * 1 > x, \\ \Lambda & \text{в противном случае.} \end{cases}$$

Однако

$$(z \leqslant w) \equiv (\widehat{z} \ominus \widehat{w} = \Lambda), \quad (z > w) \equiv (\widehat{z} \ominus \widehat{w} \neq \Lambda).$$

Отсюда получаем

$$g(x,z) = z \cdot \overline{\operatorname{sg}}(\widehat{z} \ominus \widehat{x}) \cdot \operatorname{sg}(\widehat{\widehat{z} * 1} \ominus \widehat{x}).$$

Таким образом, для любой функции $f(\bar{x}) \in K$ имеем $\widehat{f}(\bar{x}) \in K^d$. Тогда

$$f(\bar{x}) = |\widehat{f}(\bar{x})| \in K^d.$$

Классы K^d и BPC отличаются лишь тем, что в определении первого ограниченная конкатенация перебирает слова $z \leqslant y$, а в определении второго — слова $z \sqsubseteq y$. На самом деле эта аналогия верна и в арифметическом определении классов. В [15] введен класс

$$\mathrm{BSSM} = [\, x+1, \quad x \dot{-} y \,]_{\mathrm{S},\, \mathrm{SUM}_{\sqsupset},\, \mathrm{PROD}_{\sqsupset}},$$

где операции $\mathrm{SUM}_{\sqsupset}$ и $\mathrm{PROD}_{\sqsupset}$ имеют следующий вид (z пробегает все суффиксы слова y):

$$\sum_{y \sqsupset z} g(\bar{x},\, z), \qquad \prod_{y \sqsupset z} g(\bar{x},\, z).$$

Из результатов § 1, работы [15] и того, что $\mathrm{BPC} = \mathrm{TC}^0$, следует, что $\mathrm{BSSM} = \mathrm{BPC}$. Определение BSSM отличается от определения K лишь множеством значений, которое пробегают «счетчики» операций ограниченного суммирования и мультиплицирования. Определенная аналогия между классами K и TC^0 есть и в области базисов по суперпозиции.

§ 4. Конечные базисы по суперпозиции

При изучении индуктивных способов задания классов функций естественной целью является построение наиболее простых описаний. В частности, интерес представляет получение индуктивных описаний классов, использующих только операцию суперпозиции. В случае когда исходное множество функций в подобном описании конечно, говорят о конечном базисе по суперпозиции.

Самый «лучший» базис по суперпозиции, состоящий только из простых арифметических функций, построен в классе K функций, элементарных по Кальмару [10, 14]:

$$K = [\, x+y, \quad x \dot{-} y, \quad \lfloor x/y \rfloor, \quad 2^x \,].$$

Подобный по простоте используемых функций базис, помимо класса K, известен только в одном из широко изучаемых классов:

$$\mathrm{TC}^0 = [\, x + y, \quad x \dot{-} y, \quad x \wedge y, \quad \lfloor x/y \rfloor, \quad 2^{\lfloor \log_2 x \rfloor^2} \,].$$

Основная часть параграфа посвящена этому результату. Кроме того, будут рассмотрены некоторые другие базисы в классах AC^0 и TC^0.

4.1. Базис по суперпозиции в классе BPC. Прежде чем переходить к основному результату параграфа, обсудим более простые методы построения базисов. В этом разделе мы продемонстрируем, как можно построить базис в классе TC^0 (и его «расширениях») на основе словарного определения этого класса — BPC.

Пусть $f_1, \ldots, f_l$ — произвольные (не обязательно вычислимые) функции на множестве $\{1, 2\}^*$. Будем обозначать

$$\mathrm{BPC}[f_1, \ldots, f_l] = [\, \{1, \quad 2, \quad x * y, \quad x \ominus y\} \cup \{f_1, \ldots, f_l\} \,]_{\mathrm{S},\, \mathrm{CON}_\sqsubseteq}.$$

Т е о р е м а 9 [13, 16]. *Для произвольных функций $f_1, \ldots, f_l$ множество* $\mathrm{BPC}[f_1, \ldots, f_l]$ *имеет конечный базис по суперпозиции.*

Д о к а з а т е л ь с т в о. Мы сформулируем лишь основную идею доказательства.

Обозначим через $\mathrm{doub}(x)$ функцию, которая удваивает каждую букву слова x: если $x = a_1 \ldots a_m$, то $\mathrm{doub}(x) = a_1 a_1 \ldots a_m a_m$ (полагаем также $\mathrm{doub}(\Lambda) = \Lambda$). Для любой функции $f(x_1, \ldots, x_n)$ обозначим через $\widehat{f}(x)$ произвольную (словарную) функцию, которая удовлетворяет следующим двум условиям.

1. Слово $\widehat{f}(x)$ имеет вид $1212 * w_1 * 1212 * w_2 * 1212 * \ldots * 1212 * w_k * 1212$, где, в свою очередь, каждое слово w_i имеет вид

$$\mathrm{doub}(z_1 * 1) * 12 * \ldots * 12 * \mathrm{doub}(z_n * 1) * 12 * \mathrm{doub}(f(z_1, \ldots, z_n) * 1).$$

2. Среди наборов $(z_1, \ldots, z_n)$, определяющих слова w_i, присутствуют все наборы, у которых слова $z_1, \ldots, z_n$ являются подсловами слова x (порядок наборов может быть произвольным, при этом допускается повторение наборов).

Отметим, что в выражениях $\mathrm{doub}(z_i * 1)$, $\mathrm{doub}(f(z_1, \ldots, z_n) * 1)$ добавляемая справа 1 служит лишь для того, чтобы в случае пустого слова сохранить основную структуру слова w_i и иметь возможность затем из слова w_i «извлечь» аргументы $z_1, \ldots, z_n$ и значение функции $f(z_1, \ldots, z_n)$.

Функцию $\widehat{f}(x)$ можно назвать «производящей» функцией для $f(x_1, \ldots, x_n)$; операции суперпозиции и ограниченной префиксной конкатенации над функциями f в классе $\mathrm{BPC}[f_1, \ldots, f_l]$ мы изобразим в виде функций класса BPC, действующих на соответствующие «производящие» функции.

Будем пользоваться функциями, принадлежность которых классу BPC доказана в § 3. Определяем введенную выше функцию $\mathrm{doub}(x)$:

$$\mathrm{doub}(x) = \operatorname*{Con}_{z \sqsubseteq x}(\mathrm{last}(z) * \mathrm{last}(z)).$$

Пусть $\mathrm{EVEN}(x)$, $\mathrm{ODD}(x)$ — предикаты, определяющие четность (нечетность) длины слова x. Имеем

$$\mathrm{EVEN}(x) \equiv (\exists z)_{z \sqsubset x}(|z * z| = |x|), \qquad \mathrm{ODD}(x) \equiv \neg\mathrm{EVEN}(x).$$

Обозначим через $\mathrm{comress}(x)$ функцию, которая удовлетворяет тождеству

$$\mathrm{comress}(\mathrm{doub}(x)) = x.$$

Для функции comress получаем формулу

$$\mathrm{compress}(x) = \operatorname*{Con}_{z \sqsubset x}(\mathrm{last}(z) \cdot \mathrm{sg}(\chi_{\mathrm{ODD}}(z))).$$

Напомним, что $\mathrm{inv}(x)$ — это функция, которая дает инверсию слова x (т. е. слово, составленное из символов слова x, записанных в обратном порядке), и эта функция входит в класс BPC. По аналогии с операцией ограниченной префиксной конкатенации и с использованием функции inv можно ввести операцию ограниченной суффиксной конкатенации:

$$\operatorname*{Con}_{y \sqsupset z} g(\bar{x},\ z) = \operatorname*{Con}_{z \sqsubset \mathrm{inv}(y)} g(\bar{x},\ \mathrm{inv}(z)).$$

Комбинация операций $\mathrm{Con}_{z \sqsubset y}$ и $\mathrm{Con}_{y \sqsupset z}$ позволяет определить операцию $\mathrm{Con}_{z \subseteq y}$, в которой конкатенация проводится по всем подсловам z слова y (пустое слово учитывается один раз, каждое другое подслово учитывается по числу вхождений в y):

$$\operatorname*{Con}_{z \subseteq y} g(\bar{x},\ z) = g(\bar{x},\ \Lambda) * \operatorname*{Con}_{w \sqsubset y} \operatorname*{Con}_{w \sqsupset z}(g(\bar{x},\ z) \cdot \mathrm{sg}(z)).$$

Имея все три операции $\mathrm{Con}_{z \sqsubset y}$, $\mathrm{Con}_{y \sqsupset z}$, $\mathrm{Con}_{z \subseteq y}$ и образованные выше функции класса BPC, можно изобразить применение операций суперпозиции и ограниченной префиксной конкатенации к функциям класса BPC в виде действия подходящих функций класса BPC на соответствующие «производящие» функции (также из класса BPC). На заключительном этапе в классе BPC потребуется функция $\mathrm{Yield}(y, v)$, которая позволяет из «производящей» функции $\widehat{f}(x)$ «извлечь» значение функции f. Именно, если

$$v = \mathrm{doub}(z_1 * 1) * 12 * \ldots * 12 * \mathrm{doub}(z_n * 1)$$

(параметр n здесь не фиксирован) и значение $f(z_1, \ldots, z_n)$ «содержится» в $\widehat{f}(x)$, то должно быть

$$\mathrm{Yield}(\widehat{f}(x),\ v) = f(z_1, \ldots, z_n).$$

Чтобы определить в классе BPC функции, изображающие применение суперпозиции к производящим функциям, воспользуемся известным результатом А. И. Мальцева [7, 8], сводящим выполнение произвольной суперпозиции функций к последовательному выполнению четырех алгебраических операций (частные случаи операции суперпозиции):

1. Операция циклической перестановки переменных, которая при $n \geqslant 2$ определяется равенством

$$f(x_1, \ldots, x_n) = g(x_2, \ldots, x_n, x_1).$$

2. Операция транспозиции первых двух переменных, которая при $n \geqslant 2$ определяется равенством

$$f(x_1, \ldots, x_n) = g(x_2, x_1, x_3, \ldots, x_n).$$

3. Операция отождествления первых двух переменных, которая при $n \geqslant 2$ определяется равенством

$$f(x_1, \ldots, x_{n-1}) = g(x_1, x_1, x_2 \ldots, x_{n-1}).$$

4. Операция подстановки одной функции вместо первой переменной другой функции, которая при любых $m, n \geqslant 1$ определяется равенством

$$f(x_1, \ldots, x_{m+n-1}) = h(g(x_1, \ldots, x_m), x_{m+1}, \ldots, x_{m+n-1})$$

(здесь g, h — данные функции, а функция f получается применением рассматриваемой операции).

Заметим, что для операций 1, 2 в качестве функции $\widehat{f}(x)$ можно взять функцию $\widehat{g}(x)$. Для операций 3, 4 необходимо строить «преобразующие» функции $\mathrm{Sup}_1(y)$ и $\mathrm{Sup}_2(y, z)$.

При построении функции $\mathrm{Concat}(y)$, отвечающей операции ограниченной префиксной конкатенации, необходимо предварительно преобразовать аргумент y — дело в том, что в подаваемом на вход функции $\mathrm{Concat}(y)$ аргументе $\widehat{g}(x)$ «содержащиеся» значения $g(z_1, \ldots, z_n)$ могут повторяться. При получении нужного результата эта особенность аргумента $\widehat{g}(x)$ доставляет определенные технические трудности. Поэтому предварительно следует провести «проектирование» слова $\widehat{g}(x)$, освободившись в нем от повторяющихся значений функции g. В принципе такая операция в рамках класса BPC не является очень сложной: необходимо выделить в слове $\widehat{g}(x)$ первые вхождения подслов вида

$$\mathrm{doub}(z_1 * 1) * 12 * \ldots * 12 * \mathrm{doub}(z_n * 1) * 12 * \mathrm{doub}(g(z_1, \ldots, z_n) * 1)$$

и затем удалить из слова $\widehat{g}(x)$ все остальные вхождения этих слов. Соответствующую функцию класса BPC обозначим через $\mathrm{Proj}(y)$.

Предположим, что к функции $\widehat{g}(x)$ уже применена функция Proj и, следовательно, каждое значение функции $g(\bar{x}, y)$ присутствует в слове $\widehat{g}(x)$ только один раз. В этом случае нетрудно понять, как можно определить в классе BPC такую функцию $\mathrm{Concat}(y)$, что если для некоторых $x_1, \ldots, x_n, y$ слово $\widehat{g}(x)$ «содержит» все значения $g(x_1, \ldots, x_n, z)$, где $z \sqsubseteq y$, то в слово $\mathrm{Concat}(\widehat{g}(x))$ входит подслово

$$1212 * \mathrm{doub}(x_1 * 1) * 12 * \ldots * 12 * \mathrm{doub}(x_n * 1) * 12 * \mathrm{doub}(y * 1) * 12 * \\ * \mathrm{doub}((\underset{z \sqsubseteq y}{\mathrm{Con}}\, g(x_1, \ldots, x_n, z)) * 1) * 1212.$$

Таким образом, конечный базис по суперпозиции в классе $\mathrm{BPC}[f_1, \ldots, f_l]$ можно составить из функций вида $\widehat{f}$, где f принадлежит объединению множеств $\{1,\ 2,\ x*y,\ x \ominus y\}$ и $\{f_1, \ldots, f_l\}$, функций

$$\mathrm{Yield}(x_1, x_2), \quad \mathrm{Sup}_1(x), \quad \mathrm{Sup}_2(x_1, x_2), \quad \mathrm{Proj}(x), \quad \mathrm{Concat}(x),$$

а также функций $1,\ 2,\ x*y,\ \mathrm{doub}(x)$.

В [15] похожим методом построен базис по суперпозиции в классе BSSM, который задан с помощью арифметических операций, но совпадает с классом BPC (см. конец § 3).

4.2. Простой базис по суперпозиции в классе FOM. Этот раздел посвящен результату, полученному С. А. Волковым: в классе FOM существует базис по суперпозиции, состоящий из простых арифметических функций. Ряд предварительных построений был проделан в [2]. Основная теорема сформулирована в [3] и доказана в рукописи [4] и кандидатской диссертации [5]. Мы приводим полное доказательство этого результата.

Положим

$$T = \{x + y, \quad x \dot{-} y, \quad x \wedge y, \quad \lfloor x/y \rfloor, \quad 2^{\lfloor \log_2 x \rfloor^2}\}.$$

Теорема 10 [3–5].

$$\mathrm{FOM} = [T].$$

Из приведенных в § 1 результатов следует, что класс FOM замкнут относительно операции суперпозиции, а функции $x + y$, $x \dot{-} y$, $x \wedge y$, $\lfloor x/y \rfloor$, $2^{\lfloor \log_2 x \rfloor^2}$ принадлежат классу FOM. Поэтому $[T] \subseteq \mathrm{FOM}$. Дальнейшее доказательство будет посвящено установлению включения $\mathrm{FOM} \subseteq [T]$.

Элементарные функции и операции. Прежде всего докажем принадлежность классу $[T]$ ряда элементарных арифметических функций.

Утверждение 10. *Константы и функции* $\mathrm{sg}(x)$, $\max(x, y)$, xy, $\mathrm{rm}(x, y)$ *входят в* $[T]$.

Доказательство. Имеем

$$0 = x \dot{-} x, \quad 1 = 2^{\lfloor \log_2 0 \rfloor^2}, \quad 2 = 1 + 1, \quad 3 = 2 + 1, \quad \ldots,$$
$$\mathrm{sg}(x) = 1 \dot{-} (1 \dot{-} x), \qquad \max(x, y) = (x + y) \dot{-} (x \dot{-} (x \dot{-} y)).$$

Используя известное тождество

$$\lfloor \lfloor A/x \rfloor / y \rfloor = \lfloor A/xy \rfloor \quad (A \in \mathbb{N}_0, \ x, y \in \mathbb{N}),$$

выразим умножение с помощью деления:

$$xy = \left\lfloor \frac{A}{\lfloor \lfloor A/x \rfloor / y \rfloor} \right\rfloor,$$

где A — достаточно быстро растущая функция. При $x, y > 0$ для выполнения указанного равенства достаточно выбрать $A > x^2y^2 + xy$. Например, подойдет функция $A = g(g(x + y))$, где $g(x) = 2^{\lfloor \log_2 (x+x) \rfloor}$. В случае $x = 0$ или $y = 0$ указанная формула дает $xy = 0$.

Наконец, $\mathrm{rm}(x, y) = (x \dot{-} y \lfloor x/y \rfloor) \cdot \mathrm{sg}(y)$.

Через $\mathrm{ssqrt}(y)$ будем обозначать функцию, удовлетворяющую условию $\mathrm{ssqrt}(2^{2x}) = 2^x$. Здесь и далее считаем, что при задании функции подобным образом ее значения для остальных входных данных не важны для дальнейших построений.

Утверждение 11. *Функция* $\mathrm{ssqrt}(y)$ *принадлежит классу* $[T]$.

Доказательство. Приводимое ниже доказательство основано на построениях С. Маззанти [49] и является более простым, нежели оригинальное доказательство из [4].

Пусть $y = 2^{2x}$. Необходимо получить 2^x. Имеем $2^{4x} = y^2$ и $2^{4x^2} = 2^{\lfloor \log_2 y \rfloor^2}$. Нетрудно видеть, что при $x > 0$

$$2^{4x} \equiv 2 \pmod{2^{4x} - 2} \qquad \text{и} \qquad 2^{4x^2} \equiv (2^{4x})^x \equiv 2^x \pmod{2^{4x} - 2}.$$

Следовательно, при $x > 0$ получаем

$$2^x = \mathrm{rm}(2^{4x^2},\ 2^{4x} \dot{-} 2) = \mathrm{rm}(2^{\lfloor \log_2 y \rfloor^2},\ y^2 \dot{-} 2).$$

При $x = 0$ (т. е. при $y = 1$) указанное выражение дает 0. Чтобы получить желаемое значение 1, нужно прибавить к полученному выражению $2 \dot{-} y$.

В рамках следующего утверждения будем называть полиномами функции, которые получаются суперпозициями констант и функций $x + y$, $x \dot{-} y$, xy. Отметим, что функции $\mathrm{sg}(x)$ и $\max(x, y)$ в этом смысле тоже считаются полиномами.

У т в е р ж д е н и е 12. *Пусть $p(x_1, \ldots, x_n)$ — полином. Тогда*

$$g_p(x_1, \ldots, x_n) = 2^{p(\lfloor \log_2 x_1 \rfloor, \ldots, \lfloor \log_2 x_n \rfloor)} \in [T].$$

Д о к а з а т е л ь с т в о. Сначала выразим функции $2^{\lfloor \log_2 x \rfloor}$ и $2^{\lfloor \log_2 x \rfloor \cdot \lfloor \log_2 y \rfloor}$. Имеем

$$2^{\lfloor \log_2 (2x) \rfloor^2} = 2^{(\lfloor \log_2 x \rfloor + 1)^2} = 2^{\lfloor \log_2 x \rfloor^2 + 2\lfloor \log_2 x \rfloor + 1}.$$

Отсюда

$$2^{2\lfloor \log_2 x \rfloor} = \left\lfloor \frac{2^{\lfloor \log_2 (2x) \rfloor^2}}{2 \cdot 2^{\lfloor \log_2 x \rfloor^2}} \right\rfloor \qquad \text{и} \qquad 2^{\lfloor \log_2 x \rfloor} = \mathrm{ssqrt}(2^{2\lfloor \log_2 x \rfloor}).$$

Аналогично, используя функцию $2^{\lfloor \log_2 x \rfloor}$, формируем в $[T]$ выражение

$$2^{(\lfloor \log_2 (2^{\lfloor \log_2 x \rfloor} \cdot 2^{\lfloor \log_2 y \rfloor}) \rfloor)^2} = 2^{(\lfloor \log_2 x \rfloor + \lfloor \log_2 y \rfloor)^2} = 2^{\lfloor \log_2 x \rfloor^2 + 2\lfloor \log_2 x \rfloor \cdot \lfloor \log_2 y \rfloor + \lfloor \log_2 y \rfloor^2}.$$

Деля это выражение на $2^{\lfloor \log_2 x \rfloor^2} \cdot 2^{\lfloor \log_2 y \rfloor^2}$, получаем функцию $2^{2\lfloor \log_2 x \rfloor \cdot \lfloor \log_2 y \rfloor}$. Далее приходим к равенству

$$2^{\lfloor \log_2 x \rfloor \cdot \lfloor \log_2 y \rfloor} = \mathrm{ssqrt}(2^{2\lfloor \log_2 x \rfloor \cdot \lfloor \log_2 y \rfloor}).$$

Строить произвольную функцию g_p будем с помощью индукции по построению полинома p. Если p равен константе, то g_p — тоже константа. Если $p(x) = x$, то $g_p(x) = 2^{\lfloor \log_2 x \rfloor}$. Если $p = p_1 + p_2$, то $g_p = g_{p_1} \cdot g_{p_2}$. Если $p = p_1 \dot{-} p_2$, то $g_p = \lfloor g_{p_1} / g_{p_2} \rfloor + \mathrm{sg}(g_{p_2} \dot{-} g_{p_1})$. Если $p = p_1 \cdot p_2$, то $g_p = 2^{\lfloor \log_2 g_{p_1} \rfloor \cdot \lfloor \log_2 g_{p_2} \rfloor}$.

Пусть при $l > 0$

$$s_1(n, l) = \sum_{x=0}^{n-1} 2^{xl}.$$

Для выражения сумм в $[T]$ будем пользоваться следующим утверждением.

У т в е р ж д е н и е 13. *Имеет место равенство*

$$s_1(n, l) = \frac{2^{nl} - 1}{2^l - 1},\ l > 0.$$

Функция $s_1(\lfloor \log_2 y \rfloor,\ \lfloor \log_2 z \rfloor)$ принадлежит классу $[T]$.

Доказательство. Равенство следует из формулы суммы членов геометрической прогрессии. Нетрудно видеть, что каждое вычитание в формуле можно заменить на $\dot{-}$, а деление — на целую часть от деления. Поскольку в показателях степеней имеются только простейшие полиномы, в формуле для $s_1(\lfloor\log_2 y\rfloor,\ \lfloor\log_2 z\rfloor)$ в показателях будут встречаться только полиномы от логарифмов, а такие степени содержатся в $[T]$.

При $x_0, \ldots, x_{n-1} < 2^l$, $l \geqslant 1$ будем использовать обозначение $\langle x_0, \ldots, x_{n-1};\ l\rangle$ для суммы

$$\sum_{i=0}^{n-1} x_i \cdot 2^{li}.$$

Указанная сумма однозначно кодирует набор $(x_0, \ldots, x_{n-1})$, храня двоичные записи чисел x_i в l-битовых блоках двоичной записи числа $\langle x_0, \ldots, x_{n-1};\ l\rangle$ (блоки размещаются справа налево по возрастанию индексов). Число $\langle x_0, \ldots, x_{n-1};\ l\rangle$ будем называть кодом длины l набора $(x_0, \ldots, x_{n-1})$.

При $x < 2^l$ положим

$$\mathrm{rep}(x, n, l) = \langle \underbrace{x, \ldots, x}_{n};\ l\rangle.$$

При $n = 0$ или $l = 0$ доопределяем эту функцию числом 0.

Утверждение 14. *Функция* $\mathrm{rep}(x,\ \lfloor\log_2 n\rfloor,\ \lfloor\log_2 l\rfloor)$ *принадлежит классу* $[T]$.

Доказательство. Легко видеть, что

$$\mathrm{rep}(x, n, l) = \sum_{i=0}^{n-1} x 2^{li} = x \left\lfloor \frac{2^{ln} \dot{-} 1}{2^l \dot{-} 1} \right\rfloor = x \cdot s_1(n, l).$$

Утверждение 15. *Функция* $\lfloor\log_2 x\rfloor$ *принадлежит классу* $[T]$.

Доказательство. Воспользуемся техникой подсчета количества единиц в двоичной записи числа при помощи умножения (аналогичная техника используется в теореме 6 из § 3, а изначально она была предложена в [40]).

Положим $l = \lfloor\log_2 x\rfloor$ и $A = (\mathrm{rep}(1, l, l))^2$. Преобразуем A следующим образом:

$$(\mathrm{rep}(1, l, l))^2 = \left(\sum_{i=0}^{l-1} 2^{il}\right)^2 = \sum_{0 \leqslant i,j < l} 2^{(i+j)l} = \sum_{p=0}^{2l-2} \left(\sum_{\substack{0 \leqslant i,j < l \\ i+j=p}} 1 \right) 2^{pl} = \sum_{p=0}^{2l-2} a_p 2^{pl}.$$

Нетрудно видеть, что коэффициенты a_p не превосходят $l < 2^l$. Тогда $A = \langle a_0, \ldots, a_{2l-2};\ l\rangle$. При этом $a_{l-1} = l$. Для выделения этого числа воспользуемся делением на степени двойки:

$$l = \mathrm{rm}(\lfloor A/2^{l(l-1)}\rfloor,\ 2^l).$$

В показателях степеней встречаются только полиномы от $l = \lfloor\log_2 x\rfloor$, поэтому эти степени содержатся в $[T]$. При $l = 0$ приведенная формула дает 0.

Нам потребуется выразить в $[T]$ еще один вид суммирования. Пусть при $l > 0$

$$s_2(n, l) = \sum_{x=0}^{n-1} x 2^{xl}.$$

Утверждение 16. *Имеет место равенство*

$$s_2(n,l)=\frac{n2^{nl}(2^l-1)-2^l(2^{nl}-1)}{(2^l-1)^2},\ l>0.$$

Функция $s_2(\lfloor\log_2 y\rfloor,\ \lfloor\log_2 z\rfloor)$ *принадлежит классу* $[T]$.

Доказательство. Равенство получается стандартными методами суммирования (например, сведением к вычислению $s_1(n,l)$ через расширение области значений переменной l на пространство $\mathbb{R}$ и интегрирование суммы по этой переменной). Нетрудно видеть, что каждое вычитание можно заменить на $\dot{-}$, а каждое деление — на $\lfloor x/y\rfloor$. В показателях степеней встречаются только полиномы от n,l, поэтому при подстановке логарифмов эти степени будут принадлежать классу $[T]$. Переменная n также встречается не в показателе степени и требует использования функции $\lfloor\log_2 x\rfloor$.

Производящие функции. Для выражения произвольной функции из FOM суперпозициями функций из T мы воспользуемся методом производящих функций. Используем сокращения $\bar{x}=(x_1,\ldots,x_n)$ и $\bar{X}=(X_1,\ldots,X_n)$.

Пусть $h(\bar{X};\ y_1,\ldots,y_m)$ — функция, зависящая от словарных переменных $\bar{X}$ одинаковой длины со значениями из $\{w\in\{0,1\}^*\colon |w|\geqslant 2\}$ и k-переменных $y_1,\ldots,y_m$ со значениями из $\{0,\ldots,|\bar{X}|^k-1\}$, причем $h(\bar{X};\ y_1,\ldots,y_m)<2^{|\bar{X}|^k}$.

Производящей функцией для h назовем функцию

$$f_h(\bar{x},z)=\sum_{0\leqslant y_1,\ldots,y_m<l} h(\bar{X};\ y_1,\ldots,y_m)\cdot 2^{y_1 l+y_2 l^2+\ldots+y_m l^m},$$

где $z\geqslant 2^{\max(\mathrm{len}(x_1),\ldots,\mathrm{len}(x_n),2)}$, $l=\lfloor\log_2 z\rfloor^k$, а каждое слово X_i представляет собой двоичную запись числа x_i, дополненную при необходимости незначащими нулями для достижения длины $\lfloor\log_2 z\rfloor$.

Производящая функция получает на вход информацию о наборе $\bar{X}$: переменная z определяет длину каждого слова X_i (она равна $|\bar{X}|=\lfloor\log_2 z\rfloor$), а переменные $\bar{x}$ определяют содержимое этих слов. Значение производящей функции кодирует m-мерную матрицу значений функции h при данном $\bar{X}$ на всех возможных наборах $y_1,\ldots,y_m$ (каждая переменная y_i пробегает значения $0,\ldots,l-1$, где $l=|\bar{X}|^k$). Каждое значение h кодируется l битами.

Отметим, что значение производящей функции можно представить в виде

$$f_h(\bar{x},z)=\langle h_0,\ldots,h_{l^m-1};\ l\rangle,$$

где значения $h_0,\ldots,h_{l^m-1}$ функции h на всех наборах $(y_1,\ldots,y_m)$ соответствующим образом упорядочены.

Производящей функцией f_ρ предиката $\rho(\bar{X};\ y_1,\ldots,y_m)$ с k-переменными $y_1,\ldots,y_m$ назовем производящую функцию его характеристической функции.

Далее мы докажем, что суперпозициями функций из T можно выразить ряд технических функций, осуществляющих преобразования кодов наборов и производящих функций. Большая часть этих выражений получена в [2].

Ограниченное увеличение длины кода двоичных значений. Определим $\mathrm{incrx}(x,n,l)$ как функцию, для которой при $x=\langle x_0,\ldots,x_{n-1};\ 1\rangle$ и $l\geqslant n+1$ выполняется равенство

$$\mathrm{incrx}(x,n,l)=\langle x_0,\ldots,x_{n-1};\ l\rangle.$$

Утверждение 17. *Функция* $\mathrm{incrx}(x,\ \lfloor\log_2 n\rfloor,\ \lfloor\log_2 l\rfloor)$ *принадлежит классу* $[T]$.

Доказательство. Для вычисления $\mathrm{incrx}(x,n,l)$ достаточно n раз продублировать набор x и очистить лишние биты (при таком способе выражения новая длина не может быть меньше $n+1$). Пусть $A=\mathrm{rep}(x,\ n,\ l\dot{-}1)$ и $B=\mathrm{rep}(1,\ n,\ l)$. Тогда

$$A\wedge B=(x_{n-1}\underbrace{0\ldots0}_{l-1}x_{n-2}\ldots x_2\underbrace{0\ldots0}_{l-1}x_1\underbrace{0\ldots0}_{l-1}x_0)_2=\langle x_{n-1},\ldots,x_0;\ l\rangle.$$

В функцию $\mathrm{rep}(x,n,l)$ подставляются только полиномы от n,l, поэтому при подстановке логарифмов получится функция из $[T]$.

Изменение способа кодирования n-мерной матрицы двоичных значений. Пусть

$$x_f(q,\ k_1,\ldots,k_n)=\sum_{0\leqslant i_1,\ldots,i_n<q}f(i_1,\ldots,i_n)\cdot 2^{k_1i_1+\ldots+k_ni_n},$$

где f — функция, принимающая только значения 0 и 1, $q\geqslant 2$, а числа $k_1,\ldots,k_n$ таковы, что отображение

$$k_1i_1+\ldots+k_ni_n\colon\ \{0,\ \ldots,\ q-1\}^n\to\mathbb{N}_0$$

является инъективным. При каждом $n\geqslant 1$ через swap_n будем обозначать функцию, для которой при любой функции $f(x_1,\ldots,x_n)$, принимающей значения 0 и 1, любых значениях $q,k_1,\ldots,k_n$, удовлетворяющих описанным выше условиям, и при $x=x_f(q,\ k_1,\ldots,k_n)$ выполняется равенство

$$\mathrm{swap}_n(x,\ q,\ k_1,\ldots,k_n,\ m_1,\ldots,m_n)=\sum_{0\leqslant i_1,\ldots,i_n<q}f(i_1,\ldots,i_n)\cdot 2^{m_1i_1+\ldots+m_ni_n}.$$

Числа $f(i_1,\ldots,i_n)$ в правой части равенства однозначно определяются по $x,\ q,\ k_1,\ldots,k_n$.

Утверждение 18. *Функция*

$$\mathrm{swap}_n(x,\ \lfloor\log_2 q\rfloor,\ \lfloor\log_2 k_1\rfloor,\ldots,\lfloor\log_2 k_n\rfloor,\ \lfloor\log_2 m_1\rfloor,\ldots,\lfloor\log_2 m_n\rfloor)$$

принадлежит классу $[T]$.

Доказательство. Будем использовать сокращения $i=(i_1,\ldots,i_n)$ и $j=(j_1,\ldots,j_n)$. Введем целочисленную функцию

$$\varphi(i,j)=k_1(i_1-j_1)+\ldots+k_n(i_n-j_n)$$

(i,j принимают значения из $\{0,\ldots,q-1\}^n$, а функция $\varphi(i,j)$ может принимать отрицательные значения). Отметим, что в силу условий на $k_1,\ldots,k_n$ равенство наборов $i=j$ выполняется тогда и только тогда, когда $\varphi(i,j)=0$.

Центральным элементом рассмотрения будет сумма

$$S=\sum_{\substack{0\leqslant i_1,\ldots,i_n<q\\0\leqslant j_1,\ldots,j_n<q}}f(i_1,\ldots,i_n)2^{W+m_1j_1+\ldots+m_nj_n+p\varphi(i,j)},$$

где W и p — достаточно быстро растущие полиномы, которые будут определены позже.

Для выражения этой суммы заметим, что значение x можно рассматривать как код $\langle x_0, \ldots, x_{t-1};\ 1\rangle$ соответствующего набора, где $t = q(k_1 + \ldots + k_n)$. Тогда при $p > t$ сумма

$$S_0 = \sum_{0 \leqslant i_1, \ldots, i_n < q} f(i_1, \ldots, i_n) \cdot 2^{p(k_1 i_1 + \ldots + k_n i_n)}$$

будет равна $\mathrm{incrx}(x, t, p)$. Чтобы получить S, нужно умножить S_0 на сумму

$$S_1 = \sum_{0 \leqslant j_1, \ldots, j_n < q} 2^{W + (m_1 - pk_1) j_1 + \ldots + (m_n - pk_n) j_n},$$

равную при $W = na$ величине

$$\prod_{r=1}^{n} \sum_{j_r=0}^{q-1} 2^{a + (m_r - pk_r) j_r}.$$

Выбирая $a = pq(k_1 + \ldots + k_r)$, добиваемся того, что все показатели степеней в этом выражении неотрицательны. Предполагая, что $p > m_1, \ldots, m_r$, вычисляя суммы геометрических прогрессий и избавляясь от дробных значений умножением числителя и знаменателя на соответствующее число, получим

$$S_1 = \prod_{r=1}^{n} \frac{2^{a+pk_r} - 2^{a+qm_r-(q-1)pk_r}}{2^{pk_r} - 2^{m_r}} = \prod_{r=1}^{n} \left\lfloor \frac{2^{a+pk_r} \dot{-} 2^{a+qm_r \dot{-} (q \dot{-} 1)pk_r}}{2^{pk_r} \dot{-} 2^{m_r}} \right\rfloor.$$

Имея теперь сумму $S = S_0 \cdot S_1$, мы хотим избавиться от всех слагаемых, у которых $i \neq j$ (т. е. $\varphi(i, j) \neq 0$). Сумма всех слагаемых, у которых $\varphi(i, j) < 0$, не превосходит

$$q^{2n} 2^{W + q(m_1 + \ldots + m_n) - p} \leqslant 2^{W + q^{2n} + q(m_1 + \ldots + m_n) - p}.$$

Выбирая тогда

$$p = q^{2n} + q(m_1 + \ldots + m_n + 1) + t\ >\ t,\ m_1, \ldots, m_r,$$

получим, что эта сумма меньше 2^W. С другой стороны, выполняется $m_1 j_1 + \ldots + m_n j_n < p$. Это значит, что все слагаемые S, у которых $\varphi(i, j) = 0$, кратны 2^W, но меньше, чем 2^{W+p}. В свою очередь, слагаемые с $\varphi(i, j) > 0$ кратны 2^{W+p}. Следовательно,

$$\mathrm{rm}(\lfloor S/2^W \rfloor,\ 2^p) = \sum_{\substack{0 \leqslant i, j < q \\ \varphi(i,j) = 0}} f(i) 2^{m_1 j_1 + \ldots + m_n j_n + p\varphi(i,j)} = \sum_{0 \leqslant j_1, \ldots, j_n < q} f(j) \cdot 2^{m_1 j_1 + \ldots + m_n j_n}.$$

Число n есть константа. В показателях степеней встречаются только полиномы от q, $k_1, \ldots, k_n$, $m_1, \ldots, m_n$, поэтому при подстановке логарифмов образующиеся степени будут входить в $[T]$.

Основной способ использования функций swap_n — выражение перестановки k-переменных предикатов в их производящих функциях. Утверждение ниже следует из определений производящих функций и функции swap_n, а также из доказанного выше утверждения.

Утверждение 19. *Пусть* $\rho'(\bar{X};\ y_1, \ldots, y_m) = \rho(\bar{X};\ y_{j_1}, \ldots, y_{j_m})$, *где* ρ — *предикат с* k-*переменными* $y_1, \ldots, y_m$ *и* $(j_1, \ldots, j_m)$ — *перестановка чисел* $1, \ldots, n$. *Тогда при* $l = \lfloor \log_2 z \rfloor^k$ *выполняется равенство*

$$f_{\rho'}(\bar{x}, z) = \mathrm{swap}_m(f_\rho(\bar{x}, z),\ l,\ l, \ldots, l^m,\ l^{j_1}, \ldots, l^{j_m}).$$

При этом если $f_\rho \in [T]$, *то и* $f_{\rho'} \in [T]$.

Изменение длины кода двоичных значений. Определим $\mathrm{incr}(x, n, l)$ и $\mathrm{decr}(x, n, l)$ как функции, для которых при $x_0, \ldots, x_{n-1} \in \{0, 1\}$ выполняются равенства

$$\mathrm{incr}(x, n, l) = \langle x_0, \ldots, x_{n-1};\ l\rangle, \text{ где } x = \langle x_0, \ldots, x_{n-1};\ 1\rangle,$$
$$\mathrm{decr}(x, n, l) = \langle x_0, \ldots, x_{n-1};\ 1\rangle, \text{ где } x = \langle x_0, \ldots, x_{n-1};\ l\rangle.$$

Отметим, что функция incr, в отличие от incrx, не имеет ограничений на значение l.

Эти две функции можно выразить с помощью «вырожденного» варианта swap_n.

Утверждение 20. *Функция* $\mathrm{incr}(x, \lfloor\log_2 n\rfloor, \lfloor\log_2 l\rfloor)$ *и функция* $\mathrm{decr}(x, \lfloor\log_2 n\rfloor, \lfloor\log_2 l\rfloor)$ *принадлежат классу* $[T]$.

Доказательство. Имеем равенства $\mathrm{incr}(x, n, l) = \mathrm{swap}_1(x, n, 1, l)$ и $\mathrm{decr}(x, n, l) = \mathrm{swap}_1(x, n, l, 1)$.

Логические операции над кодами двоичных наборов. Определим $\mathrm{not}(x, n)$ и $\mathrm{or}(x, y, n)$ как функции, для которых при $x = \langle x_0, \ldots, x_{n-1};\ 1\rangle$ и $y = \langle y_0, \ldots, y_{n-1};\ 1\rangle$ выполняется

$$\mathrm{not}(x, n) = \langle \overline{x}_0, \ldots, \overline{x}_{n-1};\ 1\rangle,$$
$$\mathrm{or}(x, y, n) = \langle x_0 \vee y_0,\ \ldots,\ x_{n-1} \vee y_{n-1};\ 1\rangle.$$

Утверждение 21. *Функции* $\mathrm{not}(x, \lfloor\log_2 n\rfloor)$ *и* $\mathrm{or}(x, \lfloor\log_2 n\rfloor)$ *принадлежат классу* $[T]$.

Доказательство. Имеем

$$\mathrm{not}(x, n) = (2^n \dot{-} 1) \dot{-} x \quad \text{и} \quad \mathrm{or}(x, y, n) = \mathrm{not}(\mathrm{not}(x, n) \wedge \mathrm{not}(y, n),\ n).$$

В показателе степени встречается только n, поэтому при подстановке логарифма полученная степень будет входить в $[T]$.

Одновременное сравнение чисел в наборах. Определим $\mathrm{cmp}(x, y, n, l)$ и $\mathrm{cmpeq}(x, y, n, l)$ как функции, для которых при $x = \langle x_0, \ldots, x_{n-1};\ l\rangle$ и $y = \langle y_0, \ldots, y_{n-1};\ l\rangle$ выполняется

$$\mathrm{cmp}(x, y, n, l) = \langle (x_0 \geqslant y_0),\ \ldots,\ (x_{n-1} \geqslant y_{n-1});\ 1\rangle,$$
$$\mathrm{cmpeq}(x, y, n, l) = \langle (x_0 = y_0),\ \ldots,\ (x_{n-1} = y_{n-1});\ 1\rangle$$

(здесь истинному значению предиката соответствует 1, ложному — 0).

Утверждение 22. *Функция* $\mathrm{cmp}(x, y, \lfloor\log_2 n\rfloor, \lfloor\log_2 l\rfloor)$ *и функция* $\mathrm{cmpeq}(x, y, \lfloor\log_2 n\rfloor, \lfloor\log_2 l\rfloor)$ *входят в* $[T]$.

Доказательство. Будем вычислять $\mathrm{cmp}(x, y, n, l)$ с помощью поблокового вычитания. Для сведения его к простому вычитанию преобразуем числа. Вставим слева от каждого бита каждого числа x_i и y_i по нулю. Получим

$$x' = \mathrm{incr}(x, nl, 2) = \langle x'_0, \ldots, x'_{n-1};\ 2l\rangle, \quad y' = \mathrm{incr}(y, nl, 2) = \langle y'_0, \ldots, y'_{n-1};\ 2l\rangle,$$

причем $x_i \geqslant y_i \iff x'_i \geqslant y'_i$, а старшие разряды всех чисел x'_i, y'_i нулевые. Допишем в старшие разряды x'_i единицы: $\widehat{x} = x' + \mathrm{rep}(2^{2l-1},\ n,\ 2l)$. Тогда число

$$C = \widehat{x} \dot{-} y' = \langle c_0, \ldots, c_{n-1};\ 2l\rangle$$

содержит в старшем разряде каждого числа c_i результат сравнения $x'_i \geqslant y'_i$. Очищаем остальные биты, переносим результаты в младшие разряды чисел, формируем код длины 1 и получаем равенство

$$\mathrm{cmp}(x, y, n, l) = \mathrm{decr}(\lfloor (C \wedge \mathrm{rep}(2^{2l-1},\ n,\ 2l))/2^{2l-1} \rfloor,\ n,\ 2l).$$

Далее получаем

$$\mathrm{cmpeq}(x, y, n, l) = \mathrm{cmp}(x, y, n, l) \wedge \mathrm{cmp}(y, x, n, l).$$

В показателях встречаются только полиномы от n, l, поэтому при подстановке логарифмов образующиеся степени будут входить в $[T]$.

Одновременное вычисление нескольких битовых сумм. Пусть $x_i = \langle x_{i,0}, \ldots, x_{i,k-1};\ l \rangle$ при $i = \overline{0,\ n-1}$, причем $k < 2^l$ и $x_{i,j} \in \{0, 1\}$ при всех значениях i, j. Пусть далее $x = \langle x_0, \ldots, x_{n-1};\ lk \rangle$. Определим $\mathrm{sum}(x, n, l, k)$ как функцию, для которой при удовлетворяющих указанным условиям x, n, l, k выполняется равенство

$$\mathrm{sum}(x, n, l, k) = \langle s_0, \ldots, s_{n-1};\ lk \rangle,$$

где $s_i = x_{i,0} + \ldots + x_{i,k-1}$ — сумма всех чисел (нулей и единиц) в наборе x_i, $i = \overline{0,\ n-1}$. Требование $k < 2^l$ означает, что двоичные записи чисел s_i имеют длину не более l.

Утверждение 23. *Функция* $\mathrm{sum}(x, \lfloor \log_2 n \rfloor, \lfloor \log_2 l \rfloor, \lfloor \log_2 k \rfloor)$ *принадлежит классу* $[T]$.

Доказательство. Воспользуемся техникой подсчета количества единиц в двоичной записи числа при помощи умножения. Представим x в виде $\langle u_0, \ldots, u_{nk-1};\ l \rangle$, где $u_i \in \{0, 1\}$. Рассмотрим произведение $A = x \cdot \mathrm{rep}(1, k, l)$ и преобразуем его следующим образом:

$$x \cdot \mathrm{rep}(1, k, l) = \left(\sum_{i=0}^{nk-1} u_i 2^{il} \right) \left(\sum_{j=0}^{k-1} 2^{jl} \right) = \sum_{\substack{0 \leqslant i < nk \\ 0 \leqslant j < k}} u_i 2^{(i+j)l} = \sum_{p=0}^{k(n+1)-2} \left(\sum_{\substack{0 \leqslant i < nk \\ 0 \leqslant j < k \\ i+j=p}} u_i \right) 2^{pl}.$$

Обозначим через z_p множитель перед 2^{pl} в последней сумме. Видно, что каждое z_p не превосходит $k < 2^l$. Поэтому A можно представить в виде $\langle z_0, \ldots, z_{k(n+1)-2};\ l \rangle$. Заметим, что

$$z_{tk+k-1} = \sum_{i=tk}^{tk+k-1} u_i = \sum_{i=0}^{k-1} x_{t,i} = s_t \quad (t \in \{0, \ldots, n-1\}).$$

Осталось сдвинуть каждое число s_t в младшие разряды t-го kl-битового блока и обнулить все остальные значения z_j:

$$\mathrm{sum}(x, n, l, k) = \lfloor A/2^{(k-1)l} \rfloor \wedge \mathrm{rep}(\mathrm{rep}(1, l, 1),\ n,\ kl).$$

В показателях степеней встречаются только полиномы от n, l, k, поэтому при подстановке логарифмов полученные степени будут входить в $[T]$.

Обращение двоичной записи. Определим $\mathrm{reverse}(x, n)$ как функцию, для которой при x и n, удовлетворяющих равенству $x = \langle x_0, \ldots, x_{n-1};\ 1 \rangle$, выполняется

$$\mathrm{reverse}(x, n) = \langle x_{n-1}, \ldots, x_0;\ 1 \rangle.$$

Утверждение 24. *Функция* $\mathrm{reverse}(x, \lfloor \log_2 n \rfloor)$ *принадлежит классу* $[T]$.

Доказательство. Для реализации $\mathrm{reverse}(x, n)$ повторим n раз запись x и выделим значения $x_{n-1}, \ldots, x_0$ в нужном порядке. Сформируем число

$$Y = \mathrm{rep}(x, n, n) \wedge (\mathrm{rep}(1,\ n,\ n \dot{-} 1) \cdot 2^{n-1}) = \\ = (x_0 \underbrace{0 \ldots 0}_{n-2} x_1 \ldots x_{n-2} \underbrace{0 \ldots 0}_{n-2} x_{n-1} \underbrace{0 \ldots 0}_{n-1})_2.$$

Тогда $\mathrm{reverse}(x, n) = \mathrm{decr}(\lfloor Y / 2^{n-1} \rfloor,\ n,\ n \dot{-} 1)$. В показателях степеней встречаются только полиномы от n, поэтому при подстановке логарифмов полученные степени входят в $[T]$.

Выражение функций из FOM. Следующее утверждение является центральным в доказательстве.

Утверждение 25. *Пусть предикат* $\rho(\bar{X};\ y_1, \ldots, y_r)$ k-FOM*-определим. Тогда его производящая функция* $f_\rho(\bar{x}, z)$ *принадлежит классу* $[T]$.

Доказательство. Пусть предикат ρ задан FOM-формулой Ф. Можно считать, что связанные кванторами переменные Ф различны и представляют собой переменные $y_{r+1}, \ldots, y_m$.

Рассмотрим предикат $\rho'(y_1, \ldots, y_m) \equiv \rho(y_1, \ldots, y_r)$. От переменных $y_{r+1}, \ldots, y_m$ он зависит фиктивно. Будем доказывать включение $\rho' \in [T]$ индукцией по построению формулы Ф. Используем сокращение $\bar{y} = (y_1, \ldots, y_m)$ и обозначение $l = \lfloor \log_2 z \rfloor^k$. Считаем, что все подформулы Ф задают предикаты от переменных $\bar{X};\ \bar{y}$ (от переменных, не встречающихся в подформулах, предикаты зависят фиктивно).

1. Переменные и их экспоненты: $h_1(\bar{X};\ \bar{y}) = y_i,\ h_2(\bar{X};\ \bar{y}) = 2^{y_i}$.

   Отметим, что $y_i,\ 2^{y_i} < 2^l$. В первом случае строим функцию $f_{y_i}(\bar{x}, z)$ из $[T]$, равную

$$\sum_{0 \leqslant y_1, \ldots, y_m < l} y_i 2^{y_1 l + y_2 l^2 + \ldots + y_m l^m} = \\ = \left(\sum_{y_i = 0}^{l-1} y_i 2^{y_i l^i} \right) \prod_{\substack{j = \overline{1,m} \\ j \neq i}} \sum_{y_j = 0}^{l-1} 2^{y_j l^j} = s_2(l,\ l^i) \prod_{\substack{j = \overline{1,m} \\ j \neq i}} \mathrm{rep}(1,\ l,\ l^j).$$

   Во втором случае аналогично получаем

$$f_{2^{y_i}}(\bar{x}, z) = \left(\sum_{y_i = 0}^{l-1} 2^{y_i} 2^{y_i l^i} \right) \prod_{\substack{j = \overline{1,m} \\ j \neq i}} \sum_{y_j = 0}^{l-1} 2^{y_j l^j} = \\ = \mathrm{rep}(1,\ l,\ l^i + 1) \prod_{\substack{j = \overline{1,m} \\ j \neq i}} \mathrm{rep}(1,\ l,\ l^j) \in [T].$$

2. Элементарные предикаты $(y_i \leqslant y_j)$, $\mathrm{BIT}(y_i, y_j)$, $X_i(y_j)$.

Напомним, что значение производящей функции можно представить в виде

$$f_h(\bar{x}, z) = \langle h_0, \ldots, h_{l^m-1};\ l\rangle,$$

где значения h на всех наборах $y_1, \ldots, y_m$ соответствующим образом упорядочены.

Тогда

$$f_{(y_i \leqslant y_j)}(\bar{x}, z) = \mathrm{incr}(\mathrm{cmp}(f_{y_j}(x, z),\ f_{y_i}(x, z),\ l^m, l),\ l^m,\ l) \in [T].$$

Для вычисления $f_{\mathrm{BIT}(y_i, y_j)}$ сформируем $A = f_{y_i}(\bar{x}, z) \wedge f_{2^{y_j}}(\bar{x}, z)$. Значение A можно представить в виде $\langle a_0, \ldots, a_{l^m-1};\ l\rangle$, где неравенство $a_i \neq 0$ выполняется тогда и только тогда, когда при соответствующих номеру i значениях переменных $\bar{y}$ верно $\mathrm{BIT}(y_i, y_j)$. Следовательно,

$$f_{\mathrm{BIT}(y_i, y_j)}(\bar{x}, z) = \mathrm{incr}(\mathrm{not}(\mathrm{cmpeq}\,(A, 0,\ l^m, l),\ l^m),\ l^m,\ l) \in [T].$$

Выражение $f_{X_i(y_j)}$ в $[T]$ получается заменой $f_{y_i}(\bar{x}, z)$ в выражении $f_{\mathrm{BIT}(y_i, y_j)}$ на выражение $\mathrm{rep}(\mathrm{reverse}(x_i, \lfloor \log_2 z \rfloor),\ l^m,\ l)$. Здесь учитывается то, что переменная x_i одинакова при всех значениях $\bar{y}$, а ее биты нумеруются слева направо, а не справа налево.

3. Логические операции $\&, \vee, \neg$.

Пусть $f_{\Phi_1}, f_{\Phi_2} \in [T]$. Нетрудно видеть, что

$$f_{\Phi_1 \& \Phi_2}(\bar{x}, z) = f_{\Phi_1}(\bar{x}, z) \wedge f_{\Phi_2}(\bar{x}, z),$$
$$f_{\Phi_1 \vee \Phi_2}(\bar{x}, z) = \mathrm{or}(f_{\Phi_1}(\bar{x}, z),\ f_{\Phi_2}(\bar{x}, z),\ l^{m+1}),$$
$$f_{\neg \Phi_1}(\bar{x}, z) = \mathrm{incr}(\mathrm{not}(\mathrm{decr}(f_{\Phi_1}(x, z), l^m, l),\ l^m),\ l^m,\ l).$$

4. Навешивание кванторов $\exists, \forall, M$.

Пусть $f_\Phi \in [T]$ и навешивается квантор $(Q\, y_i)$. По утверждению 19 с использованием функции swap_m переставим переменные так, чтобы навешивание квантора происходило по первой переменной (далее обозначаем ее y_1).

Навешивание всех трех кванторов будем моделировать одним и тем же способом: при каждом наборе значений $y_2, \ldots, y_m$ подсчитываем сумму значений предиката по всем значениям y_1 и сравниваем эту сумму с порогом. Порог $p = 1$ для $Q = \exists$, $p = l$ для $Q = \forall$ и $p = \lfloor l/2 \rfloor + 1$ для $Q = M$ (в любом случае p выразимо в $[T]$). Если сумма больше или равна p, то результат — истина, в противном случае — ложь.

Исходную производящую функцию можно представить в виде

$$f_\Phi(x, z) = \langle h_0, \ldots, h_{l^{m-1}-1};\ l^2\rangle, \quad h_i = \langle h_{i,0}, \ldots, h_{i,l-1};\ l\rangle \quad (i = \overline{1,\ l^{m-1}-1}).$$

Код h_i содержит набор значений предиката на всех значениях переменной y_1 при фиксированных значениях всех остальных переменных. Все числа $h_{i,j}$ равны 0 или 1, а количество чисел в одном блоке h_i равно $l < 2^l$. Поэтому можно одновременно вычислить суммы значений

предиката по всем значениям y_1 при каждом фиксированном наборе значений $y_2, \ldots, y_m$:

$$S = \mathrm{sum}(f_\Phi(x, z),\ l^{m-1},\ l,\ l).$$

Для сравнения сумм с порогом формируем $P = \mathrm{rep}(p,\ l^{m-1},\ l^2)$ и производим сравнения:

$$C = \mathrm{incr}(\mathrm{cmp}(S,\ P,\ l^{m-1},\ l^2),\ l^{m-1},\ l^2) = \langle v_0, \ldots, v_{l^{m-1}-1};\ l^2\rangle,$$

где v_i — результат сравнения и итоговое значение предиката после навешивания квантора при соответствующих значениях переменных $y_2, \ldots, y_m$. Осталось продублировать эти результаты для всех значений переменной y_1, которая стала фиктивной:

$$f_{(Qy_1)\Phi}(\bar{x}, z) = C \cdot \mathrm{rep}(1, l, l) \in [T].$$

Теперь, по утверждению 19, вернем исходный порядок переменных с помощью функции swap_m.

Осталось избавиться от фиктивных переменных. Имеем $\rho(y_1, \ldots, y_r) \equiv \equiv \rho'(y_1, \ldots, y_m)$ и $f_{\rho'} \in [T]$. Тогда $f_\rho(\bar{x}, z) = \mathrm{rm}(f_{\rho'}(\bar{x}, z),\ 2^{l^{r+1}}) \in [T]$.

Наконец, можно доказать принадлежность классу $[T]$ произвольной функции из FOM.

Утверждение 26. *Пусть* $f(\bar{x}) \in \mathrm{FOM}$. *Тогда* $f(\bar{x}) \in [T]$.

Доказательство. По утверждению 3 из § 1 выберем k с условием $\mathrm{len}(f(x_1, \ldots, x_n)) \leqslant q^k$, где $q = \max(\mathrm{len}(x_1), \ldots, \mathrm{len}(x_n),\ 2)$, и k-FOM-определимый предикат $F(\bar{X};\ y)$ такой, что

$$\mathrm{BIT}(f(\bar{x}),\ y) \equiv F(\bar{X};\ y),$$

где каждое слово X_i представляет собой двоичную запись числа x_i, дополненную незначащими нулями для достижения длины q. Для производящей функции предиката F имеем

$$f_F(\bar{x}, z) = \sum_{y=0}^{l-1} F(\bar{X};\ y)2^{yl} = \sum_{y=0}^{l-1} \mathrm{BIT}(f(\bar{x}),\ y)2^{yl} \in [T],$$

где $l = \lfloor \log_2 z \rfloor^k$. Тогда при подстановке $2^{\max(\lfloor \log_2 x_1 \rfloor + 1, \ldots,\ \lfloor \log_2 x_n \rfloor + 1,\ 2)}$ вместо переменной z получим $f(\bar{x}) = \mathrm{decr}(f_F(\bar{x}, z),\ l,\ l)$.

Из теоремы 10 и вхождения функции $x^{\lfloor \log_2 y \rfloor}$ в класс FOM (см. теорему 1 из § 1) нетрудно также вывести, что базис по суперпозиции в классе FOM образует система функций

$$\{x + y,\quad x \dot{-} y,\quad x \wedge y,\quad \lfloor x/y \rfloor,\quad x^{\lfloor \log_2 y \rfloor}\}.$$

4.3. Базисы по суперпозиции к классе AC^0. Ряд результатов, продолжающих линию теоремы 10, получил С. Маззанти. В работе [49] доказательство теоремы 10 переложено на технику индуктивного определения класса TC^0, использующего операцию CRN (см. § 3).

Основу используемой техники работы [49] составляют производящие функции следующего вида:

$$f_g\left(\begin{array}{c}\langle x_{1,1},\ldots,x_{1,n};\ l\rangle,\\ \ldots,\\ \langle x_{k,1},\ldots,x_{k,n};\ l\rangle,\ 2^n,\ 2^l\end{array}\right)=\langle g(x_{1,1},\ldots,x_{k,1}),\ldots,g(x_{1,n},\ldots,x_{k,n});\ l\rangle.$$

Эта функция применяет функцию g одновременно к нескольким наборам аргументов (записанных в наборе кодов «по столбцам») и составляет код набора полученных значений.

Приведем некоторые общие соображения касательно возможности получения базиса в классе AC^0. Построенный в теореме 10 базис по суперпозиции в классе TC^0 (FOM) содержит единственную функцию, которая не принадлежит AC^0, — функцию $\lfloor x/y\rfloor$.

В доказательстве теоремы 10 функция $\lfloor x/y\rfloor$ используется при делении на степени двойки и для получения некоторых видов суммирования (эти операции выразимы в AC^0 с помощью, например, операции CRN). В произвольной форме функция $\lfloor x/y\rfloor$ применяется при получении функции xy. Но сама функция умножения существенно используется только при получении функции $\lfloor\log_2 x\rfloor\in\mathrm{AC}^0$, а также при подсчете битовых сумм и вычислении функции swap_m (которые теряют свою важность при отсутствии необходимости моделировать работу мажоритарного квантора).

Таким образом, можно ожидать, что модификация доказательства теоремы 10 с добавлением новых функций в базис приведет к получению базиса по суперпозиции в классе AC^0. Подобные преобразования были применены в работе [51] к доказательству из [49].

Напомним, что при $x_0,\ldots,x_{n-1}<2^l,\ l\geqslant 1$ код набора $(x_0,\ldots,x_{n-1})$ определяется как

$$\langle x_0,\ldots,x_{n-1};\ l\rangle=\sum_{i=0}^{n-1}x_i\cdot 2^{li}.$$

Расширим это определение на случай произвольных x_i и l следующим образом:

$$\langle x_0,\ldots,x_{n-1};\ l\rangle=\sum_{i=0}^{n-1}\mathrm{rm}(x_i,\ 2^l)\cdot 2^{li}.$$

Пусть

$$\begin{aligned}&\mathrm{ar2l}(l)=\langle 0,\ 1^2,\ 2^2,\ \ldots,\ (\mathrm{len}(l)-1)^2;\ \mathrm{len}(l)\rangle,\\ &\mathrm{convl}(x,l,r,n)=\langle x_1,\ldots,x_{\mathrm{len}(n)};\ \mathrm{len}(r)\rangle,\\ &\mathrm{repl}(x,l,n)=\langle\underbrace{x,\ldots,x}_{n};\ \mathrm{len}(l)\rangle,\end{aligned}$$

где x_i — это i-й справа блок из $\mathrm{len}(l)$ битов двоичной записи x. Отметим, что функция $\mathrm{repl}(x,l,n)$ аналогична функции $\mathrm{rep}(x,n,l)$ и суммированию

$s_1(n,l)$ из доказательства теоремы 10, а функция $\mathrm{ar2l}(l)$ схожа с суммированием $s_2(n,l)$. Функция convl меняет длину кода, и ее можно считать обобщением функций $\mathrm{incr}(x,n,l)$ и $\mathrm{decr}(x,n,l)$.

Теорема 11 [51].

$$\mathrm{AC}^0 = [\,1,\quad x+y,\quad x\dot{-}y,\quad x\wedge y,\quad x *_b y,\quad \mathrm{len}(x),\quad \lfloor x/2^{\mathrm{len}(y)}\rfloor,$$
$$\mathrm{ar2l}(l),\quad \mathrm{repl}(x,l,n),\quad \mathrm{convl}(x,l,r,n)\,].$$

В [52] получены другие базисы в классах AC^0 и TC^0. Будем говорить, что числа x,n,l удовлетворяют условию AC^0_SUM, если $n>0$, $l>0$ и выполняется хотя бы одно из требований:

1. $x=\langle x_1,\ldots,x_n;\ \mathrm{len}(l)\rangle$, где $x_i<2^{\mathrm{len}(l)}$ и $x_i\wedge x_j=0$ при $i\neq j$;
2. $n=l$ и $x=\langle 0,\ 1,\ \ldots,\ \mathrm{len}(l)-1;\ \mathrm{len}(l)\rangle$;
3. $n=l$ и $x=\langle \underbrace{1,\ldots,1}_{\mathrm{len}(l)};\ \mathrm{len}(l)\rangle$.

Обозначим

$$\mathrm{p}(x,n,l)=x\cdot\sum_{i=0}^{\mathrm{len}(n)-1}2^{\mathrm{len}(l)\cdot i}$$

и

$$\mathrm{rp}(x,n,l)=\begin{cases}\mathrm{p}(x,n,l), & \text{если } x,n,l \text{ удовлетворяют условию } \mathrm{AC}^0\text{\_SUM},\\ 0 & \text{в ином случае.}\end{cases}$$

Теорема 12 [52].

$$\mathrm{AC}^0=[\,s_1(x),\quad x\dot{-}y,\quad x\wedge y,\quad \lfloor x/2^{\mathrm{len}(y)}\rfloor,\quad \mathrm{rp}(x,n,l)\,],$$
$$\mathrm{TC}^0=[\,s_1(x),\quad x\dot{-}y,\quad x\wedge y,\quad \lfloor x/2^{\mathrm{len}(y)}\rfloor,\quad \mathrm{p}(x,n,l)\,].$$

В этой теореме базисы классов AC^0 и TC^0 отличаются только одной функцией, причем фактически отличается только область определения этой функции. Кроме того, базис в TC^0 здесь построен без использования функции $\lfloor x/y\rfloor$, доказательство принадлежности которой классу TC^0 очень нетривиально (см. теорему 1 из § 1).

Наличие базиса по суперпозиции в классе TC^0 и особенно в классе AC^0 позволяет строить базисы и в более крупных сложностных классах. Многие известные сложностные классы имеют AC^0-полные проблемы [26]. Если добавить к базису класса AC^0 производящую функцию (введенного в [49] типа) для AC^0-полного в A предиката ρ, то будет получен базис в классе A (производящая функция ρ требуется для построения функций из A; при построении предикатов достаточно использовать сам предикат ρ).

В [49, 51, 52] таким образом построены базисы для класса NC^1 и функциональных аналогов классов L, P, PSPACE.

СПИСОК ЛИТЕРАТУРЫ

1. Бельтюков А. П. Машинное описание и иерархия начальных классов Гжегорчика // Зап. научн. сем. ЛОМИ. — 1979. — Т. 88. — С. 30–46.
2. Волков С. А. Экспоненциальное расширение класса функций, элементарных по Сколему, и ограниченные суперпозиции арифметических функций // Математические вопросы кибернетики. Вып. 16. — М.: ФИЗМАТЛИТ, 2007. — С. 163–190.
3. Волков С. А. Порождение некоторых классов рекурсивных функций суперпозициями простых арифметических функций // Докл. АН. — 2007. — Т. 415, № 4. — С. 439–440.
4. Волков С. А. Простой базис по суперпозиции в классе FFOM и классы, получаемые ограниченными суперпозициями // Персональная страница С. А. Волкова, 2007. URL: http://volkov-articles.narod.ru/ffom_manyfloors.pdf (Дата обращения 03.07.2023).
5. Волков С. А. Конечные базисы по суперпозиции в классах элементарных функций: Дисс. ... канд. физ.-мат. наук: 01.01.09 — М.: МГУ, 2009.
6. Косовский Н. К. Основы теории элементарных алгоритмов. — Л.: Изд-во ЛГУ им. А. А. Жданова, 1987.
7. Мальцев А. И. Итеративные алгебры и многообразия Поста // Алгебра и логика. — 1966. — Т. 5, № 2. — С. 5–24.
8. Мальцев А. И. Итеративные алгебры Поста. — Новосибирск: Изд-во НГУ, 1976. — 100 с.
9. Марченков С. С. Элементарные рекурсивные функции. — М.: МЦНМО, 2003. — 112 с.
10. Марченков С. С. Суперпозиции элементарных арифметических функций // Дискретный анализ и исследование операций. — 2006. — Т. 13, № 4. — С. 33–48.
11. Марченков С. С. Ограниченная монотонная рекурсия и МГ-автоматы // Программирование. — 2013. — № 6. — С. 3–11.
12. Марченков С. С. Об элементарных словарных функциях, получаемых на основе ограниченной префиксной конкатенации // Дискретная математика. — 2015. — Т. 27, № 3. — С. 44–55.
13. Марченков С. С. Операция ограниченной префиксной конкатенации и конечные базисы по суперпозиции // Дискретная математика. — 2016. — Т. 28, № 4. — С. 91–99.
14. Марченков С. С. Классы элементарных рекурсивных функций. — М.: ФИЗМАТЛИТ, 2017. — 136 с.
15. Марченков С. С. Об операциях ограниченного суффиксного суммирования и мультиплицирования // Дискретный анализ и исследование операций. — 2017. — Т. 24, № 4. — С. 60–76.
16. Марченков С. С. Операция ограниченной префиксной конкатенации и конечные базисы по суперпозиции // Вестник МГУ. Серия 15. Вычислительная математика и кибернетика — 2022. — № 4. — С. 28–35.
17. Марченков С. С., Савицкий И. В. Машины в теории вычислимых функций. — М.: МАКС Пресс, 2018. — 88 с.
18. Непомнящий В. А. Рудиментарные предикаты и тьюринговы вычисления // Докл. АН СССР. — 1970. — Т. 195, № 2. — С. 282–284.
19. Орлов Е. А. Словарное определение класса функций, элементарных по Кальмару. Дипломная работа. Факультет вычислительной математики и кибернетики МГУ, 2012.
20. Разборов А. А. Нижние оценки размера схем ограниченной глубины в полном базисе, содержащем функцию логического сложения // Матем. заметки. — 1987. — Т. 41, № 4. — С. 598–607.
21. Савицкий И. В. О совпадении сложностных классов BPC и TC^0 // Вестник МГУ. Серия 15. Вычислительная математика и кибернетика. — 2022. — № 4. — С. 36–45.
22. Храпченко В. М. О сложности реализации симметрических функций формулами // Матем. заметки. — 1972. — Т. 11, № 1. — С. 109–120.
23. Яблонский С. В. Введение в дискретную математику. — М.: Высшая школа, 2008. — 384 с.
24. Ajtai M. Σ^1_1-Formulae on finite structures // Annals of Pure and Applied Logic. — 1983. — V. 24, N1. — P. 1–48.
25. Allender E., Loui M. C., Regan K. W. Complexity classes // Algorithms and Theory of Computation Handbook. General Concepts and Techniques / Ed. M. J. Atallah, M. Blanton. — Boca Raton, Fla: Taylor & Francis, 2009. — P. 597–620.

26. Allender E., Loui M. C., Regan K. W. Reducibility and completeness // Algorithms and Theory of Computation Handbook. General Concepts and Techniques / Ed. M. J. Atallah, M. Blanton. — Boca Raton, Fla: Taylor & Francis, 2009. — P. 621–650.
27. Mix Barrington D. A., Immerman N., Straubing H. On uniformity within NC^1 // J. of Computer and System Sci. — 1990. — V. 41, N3. — P. 274–306.
28. Beame P., Cook S., Hoover J. Log depth circuits for division and related problems // SIAM Journal of Computing. — 1986. — V. 15, N4. — P. 994–1003.
29. Bellantoni S., Cook S. A new recursion-theoretic characterization of the polytime functions // Comput. Complexity. — 1992. — V. 2, N2. — P. 97–110.
30. Borodin A. On Relating Time and Space to Size and Depth // SIAM J. Comput. — 1977. — V. 6, N4. — P. 733–744.
31. Buss S. R. The graph of multiplication is equivalent to counting // Information Processing Letters. — 1992. — V. 41, N4. — P. 199–201.
32. Chandra A. K., Stockmeyer L., Vishkin U. Constant depth reducibility // SIAM J. Comput. — 1984. — V. 13, N2. — P. 423–439.
33. Chiu A., Davida G., Litow B. Division in logspace-uniform NC^1 // RAIRO-Theor. Inf. Appl. — 2001. — V. 35, N3. — P. 259–275.
34. Clote P. G. Sequential, machine-independent characterizations of the parallel complexity classes $AlogTIME$, AC^k, NC^k and NC // Feasible Mathematics. — Boston, MA: Birkhäuser Boston, 1990. — P. 49–69.
35. Clote P. G., Takeuti G. First order bounded arithmetic and small boolean circuit complexity classes // Feasible Mathematics II. — Boston, MA: Birkhäuser Basel, 1995. — P. 154–218.
36. Clote P. G. Computation models and function algebras // Handbook of Computability Theory. — Amsterdam: Elsevier Science B. V., 1999. — P. 589–681.
37. Cook S. A. Deterministic CFL's are accepted simultaneously in polynomial time and log squared space // Proceedings of the eleventh annual ACM symposium on Theory of computing. — NY: Association for Computing Machinery, 1979. — P. 338–345.
38. Cook S. A. A taxonomy of problems with fast parallel algorithms // Information and Control. — 1985. — V. 64, N1–3. — P. 2–22.
39. Fagin R., Klawe M. M., Pippenger N. J., Stockmeyer L. Bounded-depth, polynomial-size circuits for symmetric functions // Theoretical Computer Science. — 1985. — V. 36. — P. 239–250.
40. Furst M., Saxe J. B., Sipser M. Parity, circuits, and the polynomial-time hierarchy // Mathematical Systems Theory. — 1984. — V. 17. — P. 12–27.
41. Grzegorczyk A. Some classes of recursive functions // Rozprawy Matematyzne. — Warzawa, 1953. — Vol. 4. — P. 1–46. [Имеется перевод: Гжегорчик А. Некоторые классы рекурсивных функций // Проблемы математической логики: сложность алгоритмов и классы вычислимых функций. — М.: Мир, 1970. — С. 9–49.]
42. Håstad J. Almost optimal lower bounds for small depth circuits // Proceedings of the 18th Annual ACM Symposium on Theory of Computing STOC '87. — Berkeley, CA: ACM Press, 1986. — P. 6–20.
43. Hesse W., Allender E., Mix Barrington D. A. Uniform constant-depth threshold circuits for division and iterated multiplication // J. of Computer and System Sci. — 2002. — V. 65, N4. — P. 695–716.
44. Immerman N. Expressibility as complexity measure: results and directions. Tech. Report. — Yale University Departament of Computer Science. — 1987.
45. Immerman N. Expressibility and Parallel Complexity // SIAM J. Comput. — 1989. — V. 18, N3. — P. 625–638.
46. Immerman N. Descriptive Complexity. — Springer, 1998.
47. Lind J., Meyer A. R. A characterization of log-space computable functions // SIGACT News. — 1973. — V. 5, N3. — P. 26–29.
48. Kalmar L. Ein einfaches Beispiel für unentscheibares Problem // Matematikai és fizikai lapok. — 1943. — Bd. 50. — S. 1–23.
49. Mazzanti S. CRN elimination and substitution bases for complexity classes // Fundamenta Informaticae. — 2012. — N1. — P. 29–58.
50. Mazzanti S. Iteration on notation and unary functions // Math. Log. Quart. — 2013. — V. 59, N6. — P. 415–434.
51. Mazzanti S. Bases for AC^0 and other complexity classes // Fundamenta Informaticae. — 2015. — N4. — P. 433–460.

52. Mazzanti S. New substitution bases for complexity classes // Math. Log. Quart. — 2019. — V. 66, N1. — P. 1–14.
53. Murray C., Williams R. Circuit lower bounds for nondeterministic quasi-polytime: an easy witness lemma for NP and NQP // Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing. — Los Angeles, CA: ACM, 2018. — P. 890–901.
54. Pippenger N. On simultaneous resource bounds (preliminary version) // 20th Annual Symposium on Foundations of Computer Science. — 1979. — P. 307–311.
55. Parberry I., Schnitger G. Parallel computation with threshold functions // J. of Computer and System Sci. — 1988. — V. 36, N3. — P. 278–302.
56. Quine W. V. Concatenation as a basis for arithmetic // The Journal of Symbolic Logic. — 1946. — V. 11, N4. — P. 105–114.
57. Ritchie R. W. Classes of predictably computable functions // Trans. Am. Math. Soc. — 1963. — V. 106, N1. — P. 139–173. [Имеется перевод: Ричи Р. В. Классы предсказуемо вычислимых функций // Проблемы математической логики: сложность алгоритмов и классы вычислимых функций. — М.: Мир, 1970. — С. 50–93.]
58. Ruzzo W. L. On uniform circuit complexity // J. of Computer and System Sci. — 1981. — V. 22, N3. — P. 365–383.
59. Smolensky R. Algebraic methods in the theory of lower bounds for Boolean circuit complexity // Proceedings of the 19th annual ACM conference on Theory of computing STOC '87. — NY: ACM Press, 1987. — P. 77–82.

Поступило в редакцию 4 VII 2023