Теормин по ДГДМиК

Иван Кобзев

январь 2020

ИДЁМ ПОДТВЕРЖДАТЬ

Содержание

В центре страницы ссылка на программу курса нашего учебного года. Основное отличие от предыдущих лет в том, что в третьей части билет по инвариантным классам передвинут с предпоследнего на второй. Надеюсь, что и в последующие годы существенных изменений не будет.

Почти вся наша группа пришла подтверждать оценку, и этот теормин тоже, в первую очередь, расчитан на подтверждающих. В нём почти нет идей доказательств, он состоит из определений и формулировок основных утверждений, которые иногда удавалось связать логически.

Ссылка на программу курса 2019-2020. Список билетов на третьей странице

Генерал раздал всем автоматы.
А кому не хватило — сапёрные
лопатки.

ДМБ

1 Конечные автоматы

1 Конечный автомат-распознаватель, конечно-автоматное множество.

Определение 1.1. *Алфавит* A - множество $\{a_1, \ldots, a_n\}$, называемых буквами алфавита. *Слово* - конечная последовательность букв без пропусков одна за другой. Множество всех слов обзначаем A^*. Кроме того, в A^* включаем Λ - пустое слово. *Конкатенация* - запись двух слов одно за другим. $\bar{a}^n$ - конкатенация n экземпляров слова $\bar{a}$. Положим $\bar{a}^0 = \Lambda$.

Мы считаем, что автомат может читать любые слова из A^*. В силу неопределённости длины слова, автомат читает его побуквенно в дискретные моменты времени. С чтением каждой буквы он переходит в новое состояние. Функцию, которая управляет переходом из состояния в состояние, назовём *функцией переходов*. Она по нынешнему состоянию и считанной букве определяет следующее состояние. Считаем, что при принятии на вход Λ, автомат ничего не делает и остаётся в том же состоянии.

Тем самым, обобщив всё вышесказанное, придём к понятию простейшего автомата.

Определение 1.2. *Простейший автомат* задаётся входным алфавитом A, конечным множеством состояний Q и функцией переходов $f : A \times Q \to Q$. Символически изобразим $\mathcal{A} = (A, Q, f)$.

Если считать, что в момент t автомату приходит буква $x(t)$, а состояние, в котором он после этого будет находиться, - $q(t)$, то, задав дополнительно начальное состояние, получим систему, отражающую работу автомата во времени.

$$\begin{cases} q(t) = f(x(t), q(t-1)) \\ q(0) = q_1 \end{cases}$$

Автомат с заданным начальным состоянием обычно называют *инициальным*, а сами уравнения - *каноническими уравнениями автомата* A, который, заметим, теперь задаётся четвёркой (A, Q, f, q_1).

Существует несколько способов графического задания инициального автомата. Один из наиболее распространённых - *диаграмма Мура*. Она представляет собой $r = |Q|$ точек на плоскости, соответствующих состояниям. Из каждой точки проводится ровно $k = |A|$ направленных дуг в другие точки диаграммы или в себя, соответствующих буквам. Дуга из q_i-го состояния в q_j-е, соответствующая букве a_i, означает, что автомат, находящийся

в q_i-м состоянии по букве a_i перейдёт в состояние q_j, то есть $q_j = f(a_i, q_i)$. Начальное состояние обычно помечают символом $*$.

У описанного автомата есть один недостаток: он не умеет "отвергать"или "допускать"(распознавать) слова. Избавимся от него введением нового множества $F \subseteq Q$ *заключительных состояний*. Теперь будем считать, что слово *допускается* автоматом, если после его подачи автомат окажется в состоянии $q_f \in F$. В противном случае считаем, что слово автоматом *отвергается*.

Таким образом, мы можем дать строгое

Определение 1.3. *Автомат без выхода* (дальше просто автомат) - математическая модель распознающего устройства с конечной памятью, задающаяся набором (A, Q, f, q_1, F). Конечно, может быть, что $q_1 \in F$.

Кроме того, введём

Определение 1.4. *Конечно-автоматное множество* $D(\mathcal{A})$ - множество слов из A^*, допускаемых автоматом $\mathcal{A}$.

Дальше у Марченкова следует несколько примеров, которые я опускаю. Их спокойно можно воспроизвести, придумав диаграмму Мура с заданным начальным и конечными состояниями, по которой строится конечно-автоматное множество.

2 Правоинвариантное отношение эквивалентности, связь с конечно-автоматными множествами.

Пусть в заданном автомате (A, Q, f, q_1, F) достижимо любое состояние $q_i \in Q$, то есть найдётся некоторое слово $\bar{a} \in A^*$, при подаче которого автомат переходит из состояния q_1 в состояние q_i.

Разобьём все слова в A^* на r непересекающихся классов, при этом в одном классе K_i оказываются слова, переводящие автомат в состояние q_i после их подачи. Очевидно, что такое разбиение на классы порождает отношение эквивалентности $\sim$ (рефлексивность, симметричность, транзитивность проверим ручками). Назовём $\sim$ *конечно-автоматной эквивалентностью*, порождённой автоматом $\mathcal{A}$.

Конечно-автоматная эквивалентность обладает свойством *правой инвариантности*. Это означает, что $\forall c \in A^* : \bar{a} \sim \bar{b} \Longrightarrow \bar{a}\bar{c} \sim \bar{b}\bar{c}$.

Теперь рассмотрим некоторое правоинвариантное отношение эквивалентности на A^*. Оно разобьёт A^* на несколько попарно непересекающихся классов. Будем говорить, что отношение имеет *конечный индекс*, если множество этих классов конечно.

Верна следующая

Теорема 2.1. *Любой конечный инициальный автомат порождает правоинвариантное отношение эквивалентности конечного индекса. И наоборот, любое правоинвариантное отношение эквивалентности конечного индекса является конечно-автоматной эквивалентностью.*

Учитывая определение конечно-автоматного множества и эту теорему, мы приходим к следующей теореме, задающей связь правоинвариантного отношения эквивалентности и конечно-автоматных множеств.

Теорема 2.2. *Любое непустое конечно-автоматное множество есть объединение некоторого числа классов подходящего правоинвариантного отношения эквивалентности конечного индекса. Обратно, объединение любого числа классов правоинвариантного отношения конечного индекса эквивалентности есть конечно-автоматное множество.*

Эту теорему обычно используют для доказательства непринадлежности множества классу конечно-автоматных, поскольку она даёт описание конечно-автоматных множеств, не опираясь на понятие автомата.

3 Замкнутость класса конечно-автоматных множеств относительно теоретико-множественных операций.

Теорема 3.1. *Операции дополнения, объединения и пересечения сохраняют конечную автоматность множества.*

Соответственно, через эти операции можно выразить другие, а тем самым доказать, что они тоже сохраняют конечную автоматность. Например $A \setminus B = X \cap \bar{Y}$, $A \triangle B = (A \setminus B) \cup (B \setminus A)$.

4 Недетерминированные автоматы, процедура детерминизации.

Определение 4.1. *Недетерминированный автомат* - автомат, так же задающийся набором (A, Q, f, q_1, F), но теперь функция f, вообще говоря, не является однозначной. То есть теперь $f : A \times Q \to 2^Q \setminus \{\varnothing\}$ и по одной и той же букве из одного и того же состояния мы можем перейти в разные.

Конечно, недетерминированные автоматы не встречаются на практике, но при этом эта абстракция часто позволяет проще решать некоторые задачи.

Считаем, что недетерминированный автомат допускает слово, если под действием слова автомат может попасть хотя бы в одно из заключительных состояний.

Теорема 4.2. *Класс множеств, допускаемых конечными недетерминированными автоматами, совпадает с классом конечно-автоматных множеств.*

Естественно, любое конечно-автоматное является допустимым для некоторого недетерминированного автомата (просто потому, что любой конечный автомат можно считать недетерминированным). Обратно доказывается построением детерминированного автомата $\mathcal{A}'$ с тем же допускаемым

множеством: $D(\mathcal{A}') = D(\mathcal{A})$. Процедура детерминизации заключается в построении новых состояний $q'_1, \ldots, q'_{2^r-1}$, являющихся всеми непустыми подмножествами исходного множества состояний, и переопределении функции перехода: $f'(a_i, q'_j) = q'_l$, где $q'_j = \{q_{j_1}, \ldots q_{j_s}\}, q'_l = f(a_i, q_{j_1}) \cup \ldots \cup f(a_i, q_{j_s})$.

5 Операции произведения и итерации. Замкнутость класса конечно-автоматных множеств относительно операций произведения и итерации.

Определение 5.1. Пусть $X, Y \subseteq A^*$. *Произведением* множеств X, Y называется множество

$$X \cdot Y = \bigcup_{\bar{a} \in X, \bar{b} \in Y} \bar{a}\bar{b}.$$

Естественно обозначать произведение множества X на себя n раз n-й степенью X, то есть $X^n = X \cdot X \cdot \ldots \cdot X$ (n раз). Причём положим для удобства $X^0 = \{\Lambda\}$.

Определение 5.2. Пусть $X \subseteq A^*$. *Итерацией* множества $X \neq \varnothing$ называется

$$X^* = \bigcup_{i=0}^{\infty} X^i.$$

Кроме того, положим итерацию пустого множества равной пустому множеству

Из определения понятно, что итерация состоит из тех слов, которые получаются конкатенацией некоторого количества слов из X. Соответственно, если $\bar{a} \in X, \bar{a} \neq \Lambda$, то итерация - бесконечное множество, поскольку в него входят все степени $\bar{a}, \bar{a}^2, \bar{a}^3, \ldots$. Если же $X = \{\Lambda\}$, то $X^* = \{\Lambda\}$.

Теорема 5.3. *Операция произведения сохраняет конечную автоматность множеств.*

Идея доказательства состоит в том, чтобы присоединить в конечных состояниях к автомату $\mathcal{A}$, который допускает множество A, автомат $\mathcal{B}$, допускающий множество B, и получить некоторый недетерминированный автомат $\mathcal{C}$, допускающий множество $D(\mathcal{A}) \cdot D(\mathcal{B})$. Проблема может заключаться в том, что полученный автомат, вообще говоря, может быть проходим в обоих направлениях, а операция произведения некоммутативна, поэтому сначала автомат $\mathcal{B}$ модифицируется таким образом, чтобы нельзя было попасть в его начальное состояние ни по какому переходу. Тогда, придя в какое-то конечное состояние первого автомата, мы можем быть уверены, что, продолжая работу как второй автомат, мы не вернёмся ни в одно из состояний автомата $\mathcal{A}$.

Теорема 5.4. *Операция итерации сохраняет конечную автоматность множеств.*

Если множество $X = D(\mathcal{A}) = \{\Lambda\}$, то итерация по определению совпадает с X. Если же есть некоторое непустое слово среди допустимых слов автомата $\mathcal{A}$, то идея заключается в следующем: будем строить недетерминированный автомат $\mathcal{C}$, допускающий множество X^*, отождествляя заключительные состояния автомата $\mathcal{A}$ с его начальным состоянием q_1.

6 Регулярные выражения и регулярные множества.

Определение 6.1. Введём понятия *регулярного выражения* над алфавитом A и задаваемого им *регулярного множества* в алфавите A индуктивно.

1. $\varnothing$ является регулярным выражением над алфавитом A и задаёт пустое регулярное множество.
2. Λ является регулярным выражением над алфавитом A и задаёт регулярное множество $\{\Lambda\}$.
3. $a_i \in A$ является регулярным выражением над алфавитом A и задаёт регулярное множество из однобуквенного слова $\{a_i\}$.
4. Пусть α, β - регулярные выражения над алфавитом A, задающие регулярные множества X, Y соответственно. Тогда $(\alpha \cup \beta), (\alpha\beta), (\alpha)^*$ - регулярные выражения над алфавитом A, задающие, соответственно, регулярные множества $X \cup Y, X \cdot Y, X^*$.

7 Теорема Клини.

Теорема 7.1 (Клини)**.** *Класс конечно-автоматных множеств совпадает с классом регулярных множеств.*

Любое регулярное, очевидно, является конечно-автоматным, в силу определения и замкнутости конечно-автоматных относительно операций $\cup, \cdot, ^*$.

Остаётся доказать, что любое конечно-автоматное множество является регулярным. Пусть $\mathcal{A} = (A, Q, f, q_1, F), X = D(\mathcal{A}), Q = \{q_1, \ldots, q_r\}, F = \{q_{j_1}, \ldots, q_{j_s}\}$. Заметим, что $X = \bigcup\limits_{i=1}^{s} D(\mathcal{A}_i)$, где автомат $\mathcal{A}_i = (A, Q, f, q_1, \{q_{j_i}\})$. Вследствие этого можно считать, что множество F является одноэлементным, то есть $F = \{q_t\}$.

Идея заключается в следующем. Для любых $0 \le k \le r, 0 \le i, j \le r$ обозначим Z_{ij}^k - множество слов, переводящих автомат из состояния q_i в состояние q_j, используя только состояния $\{q_1, \ldots, q_k\}$. Тогда, если мы по индукции по k докажем регулярность всех Z_{ij}^k, то, поскольку $X = Z_{ij}^r$, регулярность X будет доказана.

8 Детерминированные функции. Задание детерминированных функций деревьями. Вес дерева.

Рассмотрим множество $E_{t,k}$ всех k-значных последовательностей α, где $\alpha = \{\alpha(1), \alpha(2), \ldots\}, \alpha(i) \in E_k$. Обозначим через $P_{t,k}$ множество функций от

$n = 1, 2, 3, \ldots$ переменных, определённых на наборах $(\alpha_1, \ldots, \alpha_n), \alpha_i \in E_{t,k}$, и принимающих значения из $E_{t,k}$. То есть подобные функции принимают некоторое количество последовательностей и выдают другую последовательность. Понятно, что можно записывать в векторной форме $f(X)$, где $X = (x_1, x_2, \ldots, x_n)$.

Теперь сделаем финт ушами и будем рассматривать набор последовательностей как последовательность наборов:

$$(\{\alpha_1(1), \alpha_1(2), \ldots\}, \{\alpha_2(1), \alpha_2(2), \ldots\}, \ldots, \{\alpha_n(1), \alpha_n(2), \ldots\}) =$$

$$\{(\alpha_1(1), \alpha_2(1), \ldots, \alpha_n(1)), (\alpha_1(2), \alpha_2(2), \ldots, \alpha_n(2)), \ldots\}.$$

Тем самым мы можем рассматривать функцию от n переменных из $P_{t,k}$ как функцию от одной переменной-вектора X из $P_{t,N}, N = k^n$.

Определение 8.1. Функция $f(X) \in P_{t,N}$ называется детерминированной, если для любых последовательностей α, β:

$$\alpha(1) = \beta(1), \ldots, \alpha(m) = \beta(m)$$

значения функций $\gamma = f(\alpha), \delta = f(\beta)$ тоже совпадают в первых m членах, то есть

$$\gamma(1) = \delta(1), \ldots, \gamma(m) = \delta(m).$$

Проинтерпретировать детерминированную функцию можно как некоторый преобразователь, который имеет n входов $x_1, \ldots, x_n$ и один выход f. На вход подаются последовательности, а значение функции на выходе в любой момент времени m зависит только от значений последовательностей в моменты времени $t = 1, 2, \ldots, m$ (то есть зависит только от значений в прошлом и не зависит от значений в будущем).

Будем представлять детерминированные функции с помощью дерева с некоторым корнем ξ_0 и N рёбрами, исходящими из каждой вершины. Тогда проход по этому дереву, очевидно, можно интерпретировать как последовательность чисел от 0 до $N - 1$, каждое из которых можно представить как n-значную запись этого числа в k-ичной системе счисления (поскольку $N = k^n$). Теперь мы можем рассматривать любое ребро с точки зрения номеров рёбер $\alpha(1), \ldots, \alpha(m)$, по которым надо пройти, чтобы дойти от корня до данного ребра. Тогда, вписав над этим ребром значение детерминированной функции, соответствующее данным номерам рёбер, мы получим представление детерминированной функции как занумерованного дерева.

Определим отношение эквивалентности на поддеревьях исходного дерева как совпадение нумераций при наложении одного поддерева на другое. Тогда число классов эквивалентности, на которое разбивается множество поддеревьев, называется *весом дерева* и соответственно *весом детерминированной функции.*

9 Канонические уравнения, векторная и скалярная формы канонических уравнений.

Детерминированные функции конечного веса будем называть *ограниченно-детерминированными* (у Марченкова *конечно-автоматными*) функциями. Понятно, что можно занумеровать вершины дерева в соответствии с отношением эквивалентности из предыдущего билета, после чего обрезать каждую ветвь до вершины с минимальным номером j таким, что $x_j = x_i$ для некоторого $i < j$. Тогда для о.-д. функций получаем *усечённое дерево*, отождествляя вершины с одинаковыми номерами которого, получим *диаграмму Мура*.

Аналогично каноническим уравнениям работы автомата, введённым в первом билете, можем ввести канонические уравнения для функции, учитывая ещё выходные величины:

$$\begin{cases} Z(t) = \mathcal{F}(X(t), Q(t-1) \\ Q(t) = \mathcal{G}(X(t), Q(t-1)) \\ Q(0) = 0 \end{cases}$$

Здесь Q описывает состояния, Z - выходные значения, а X - входные величины. Это запись канонических уравнений в векторной форме. Довольно просто перейти от векторной формы к скалярной. Положим $l = [\log_k r]$ (необходимо, чтобы $k^l \geq r$, тогда состояния могут быть закодированы словами длины l в алфавите E_k). Тогда

$$\begin{cases} z(t) = F'(x_1(t), \ldots, x_n(t), q_1(t-1), \ldots, q_l(t-1)) \\ q_1(t) = G'_1(x_1(t), \ldots, x_n(t), q_1(t-1), \ldots, q_l(t-1)) \\ \ldots \\ q_l(t) = G'_l(x_1(t), \ldots, x_n(t), q_1(t-1), \ldots, q_l(t-1)) \\ q_1(0) = \ldots = q_l(0) = 0. \end{cases}$$

10 Замкнутость класса конечно-автоматных функций относительно операции суперпозиции.

Обозначим класс детерминированных функций $P_{\partial,k}$, а класс конечно-автоматных - $P_{ka,k}$. Операция суперпозиции вводится как обычно:

$$f(x_1, \ldots, x_n) = g_0(g_1(x_1, \ldots, x_n), \ldots, g_m(x_1, \ldots, x_n)).$$

Теорема 10.1. *Класс $P_{\partial,k}$ замкнут относительно операции суперпозиции*

Теорема 10.2. *Класс $P_{ka,k}$ замкнут относительно операции суперпозиции.*

Доказываем подстановкой суперпозиции в канонические уравнения.

11 Зависимость с запаздыванием. Операция введения обратной связи.

Будем говорить, что детерминированная функция $y = f(x_1, \ldots, x_n)$ зависит от переменной x_i *с запаздыванием*, если значение $y(t)$ зависит от значений $x_j(1), \ldots, x_j(t)$ при $j \neq i$ и $x_i(1), \ldots, x_i(t-1)$, а от значения $x_i(t)$ не зависит.

Операция *введения обратной связи* заключается в следующем: пусть $\{f_1(x_1, \ldots, x_n), \ldots, f_m(x_1, \ldots, x_n)\}$ - система детерминированных функций, и функция f_j зависит с запаздыванием от переменной x_i. Пусть даны канонические уравнения и выход

$$y_j(t) = F_j(x_1(t), \ldots, x_{i-1}(t), x_{i+1}(t), \ldots, x_n(t), q_1(t-1), \ldots, q_l(t-1))$$

Операция обратной связи получается при подстановке правой части этого уравнения во все остальные уравнения системы вместо $x_i(t)$, а само это уравнение вычёркивается.

12 Существование конечных полных систем в классе конечно-автоматных функций.

Теорема 12.1. *При любом $k \geq 2$ класс $P_{ka,k}$ содержит конечную систему функций, полную относительно операций суперпозиции и введения обратной связи.*

Замечание 12.1.1. Такой системой случит система функций $\{f_v, \zeta\}$, где f_v - функция Вебба(стрелка Пирса), а ζ - функция единичной задержки, которая по входу $x(1)x(2)x(3)\ldots$ выдаёт $0x(1)x(2)\ldots$.

Алан Тьюринг сидит дома. Звонит телефон, Алан берет трубку:
— Алан, у нас проблема!
— Что случилось?
На заднем фоне слышится крик машины Тьюринга — "МОЯ ОСТАНОВОЧКА"

типа анек

2 Машины Тьюринга и вычислимые функции

13 Машины Тьюринга. Функции, вычислимые на машинах Тьюринга.

Машина Тьюринга состоит из ленты, считывающе-записывающей головки и управляющего устройства. Лента бесконечна в обе стороны и разбита на клетки. В каждую клетку может быть записана одна из букв конечного алфавита $A = \{a_0, a_1, \ldots, a_k\}$. Обычно выделяется пустой символ a_0. Головка может двигаться влево и вправо, читать символ и записывать в обозреваемую клетку. Управляющее устройство организует перемещение головки и запись символов в клетки. Оно может находиться в одном из конечного числа состояний $q_0, q_1, \ldots, q_r$. Выделяются начальное состояние q_1 и заключительное q_0.

Программа машины состоит из команд вида

$$a_i q_j \to a_l D q_s, D \in \{L, R, S\}.$$

Команда с заданной левой частью ровно одна.

Работа заключается обычно в том, что на ленту записывается некоторое слово, остальные клетки помечаются символами a_0, головка ставится на первую букву слова, а состояние принимает значение q_1.

Условимся представлять число $m \in \mathbb{N}$ (натуральные здесь включают 0!!!) на ленте словом, состоящем из $m+1$ символа 1. Тогда алфавит обязательно должен содержать 1, 0 как пустой символ и, возможно, что-то ещё. *Основным кодом набора* будем называть представление набора чисел как описано выше с разделяющими символами 0.

Пусть $f(x_1, \ldots, x_n) : \mathbb{N} \to \mathbb{N}$, $\mathcal{M}$ - машина Тьюринга с алфавитом A, содержащим $0, 1$. Будем говорить, что *машина вычисляет функцию*, если для любого набора выполняются следующие условия:

1. Если значение $f(m_1, \ldots, m_n)$ определено, то машина через конечное количество тактов остановится, а на ленте будет представлено значение функции.

2. Если значение не определено, то машина либо не остановится никогда, либо остановится, но на ленте не будет представлено в основном коде ни одно число из $\mathbb{N}$.

Если в первом пункте требовать остановки на первой единице основного кода значения функции, а во втором исключительно бесконечность работы машины, то такое понятие мы будем называть *правильным вычислением функции*.

14 Операции композиции и итерации над машинами Тьюринга.

Композиция машин Тьюринга $\mathcal{M}_1, \mathcal{M}_2$ - машина $\mathcal{M}$, которая на исходном слове работает как первая машина, а если первая машина заканчивает работу, то при конечной конфигурации запускается вторая машина. Результат - либо бесконечная работа, либо результат после остановки машины $\mathcal{M}_2$. Чтобы получить машину $\mathcal{M}$, достаточно заменить заключительное состояние машины $\mathcal{M}_1$ начальным состоянием машины $\mathcal{M}_2$, а потом объединить множества команд.

Итерация машины Тьюринга - формализация идеи многократного повторения одного и того же алгорима. Необходимо многократно повторять машину до тех пор, пока не будет выполнено некоторое условие(предикат) p. Условие можно задать как некоторые заключительные состояния машины. Тогда все остальные состояния надо просто заменить на q_1 и получить новую машину.

15 Моделирование машин Тьюринга.

Нам хочется[1] смоделировать работу машины Тьюринга, работающей в алфавите $A = \{0, 1, a_2, \ldots, a_k\}$ на машине Тьюринга с алфавитом $\{0, 1\}$. Идея очень проста: закодируем буквы алфавита A словами в алфавите $\{0, 1\}$, а затем оперировать с этими кодами так же, как исходная машина оперирует с буквами. Кодируем словами одной длины l. Будем считать, что буква 0 кодируется l нулями, а буква 1 - l единицами. Дальше к состояниям надо добавить ещё три группы состояний, первая из которых будет расшифровывать код, помня текущее состояние машины, вторая - моделировать замену значения в ячейке, а третья - сдвигать головку на l клеток влево или вправо или стоять на месте.

16 Универсальная машина Тьюринга.

Пусть $f_0(x), f_1(x), \ldots$ - последовательность функций, заданных на $\mathbb{N}$. Функция $U(n, x)$ называется *универсальной*, если:

1. $\forall n \in \mathbb{N}\, \exists m : U(n, x) = f_m(x)$.
2. $\forall m \exists n : U(n, x) = f_m(x)$.

[1] естественно, нам хочется сдать ДГДМиК и всё. Здесь и везде дальше подобные фразы будут означать исключительно моральную подготовку читателя к уверенному ответу.

Определение 16.1. Назовём машину Тьюринга $\mathcal{U}$ *универсальной*, если он вычисляет функцию $U(n, x)$, являющуюся универсальной для последовательности функций одной переменной, вычислимых на машинах Тьюринга с ленточным алфавитом $\{0, 1\}$.

Теорема 16.2. *Универсальная машина Тьюринга существует.*

17 Операции суперпозиции, примитивной рекурсии и минимизации.

1. *Суперпозиция*: $f(x_1, \ldots, x_n) = g_0(g_1(x_1, \ldots, x_n), \ldots, g_m(x_1, \ldots, x_n))$. В случае частично определённых функций считаем, что значение определено только в случае, когда определены все значения функций g_i.

2. *Примитивная рекурсия*: пусть имеются (частично определённые) функции $g(x_1, \ldots, x_{n-1}), h(x_1, \ldots, x_{n+1})$. Будем говорить, что функция $f(x_1, \ldots, x_n)$ получена из этих функций с помощью операции примитивной рекурсии по переменной x_n, если $\forall x_1, \ldots, x_n$:

$$\begin{cases} f(x_1, \ldots, x_{n-1}, 0) = g(x_1, \ldots, x_{n-1}) \\ f(x_1, \ldots, x_{n-1}, x_n + 1) = h(x_1, \ldots, x_{n-1}, x_n, f(x_1, \ldots, x_n)). \end{cases}$$

   В случае частично определённых функций мы считаем, что значение $f(a_1, \ldots, a_n)$ определено только в случае, когда определено значение $g(a_1, \ldots, a_{n-1})$ и $\forall b < a_n$ определено $h(a_1, \ldots, a_{n-1}, b, f(a_1, \ldots, a_{n-1}, b))$.

3. *Минимизация*: пусть $g(x_1, \ldots, x_n)$ - частично определённая функция. Определим результат применения операции минимизации μ к функции g:

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

   Считаем, что значение определено и равно числу b тогда и только тогда, когда:

   - $g(a_1, \ldots, a_{n-1}, b)$ определено и равно a_n;
   - $\forall z < b$ значение $g(a_1, \ldots, a_{n-1}, z)$ определено и отлично от a_n.

   Понятно, что с помощью операции минимизации находится наименьшее по y решение уравнения

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

18 Замкнутость класса функций, вычислимых на машинах Тьюринга, относительно операций суперпозиции, примитивной рекурсии и минимизации.

Определение 18.1. Назовём функцию *частично рекурсивной*, если её можно получить из функций $0, x + 1, I$ (где I - класс функций I_i^n, равных значению своей i-й переменной) с помощью конечного числа суперпозоций, примитивных рекурсий и минимизаций.

Теорема 18.2 (Тезис Чёрча)**.** *Класс алгоритмически вычислимых функций (вычислимых на машине Тьюринга) совпадает с классом частично рекурсивных функций.*

Теорема 18.3. *Любая частично рекурсивная функция вычислима на машине Тьюринга.*

19 Класс примитивно-рекурсивных функций. Простейшие примитивно-рекурсивные функции.

Выделим в классе частично рекрсивных функций подкласс F_{np} *примитивно рекурсивных функций.* Функция называется примитивно рекурсивной, если её можно получить из всё тех же функций $0, x+1, I$, но теперь уже только операциями суперпозиции и примитивной рекурсии.

Примеры:

1. 0 - примитивно рекурсивная функция. Любая константа получается суперпозицией функции-константы 0 и функции $x+1$.
2. Примитивная рекурсия $\begin{cases} sum(x_1, 0) = x_1 \\ sum(x_1, x_2+1) = sum(x_1, x_2) + 1 \end{cases}$ определяет функцию суммы $x_1 + x_2$.
3. Примитивная рекурсия $\begin{cases} prod(x_1, 0) = 0 \\ prod(x_1, x_2+1) = prod(x_1, x_2) + x_1 \end{cases}$ определяет функцию произведения $x_1 * x_2$.
4. Примитивная рекурсия $\begin{cases} pow(x_1, 0) = 1 \\ pow(x_1, x_2+1) = pow(x_1, x_2) \cdot x_1 \end{cases}$ определяет степень $x_1^{x_2}$.

Кроме того, положим $x \dot{-} y = \begin{cases} x - y, x \geq y \\ 0, x < y \end{cases}$ Доказываем примитивную рекурсивность, сначала устанавливая её для $x \dot{-} 1$, а потом уже для $x \dot{-} y$.

Из этих функций легко получаются функции $\max, \min, sg, \overline{sg}$, которые тоже являются примитивно-рекурсивными.

20 Класс частично-рекурсивных функций. Примеры частично-рекурсивных функций.

Повторим

Определение 20.1. Назовём функцию *частично рекурсивной*, если её можно получить из функций $0, x+1, I$ (где I - класс функций I_i^n, равных значению своей i-й переменной) с помощью конечного числа суперпозоций, примитивных рекурсий и минимизаций.

Примеры:

1. $g_1(x) = x + 1 \Longrightarrow f_1(x) = (\mu y)(g_1(y) = x) = \begin{cases} x - 1, x > 0 \\ \text{не определено}, x = 0 \end{cases}$

2. $g_2(x) = a \Longrightarrow f_2(x) = (\mu y)(g_2(y) = x) = \begin{cases} 0, x = a \\ \text{не определено}, x \neq a \end{cases}$

3. $g_3(x) = f_1(x) \Longrightarrow f_3(x) = (\mu y)(f_1(y) = x)$ не определена при любом значении x.

4. $f_4(x) = (\mu y)(y^2 = x) = \begin{cases} k, x = k^2 \\ \text{не определено}, x \neq k^2 \end{cases}$

5. $f_5(x) = (\mu y)(2^y = x) = \begin{cases} \log_2 x, x = 2^k \\ \text{не определено}, x \neq 2^k \end{cases}$

21 Частичная рекурсивность вычислимых функций. Формула Клини.

Хотим показать, что любая вычислимая функция является частично рекурсивной. Это утверждение по сути является прямым следствием следующей

Теорема 21.1 (формула Клини)**.** *Любая вычислимая на машине Тьюринга функция $f(x_1, \ldots, x_n)$ может быть представлена в виде*

$$f(x_1, \ldots, x_n) = G(x_1, \ldots, x_n, (\mu y)(H_f(x_1, \ldots, x_n, y) = 0)),$$

где функции $G(x_1, \ldots, x_n, y)$ и $H_f(x_1, \ldots, x_n, y)$ примитивно рекурсивны.

После этого, учитывая, что в билете 2.6 мы сформулировали и (уже не мы, но всё же) доказали, что любая частично рекурсивная функция является вычислимой, можем считать, что тезис Чёрча о совпадении классов вычислимых и частично рекурсивных функций доказан.

22 Классы P и NP. Примеры задач из класса NP.

Определение 22.1. Пусть A, B - конечные алфавиты, $f : A^* \to B^*$, $T(n) : \mathbb{N} \to \mathbb{N}$. Будем говорить, что функция f *вычислима за время* $T(n)$, если существует машина Тьюринга, которая вычисляет функцию f и при этом $\forall \bar{a} = a_1 a_2 \ldots a_n \in A^*$ число шагов, которое машина затрачивает на получение результата $f(\bar{a})$ не превосходит $T(n)$.

Определение 22.2. Назовём функцию *полиномиально вычислимой*, если существуют машина Тьюринга и полином $p(n)$ с натуральными коэффициентами такие, что машина вычисляет функцию за время $p(n)$.

Мы рассматриваем задачи распознавания множеств, то есть вычисления их характеристических функций (должны выдать 1, если элемент лежит в множестве, и 0 иначе). Обозначим через P класс всех множеств, характеристические функции которых вычислимы за полиномиальное время.

Определение 22.3. Пусть $L_1 \subseteq A^*, L_2 \subseteq B^*$. Говорим, что L_1 *полиномиально сводится* к множеству L_2, если существует полиномиально вычислимая функция $f : A^* \to B^*$ такая, что

$$(\forall \bar{a})(\bar{a} \in L_1 \iff f(\bar{a}) \in L_2).$$

Лемма 22.4. *Если множество L_1 полиномиально сводится к $L_2 \in P$, то $L_1 \in P$.*

Введём понятие *недетерминированной машины Тьюринга.* Аналогично автоматам, назовём машину Тьюринга недетерминированной, если в её программе может присутствовать несколько команд с одинаковыми левыми частями. Соответственно, после начала работы в каждый момент времени мы можем находиться в одной из конфигураций, определяемых правыми частями, и вынуждены рассматривать все возможные конфигурации одновременно. Точно так же, как и для автоматов, назовём множеством $D(\mathcal{M})$ совокупность всех слов $\bar{a}$, для которых есть какое-то вычисление, завершающееся в заключительном состоянии. Если $\bar{a} \in D(\mathcal{M})$, то говорим, что машина допускает(распознаёт) слово. Множество $D(\mathcal{M})$ называют допускаемым.

Соответственно, будем говорить, что недетерминированная машина Тьюринга допускает множество $D(\mathcal{M})$ за время $T(n)$, если для любого допустимого слова длины n существует вычисление длины не более $T(n)$.

Обозначим через NP класс всех множеств, распознаваемых на недетерминированных машинах Тьюринга за полиномиальное время.

Примеры будем формулировать в виде задач, объекты, являющиеся их решениями, образуют искомое множество. Итак:

1. Задача ВЫПОЛНИМОСТЬ (ВЫП).

   Вход: формула в виде КНФ.

   Вопрос: является ли формула выполнимой, то есть существует ли набор, на котором формула равна 1.

2. Задача КЛИКА.

   Вход: неориентированный граф G и натуральное число k.

   Вопрос: существует ли в графе клика (полный подграф) размера k (с k вершинами).

3. Задача ГАМИЛЬТОНОВ ЦИКЛ.

   Вход: неориентированный граф G.

   Вопрос: существует ли в графе гамильтонов цикл, то есть цикл, проходящий через каждую вершину графа ровно один раз.

23 NP-полнота. Теорема Кука.

Очевидно, что имеет место вложение $P \subseteq NP$. До сих пор открытым вопросом является равенство или неравенство этих классов. Нам хочется определить несколько ключевых задач в классе NP - тех, которые содержат в себе все задачи из NP. Поэтому введём два определения.

Определение 23.1. Назовём множество L *NP-трудным*, если любое множество из класса NP полиномиально сводится к множеству L.

Определение 23.2. Назовём множество L *NP-полным*, если оно NP-трудно и $L \in NP$.

Ответ на вопрос, существуют ли NP-полные множества, даёт

Теорема 23.3 (Кука)**.** *Задача ВЫП является NP-полной.*

24 NP-полнота задачи 3-ВЫП.

Назовём КНФ, у которой в любой дизъюнкции не более трёх слагаемых, 3-КНФ. Задача 3-ВЫП - вариант задачи ВЫП, только множество рассматриваемых КНФ ограничено 3-КНФ.

Теорема 24.1. *Задача 3-ВЫП является NP-полной.*

25 Полиномиальная разрешимость задачи 2-ВЫП.

Аналогично предыдущему билету введём понятия 2-КНФ и задачи 2-ВЫП.

Теорема 25.1. *Задача 2-ВЫП является полиномиально разрешимой, то есть принадлежит классу P.*

Всей удали у него - что за ложкой потеть.

Русская пословица

3 Сложность реализации функций алгебры логики

26 Задача синтеза схем для функций (операторов) из специального класса, мощностные нижние оценки функции Шеннона для их сложности в случае невырожденного (ненулевого, квазиинвариантного) класса.

Для множества ФАЛ $Q, Q \subseteq P_2$ и для натурального n будем обозначать $Q(n) = Q \cap P_2(n)$. При этом само множество Q и связанную с ним последовательность $Q(1), Q(2), \ldots$ будем называть *классом ФАЛ*. Аналогично, для операторов $Q(1), Q(2), \ldots$, где $Q(n) \subseteq P_2^m(n), m = m_Q(n)$ называется классом операторов[2]. Будем предполагать, что все множества $Q(n)$ непусты и, как правило, $|Q(n)| \geq 3$.

Пусть задан класс ФАЛ или операторов Q, класс схем $\mathcal{U}$ и функционал сложности $\mathcal{L}$.

Определение 26.1. Функцией Шеннона для класса ФАЛ или операторов Q при их реализации в классе схем $\mathcal{U}$ относительно функционала сложности $\mathcal{L}$ называется функция натурального аргумента

$$\mathcal{L}(Q(n)) = \max_{f \in Q(n)} \mathcal{L}(f),$$

где $\mathcal{L}(f)$ - минимальная $\mathcal{L}$-сложность схем из $\mathcal{U}$, реализующих (систему) ФАЛ f.

Будем рассматривать класс $\mathcal{U}^C$ - класс СФЭ над базисом $Б_0 = \{\&, \vee, \neg\}$, а в качестве $\mathcal{L}(\Sigma)$ - функционал, равный числу элементов в СФЭ Σ. Для класса ФАЛ или операторов Q' и класса ФАЛ Q'' введём функции

$$\mathcal{J}(|Q'(n)|) = \frac{\log |Q'(n)|}{\log\log |Q'(n)|} \text{ и } \sigma_{Q''}(n) = \frac{\log |Q''(n)|}{2^n},$$

где $n = 1, 2, \ldots$. При этом последовательность $\{\sigma_{Q''}(n)\}$ будем называть *мощностной последовательностью* класса Q''.

Определение 26.2. Класс ФАЛ(операторов) называется

1. *невырожденным*, если $n + m_Q(n) = o(\mathcal{J}(|Q(n)|))$;

[2]Понятия не имею, о чём идёт речь

2. *строго невырожденным*, если $\log n = o(log|Q(n)|)$;

3. *ненулевым*, если $\varliminf_{n\to\infty} \sigma_Q(n) > 0$ (соответственно, нулевым, если верхний предел равен пределу и равен нулю).

Лемма 26.3. *Если Q - невырожденный класс ФАЛ(операторов), то $L^C(Q(n)) \gtrsim \mathcal{J}(|Q(n)|)$, а если Q - строго невырожденный класс ФАЛ, то $L^K(Q(n)) \gtrsim \mathcal{J}(|Q(n)|)$.*

Следствие. Для всякого ненулевого класса ФАЛ Q выполнено неравенство $L^C(Q(n)) \gtrsim \sigma_Q(n)\frac{2^n}{n}$.

Определение 26.4. Класс ФАЛ Q называется *квазиинвариантным*, если для некоторого $n_0 \geq 2$, для любого $n \geq n_0$ и для произвольной ФАЛ $f \in Q(n)$ при любом $\sigma \in B$ ФАЛ $f(x_1, \ldots, x_{n-1}, \sigma) \in Q(n-1)$. Минимальное такое n_0 называется *порогом* квазиинвариантного класса Q.

Лемма 26.5. *Пусть Q - квазиинваринатный класс ФАЛ. Тогда существует предел*

$$\sigma_Q = \lim_{n\to\infty} \sigma_Q(n) = \lim_{n\to\infty} \frac{\log |Q(n)|}{2^n},$$

где $0 \leq \sigma_Q \leq 1$. Этот предел называется мощностной характеристикой класса Q.

Следствие. Квазиинвариантный класс является нулевым(ненулевым) тогда и только тогда, когда мощностная характеристика равна нулю (больше нуля).

Лемма 26.6. *Для ненулевого квазиинвариантного класса ФАЛ Q верно $L^C(Q(n)) \gtrsim \sigma_Q \cdot \frac{2^n}{n}$.*

27 Инвариантные классы функций С.В. Яблонского, их описание на языке базовых множеств и порождающих элементов. Теорема о числе инвариантных классов (фрагменты доказательства).

Рассмотрим операции над ФАЛ:

1. добавление и изъятие фиктивных БП.

2. переименование БП без отождествления.

3. подстановка констант $0, 1$ вместо БП.

Если функция g получена из функции f применением операции 3 (1-3), то говорят, что g - *подфункция*(*псевдоподфункция*) f. Сама же f, соответственно, *надфункция*(*псевдонадфункция*). Для множества функций F обозначим через $F^{\urcorner}$ и $F_{\lrcorner}$ множество псевдонадфункций и псевдоподфункций для F соответственно.

Множество ФАЛ $Q \subseteq P_2$ называется *инвариантным классом* ФАЛ, если оно замкнуто относительно операций (1-3). Множества $\{0\}, \{1\}, \{0,1\}$ называются *тривиальными инвариантными классами*. Инвариантный класс является квазиинвариантным классом с порогом 2, а значит $\sigma_Q(n)$ - монотонно не возрастающая последовательность.

Можно показать, что единственный инвариантный класс с характеристикой $\sigma_Q = 1$ - это класс P_2.

Лемма 27.1. *Существует континуум различных инвариантных классов с характеристикой* 0.

Более того, верна

Теорема 27.2. *Существует континуум инвариантных классов с характеристикой* $0 \leq \sigma < 1$.

Определение 27.3. Множество F называется *базовым множеством* инвариантного класса Q, если $F_\lrcorner = Q$. Базовое множество называется *базой*, если любое его собственное подмножество уже не является базовым.

Для всякого инвариантного класса достаточно задать его базовое множество.

Существует и другой способ задания инвариантного класса.

Определение 27.4. Пусть Q - нетривиальный отличный от P_2 инвариантный класс. Функция $g \in P_2$ называется *порождающим элементом* класса Q тогда и только тогда, когда $g \notin Q$, а любая собственная подфункция (подфункция g, не совпадающая с самой g) лежит в Q. *Порождающим множеством* называется всякое максимальное по включению множество попарно не конгруэнтных порождающих элементов Q.

Лемма 27.5. *Пусть Q - нетривиальный отличный от P_2 инвариантный класс, а G - его порождающее множество. Тогда $Q = P_2 \setminus (G^\urcorner)$.*

28 Синтез схем на основе модификации асимптотически наилучшего метода. Стандартные классы и стандартность класса функций равных нулю на всех наборах значений переменных, номера которых больше заданного числа.

Определение 28.1. Класс ФАЛ(операторов) Q называется *стандартным*, если

$$L^C(Q(n)) \lesssim \mathcal{J}(|Q(n)|) + O(n + m(n)).$$

Рассмотрим множество $P_2(n,t)$, включающее в себя все ФАЛ из $P_2(n)$, обращающиеся в 0 на наборах с номерами $t, t+1, \ldots, 2^n - 1$. Его мощность, очевидно, равна 2^t.

Лемма 28.2. *Для любой функции $r = r(n) \geq 1$ соответствующий класс $Q(n) = P_2(n, r(n))$ является стандартным относительно функционала сложности L схем класса U^C, то есть $L^C(Q(n)) \lesssim \frac{r}{\log r} + O(n)$.*

Следствие. Если $n = o(\frac{r}{\log r})$, то $Q(n) = P_2(n, r(n))$ - стандартный невырожденный класс ФАЛ, для которого асимптотически

$$L^C(Q(n)) \sim \frac{r}{\log r}.$$

29 Асимптотически наилучший метод синтеза схем для ненулевых квазиинвариантных классов, их стандартность.

Лемма 29.1. *Для всякого квазиинвариантного класса Q и $n = 1, 2, \ldots$*

$$\begin{cases} L^C(Q(n)) \sim \sigma_Q \frac{2^n}{n}, \textit{при } \sigma_Q > 0 \\ \\ L^C(Q(n)) = o(\frac{2^n}{n}), \textit{при } \sigma_Q = 0 \end{cases}$$

30 Общее описание принципа локального кодирования, его применение для доказательства стандартности класса самодвойственных функций.

Принцип локального кодирования: пусть Q - класс операторов и пусть для каждого натурального n определено кодирование $\Pi = \Pi_n : F \to \pi(F), F \in Q(n), |\pi(F)| = d(n)$. В каждом коде выделим куски π_i, составленные из подряд идущих разрядов кода π и длины не более, чем $\lambda = \lambda(n)$. Пусть для кодирования выполнено свойство локальности, то есть для вычисления F на произвольном наборе σ достаточно знать лишь один кусок кода $\pi_{i(\sigma)}$, задаваемый какими-то координатами (например позицией его первого разряда и длиной).

Пусть оператор кодирования $A^{(1)} = A^{(1)}_n$ по σ вычисляет $\pi_{i(\sigma)}$, а оператор декодирования $A^{(2)} = A^{(2)}_n$ по куску $\pi_{i(\sigma)}$ и, возможно, набору σ вычисляет $F(\sigma)$. Искомая схема, реализующая F и построенная на основе локального кодирования, состоит из подсхем $A^{(1)}, A^{(2)}$ и основного блока $\mathcal{O} = \mathcal{O}_n$, который по координатам куска $\pi_{i(\sigma)}$ выдаёт сам этот кусок.

Если выполнены следующие соотношения:

$$L^C(A^{(1)}_n) = o(\mathcal{J}(|Q(n)|)), L^C(A^{(2)}_n) = o(\mathcal{J}(|Q(n)|)),$$

$$L^C(O_n) \lesssim \mathcal{J}(|Q(n)|),$$

то искомая СФЭ Σ_n может быть выбрана так, что $L(\Sigma_n) \lesssim \mathcal{J}(|Q(n)|)$. Значит, в силу леммы 1.1 в случае невырожденности класса Q выполняется $L^C(Q(n)) \sim \mathcal{J}(|Q(n)|)$, которое означает стандартность класса Q.

Лемма 30.1. *Класс самодвойственных ФАЛ является невырожденным ФАЛ, и, тем самым,*

$$L^C(\mathcal{S}(n)) \sim \frac{2^{n-1}}{n}.$$

31 Применение принципа локального кодирования для доказательства стандартности невырожденных классов симметрических операторов и операторов, связанных с вычислением функции на нескольких последовательных наборах.

Под (n,m)-оператором будем понимать систему ФАЛ $F = (f_1, \ldots, f_m) \in P_2^m(n)$. Обозначим через S класс симметрических ФАЛ.

Лемма 31.1. *Если для последовательности $m = m(n) \in \mathbb{N}$ верно $\log n = o(m), \log m = o(\log n)$, то класс операторов Q, для которого $Q(n) = S^m(n)$ является невырожденным и стандартным.*

Лемма 31.2. *Для постоянной последовательности $m = m(n) \geq 2$ класс операторов Q, для которого $Q(n)$ состоит из всех (n,m)-операторов $F = (f_1, \ldots, f_m)$ таких, что $f_i(\beta) = f_1(\alpha)$, $\forall i \in [2,m], \alpha \in B^n, \nu(\beta) - \nu(\alpha) \equiv i-1 (\mod 2^n)$, является стандартным*[3].

32 Задача синтеза схем для не всюду определённых функций. Особенности получения нижней мощностной оценки соответствующей функции Шеннона, формулировка теоремы о её асимптотическом поведении.

Отображение $f : B^n \to [0,2]$ будем называть *не всюду определённой* ФАЛ от n БП, а множество $\delta(f) = f^{-1}(0,1)$ будем считать её *областью определённости. Доопределением* считается любая ФАЛ из $P_2(n)$, совпадающая с f на $\delta(f)$, а под сложностью $L^C(f)$ реализации будем понимать наименьшую из сложностей доопределений.

Обозначим $\widehat{P}_2(n)$ множество не всюду определённых ФАЛ от БП $X(n) = (x_1, \ldots, x_n)$. Для любого $t \in [0, 2^n]$ введём его подкласс $\widehat{P}_2(n,t)$, состоящий из тех $f \in \widehat{P}_2(n)$, для которых $|\delta(f)| = t$.

Функция Шеннона определяется стандартно:

$$L^C(\widehat{P}_2(n,t)) = \max_{f \in \widehat{P}_2(n,t)} L^C(f).$$

Лемма 32.1. *Если $n \log n = o(t(n))$, то*

$$L^C(\widehat{P}_2(n, t(n))) \gtrsim \frac{t(n)}{\log t(n)}.$$

[3] ν - соответствующая десятичная запись двоичного числа

Теорема 32.2. *Если* $n \log^2 n = o(t)$, *то* $L^C(\widehat{P}_2(n,t)) \sim \frac{t}{\log t}$.

33 Асимптотически наилучший метод синтеза схем для не всюду определённых функций в случае их «сильной» определённости.

Лемма 33.1. *Если* $t = t(n) \sim n$, *то*

$$L^C(\hat{P}_2(n,t)) \lesssim \frac{t(n)}{\log t(n)}.$$

34 Лемма о линейном разделяющем операторе. Асимптотически наилучший метод синтеза схем для не всюду определённых функций в случае их «средней» и «слабой» определённости.

Пусть $n, s \in \mathbb{N}, s \leq n, A$ - произвольное множество наборов куба B^n и $|A| \leq 2^s$. Будем говорить, что (n,s)-оператор $\psi \in P_2^s(n)$ является *оператором разделения*(*оператором хэширования*) для A, если $\psi(\alpha) \neq \psi(\beta)$ для любых различных наборов $\alpha, \beta \in A$. Обозначим через Λ класс линейных ФАЛ с нулевым свободным членом и будем выбирать нужные нам операторы разделения из $\Lambda^s(n)$.

Лемма 34.1. $\forall A \subseteq B^n, s \leq n\ \exists \psi \in \Lambda^s(n)$, *разделяющий некоторое множество* $A' \subseteq A$ *такое, что* $|A'| \geq t - \frac{t(t-1)}{2^{s+1}}, t = |A|$.

Лемма 34.2. $2^{\frac{n}{3}} \leq t \leq \frac{2^n}{n^5} \Longrightarrow L^C(\hat{P}_2(n,t)) \lesssim \frac{t}{\log t}$.

Лемма 34.3. $t \leq 2^{\frac{n}{2}}, n \log^2 n = o(t) \Longrightarrow L^C(\hat{P}_2(n,t)) \lesssim \frac{t}{\log t}$.

35 Лемма о цепях и сечениях π-схем. Верхние оценки сложности реализации линейных функций в классе π-схем.

Будем понимать под контактной схемой $(1,1)$-КС из неориентированных контактов.

Результат последовательного(параллельного) соединения КС Σ_1, Σ_2 будем обозначать $\Sigma_1 \cdot \Sigma_2 (\Sigma_1 \parallel \Sigma_2)$. Определим *простейшую* π*-схему* как КС из одного контакта, а затем индуктивно определим π*-схему* как последовательное или параллельное соединение π-схем.

Для множества C из контактов $x_{j_1}^{\sigma_1}, \ldots, x_{j_t}^{\sigma_t}$ положим

$$K(C) = x_{j_1}^{\sigma_1} \cdot \ldots \cdot x_{j_t}^{\sigma_t}$$

$$J(C) = x_{j_1}^{\overline{\sigma_1}} \vee \ldots \vee x_{j_t}^{\overline{\sigma_t}}.$$

[4] Полюсом КС назовём контакт, либо в который всё входит, либо из которого всё выходит. Простой цепью будем считать цепь C в контактной схеме, соединяющую полюса, а проводимостью - свойство цепи, при котором $K(C) \neq 0$.

Сечение S, разделяющее две вершины КС, - множество контактов, без которых не будет пути между этими двумя вершинами. Тупиковым будем считать сечение, из которого нельзя выбросить контакт так, чтобы оно осталось сечением, а свойство отделимости соответствует $J(S) \neq 1$.

Обозначим $\mathcal{C}(\Sigma)$ множество проводящих простых цепей, соединяющих полюса КС, а $\mathcal{S}(\Sigma)$ - множество отделимых тупиковых сечений, разделяющих полюса

Лемма 35.1. *Для π-схемы Σ любая цепь $C \in \mathcal{C}(\Sigma)$ и любое сечение $S \in \mathcal{S}(\Sigma)$ имеют ровно один общий контакт.*

Лемма 35.2. *При $n \geq 1$ для линейной ФАЛ $l_n^\sigma, \sigma \in B$ верно:*

$$L^\pi(l_n^\sigma) \leq 4n^2.$$

36 Теорема Храпченко, нижние оценки сложности линейной функции в классе π-схем

Пусть $\mathcal{N}', \mathcal{N}'' \subseteq B^n, \mathcal{N}' \cap \mathcal{N}'' = \varnothing$. Обозначим через $\mathcal{R}(\mathcal{N}', \mathcal{N}'')$ множество всех пар (α, β), где наборы α, β из куба B^n, соседние по некоторой БП и $\alpha \in \mathcal{N}', \beta \in \mathcal{N}''$.

Теорема 36.1 (Храпченко)**.** *Для любой ФАЛ $f \in P_2(n)$, любых $\mathcal{N}' \subseteq N_f, \mathcal{N}'' \subseteq \overline{N}_f$ (N_f - множество наборов, на которых функция равна 1) справедливо неравенство:*

$$L^\pi(f) \geq \frac{|\mathcal{R}(\mathcal{N}', \mathcal{N}'')|^2}{|\mathcal{N}'| \cdot |\mathcal{N}''|}.$$

Следствие. При $n \geq 1$ для линейной ФАЛ $l_n^\sigma, \sigma \in B$ выполнены неравенства

$$n^2 \leq L^\pi(l_n^\sigma) \leq 4n^2.$$

37 Схемная и алгоритмическая сложность функций, гипотеза С.В. Яблонского и теорема Дж. Сэвиджа.

Определение 37.1. Функцию $f_n(x_1, \ldots, x_n)$ назовём *сложной*, если $L(f_n) = L(n)$. Бесконечную последовательность булевых функций $f_1(x_1), f_2(x_1, x_2), \ldots$ назовём *сложной*, если $\forall N\ \exists n \geq N : f_n(x_1, \ldots, x_n)$ является сложной.

[4] В лекциях Ложкина сигмы в $J(C)$ входят без отрицаний. В его же лекциях, но по ОКам, во второй части, есть то же самое определение, но с отрицаниями. Что выбрать - решать вам.

Определение 37.2. Алгоритм, строящий бесконечную последовательность булевых функций $\{f_i(x_1, \ldots, x_i)\}_{i=1}^{\infty}$ называется *правильным*, если он строит все функции миинимального инвариантного класса, содержащего эту последовательность.

Теорема 37.3 (Яблонский)**.** *Любой правильный алгоритм, строящий сложную последовательность функций из P_2, строит всё множество P_2.*

Теорема 37.4 (Сэвидж)**.** *Пусть машина Тьюринга работает на словах длины n не более $T(n)$ тактов. Тогда её можно моделировать СФЭ сложности $O(T^2(n))$.*

Как она ни пыталась, она не могла найти тут ни тени смысла, хотя все слова были ей совершенно понятны.

Л. Кэрролл "Алиса в стране чудес"

4 Заметки об экзамене

Я сдавал Марченкову. Милейший дедуля, корректный, добрый и приятно спрашивающий. По его собственным словам знания определений гарантируют оценку 4, так что всем советую идти к нему. Любимый вопрос - операция минимизации. Оценки в ложкинской, третьей, части обычно спрашивает либо сам Ложкин, либо ещё один-два препода, среди которых, например, Нагорный. Остальные ограничиваются формулировками именных теорем, леммы о цепях и сечениях[5] и, возможно, какими-нибудь оценками сложности для определённых классов функций (например, у меня Марченков спросил, какую нижнюю оценку даёт теорема Храпченко для линейных функций). Ну а в остальном всё обычно, упор делаем на именные теоремы и ключевые определения, такие как определение детерминированной функции.

А кроме того, группа не поленилась и скинула вопросы, которые им задавали, так что вот:

1. Подтверждение 5 Нагорному
    - Зависимость с запаздыванием
    - Теорема Клини
    - Является ли $\{1^n01^n\}$ конечно-автоматным множеством
    - Примитивная рекурсивность функции, равной количеству натуральных делителей
    - Теорема кого-то об оценке нелулевого квазиинвариантного класса.
2. Подтверждение 4 Нагорному
    - Теорема Храпченко
    - NP-полнота
    - Теорема Клини + все определения к ней
    - Построить автомат, который допускает $\{0, 010, 1^n0\}$.

[5]Кстати, что касается сноски на странице 27. У меня Марченков спросил, что такое $K(C)$ и $J(C)$, в $J(C)$ я написал с отрицаниями, он принял.

3. Повышение с 3 до 4 у аспиранта, который не Вася (с вытянутым лицом)
    - Что такое инвариантный класс, какие из известных классов относятся к инвариантным, объяснить почему класс T_0 неинвариантный.
4. Подтверждение 4 всё тому же Нагорному
    - Найти оценку с помощью Храпченко для $S^{\{2,3\}}(5)$
    - Доказать что функция $\max - \min$ частично рекурсивная.
5. Повышение с 3 до 4 у аспирантки
    - Теорема Клини, конечно-автоматные и регулярные множества.
    - Задача ВЫП и теорема Кука, 3-ВЫП и теорему про 2-ВЫП (о полиномиальной разрешимости).
    - Полиномиальная сводимость множеств, NP-полнота.
    - Определения из 26-го билета (которые про нижние оценки для невырожденных классов).
    - Определения из билета про МТ, выполнимые функции и полную систему в классе КА-функций.
6. Марченков, подтверждение 5
    - Квазиинвариантные функции.
    - Операция минимизации.
    - Доказать, что для $\{0^n1^n\}$ не существует правоинвариантного отношения эквивалентности.
    - Является ли функция, определенная только в 15 рандомных точках, частично-рекурсивной?
7. Подтверждение 4 у Васи
    - Определения из первых двух частей.
    - Теорема Клини.
    - Теорема Храпченко.
8. Всё то же подтверждение 4 у всё того же Васи
    - Машина Тьюринга.
    - Частично-рекурсивные, примитивно-рекурсивные функции.
    - Привести пример частично-рекурсивной функции, которая не является примитивно-рекурсивной.
    - Правоинвариантное отношение эквивалентности и его связь с конечно-автоматными множествами.

- Гипотеза Яблонского.
- Теорема Кука.
- Задача синтеза.

9. Подтверждение 4 у аспирантки
    - Правоинвариантная эквивалентность.
    - Универсальная машина Тьюринга.
    - NP полнота и теорема Кука.
    - Лемма о цепях и сечениях.
    - Теорема Храпченко.
    - Детерминизация недетерминированного автомата.
    - Если есть две задачи из P, то верно ли что они P-сводятся друг к другу?
10. Подтверждение 5 у Марченкова
    - Теорема Сэвиджа.
    - Регулярное выражение и регулярное множество. Является ли регулярным множество $\{0^n 1^n 2^n, n \in \mathbb{N}\}$.
    - Резольвента.