МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение
высшего образования

«МИРЭА – Российский технологический университет»

РТУ МИРЭА

ЛЕКЦИОННЫЕ МАТЕРИАЛЫ

Теория алгоритмов и рекурсивных функций

Уровень	бакалавриат
Форма обучения	Очная
Направление(-я) подготовки	09.03.03 «Прикладная информатика»
Институт	информационных технологий (ИТ)
Кафедра	Прикладная математика (ПМ)
Лектор	докт. физ.-мат. наук, проф. Людковский Сергей Викторович

Используются в данной редакции с учебного года 2020/21

(учебный год цифрами)

Проверено и согласовано «____» _____20___г. _____ _____

(подпись директора Института/Филиала с расшифровкой)

Москва 2021 г.

Содержание

1. Лекция 1. Эффективная вычислимость. Основные понятия теории алгоритмов.

На практике часто встречаются вычислительные процедуры. Для их выполнения служат алгоритмы. Можно определить алгоритм как систематизированный прием счета для той или иной задачи. То есть алгоритм точно описывает вычислительный процесс, иными словами алгоритмический процесс. Он начинается с исходных данных и заканчивается получением результата. Для выполнения алгоритмического процесса в наше время чаще всего используют компьютеры. Вычислительная машина осуществляет алгоритм с помощью компьютерной программы, которая пишется на понятном машине языке программирования и сообщает машине порядок определенных действий, предписываемых алгоритмом. Однако, исполнителем вычислений по алгоритму может выступать и человек, если расчеты не слишком громоздки или утомительны.

В качестве примеров алгоритмов можно упомянуть алгоритм Евклида нахождения наибольшего общего делителя двух целых чисел; алгоритм разложения произвольного натурального числа на простые множители; алгоритм извлечения квадратного корня из натурального числа; алгоритм Гаусса для решения системы линейных алгебраических уравнений; существуют многочисленные алгоритмы для нахождения собственных значений и собственных векторов квадратных матриц; имеются многообразные алгоритмы для приближенного вычисления определенных интегралов с заданной точностью и так далее.

Однако встречаются проблемы в математике требующие построения их алгоритмических решений, которые не всегда выполнимы или приводят к необходимости глубоких исследований для их создания. Конечно, не всякая математическая проблема имеет какое-либо алгоритмическое решение. Как доказал советский математик Ю.В. Матиясевич в 1970 году, не существует алгоритма распознавания имеются ли решения в целых числах произвольно заданного диофантова уравнения.

Тем самым он решил десятую проблему Гильберта. Это была одна из проблем, сформулированных Д. Гильбертом в 1900 году на Международном математическом конгрессе в Париже.

Раздел математики, посвященный созданию алгоритмов и доказательству их существования или отсутствия, называется теорией алгоритмов.

Всем алгоритмам, присущи дискретность, элементарность шагов, детерминированность, направленность и массовость их применения.

Любой алгоритм оперирует с соответствующим ему множеством возможных исходных данных.

Истоки развития теории алгоритмов начинаются с доказательства известным математиком К. Гёделем теорем математической логики и теории алгоритмов о неполноте формальных систем. Примером одной из них является арифметика. Первая теорема в этой области математики была доказана в 1931 году. В ту пору на основе этих теорем возникло предположение, что многие математические проблемы невозможно разрешить алгоритмическим образом. К ним относятся проблемы выводимости в исчислении предикатов. Впоследствии это привело к необходимости стандартизации понятия алгоритма и более глубокого его изучения. Первые стандартизованные варианты этого понятия были разработаны в 30-х годах XX века в работах А. Тьюринга, А. Чёрча и Э. Поста. Предложенные ими машина Тьюринга, машина Поста и лямбда-исчисление Чёрча оказались эквивалентными друг другу. Опираясь на основополагающие работы Гёделя, другой известный математик С. Клини ввёл понятие рекурсивной функции. Впоследствии было выяснено, что оно оказалось эквивалентным понятиям упомянутым выше.

Одним из наиболее удачных стандартизованных вариантов алгоритма является введённое А. А. Марковым понятие нормального алгоритма. Оно было разработано десятью годами позже работ Тьюринга, Поста, Чёрча и Клини в связи с доказательством алгоритмической неразрешимости ряда алгебраических проблем.

Каждую задачу из бесконечного множества задач можно закодировать некоторым словом некоторого алфавита, а решение задачи - каким-то другим словом того же алфавита. В результате получим функцию, заданную на некотором подмножестве множества всех слов того же алфавита. Решить какую-либо задачу – значит найти значение этой функции на слове, кодирующем данную задачу. Иметь алгоритм решения всех задач данного класса – значит, иметь единый способ, позволяющий за конечное число шагов вычислять значения построенной функции для любых значений аргумента из ее области определения. Таким образом, алгоритмическая проблема – это проблема о вычислении значений функции, заданной в некотором алфавите.

Каждая функция, для вычисления которой существует какой-нибудь алгоритм, оказывалась вычислимой посредством некоторой машины Тьюринга. Это

дало повод Тьюрингу высказать гипотезу. Эта гипотеза была названа основной гипотезой теории алгоритмов. Иначе её называют тезисом Тьюринга.

Она состоит в следующем. Для нахождения значений функции, заданной в некотором алфавите, тогда и только тогда существует какой-нибудь алгоритм, когда функция является вычислимой по Тьюрингу. Иными словами, когда эта функция может быть вычислена на соответствующей машине Тьюринга.

Данный тезис является аксиомой. Этот тезис не может доказан методами математики. Это вызвано тем, что он не имеет внутреннего математического характера. В самом деле, одной из сторон тезиса является понятие алгоритма. Однако, это понятие не является точным математическим понятием. Следует отметить, что рассматриваемый тезис может быть опровергнут, если будет найдена функция, которая вычислима с помощью какого-нибудь алгоритма, но не вычислима на машине Тьюринга.

Теория нормальных алгоритмов была разработана русским математиком А.А. Марковым в конце 1940-х – начале 1950-х гг. XX в. Эти алгоритмы представляют некоторые правила по переработке слов в каком-либо алфавите, так что исходные данные и результаты являются словами в некотором алфавите.

Для определения нормального алгоритма Маркова вводится произвольный алфавит - конечное непустое множество символов, при помощи которых описывается алгоритм и данные. В алфавит также включается пустой символ. Под словом понимается любая последовательность непустых символов алфавита либо пустой символ, который обозначает пустое слово. Всякий нормальный алгоритм Маркова определяется конечным упорядоченным множеством пар слов алфавита, называемых подстановками. В паре слов подстановки левое (первое) слово непустое, а правое (второе) слово может быть пустым символом. Для наглядности левое и правое слово разделяются стрелкой.

В качестве исходных данных алгоритма берется какая-либо непустая строка символов. Работа нормального алгоритма Маркова заключается в последовательности весьма однотипных шагов. Каждый шаг работы алгоритма можно более подробно описать следующим образом.

В упорядоченной последовательности подстановок ищем самую первую подстановку, левое слово которой входит в строку данных.

В строке данных ищем самое первое (левое) вхождение левого слова найденной подстановки.

Это вхождение в строке данных заменяем на правое слово найденной подстановки (преобразование данных).

Данный шаг работы алгоритма повторяется до тех пор, пока либо не возникнет ситуация, когда шаг не сможет быть выполнен из-за того, что ни одна подстановка не подходит (левое слово любой подстановки уже не входит в строку данных) - правило остановки; либо не будет установлено, что процесс подстановок не может остановиться.

В первом рассмотренном выше случае строка данных, получившаяся при остановке алгоритма, является выходной (результатом) и алгоритм применим к входным данным, а во втором случае алгоритм не применим к входным данным.

Таким образом, всякий нормальный алгоритм Маркова определяет функцию, которую мы назовем нормальной (или вычислимой по Маркову), которая может быть частичной и которая в области определения входному слову ставит в соответствие выходное слово.

При рассмотрении алгоритмов в качестве исходных данных фигурируют конструктивные объекты, которые строятся из исходных неделимых элементов по предписанным им правилам.

Примеры конструктивных объектов

1. Имеется конечный алфавит Σ. Исходя из него задается множество Σ^* всех слов в алфавите Σ. Слово строится в видео конечной последовательности букв. В множеств0 Σ^* обычно выключают также пустое слово ∧.
2. Множество неотрицательных целых чисел $N_0 = \{0,1,2 \ldots\}$ можно записать как конструктивный объект, исходя из однобуквенного алфавита $\{1\}$. Тогда 0 – это пустое слово, 1 – это слово «1», 2 – слово «11» и т.д. Такой вид записи называется унарным.
3. Если использовать алфавит {1, –, 1}, то каждое рациональное число представимо словом, составленным из этих букв. Слово имеет вид X / Y, где X – целое число, а Y – натуральное число, отличное от 0 и 1, где числа X и Y являются взаимно простыми.

В программировании используют термин тип данных вместо типа конструктивных объектов.

Типы конструктивных объектов подразделяются на конечные и бесконечные. Естественно, что за конечное число шагов строится конечное число объек-

тов данного типа. Бесконечность типа понимается в том смысле, что последующие шаги приводят к новым объектам по сравнению с предыдущими. Например, типы конструктивных объектов Σ^*, $\mathbb{N}$, $\mathbb{Z}$, $\mathbb{Q}$ бесконечны.

2. Лекция 2. Нормальные алгоритмы Маркова.

Непустое конечное множество символов для удобства назовём алфавитом, а символы алфавита называются буквами.

Достаточно рассматривать только конечные алфавиты. Если имеется бесконечный алфавит, то каждую его букву можно записать в алфавите $\{p, q\}$, содержащим лишь две буквы. Для этой цели достаточно условиться букву a_n изображать в виде конечной последовательности $p\ q\ \ldots\ q\ p$, в которой q стоят подряд и фигурируют n раз. Можно полагать, что буквы всех алфавитов черпаются из одной и той же счётной последовательности $S_0, S_1, \ldots$.

При этом словом в алфавите *A* называют любую конечную последовательность букв алфавита *A*. Пустая последовательность букв называется *пустым словом.* Обозначим его «Λ».

Для двух слов $P = S_{k_1}, \ldots, S_{k_n}$ и $Q = S_{i_1}, \ldots, S_{i_m}$ через PQ обозначим их соединение $PQ = S_{k_1}, \ldots, S_{k_n}\ S_{i_1}, \ldots, S_{i_m}$. Тогда «$P \wedge = \wedge P = P$» и «$(P_1P_2)P_3 = P_1(P_2P_3)$» для слов P, P_1, P_2, P_3.

Если алфавит B содержится в алфавите A, то пишем $B \subseteq A$. При этом A называется расширением алфавита B.

Функция $f(x_1, \ldots, x_n)$ называется эффективно вычислительной, если существует некоторая механическая процедура, согласно которой можно вычислить её значения $f(k_1, \ldots, k_n)$ для произвольных значений $k_1, \ldots, k_n$ аргументов.

Если задана эффективно вычислимая функция с областью определения, содержащейся в множестве всех слов алфавита A, а значениями функции тоже являются слова алфавита A, то говорят, что задан алгоритм в алфавите A. Если

слово *P* алфавита *A* содержится в области определения алгоритма *U,* то условимся, что алгоритм *U* применим к слову *P*. Для расширения *A* алфавита *B* некоторый алгоритм в алфавите *A* называется алгоритмом над алфавитом *B.*

А.А. Марков исследовал алгоритмы, которые строятся из описанных ниже элементарных операций. Рассмотрим подстановку слова Q вместо слова P в алфавите A. Тогда выражения $P \to Q$ и $P \to * Q$ называются формулами подстановки в алфавите A. Считается, что стрелка → и точка * не есть буквы алфавита A. В частности, слова P и Q могут быть пустыми. Формула подстановки $P \to Q$ носит название простой, а формула подстановки $P \to * Q$ – заключительная подстановка. Обозначим $P \to (*) Q$ одну из формул подстановки $P \to Q$ или $P \to * Q$. Если задан некоторый конечный список формул подстановки в алфавите A $\begin{cases} P_1 \to (*) Q_1 \\ \dots \\ P_m \to (*) Q_m \end{cases}$, то он называется схемой алгоритма. Говорят, что слово B входит в слово C, если имеются слова D и E такие, что C=DBE. При этом D и E могут быть и пустыми словами. Если имеется некоторое слово B в алфавите A, то может оказаться

1. Ни одно из слов $P_1, \dots, P_r$ не входит в B
2. Имеются слова $P_{j1}, \dots, P_{jk}$ входящие в B

В первом случае тогда пишут $U: P \sqsupset$. Возьмём наименьшее целое число k такое, что P_k входит в B, положим слово S равным слову, полученному в результате замены самого левого вхождения P_k в B на слово Q_k. Такое соотношение между словами B и S обозначим $U: B \vdash S$. Если при этом подстановка $P_k \to (\cdot) Q_k$ – простая, то говорят, что алгоритм U просто переводит слово B в слово S. Для заключительной постановки $P_k \to (\cdot) Q_k$ используют тогда обозначения $U: B \vdash \cdot S$, и говорят, что алгоритм U заключительно переводит слово B в слово S. Предположим, что имеется последовательность слов $S_0, \dots, S_m$ в алфавите A, для которой $B = S_0, S = S_m$, $U: S_j \vdash S_{j+1}$ для любого $j = 0,1, \dots, k-2$, либо $U: S_{m-1} \vdash S_m$, либо $U: S_{m-1} \vdash \cdot S_m$, тогда пишут $U: B \mid = S$ или $U: B \mid = \cdot S$ соответственно. Предположим, что U(B) = S в том и только том случае, когда

либо $U{:}\,B\mid=\cdot S$, либо $U{:}\,B\mid=S$ и $U{:}\,S\sqsupset$. Такой алгоритм U называется нормальным алгоритмом или алгоритмом Маркова в алфавите A. Опишем детальнее работу алгоритма U. Для произвольного слова B в алфавите A находят первую формулу подстановки $S_m{-}{\to}\,(.)Q_m$, где S_m входит в B. Потом слово Q_m подставляют вместо самого левого вхождения слова S_m в слово B.

Если $S_m{-}{\to}\,(.)Q_m$ заключительная формула, то работа алгоритма заканчивается с некоторым результатом R_1. Если формула подстановки оказалась простой, то к промежуточному результату применяют аналогичный поиск и т.д.

Пусть V_φ обозначает алгоритм соответствующий M и ϕ таким образом, что $V_\varphi(\overline{(k_1\ldots k_n)})=\varphi\overline{(k_1\ldots k_n)}$, когда хотя бы одна из частей равенства определена. Причем, полагаем, что алгоритм V_φ неприменим к словам не имеющим типа $\overline{(k_1\ldots k_n)}$. Говорят, что частичная функция ϕ вычислима по Маркову, если имеется алгоритм U над M, который вполне эквивалентен V_φ относительно M.

Нормальный алгоритм Маркова называется замкнутым, если в его схеме содержится формула подстановки $\Lambda\to*\,Q$ типа. Для такого алгоритма возможен лишь заключительный обрыв.

Для произвольного алгоритма U через U^*обозначим алгоритм полученный из U добавлением к его схеме в конце формулы подстановки $\Lambda\to*\,\Lambda*$. Можно усмотреть, что алгоритм U^* замкнут и вполне эквивалентен алгоритму U относительно алфавита, выбранного для алгоритма U.

Возьмем два нормальных алгоритма U и V в алфавите A. Любой букве $a\in A$ сопоставим новую букву $\bar{a}$, которую назовем двойником буквы a. Алфавит, состоящий из двойников букв, обозначим $\bar{A}$. Возьмем еще две произвольные буквы α и β, которые не принадлежат $A\cup\bar{A}$. Схему нормального алгоритма U^*преобразуем в схему G_U путем замены точки в каждой заключительной формуле подстановки буквой α. Введем также схему G_V, которая образуется заменой

в схеме алгоритма V^* всех букв алфавита A их двойниками, а всех точек – буквами β. Тогда формулы подстановки вида $\Lambda \to Q$ и $\Lambda \to* Q$ заменяются на $\alpha \to \alpha Q$ и $\beta \to \alpha\beta Q$ соответственно. В итоге получается схем.

$$\begin{cases} a\alpha = \alpha a \quad (a \in A) \\ \alpha a \to a\overline{a} \quad (a \in A) \\ \overline{a}\, b \to \overline{a}\,\overline{b} \quad (a, b \in A) \\ \overline{a}\,\beta \to \beta\overline{a} \quad (a \in A) \\ \beta\overline{a} \to \beta a \quad (a \in A) \\ \alpha\overline{b} \to \overline{a}\, b \quad (a, b \in A) \\ \alpha\beta \to* A \\ G_V \\ G_U \end{cases}$$

Нормальный алгоритм C, обладающий этой схемой, каждому слову B в A сопоставляет $C(B) = V\big(U(B)\big)$.

Описанный алгоритм называется композицией алгоритмов U и V. Он обозначается $V \circ U$.Для алгоритмов U_1, ..., U_n последовательная композиция обозначается:

$$U_n \circ (\ldots \circ (U_3 \circ (U_2 \circ U_1))\ldots)$$

Пусть теперь D - некоторый нормальный алгоритм в A, а E - расширение A.

К схеме алгоритма D добавим подстановки вида $b \to b$, где $b \in E \setminus A$. Полученная схема индуцирует нормальный алгоритм. Обозначим его D_E. Он не применим к словам, содержащим буквы из $E \setminus A$. Для всякого слова C в A имеем $D_E(C) \approx D(C)$. Относительно алфавита A алгоритмы D_E и D эквивалентны. Алгоритм D_E называется формальным распространением алгоритма D на алфавит E.

Для двух алгоритмов U_1 и U_2 в алфавитах A_1 и A_2 соответственно возьмём объединение $A = A_1 \cup A_2$ алфавитов и формальные распространения $U_{1,A}$ и $U_{2,A}$ алгоритмов U_1 и U_2 на алфавит A. Композицию алгоритмов $U_{1,A}$ и $U_{2,A}$ называют нормальной композицией алгоритмов U_1 и U_2 и обозначают $U_2 \circ U_1$. При $A_1 = A_2$ композиция алгоритмов совпадает с их нормальной композицией. Сам алгоритм $U_2 \circ U_1$ является нормальным над алфавитом A. При этом $U_2 \circ U_1(C) = U_2(U_1(C))$

для каждого слова C в алфавите A_1. Алгоритм $U_2 \circ U_1$ является применимым только к тем словам C в алфавите A, которые удовлетворяют ограничениям:

C - слово в алфавите A_1, U_1 применим к C, U_2 применим к $U_1(C)$.

Для расширения E алфавита A и слова C в алфавите E можно получить новое слово C^A путём стирания всех входящих в C букв из E\A. Слово C^A называется проекцией слова C на алфавит A. Тогда схема $\{ \xi \rightarrow \Lambda \ (\xi \in B - A)\}$ даёт нормальный алгоритм $U_{B,A,}$ для которого $U_{B,A}(P) = P^A$ для всякого слова P в алфавите B. Такой алгоритм $U_{B,A}$ называется проектирующим.

Предположим теперь, что имеются два алфавита A и C не имеющие общих букв. Возьмём их объединение $B = A \cup C$. По схеме $\{ca \rightarrow ac\}$ $(a \in A, c \in C)$ задаётся нормальный алгоритм $V_{A,C}$ в алфавите B.

Такой алгоритм нормален. Он называется композицией алгоритмов U и V, его также обозначают $V \circ U$.

Рассмотрим некоторый нормальный алгоритм D в алфавите A. Возьмем некоторый алфавит B, который является расширением A. Тогда можно построить алгоритм D_B вполне эквивалентный алгоритму D относительно A и называемый формальным распространением алгоритма D на алфавите B. Для его построения в схеме алгоритма D добавляют формулы подстановки вида $b \rightarrow b$, где $b \in B - A$.

Рассмотрим два нормальных алгоритма U и V в алфавитах A и C, возьмём алфавит $B = A \cup C$ и формальные распространения U_B и V_B алгоритмов U и V на алфавит B. Тогда можно образовать композицию $E = V_B \circ U_B$ Этот алгоритм E над алфавитом B называется нормальной композицией алгоритмов U и V, и она обозначается $V \circ U$. Очевидно, что $E(P) \cong V \circ U(P)$ для любого слова P в алфавите A, а также он применим к тем и только тем словам P, которые удовлетворяют трём условиям: P – слово в A, U применим к P, V применим к U(P).

Возьмём случай, когда алфавит B является расширением алфавита A, а слово P задано в алфавите A. Если в P стереть все буквы из алфавита B-A, то получится некоторое слово P^A называемое проекцией P на алфавит A. Это процедура задается проектирующим алгоритмом $V_{B,A}$ со схемой $\{\xi \to \wedge\ (\xi \in B - A)$.

Если же A и B алфавит без общих букв, $C = A \cup B$, то алгоритм $L_{A,B}(P) = P^A P^B$ в алфавите C имеет схему $\{ba \to ab(a \in A, b \in B)$

Если алфавит B есть расширение A, то алгоритм V в алфавите B называется естественным распространением алгоритма U с алфавита U с алфавита A на B тогда и только тогда, когда $V(PQ) \cong V(P)Q$ для любого слова P в A и всякого слова Q в B-A.

Теорема. Предположим, что $V_1, \dots, V_k$ – нормальные алгоритмы, A – объединение их алфавитов. Тогда существует нормальный алгоритм U над A удовлетворяющий $U(P) \simeq C_1(P)C_2(P) \dots C_k(P)$ для любого слова P в алфавите A, где C_i – естественное распространение V_i на A.

Примеры работы нормального алгоритма Маркова.

1. В качестве входные данных берётся слово cdbacab. Алгоритм состоит в следующем: ab → bd; db → ba; bba → abb; c → ʎ, где ʎ - пустой символ.

Путем преобразований исходного слова данным алгоритмом, последовательно получаются следующие слова: cdabacab → cdbacbd → cbaacbd → baacbd → baabd → babdd → bbddd. Результатом работы рассматриваемого алгоритма является слово bbddd.

Далее рассматриваются возможности реализации арифметических операций с помощью нормальных алгоритмов Маркова. Сначала обратим внимание на одно обстоятельство, связанное с работой любого нормального алгоритма Маркова. Оно состоит в том, что нужно либо вводить дополнительное правило остановки работы нормального алгоритма (иначе в примере увеличения числа на 1 алгоритм продолжит работу и снова будет увеличивать полученный результат еще на 1 и т.д. неограниченное число раз), либо перед началом работы нормального алгоритма добавлять к входной строке специальные символы, отличные от других символов строки, которые учитываются подстановками алгоритма в начале его работы и которые удаляются в конце работы алгоритма. Мы будем придержи-

ваться второго способа, как и одна из наиболее успешных реализаций нормальных алгоритмов Маркова в виде языка программирования Рефал. В качестве добавляемого символа возьмем символ "*".

Пример 2. Рассмотрим простейшую операцию увеличения десятичного числа на 1. В этом случае почти всегда необходимо увеличить последнюю цифру на 1, а последняя цифра отличается тем, что после нее идет символ "*". Поэтому первыми подстановками должны быть подстановки типа <цифра> * → <цифра + 1>. Но если это цифра 9, то ее нужно заменить 0 и увеличение на 1 перенести в предыдущий разряд. Этому отвечает подстановка 9* → *0. Наконец, если число начинается с 9 и перед этой цифрой нужно поставить 1, то этому будет отвечать подстановка ** → 1, а если это не так, то в конце работы алгоритма символы * надо стереть, что выполнит подстановка * → ʎ. Таким образом, мы получаем следующий НАМ увеличения десятичного числа на 1: 0* → 1; 1* → 2; 2* → 3; 3* → 4; 4* → 5; 5* → 6; 6* → 7; 7* → 8; 8* → 9; 9* → *0; ** → 1; * → ʎ.

Приведем работу построенного алгоритма для чисел 79 и 99:

79 → *7*0 → *80 → 80;

99 → *9*0 → **00 → 100.

Пример 3. Прежде, чем перейти к другим арифметическим операциям, рассмотрим как довольно типичный пример, используемый часто в других алгоритмах, алгоритм копирования двоичного числа. В этом случае прежде всего исходное и скопированное числа разделим символом "*". В разрабатываемом алгоритме мы будем копировать разряды числа по очереди, начиная с младшего, но нужно решить 2 проблемы: как запоминать значение символа, который мы копируем, и как запоминать место копируемого символа. Для решения второй проблемы используем символ "!", которым мы будем определять еще не скопированный разряд числа, после которого и стоит этот символ. Для запоминания значения копируемого разряда мы будем образовывать для значения 0 символ "a", а для значения 1 - символ "b". Меняя путем подстановок эти символы "a" или "b" с последующими, мы будем передвигать разряды "a" или "b" в начало копируемого числа (после "*"), но для того, чтобы пока не происходило копирование следующего разряда справа, мы перед передвижением разряда временно символ "!" заменим на символ "?", а после передвижения сделаем обратную замену. После того как все число окажется скопированным в виде символов "a" и "b", мы заменим эти символы на 0 и 1 соответственно. В результате нормальный алгоритм копирования двоичного числа можно определить следующей последовательностью подстановок:

1. Начальные пометки копирования разряда и копии числа:
   0* → 0!*; 1* → 1!*.
2. Копирование разряда с заменой пометки разряда:
   0! → ?0a; 1! → ?1b.
3. Передвижение скопированного разряда:
   a0 → 0a; a1 → 1a; b0 → 0b; b1 → 1b.
4. Остановка передвижения скопированного разряда:
   a* → *a; b* → *b.
5. Обратная замена пометки разряда:
   ? → !.
6. Обратная замена скопированного разряда:
   a → 0; b → 1.
7. Стирание символов:
   *! → ʌ.

Можно продемонстрировать работу этого алгоритма для числа 10:

*1$\underline{0*}$ → *1$\underline{0!}$* → *1?0$\underline{a*}$ → *1$\underline{?}$0 * a → *$\underline{1!}$0 * a → *?1$\underline{b0}$ * a → *?10$\underline{b*}$a → *$\underline{?}$10 * ba → *!10 * $\underline{ba}$ → $\underline{*!}$10 * 10 →10 * 10.

Приведенные примеры показывают также возможности аппарата нормальных алгоритмов Маркова по организации ветвления и цикличных процессов вычисления. Это показывает, что всякий алгоритм может быть нормализован. Это означает, что задан нормальным алгоритмом Маркова. В этом и состоит тезис Маркова, который следует интерпретировать как определение алгоритма.

Вместе с тем построение алгоритма в последнем приведенном примере подсказывает следующую методику разработки нормальным алгоритмом Маркова:

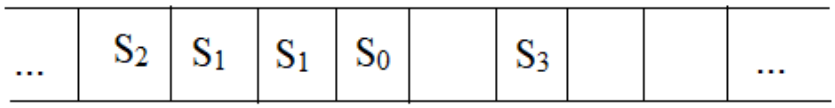
1. Произвести декомпозицию строящегося алгоритма.
2. Решение проблем реализации каждой части. В предыдущем примере:
   запоминание копируемого разряда - разряд 1 запоминается как символ "a", а разряд 0 - как символ "b";
   запоминание места копируемого разряда - пометка еще не скопированного символа дополнительным символом "!" с заменой его на символ "?" при передвижении копируемого разряда и обратной заменой после передвижения.
3. Если часть для реализации является сложной, то она также подвергается декомпозиции.
4. Сборка реализации в единый алгоритм.

3. Лекция 3. **Алгоритмы Тьюринга.**

Тьюринг в результате своих исследований понятия эффективной вычислимости выделил класс абстрактных машин пригодных для выполнения всякой «механической» вычислительной процедуры. В настоящее время они называются машинами Тьюринга.

Эти машины описываются следующим образом.

Пусть имеется лента. О ней предполагается, что она потенциально бесконечна в обе стороны и разделена на квадраты. При этом в каждый момент времени она имеет конечную длину, но всегда к ней могут быть добавлены новые квадраты как слева, так и справа.



Имеется конечное множество символов ленты S_0, …, S_n, которое называется алфавитом машины. В каждый момент всякий квадрат может быть записан не более чем одним символом. Сама машина обладает конечным множеством внутренних состояний $\{q_0, q_1, \ldots, q_m\}$. В любой момент времени она находится в точности в одном из этих состояний. Также имеется читающая головка, находящаяся на одном из квадратов ленты в каждый момент времени. Если в момент времени t читающая головка воспринимает квадрат с символом S_i в нем, а машина находится во внутреннем состоянии q_j, то действие машины однозначно определено и она совершает один из четырех актов: (1) головка стирает символ S_i и записывает на том же квадрате символ S_k; (2) головка перемещается в соседний слева квадрат; (3) головка перемещается в соседний справа квадрат; (4) машина останавливается.

В вариантах (1), (2), (3) машина переходит в новое внутренне состояние q_r и готова снова к действию в момент t+1.

Предполагается, что символ S_0 представляет пустой квадрат. Таким образом читающая головка всегда воспринимает некоторый символ алфавита.

Первые три возможных актов действия машины описываются следующими упорядоченными четверками называемыми командами: (1) $q_jS_iS_kq_r$, (2) $q_jS_iLq_r$, (3) $q_jS_iRq_r$. q_j – внутреннее состояние, S_i – воспринимаемый символ, третий символ – действие машины: S_k – написание головкой символа S_k; L – перемещение головки на один квадрат влево; R – перемещение головки на один квадрат вправо.

Если лента вкладывается в машину Тьюринга ее читающая головка помещается на один из квадратов ленты, машина приводится в одно из своих внутренних состояний. Тогда машина начинает оперировать на ленте согласно трем вариантам. Если в некоторый момент t_1 машина останавливается, то находящаяся в момент остановки лента называется результатом применения машины на данной ленте.

С каждой машиной Тьюринга Т связывается некоторый алгоритм В в алфавите А машины Т. Берется некоторое слово Р в алфавите А и записывается слева направо в квадратах чистой ленты. Потом ленту помещают в машину Т, чтобы читающая головка воспринимала самый левый квадрат, машина Т приводится во внутреннее состояние q_0. Далее машина начинает работать. Если машина Т когда-нибудь остановится, то появившееся в результате на ленте слово в А является значением алгоритма В. Такой алгоритм называется алгоритмом Тьюринга. Результирующее слово читается слева направо на ленте, причем пустые квадраты считаются заполненными символом S_0.

Итак, машиной Тьюринга называется любое конечное множество T упорядоченных четвёрок символов, удовлетворяющих условиям: (i) каждая входящая в T четвёрка принадлежит одному из трёх типов: (1) $q_jS_iS_kq_r$; (2) $q_jS_iLq_r$; (3) $q_jS_iRq_r$, (ii) никакие две четвёрки из T не имеют совпадающих первые два сим-

вола. Упорядоченные четвёрки указанных типов называются командами. Множество символов $\{S_m\}$ – алфавит машины T, q_s – внутреннее состояние, q_0 – внутреннее состояние любой машины T.

Конфигурацией машины T называется каждое слово Pq_sQ, где P – слово (возможно пустое) в алфавите МТ, q_s – внутреннее состояние МТ, Q – непустое слово в МТ.

Машина T переводит конфигурацию α в конфигурацию β, обозначается $\alpha \underset{T}{\rightarrow} \beta$, если либо (a) $\alpha = Pq_jS_iQ$, $\beta = PqS_kQ$ и $q_jS_iS_kq_r$ есть одна из команд МТ, либо (b) $\alpha = PS_sq_jS_iQ$, $\beta = Pq_rS_sS_iQ$ и q_jS_iLq – команда МТ, либо (c) $\alpha = q_jS_iQ$, $\beta = q_rS_0S_iQ$, $q_jS_iLq_r$ – команда МТ, либо (d) $\alpha = Pq_jS_iS_kQ$, $\beta = PS_iq_rS_kQ$, $q_jS_iRq_r$ – команда МТ, либо (e) $\alpha = Pq_jS_i$, $\beta = PS_iq_rS_0$, $q_jS_iRq_r$ – команда МТ.

МТ останавливается при конфигурации α, если не существует конфигурации β такой, что $\alpha \underset{T}{\rightarrow} \beta$. Это имеет место в том случае, когда q_jS_i входит в α, то среди команд МТ нет начинающейся с q_jS_i.

Вычисление МТ задается последовательностью конфигураций $\alpha_0, \dots, \alpha_m$ с $m \geq 0$ такой, что внутреннее состояние входящее в α_0 есть q_0; $\alpha_i \underset{T}{\rightarrow} \alpha_{i+1}$ при $i = 0, 1, \dots, m-1$, МТ останавливается на α_m. Тогда говорят, что вычисление начинается с α_0 и заканчивается на α_m.

Рассмотрим алфавит C, содержащий в себе алфавит A МТ. Тогда можно задать алгоритм $B_{T,C}$ в C так: для произвольных слов P и Q в C равенство $B_{T,C}(P) = Q$ выполняется ⇔ существует вычисление МТ, начинающееся с конфигурации до T и заканчивается конфигурацией вида $R_1q_jR_2$, где $R_1R_2 = Q$. Алгоритм U в алфавите D называется вычислимым по Тьюрингу, если существует МТ с алфавитом A и алфавитом C, где $A \cup D \subseteq C$, т.ч. алгоритмы $B_{T,C}$ и D вполне эквивалентны относительно D.

Запишем 1 вместо S_1, слово 1^{m+1} обозначим $\overline{m}$ для любого натурального числа m, * обозначим S_2.

Любое конечное множество T упорядоченных четвёрок символов, удовлетворяющих условиям:

i. Каждая входящая в T четвёрка принадлежит одному из трёх типов:
   1. $q_jS_iS_kq_r$
   2. $q_jS_iLq_r$
   3. $q_jS_iRq_r$

ii. Никакие две четвёрки из T не имеют совпадающими первые два символа. Упорядоченные четвёрки указанных типов называются командами. Множество символов $\{S_m\}$ – алфавит машины T, q_s – внутренние состояния. Ограничение: q_0 – внутреннее состояние любой машины T.

Конфигурацией машины T называется каждое слово Pq_sQ, где P – слово (возможно пустое) в алфавите MT, q_s – внутреннее состояние MT. Машина T переводит конфигурацию α в конфигурацию β, обозначается как $\alpha\ \overrightarrow{T}\ \beta$, если либо:

a) $\alpha = Pq_jS_iQ\ ,\ \beta = Pq_rS_kQ$ и $q_jS_iS_kq_r$ есть одна из команд Машины T, либо
b) $\alpha = PS_Sq_jS_iQ\ ,\ \beta = Pq_rS_SS_iQ$ и q_jS_iLq – команда Машины T, либо
c) $\alpha = q_jS_iQ\ , \beta = q_rS_0S_iQ\ ,\ q_jS_iLq_r$ – команда Машины T, либо
d) $\alpha = Pq_jS_iS_kQ\ ,\ \beta = PS_iq_rS_kQ$ и $q_jS_iRq_r$ – команда Машины T, либо
e) $\alpha = Pq_jS_i,\ \beta = PS_iq_rS_0$ и $q_jS_iRq_r$ – команда Машины T.

MT останавливается при конфигурации α, если не существует конфигурации β такой, что $\alpha\ \vec{T}\ \beta$.

Это имеет место в том случае, когда q_jS_i входит в α, но среди команд MT нет начинающейся с q_jS_i.

Вычисление MT задается последовательностью конфигураций $\alpha_0, \dots, \alpha_m$ с $m \geq 0$ такой, что внутреннее состояние входящее в α_0 есть q_0;

$\alpha_i\ \vec{T}\ \alpha_{i+1}$ при $i = 0,1, \dots, m-1$, MT останавливается на α_m.

Тогда говорят, что вычисление начинается с α_0 и заканчивается на α_m.

Рассмотрим алфавит C, содержащий в себе алфавит A MT. Тогда можно задать алфавит $B_{T,C}$ в C так:

Для произвольных слов P и Q в C равенство $B_{T,C}(P) = Q$ выполняется <=> существует вычисление MT, начинающееся с конфигурации q_0T и заканчивающееся конфигурацией вида $R_1q_jR_2$, где $R_1R_2 = Q$. Алгоритм U в алфавите D называется вычислимым по Тьюрингу, если существует MT с алфавитом A и алфавитом C, где $A \cup D \subseteq C$, т.ч. алгоритмы $B_{T,C}$ и D вполне эквивалентны относительно D.

Запишем 1 вместо S_1, слово 1^{m+1} обозначим $\overline{m}$ для любого натурального числа m, * обозначим S_2.

Пусть имеется алфавит A, в котором содержится 1 и *, а для всяких натуральных чисел $k_1,\dots,k_n$ и каждого слова P и данной частичной арифметической функции $f(x_1,\dots,x_n)$ равенство $B_{T,A}(\bar{k}_1, \dots, \bar{k}_n) = P$ выполняется тогда и только тогда, когда существуют слова T_1 и T_2 в алфавите { S_0 } таким, что $P = T_1$

$\overline{f(k_1, \ldots, k_n)}$ T_2. Напомним, что символ S_0 интерпретируется как изображение пустого квадрата ленты МТ. Поэтому форма T_1 $\overline{f(k_1, \ldots, k_n)}$ T_2 допустима для P. В таком случае частичная арифметическая функция $f(x_1,\ldots,x_n)$ называется вычислимой по Тьюрингу. То есть, когда существует МТ, осуществляющая вычисление рассматриваемой функции.

Предложение. Предположим, что C-расширение алфавита A, а T-машина Тьюринга с алфавитом A. Тогда существует нормальный алгоритм U над алфавитом C, который вполне эквивалентен алгоритму Тьюринга $B_{T,C}$.

Доказательство. Положим E = C ⋃ $\{q_{k0},\ldots q_{kn}\}$, где $q_{k0},\ldots,q_{kn}$ - некоторые внутренние состояния T, причем $q_{k0} = q_0$, Для построения схемы алгоритма U возьмем сначала для всех команд $q_jS_iS_kq_r$ машины Тьюринга T формулы подстановок вида q_jS_i -> q_rS_k. Возьмем произвольную команду $q_jS_iS_kq_r$ машины Тьюринга T и S ϵ C и, исходя из формулы подстановки q_jS_i -> $q_rS_0S_i$, перечислим все формулы подстановок типа $S_lq_jS_i$ -> $q_rS_lS_i$. Потом для S_l ϵ C и формулы подстановки q_jS_i -> $S_iq_rS_0$ и произвольной команды $q_jS_iRq_r$ предъявим всевозможные формулы подстановок типа $q_jS_iS_l$ -> $S_iq_rS_l$.

Для любого $q_{k_i} \in E$ возьмем формулу подстановки $q_{k_i} \rightarrow \Lambda$ и присоединим к списку формул подстановки еще $\Lambda \rightarrow q_0$.

В результате получается схема, по которой задается алгоритм U над C. Тогда $B_{T,C}(P) \simeq U(P)$ для каждого слова P в C.

Следствие. Если функция f вычислима по Тьюрингу, то она вычислима по Маркову.

Доказательство. Рассмотрим функцию $f(x_1, \ldots, x_n)$, которая вычислима по Тьюрингу.

4. Лекция 4. K-значная логика. Формулы и функции k-значной логики.

Наряду с двузначной логикой имеются k-значные логики. Потребность в них возникает уже в рассуждениях, когда возможны более двух вариантов исходов. Они также полезны для разработки алгоритмов программ, алгоритмов управления машинами, когда имеются многовариантные возможности действий и т.д.

Конечнозначные логики являются обобщениями двузначной логики. Часть свойств и результатов двузначной логики переносятся на k-значные логики, но имеются между ними и принципиальные отличия.

Берётся некоторый алфавит $U = \{u_1, u_2, \dots\}$ переменных (аргументов) и рассматриваются функции $f(u_{j_1}, \dots, u_{j_n})$, где $u_{jk} \neq u_{jl}$ при $k \neq l$, u_j определены на множестве $B_k = \{0,1, \dots, k-1\}$, $f: B_k^n \to B_k$. Каждая такая функция полностью характеризуется своей таблицей. Функцию одной переменной можно охарактеризовать подстановкой $S(x) = \begin{pmatrix} 0 & 1 & \dots & k-1 \\ i_0 & i_1 & \dots & i_{k-1} \end{pmatrix}$, где $S(h) = i_n$.

Множество всех функций k-значной логики обладают P_k. В это множество также входят постоянные функции $0,1, \dots, k-1$. Число различных наборов $(a_1, \dots, a_n)$ значений переменных $x_1, \dots, x_n$ равно k^n. Для каждого набора $(a_1, \dots, a_n)$ функция f из P_k принимает одно из значений $0,1, \dots, k-1$. Поэтому получается такой результат.

Теорема 1.

Число всевозможных функций k-значной логики, зависящих от n переменных равно k^m, где $m = k^n$.

Например, в P_3 число функций от двух переменных составляет $3^9 = 19683$. Число строк в таблицах функций n переменных возрастает с k и n как k^n. Из-за этого вместо таблиц функций часто используют алгоритмы вычисления функций . Например, $f(x_1, .., x_n) = \max(x_1, \dots, x_n)$ вычисляется как максимальное значение среди n значений переменных.

Примеры. 1. $\bar{x} = x + 1 (mod\ k)$, $\bar{x}$ - обобщение отрицания в смысле «циклического» сдвига значений;

1. $\mathrm{N}x = k - 1 - x = \sim x$ – отрицание в смысле «зеркального» отображения значений, отрицание Лукашевича;
2. $I_i(x) = \begin{cases} k-1, \text{при } x = i \\ 0, \text{при } x \neq i \end{cases}$, где $i = 0, \dots, k-1$
   $I_i(x)$ тоже обобщает некоторые свойства отрицания;

3. $f_i(x) = \begin{cases} 1, \text{при } x = i \\ 0, \text{при } x \neq i \end{cases}$ характеристическая функция.

При $i \neq k-1$ тоже обобщает отрицание.

4. $\min(x_1, x_2) -$ обобщение конъюнкции;
5. $x_1 x_2 (mod\ k) -$ второе обобщение конъюнкции;
6. $\max(x_1, x_2) -$ обобщение дизъюнкции;
7. $x_1 + x_2 (mod\ k)$.

Формулы U и B эквивалентны, ели соответствующие им функции f_u и f_B равны.

$V_K(x_1, x_2) = \max(x_1, x_2) + 1 -$ функция Вебба – аналог функции Шеффера.

Рассмотрим теперь свойства элементарных функций k-значной логики. Пусть $x_1 \circ x_2$ служит обозначением какой-либо из функций $\min(x_1, x_2)$, $x_1 x_2 (mod\ k)$, $\max(x_1, x_2)$, $x_1 + x_2 (mod\ k)$.

1. Функция $x_1 \circ x_2$ ассоциативна
   $((x_1 \circ x_2) \circ x_3 = (x_1 \circ (x_2 \circ x_3)$.
2. Функция $x_1 \circ x_2$ коммутативна
   $(x_1 \circ x_2) = (x_2 \circ x_1)$.

Также функцию $\min(x_1, x_2)$ обозначают $(x_1 \& x_2)$, а функцию $\max(x_1, x_2)$ часто обозначают $(x_1 \vee x_2)$. Операция & выполняется раньше, чем ∨, что используют для сокращения скобок в формулах.

Выполняются следующие тождества в системе $\{0,1, \dots, k-1, I_0(x), \dots, I_{k-1}(x), \min(x_1, x_2), \max(x_1, x_2)\}$.

3. Правило спуска символа I «вглубь» формулы:
   $I_b(C) = \begin{cases} k-1 \text{ при } c = b \\ 0 \text{ при } c \neq b \end{cases}$, где $b, c = 0,1, \dots, k-1$;
   $$I_b\big(I_c(x)\big) = \begin{cases} I_0(x) \vee \dots \vee I_{c-1}(x) \vee I_{c+1}(x) \vee \dots \vee I_{k-1}(x), b = 0 \\ 0 \text{ при } 0 < b < k-1 \\ I_c(x) \text{ при } b = k-1 \end{cases};$$
   $I_b(x_1 x_2) = I_b(x_1)(I_b(x_2) \vee \dots \vee I_{k-1}(x_2)) \vee I_b(x_2)(I_b(x_1) \vee \dots \vee I_{k-1}(x_1))$, где $x_1 \cdot x_2 = \min(x_1, x_2) = x_1 \& x_2$,

   $I_b(x_1 \vee x_2) = I_b(x_1)(I_0(x_2) \vee \dots \vee I_b(x_2)) \vee I_b(x_2)(I_0(x_1) \vee \dots \vee I_b(x_1))$, где $x_1 \vee x_2 = \max(x_1, x_2)$.
4. Свойства дистрибутивности:
   $(x_1 \vee x_2) x_3 = (x_1 x_3) \vee (x_2 x_3)$,
   $(x_1 x_2) \vee x_3 = (x_1 \vee x_3)(x_2 \vee x_3)$.
5. Правило исключения «чистых» вхождений переменной:

$x = 1 \cdot I_1(x) \vee 2 \cdot I_2(x) \vee \ldots \vee (k-1) \cdot I_{k-1}(x)$.

6. Правило введения переменной:
   $x_1 = x_1(I_0(x_2) \vee \ldots \vee I_{k-1}(x_2))$.
7. Правила упрощений:
   $I_b(x)I_c(x) = \begin{cases} I_b(x) \text{ при } c = b \\ 0 \text{ при } b \neq c \end{cases}$,
   где $x \cdot y = x\&y = \min(x, y)$;
   $x \vee y = \max(x, y)$.
8. $(k-1)x = x;\ 0 \cdot x = 0$
   $(k-1) \vee x = k-1;\ 0 \vee x = x$.

Важно отметить, что не все свойства булевых функций сохраняются в k-значной логике при $k \geq 3$.

Примеры.

1)$\bar{\bar{x}} \neq x$ при $k \geq 3$,

2)$\overline{\min(x_1, x_2)} \neq \max(\overline{x_1}, \overline{x_2})$ при $k \geq 3$.

Предложение 1.

Если $f(x_1, \ldots, x_2) -$ функция k-значной логики, то $f(x_1, \ldots, x_2) = (b_1, \ldots, b_n) \vee I_{b_1}(x_1)\& \ldots \& I_{b_n}(x_n)\& f(b_1, \ldots, b_n)$, (1).

Доказательство последней формулы получается непосредственной проверкой. Формула (1) является аналогом совершенной дизъюнктивной нормальной формы.

5. Лекция 5. Полные системы функций.

Определение 1. Система функций $\Omega = \{f_1, f_2, \ldots, f_n, \ldots\}$ из P_k называется функционально полной, если каждая функция из P_k может быть представлена в виде формулы через функции системы Ω.

Очевидно, что система $\Omega = P_k$ полна.

Теорема 1.

Система

$\Omega = \{0,1, \ldots, k-1, I_0(x), \ldots, I_{k-1}(x), \min(x_1, x_2), \max(x_1, x_2)\}$ является полной в P_k .

Доказательство вытекает из формулы (1) предложения 1 предыдущего параграфа.

Теорема 2.

Система $\Omega = \{\bar{x}, \max(x_1, x_2)\}$ является полной в P_k .

Доказательство. Проводится в несколько этапов.

Сначала строятся все постоянные функции. Из функции $\bar{x} = x + 1$ последовательно путем итераций получают функции $x + 2 = (x + 1) + 1, \dots, x + k - 1 = (x + k - 2) + 1, x = x + k = \big(x + (k - 1)\big) + 1.$

Тогда $\max(x, x + 1, \dots, x + k - 1) = k - 1.$

Из постоянной функции $k - 1$ остальные постоянные функции $0,1, \dots, k - 2$ получают с помощью $\bar{x} = f(x)$, так $0 = f(k - 1)$, $1 = f(0)$ и т.д.

Далее строят функции одной переменной. Функции $I_m(x)$ задаются формулами $I_m(x) = 1 + \max(x + a)$, $a \neq k - 1 - m$, где $m = 0,1, \dots, k - 1$. Для их проверки рассмотрим $x = m$, тогда левая часть равна $k - 1$, а правая часть равна

$$1 + \max(m + a) = 1 + \max(m + a) = 1 + k - 2 = k - 1$$

$a \neq k - 1 - m \qquad m + a \neq k - 1$

При $x \neq m$ левая часть равна 0, а правая составляет

$$1 + \max(x + a) = 1 + \big(x + (k - 1 - x)\big) = k = 0.$$

$a \neq k - 1 - m$

Далее рассмотрим функции $f_{s,m}(x)$ такие, что

$$f_{s,m}(x) = \begin{cases} s \text{ при } x = m \\ 0 \text{ при } x \neq m \end{cases}.$$

Тогда получается разложение

$$f_{s,m}(x) = s + 1 + max\{I_m(x); k - 1 - m\}.$$

Произвольная функция $g(x)$ одной переменной x из P_k можно тогда записать в виде

$$g(x) = max\big\{f_{g(0),0}(x), f_{g(1),1}(x), \dots, f_{g(k-1),k-1}(x)\big\}.$$

В частности, для отрицания Лукашевича $\mathrm{N}x = {\sim}x = max\{f_{k-1,0}(x), f_{k-2,1}(x), \dots, f_{0,k-1}(x)\}$.

При этом $max\{x_1, \dots, x_n\} = max\{max\{x_1, \dots, x_{n-1}\}, x_n\}$ для $n \geq 3$ получается в результате последовательных композиций из $max\{x, y\}$.

Построим теперь $\min\{x_1, x_2\}$ из уже имеющихся в доказательстве выше функций. Воспользуемся тождеством

${\sim}\min(x_1, x_2) = \max({\sim}x_1, {\sim}x_2)$, т.е.

$\min(x_1, x_2) = {\sim}\max({\sim}x_1, {\sim}x_2)$.

Поэтому из теоремы 1 следует утверждение данной теоремы.

Определение 2.

Функция Вебба называется $V_k(x_1, x_2) = \max(x_1, x_2) + 1$.

Теорема 3.

Система $B = \{V_k(x_1, x_2)\}$ полна в P_k.

Доказательство.

Заметим, что $V_k(x, x) = \max(x, x) + 1 = x + 1 = \bar{x}$.

Последовательная композиция $\bar{x} = f_1(x)$,

$f_2(x) = f_1\big(f_1(x)\big), \dots, f_k(x) = f_1(f_{k-1}(x))$ даёт $f_k(x) = x$ и $f_k\big(V_k(x_1, x_2)\big) = \max(x_1, x_2)$ в P_k.

В силу теоремы 2 получаем утверждение данной теоремы.

Определение 3.

Замыканием подмножества S в P_k называется множество $[S]$ всех функций из P_k, которые представимы в виде всевозможных композиций через функции из S.

Подмножество S в P_k называется функционально замкнутым, если его замыкание $[S]$ совпадает с S.

Примеры.

1.Само множество $S = P_k$ замкнуто.

2. Для подмножества Q в $E_k = \{0,1,\dots,k-1\}$ рассмотрим множество T_Q всех функций $f(x_1,\dots,x_n)$ принадлежащих P_k и удовлетворяющих условию $f(\alpha_1,\dots,\alpha_n) \in Q$, если $\alpha_i \in Q$ для всех $i = 1,\dots,n$. Подмножество функций (класс) T_Q замкнуто.

3. Для системы функций $B = \{\sim x, \max(x_1, x_2)\}$ возьмём подмножество $Q = \{0, k-1\}$ в E_k, где $k \geq 3$. Тогда $B \subseteq T_Q$. Подмножество T_Q не равно P_k, так как T_Q не содержит постоянных $1,\dots,k-2$. Отсюда вытекает, что система B не полна.

В следующем параграфе рассматривается полнота систем функций в частном случае при k=2. Это нужно для сравнения со случаями k-значной логики при k>2.

K-значные логики при k>2 имеют ряд специфических особенностей по сравнению со случаем k=2. Для k-значных логик при k>2 полнота систем функций и критерии распознавания полноты изложены в последующих параграфах.

6. Лекция 6. Полнота и замкнутость систем функций алгебры логики.

Определение 1. Некоторая система функций B = $\{\beta_1, \beta_2,\dots, \beta_p\dots\}$ из P_2 называется функционально полной, если каждая функция алгебра логики может быть записана с помощью формулы через функции этой системы.

Примеры: Сама система P_2 всех булевых функций является функционально полной.

Уже была доказана полнота следующих систем функций.

1. $\{\neg x, x \vee y\}$
2. $\{\neg x, x \,\&\, y\}$
3. $\{\neg x, x \rightarrow y\}$
4. $\{0, 1, x \,\&\, y, x \oplus y\}$
5. $\{x \downarrow y\}$
6. $\{x \mid y\}$

Теорема 1. Если имеются две системы функций B = $\{\beta_1, \beta_2,\dots\}$ и D = $\{g_1, g_2,\dots\}$, и система B функционально полна, а каждая функция β_j из B выражается в виде формулы через функции системы D, то системы D тоже функционально полна.

Доказательство. Возьмем произвольную функцию h из P_2, тогда в силу функциональной полноты системы B имеется формула над B такая, что h = C $[\beta_1, \beta_2,\ldots, \beta_p\ldots]$, то есть h выражается с помощью композиций функций из B. Также по условию теоремы имеются формулы над системой D такие, что $\beta_j = C_j [g_1, g_2,\ldots]$ для любого j. Поэтому

$$C [\beta_1, \beta_2,\ldots] = C [C_1 [g_1, g_2,\ldots], C_2[g_1, g_2,\ldots],\ldots]$$

или

$$C [C_1 [g_1, g_2,\ldots], C_2[g_1, g_2,\ldots],\ldots] = C^{|} [g_1, g_2,\ldots],$$

где формула со строением $C^{|}$ над системой D определяется левой частью равенства. Итак, $h = C^{|} [g_1, g_2,\ldots]$, то есть функция h выражена через функции системы D в виде формулы над системой D.

Определение 2. Если M – подмножество функций P_2, то множество [M] всех функций, которые представляются в виде формул через функции из M, называется замыканием M.

Из этого определения вытекает, что замыкание инвариантно относительно введения и удаления фиктивных переменных.

В связи с понятиями замкнутости и полноты часто рассматриваются следующие важные классы функций.

1. Класс T_0 всех функций алгебры логики $\beta(x_1,\ldots, x_n)$, которые сохраняют постоянную 0, то есть $\beta(0,\ldots, 0) = 0$.

Например, функции $\beta_1(x) = 0$, $\beta_2(x) = x$, $\beta_3(x,y) = x\&y$, $\beta_4(x,y) = x \vee y$, $\beta_5(x,y) = x \oplus y$ принадлежат классу T_0, но функции $\beta_6(x) = 1$, $\beta_7(x) = \neg x$ не входят в T_0.

В классе T_0 содержится $\frac{1}{2}2^{2^n}$ функций n переменных $x_1,\ldots,x_n$, так как первая строка таблицы каждой такой функции содержит 0. Если $\beta, \beta_1,\ldots, \beta_m \in T_0$, то $\beta(\beta_1(0,\ldots,0)\ldots, \beta_m(0,\ldots,0)) = \beta(0,\ldots,0) = 0$, следовательно, класс T_0 замкнут.

2. Класс T_1 функций алгебры логики $\beta(x_1,\ldots, x_n)$, которые сохраняют постоянную 1, что выражается равенством $\beta(1,\ldots, 1) = 1$.

Функции $\beta_1(x) = 0$ и $\beta_7(x) = \neg x$ не входят в T_1, а функции $\beta_2(x) = x$, $\beta_3(x,y) = x\&y$, $\beta_4(x,y) = x \vee y$, $\beta_6(x) = 1$ содержатся в классе T_1.

Можно легко усмотреть, что $\beta(x_1,\ldots, x_n)$ принадлежит T_1 тогда и только тогда, когда двойственная ей функция $\beta^*(x_1,\ldots, x_n) = \neg\beta(\neg x_1,\ldots, \neg x_n)$ принадлежит T_0. Поэтому классы T_0 и T_1 двойственны друг к другу. Класс T_1 также замкнут и содержит $\frac{1}{2}2^{2^n}$ функций $\beta(x_1,\ldots, x_n)$ n переменных.

3. Класс всех самодвойственных функций $\beta^* = \beta$ обозначают S.

Например, функции $\beta_2(x) = x$ и $\beta_7(x) = \neg x$ принадлежат S, а также $\beta(x_1, x_2, x_3) = x_1\&x_2 \vee x_1\& x_3 \vee x_2\& x_3$ самодвойственна.

Самодвойственная функция $\beta(x_1,\ldots,x_n)$ на противоположных наборах $(a_1,\ldots,a_n)$ и $(\neg a_1,\ldots,\neg a_n)$ принимает противоположные значения, следовательно, таблица для β полностью характеризуется ее верхней половиной. Поэтому число самодвойственных функций, зависящих от n переменных $x_1,\ldots,x_n$, равно $2^{2^{n-1}}$. Если $\beta, \beta_1,\ldots, \beta_m$ принадлежат S, то $\beta^*(\beta^*_1,\ldots,\beta^*_m) = \beta(\beta_1,\ldots,\beta_m)$, следовательно класс S замкнут.

Лемма 1. Пусть $\beta(x_1,\ldots,x_n)$ – несамодвойственная функция. Тогда существует подстановка функций x и ¬x в β, дающая несамодвойственную функцию одной переменной, то есть постоянную.

Доказательство. Поскольку $\beta \notin S$, то существует набор $(a_1,\ldots,a_n)$, для которого $\beta(\neg a_1,\ldots,\neg a_n) = \beta(a_1,\ldots,a_n)$. Возьмем функцию

$$g(x) = \beta(g_1(x),\ldots, g_n(x)),$$

где $g_k(x)=x^{a_k}$, $x^a = \begin{cases} x, \text{при } a = 1 \\ \neg x, \text{при } a = 0. \end{cases}$ Тогда выполняются равенства

$$g(0) = \beta(g_1(0),\ldots, g_n(0)) = \beta(0^{a_1}, \ldots 0^{a_n}) = \beta(\neg a_1,\ldots,\neg a_n) = \beta(1^{a_1}, \ldots, 1^{a_n}) = \beta(g_1(1),\ldots, g_n(1)) = g(1).$$

Определение 3. Если $a_j \le b_j$ для всех j = 1,…, n, то полагают, что выполнено отношение предшествования $a \le b$ для двух наборов $a = (a_1,\ldots,a_n)$ и $b = (b_1,\ldots,b_n)$.

Очевидно, что если $a \le b$ и $b \le c$, то $a \le c$.

Множество векторов из булевого куба B^n частично упорядочено отношением предшествования ≤, где B = {0,1}, а n – натуральное число.

Определение 4. Функция алгебры логики $\beta(x_1,\ldots,x_n)$ называется монотонной, если для всяких двух булевых векторов $a \le b$ из B^n выполняется неравенство $\beta(a) \le \beta(b)$.

Например, функции 0, 1, x, x&y, x V y монотонны.

Проверим, что класс M всех монотонных функций замкнут. Функция x принадлежит M. Если $\beta, \beta_1,\ldots, \beta_m$ принадлежат M и $\tilde{a}_j \le \tilde{b}_j$ при j = 1,…, m, то

$$\beta(\beta_1(\tilde{a}_1),\ldots, \beta_m(\tilde{a}_m)) \le \beta(\beta_1(\tilde{b}_1),\ldots, \beta_m(\tilde{b}_m)),$$

так как $\beta_j(\tilde{a}_j) \le \beta_j(\tilde{b}_j)$ для всех j = 1,…, m,

$$\beta(\tilde{a}) \le \beta(\tilde{b}),$$

где $\tilde{a} = (a_1,\ldots,a_n)$, $\tilde{a}_j = (a_{j1},\ldots,a_{j\,p(j)})$, $a_{ji} \in \{a_1,\ldots,a_n\}$, $a_k \le b_k$ для всех k = 1,…, n, β_j зависит от p(j) переменных.

Итак, где $\Phi = \beta(\beta_1,\ldots, \beta_m)$, следовательно, класс M монотонен.

Наборы $\tilde{a}$ и $\tilde{b}$ называются соседними, если

$$\tilde{a} = (a_1,\ldots,a_{i-1}, a_i, a_{i+1}, \ldots, a_n),$$

$$\tilde{b} = (a_1,\ldots,a_{i-1}, \neg a_i, a_{i+1}, \ldots, a_n),$$

где $\overline{a}_i = \neg a_i$.

Лемма 2. Пусть $\beta(x_1,\ldots,x_n)$ – немонотонная функция. Тогда имеется подстановка постоянных 0 и 1, и функции x такая, что из β получается функция ¬x.

Доказательство. Поскольку $\beta \notin M$, то существуют наборы $\tilde{a}_1$ и $\tilde{b}_1$, для которых $\tilde{a}_1 \leq \tilde{b}_1$ и $\beta(\tilde{a}_1) > \beta(\tilde{b}_1)$. Если наборы соседние, то получаем это неравенство для соседних наборов.

В общем случае, если $\tilde{a}_1$ и $\tilde{b}_1$ не соседние наборы, то они отличаются в k координатах, причем в $\tilde{a}_1$ эти координаты нулевые, а в $\tilde{b}_1$ они единичные, следовательно, существует цепочка соседних наборов $\tilde{a}_2,\ldots, \tilde{a}_k$, удовлетворяющих неравенством

$$\tilde{a}_1 \leq \tilde{a}_2 \leq \ldots \leq \tilde{a}_k \leq \tilde{b}_1.$$

Поскольку $\beta(\tilde{a}_1) > \beta(\tilde{b}_1)$, то по крайней мере на одной паре соседних наборов в этой цепочке выполняется $\beta(\tilde{a}) > \beta(\tilde{b})$.

Эту пару соседних наборов обозначим $\tilde{a}$ и $\tilde{b}$, где $\tilde{a} \leq \tilde{b}$. Они имеют вид

$$\tilde{a} = (a_1,\ldots,a_{i-1}, 0, a_{i+1}, \ldots, a_n),$$

$$\tilde{b} = (a_1,\ldots,a_{i-1}, 1, a_{i+1}, \ldots, a_n).$$

Возьмем функцию $g(x) = \beta(a_1,\ldots,a_{i-1}, x, a_{i+1}, \ldots, a_n)$, тогда $g(0) = \beta(\tilde{a}) > \beta(\tilde{b}) = g(1)$, следовательно, $g(0) = 1$, $g(1) = 0$. Таким образом $g(x) = \neg x$.

7. Лекция 7.Распознавание полноты.

Теорема 1.

Если система B полна в P_k, то существует конечная подсистема C в B, которая также является полной.

Доказательство.

Рассмотрим систему $B = \{f_1, f_2, \dots, f_n, \dots\}$. Поскольку $[B] = P_k$, то функция Вебба V_k выражается через функции системы B при помощи некоторой формулы $V_k(x_1, x_2) = U[f_{i_1}, \dots, f_{i_m}]$.

В силу теоремы 3 второго параграфа подсистема $\{f_{i_1}, \dots, f_{i_m}\}$ является искомой.

Теорема (о распознавании полноты) 2.

Для распознавания полноты существует алгоритм.

Доказательство.

Если задана система функций B в P_k, то согласно теореме 1 без ограничения общности можно предполагать, что B конечна, $B = \{f_1, \dots, f_q\}$.

Для описания алгоритма распознавания полноты сначала строится по индукции последовательность множеств $K_0, K_1, \dots, K_m, \dots$ функций от двух переменных x_1 и x_2.

В качестве базиса индукции полагают $K_0 = Л$, где Л обозначает пустое множество. Далее совершается индуктивный переход. Предположим, что уже построены множества $K_0, \dots, K_m$, где $K_m = \{h_1(x_1, x_2), \dots, h_{p_{(m)}}(x_1, x_2)\}$, $p(0) = 0$.

Рассмотрим всевозможные формулы типа $f_i(H_1(x_1, x_2), \dots, H_n(x_1, x_2))$, где либо $H_b(x_1, x_2) = h_{j(b)}(x_1, x_2)$, либо $H_b(x_1, x_2) = g^2_{i(b)}(x_1, x_2), j(b) \in \{1, \dots, p(m)\}$, $g^n_i(x_1, \dots, x_n) = x_i$, $i \in \{1, \dots, n\}$, $i(b) \in \{1,2\}$.

Так получается $q(p(m) + 2)^n$ формул. Если среди них получаются функции не вошедшие в K_m, то обозначим их $h_{p(m)+1}(x_1, x_2), \dots, h_{p(m+1)}(x_1, x_2)$ и зададим $K_{m+1} = K_m \cup \{h_{p(m)+1}(x_1, x_2), \dots, h_{p(m+1)}(x_1, x_2)\}$. Таким образом получается последовательность семейств функций $K_0 \subseteq K_1 \subseteq \cdots \subseteq K_m \subseteq \cdots$. Если для некоторого m выполняется равенство $K_{m+1} = K_m$, то $K_{m+i} = K_m$ для любого положительного целого числа i.

В силу теоремы 1 из §1 мощность K_i не превосходит K^{K^2} для любого i. Поэтому существует минимальное m, начиная с которого наступает стабилизация $K_{m+1} = K_m$, обозначим его S.

Тогда имеются два варианта.

1. Если K_S содержит все функции двух переменных, то функция Вебба $V_k(x_1, x_2)$ принадлежит K_S, следовательно, исходная система функций полна.

2. Если K_S не содержит всех функций двух переменных, то $[B]$ не содержит всех функций от переменных x_1 и x_2, так как $[B]_{x_1,x_2} = K_S$.
   Таким образом алгоритм состоит в том, что строится последовательность классов $K_0, \dots, K_m, \dots$ до момента стабилизации S и рассматривается класс K_S. По полученному классу K_S определяется полнота.
   Далее описывается второй подход ля решения вопроса о полноте системы функций. Обозначим через R_p некоторый класс функций $h_i(x_1, \dots, x_p)$ из P_k, зависящих от p переменных, причём такой, чтобы все функции g_i^p принадлежали R_p, где $g_i^p(x_1, \dots, x_p) = x_i,\ i = 1, \dots, p$.
   Определение 1.
   Если для любых функций $h_{j(1)}(x_1, \dots, x_p), \dots, h_{j(n)}(x_1, \dots, x_p)$ из R_p выполняется включение $f(h_{j(1)}(x_1, \dots, x_p), \dots, h_{j(n)}(x_1, \dots, x_p)) \in R_p$, то говорят, что функция $f(x_1, \dots, x_n)$ сохраняет множество R_p. Класс всех таких функций f сохраняющих R_p обозначим через $M = M(R_p)$.
   Пример 1.
   Положим $k = 2, p = 1, R_1 = \{x, \bar{x}\}$. Функции, сохраняющие множество R_1, удовлетворяют равенству
   $f(y^{\delta_1}, \dots, y^{\delta_n}) = y^{\delta}$, где

$$y^{\delta} = \begin{cases} y \text{ при } \delta = 1 \\ \bar{y} \text{ при } \delta = 0 \end{cases}.$$

В частности, $M(R_1)$ содержит самодвойственные функции $f(\overline{x_1}, \dots, \overline{x_n}) = \bar{f}(x_1, \dots, x_n)$. Поскольку $o^{\delta} = \bar{\delta}, 1^{\delta} = \delta$, то $f(0^{\delta_1}, \dots, 0^{\delta_n}) = f(\overline{\delta_1}, \dots, \overline{\delta_n}) = 0^{\delta} = \bar{\delta}$, $f(1^{\delta_1}, \dots, 1^{\delta_n}) = f(\delta_1, \dots, \delta_n) = 1^{\delta} = \delta$, следовательно, $f(\overline{\delta_1}, \dots, \overline{\delta_n}) = \bar{f}(\delta_1, \dots, \delta_n)$.

Таким образом, $M(R_1)$ совпадает с классом S самодвойственных функций.

Лемма 1.

$[M] = M$ для $M = M(R_p)$, где $M(R_p)$ и R_p даются определением 1.

Доказательство.

Из определения 1 следует, что M содержит тождественную функцию $id(x_i) = x_i$, так как $id\left(g_i^p(x_i, \dots, x_i)\right) = x_i = g_i^p(x_i, .., x_i)$. Пусть $f, f_1, \dots, f_m$ принадлежат M. Возьмём их композицию $\Phi = f(f_1, \dots, f_m)$. Тогда Φ зависит от конечного числа n переменных. Если $h_{i(1)}, \dots, h_{i(n)}$ принадлежат R_p, то $\Phi(h_{i(1)}, \dots, h_{i(n)}) = \Phi\left(id(h_{i(1)}), \dots, id(h_{i(n)})\right) = f(f_1(h_{i(1)}, \dots, h_{i(n)}), \dots, f_m(h_{i(1)}, \dots, h_{i(m)}) = f(G_1, \dots, G_m)$, где $G_j = f_j(h_{i(1)}, \dots, h_{i(n)}) \in R_p$ для любого $j = 1, \dots, m$.

Поэтому $f(G_1, \dots, G_m) \in R_p$.

Лемма 2.

Пусть класс R_p удовлетворяет ограничению $[R_p]_{x_1,\dots,x_p} = R_p$. Тогда для класса $M = M(R_p)$ выполняется равенство $M_{x_1,\dots,x_p} = R_p$.

Доказательство.

Возьмём $g(x_1, \dots, x_p) \in R_p$, $h_{j_1}, \dots, h_{j_p} \in R_p$, тогда $y(h_{j_1}, \dots, h_{j_p}) \in R_p$ так как в силу условия леммы $R_p = [R_p]_{x_1,\dots,x_p}$. Если же $f(x_1, \dots, x_p) \in M_{x_1,\dots,x_p}$, то подставим функции $g_1^p, \dots, g_p^p$ вместо $x_1, \dots, x_p$ и получим $f(g_1^p, \dots, g_p^p) \in R_p$, то есть $f(x_1, \dots, x_p) \in R_p$.

8. Лекция 8. Теорема (о функциональной полноте А.В.Кузнецова).

Существует алгоритм построения системы замкнутых классов $M_1, M_2, \dots, M_S$ в P_k таких, что M_i не содержит целиком M_j для любых $i \neq j$. Более того, подсистема функций B из P_k полна тогда и только тогда, когда она целиком не содержится ни в одном из классов $M_1, \dots, M_S$.

Доказательство.

Сначала строится система классов $M_1, \dots, M_S$. Для этого берётся система $K_1, \dots, K_l$ собственных подмножеств из P_k состоящих из функций двух переменных x_1 и x_2, удовлетворяющих условиям (1) и (2) для любого $i = 1, \dots l$:

(1) K_i содержит g_1^2 и g_2^2
(2) $[K_i]_{x_1,x_2} = K_i$

Для построения подмножеств K_i просматривают все собственные подмножества в $P_k(x_1, x_2)$ функций зависящих от двух переменных x_1 и x_2 из P_k . Это возможно, так как мощность $P_k^{(x_1,x_2)}$ равна $\mathrm{c} = k^{k^2}$, а мощность семейства всех конечных подмножеств в $P_k(x_1, x_2)$ равна 2^{c}.

Среди просматриваемых подмножеств оставляют те подмножества, которые содержат g_1^2 и g_2^2. Из оставшихся подмножеств K отбирают те K, для которых $[K]_{x_1,x_2} = K$. Последнее условие проверяют аналогично тому, как это делается в доказательстве теоремы 2 о распознавании полноты.

Рассмотрим $L_i = M(K_i)$, где $i = 1, \dots, l$, классы L_i сохранения K_i.

Из лемм 1 и 2 следует, что класс L_i замкнут и $(L_i)_{x_1,x_2} = K_i$. Отсюда вытекает, что классы L_i все попарно различны и не полны в P_k.

Из этого списка удаляют те классы, которые целиком содержатся в каком-либо L_i. В результате получается система $M_1, \dots, M_S$. Пусть подсистема B функций из P_k полна. Тогда B не содержится целиком ни одном из M_i, так как M_i – замкнутый и неполный класс. Пусть теперь подсистема B функций из P_k не содержится целиком ни в одном из классов M_i, где $i = 1, \dots, l$. Поскольку каждый класс M_i замкнут, то без ограничения общности достаточно рассмотреть случай, когда класс B замкнут. Положим $C = [B \cup \{g_1, g_2\}]$. Тогда $C = B \cup [\{g_1, g_2\}]$. Классы C и B одновременно либо полны, либо неполны, так как функция Вебба $V_k(x_1, x_2)$ либо входит в B и C, либо не содержится ни в одном из этих классов. Рассмотрим $D = C_{x_1,x_2}$. Докажем, что D содержит все функции, зависящие от двух переменных x_1 и x_2. Если бы это было не так, то $D = K_i$ и $C \subseteq L_i \subseteq M_j$ для некоторых i и j. Из $B \subseteq C$ следует, что $B \subseteq M_j$.

Получилось противоречие с условием теоремы. Итак D, а значит и C, содержит функцию Вебба $V_k(x_1, x_2)$. Отсюда следует, что класс C и класс B полны.

Замечание.

Указания в доказательстве теоремы процедура практически трудно осуществима из-за громоздких вычислений.

Система $B = \{V_k(x_1, x_2)\}$ является полной. В самом деле, в силу теоремы 3 система $B = \{\bar{x}, \max(x_1, x_2)\}$ является полной.

$\bar{x} = \max(x, x) + 1 = x + 1 \Rightarrow \bar{x} \in \Phi\{V_k(x_1, x_2)\}$

$\max(x_1, x_2) = \big(V_k(x_1, x_2)\big)^{-(k-1)} = \varphi_{k-1}\big(V_k(x_1, x_2)\big)$, где $\varphi_1(x) = \bar{x}$, $\varphi_2(x) = \bar{\bar{x}} = \varphi_1\big(\varphi_1(x)\big), \dots, \varphi_{k-1}(x) = \varphi_{k-2}\big(\varphi_1(x)\big).\Rightarrow$

$\max(x_1, x_2) \in \Phi\{V_k(x_1, x_2)\}$, Φ обозначает множество формул, полученных из данных в $\{.\}$.

9. Лекция 9. Существенные функции и критерий полноты.

Функцию $f(x_1, \dots, x_n)$ из P_k называется существенной, если она существенно зависит не менее чем от двух переменных.

Лемма 1.
Пусть $f(x_1, \dots, x_n)$ – существенная функция, которая принимает p значений, где $p \geq 3$. Если $x_1 -$ существенная переменная функции f, то существует два таких набора $(a_1, \dots, a_n)$ и $(b, a_2, \dots, a_n)$, что $f(a_1, \dots, a_n) \neq f(b, a_2, \dots, a_n)$ и функция $f(a_1, x_2, \dots, x_n)$ принимает значения отличное и от $f(a_1, \dots, a_n)$ и $f(b, a_2, \dots, a_n)$.

Доказательство:
Поскольку $x_1 -$ существенная переменная функции f, то существуют значения $\alpha_2, \dots, \alpha_n$ такие, что среди значений функции $f(1, \alpha_2, \dots, \alpha_n), \dots, f(k-1, \alpha_2, \dots, \alpha_n)$ имеется не менее двух различных.

Вариант 1.
В рассматриваемой последовательности содержатся не все l значений. Возьмем набор $(\alpha, \gamma_2, \dots, \gamma_n)$ такой, что значение $f(\alpha, \gamma_2, \dots, \gamma_n)$ не встречается среди $f(i, \alpha_2, \dots, \alpha_n)$, где $i = 0,1, \dots, k-1$.
Тогда $f(\alpha, \alpha_2, \dots, \alpha_n) \neq f(\alpha, \gamma_2, \dots, \gamma_n)$. В качестве β возьмем то, для которого $f(\beta, \alpha_2, \dots, \alpha_n) \neq f(\alpha, \alpha_2, \dots, \alpha_n)$. При этом выполняется также неравенство $f(\beta, \alpha_2, \dots, \alpha_n) \neq f(\alpha, \gamma_2, \dots, \gamma_n)$.

Вариант 2.
Среди $f(i, \alpha_2, \dots, \alpha_n)$,где $i = 0,1, \dots, k-1$, встречаются все l значения. Поскольку функция f существенна, то существует α такое, что $f(\alpha, x_2, \dots, x_n) \neq const$, иначе бы f существенно зависела лишь от одной переменной. Поэтому имеются наборы $(\alpha, \alpha_2, \dots, \alpha_n)$ и $(\alpha, \gamma_2, \dots, \gamma_n)$, для которых $f(\alpha, \alpha_2, \dots, \alpha_n) \neq f(\alpha, \gamma_2, \dots, \gamma_n)$. Тогда существует β такое, что $f(\beta, \alpha_2, \dots, \alpha_n) \neq f(\alpha, \alpha_2, \dots, \alpha_n)$ и $f(\beta, \alpha_2, \dots, \alpha_n) \neq f(\alpha, \gamma_2, \dots, \gamma_n)$, так как $l \geq 3$.
Для конечного множества G через $|G|$ обозначается число элементов в G.

Лемма 2.
Пусть $f -$ существенная функция n переменных k-значной логики, которая принимает l различных значений с $l \geq 3$. Тогда существуют n подмножеств $G_1, \dots, G_n$ множества $E_k = \{0,1, \dots, k-1\}$такие , что $1 \leq |G_1|, \dots, |G_n| \leq l-1$ $\alpha_i \in G_i, i = 1, \dots, n$, функция f принимает l значений.

Доказательство:
После перенумерации переменных без ограничения общности можно предполагать, что x_1 – существенная переменная функции f. В силу предыдущей леммы существуют три различных набора $(\alpha, \alpha_2, \dots, \alpha_n)$,
$(\beta, \alpha_2, \dots, \alpha_n), (\alpha, \gamma_2, \dots, \gamma_n)$, на которых функция f принимает три различных значения. К этим наборам добавим еще $l-3$, набора $\left(S_1^{(i)}, \dots, S_n^{(i)}\right)$, $i = 1, \dots, l-3$, на которых функция f принимает остальные $l-3$ значений. Зададим множества:
$G_1 = \left\{\alpha, \beta, S_1^{(1)}, \dots, S_1^{(l-3)}\right\}$,
$G_2 = \left\{\alpha_2, \gamma_2, S_2^{(1)}, \dots, S_2^{(l-3)}\right\}$,
….
$G_1 = \left\{\alpha_n, \gamma_n, S_n^{(1)}, \dots, S_n^{(l-3)}\right\}$.
Эти множества удовлетворяют условиям леммы.

Определение1. Если задана система наборов:
$(\alpha_1, \dots, \alpha_{i-1,}\alpha_{i,}\alpha_{i+1}, \dots, \alpha_{j-1}, \alpha_j, \alpha_{j+1}, \dots, \alpha_n)$,
$(\alpha_1, \dots, \alpha_{i-1,}\beta_{i,}\alpha_{i+1}, \dots, \alpha_{j-1}, \alpha_j, \alpha_{j+1}, \dots, \alpha_n)$,
$(\alpha_1, \dots, \alpha_{i-1,}\beta_{i,}\alpha_{i+1}, \dots, \alpha_{j-1}, \beta_j, \alpha_{j+1}, \dots, \alpha_n)$,
$(\alpha_1, \dots, \alpha_{i-1,}\alpha_{i,}\alpha_{i+1}, \dots, \alpha_{j-1}, \beta_j, \alpha_{j+1}, \dots, \alpha_n)$,
удовлетворяющая условиям $\alpha_i \neq \beta_i$ и $\alpha_j \neq \beta_j$, то она называется квадратом.

Лемма 3.
Если $f(x_1, \dots, x_n)$ –существенная функция, принимающая l значений, где $l \geq 3$, то имеется квадрат, на котором f принимает либо более двух значений, либо два значения и одно из них только в одной точке.

Доказательство.
Используя перенумерацию переменных, без ограничения общности можно полагать, что x_1 – существенная переменная функции f. В силу леммы 1 имеются три набора $(\alpha, \alpha_2, \dots, \alpha_n), (\beta, \alpha_2, \dots, \alpha_n)$ и $(\alpha, \gamma_2, \dots, \gamma_n)$, на которых функция f принимает три различных значения. Рассмотрим куб $Q = \{\alpha, \beta\} \times \{\alpha_2, \gamma_2\} \times \dots \times \{\alpha_n, \gamma_n\}$, содержащий эти наборы. Возьмем сечения этого куба гиперплоскостью $x_1 = \alpha$ и в этом сечении соединим точку $(\alpha_2, \dots, \alpha_n)$ с точкой $(\gamma_2, \dots, \gamma_n)$ цепочкой рёбер, принадлежащих этому сечению. При этом ЧТОТО ребро представляет собой отрезок, соединяющий пару соседних наборов, то есть набор отличающихся значениями ровно одной координаты. Это сечение рёбер определяет соответствующую цепочку

рёбер в гиперплоскости $x_1 = \beta$ с помощью проекции на неё. Тогда получаются пары соответствующих рёбер в гиперплоскостях $x_1 = \alpha$ и $x_1 = \beta$, образующие квадраты. Таким образом, получается цепочка квадратов R_1,…, R_j образованных посредством соединения последовательных вершин от

$$(\alpha, \alpha_2, \dots, \alpha_n) \text{ до } (\alpha, \gamma_2, \dots, \gamma_n)$$

в верхней плоскости задаваемой условием

$$x_1 = \alpha$$

и соответствующих последовательных вершин от

$$(\beta, \alpha_2, \dots, \alpha_n) \quad \text{до} \quad (\beta, \gamma_2, \dots, \gamma_n)$$

в нижней плоскости задаваемой уравнением

$$x_1 = \beta.$$

Поэтому на ребре R_1 первого квадрата функция f принимает значения $f(\alpha, \alpha_2, \dots, \alpha_n)$и $f(\beta, \alpha_2, \dots, \alpha_n)$, тогда как на последнем ребре R_j функция f не принимает хотя бы одно из этих значений. Отсюда вытекает, что существует квадрат с номером i, где $1 \le i < j$, такой, что на ребре R_i функция f принимает значения $f(\alpha, \alpha_2, \dots, \alpha_n)$ и $f(\beta, \alpha_2, \dots, \alpha_n)$, а на ребре R_{i+1} не принимает по крайней мере одно из этих значений. Таким образом, квадрат с номером i является искомым.

Замечание 1.

При $k = 2$ леммы 1 и 2 не имеют смысл, а лемма 3 не выполняется. Например, при $k = 2$ функция $f(x_1, x_2) = x_1 \oplus x_2$ принимает два значения, и оба с кратностью два, на квадрате, который является её областью определения.

Теория 1.

Пусть $k \ge 3$, а система B функций из f_k содержит все функции одной переменной, принимающие не более $k - 1$ значений. Тогда следующие условия равносильны:

(1) B полна

(2) B содержит существенную функцию $f(x_1, \dots, x_n)$, принимающую все K значений.

Доказательство(1) ⇒ (2).

Рассмотрим полную систему и предположим, что B не содержит существенной функции, принимающей все k значений. Тогда из B нельзя получить существенную функцию, принимающую все k значений. Получилось противоречие. Поэтому B содержит существенную функцию, принимающую все k значений.

(2) ⇒ (1).

Пусть выполняется условие теоремы вместе со свойством (2). Докажем по индукции, что система B полна.

1) Базис индукции. Сначала докажем, что из B можно получить все функции из P_k, принимающие два значения.
Функция f, удовлетворяющая свойству (2), в силу леммы 3 принимает не менее двух значений на некотором квадрате, причём одно из них, Z_1 – в одной точке.

$$(\alpha_1, \dots, \alpha_{i-1}, \alpha_i, \alpha_{i+1}, \dots, \alpha_{j-1}, \alpha_j, \alpha_{j+1}, \dots, \alpha_n)$$
$$(\alpha_1, \dots, \alpha_{i-1}, \beta_i, \alpha_{i+1}, \dots, \alpha_{j-1}, \alpha_j, \alpha_{j+1}, \dots, \alpha_n)$$
$$(\alpha_1, \dots, \alpha_{i-1}, \beta_i, \alpha_{i+1}, \dots, \alpha_{j-1}, \beta_j, \alpha_{j+1}, \dots, \alpha_n)$$
$$(\alpha_1, \dots, \alpha_{i-1}, \alpha_i, \alpha_{i+1}, \dots, \alpha_{j-1}, \beta_j, \alpha_{j+1}, \dots, \alpha_n),$$

где $\alpha_i \neq \beta_i$ и $\alpha_j \neq \beta_j, i <$j.
Выберем функцию y из B такую, что

$$y(x) = \begin{cases} 0 \text{ при } x = z \\ 1 \text{ при } x \neq z \end{cases}.$$

В самом деле, функция y удовлетворяет условию теоремы, следовательно, принадлежит B. Зададим функцию

$$g(x_1, x_2) = y(f(\alpha_1, \dots, \alpha_{i-1}, x_1, \alpha_{i+1}, \dots, \alpha_{j-1}, x_2, \alpha_{j+1}, \dots, \alpha_n)).$$

Данная функция $g(x_1, x_2)$ принимает два значения 0 и 1 на квадрате $\{(\alpha_i, \alpha_j), (\beta_i, \alpha_j), (\beta_i, \beta_j), (\alpha_i, \beta_j)\}$. Значение 0 она принимает лишь в одной точке этого квадрата. Обозначим эту точку (a_1, a_2).
Далее воспользуемся нормировкой функции g. В общем случае, если функция $h(x_1, \dots, x_n)$ из P_k задана на $G = G_1 \times \dots \times G_n$, то переход к функции $h'(x_1, \dots, x_n)$ на $G' = G_1' \times \dots \times G_n'$ называется нормировкой. То есть $h'(x_1, \dots, x_n) = \Psi\left(h\left(\Psi_1(x_1), \dots, \Psi_n(x_n)\right)\right)$.
При этом преобразовании предполагается, что $|G_1'| = |G_1| = l_1, \dots, |G_n'| = |G_n| = l_n$. Пусть $h(G) = \{b_0, \dots, b_{l-1}\}, h'(G') = \{b_0', \dots, b_{l-1}'\}$. Таким образом, нормировка указывает взаимно однозначные соответствия между множествами
$\{b_0, \dots, b_{l-1}\} \leftrightarrow \{b_0', \dots, b_{l-1}'\}$,
$G_1 \leftrightarrow G_1', \dots, G_n \leftrightarrow G_n'$.
Рассматриваемые соответствия осуществляются с помощью функций $\Psi(\text{x}), \Psi_1(x), \dots, \Psi_n(x)$из f_k. Эти функции всегда можно выбрать как функции подстановок. В частности, если $l, l_1, \dots, l_n \leq k - 1$, то $\Psi(\text{x}), \Psi_1(x), \dots, \Psi_n(x)$ можно выбрать как функции одной переменной, принимающие не более $k - 1$ значений функции f' на E' совпадают с кратностями соответствующих значений функции f на множестве E.
Удобно использовать нормировку в следующих формах:

(1) $\{b_0, \dots, b_{l-1}\} = \{b_0', \dots, b_{l-1}'\}$,
$\Psi(\text{x}) = x$, $G_i' = \{0, \dots, l_{i-1}\}$, при $i = 1, .., n$

(2) $G_i = G_i', \Psi_i(\text{x}) = x$ при $i = 1, \dots, n$;

$\{b'_0, \dots, b'_{l-1}\} = \{0, \dots, l-1\};$

(3) $\{b'_0, \dots, b'_{l-1}\} = \{0, \dots, l-1\},$

$G'_i = \{0, \dots, l-1\}$ при $i = 1, \dots, n.$

Случай (1) соответствует преобразованию переменных, а вариант (2) даёт преобразование значений. Варианты (1) и (2) называют также неполными нормировками. Часто в качестве фиксированных точек b'_j и $(a'_1, \dots, a'_n)$ выбирают 0 и $(0, \dots, 0)$ соответственно. В свете описания нормировки функций выберем неполную нормировку, при которой (0,0) отображается в (a_1, a_2), а квадрат $\{(0,0), (1,0), (1,1), (1,0)\}$ в $\{(\alpha_i, \alpha_j), (\beta_i, \alpha_j), (\beta_i, \beta_j), (\alpha_i, \beta_j)\}$.

Тогда приходим к функции

$g'(x_1, x_2) = g(\Psi_1(x_1), \Psi_2(x_2))$, где $\Psi_1 \in B, \Psi_2 \in B$. Очевидно, что $g'(x_1, x_2)$ совпадает с функцией $\max(x_1, x_2)$ на множестве $\{0,1\} \times \{0,1\}$. Обозначим её $x_1 \vee_{01} x_2$. Поскольку система B содержит функции $j_i(x)$такие что $j_i(x) = \begin{cases} 1 \text{ при } x = i \\ 0 \text{ при } x \neq i \end{cases}$, то $x_1 \&_{01} x_2 = j_0(j_0(x_1) \vee_{01} j_0(x_2))$ совпадает с функцией $\min(x_1, x_2)$ на множестве $\{0,1\} \times \{0,1\}$. Для произвольной функции $h(x_1, \dots, x_m)$, принимающей два значения 0 и 1 выполняется разложение $h(x_1, \dots, x_m) = \vee d_{\delta_1}(x_1) \& \dots \& j_{\delta_m}(x_m) \& h(\delta_1, .., \delta_m) \cdot (\delta_1, .., \delta_m) = \vee_{01} j_{\delta_1}(x_1) \&_{01} \dots \&_{01} j_{\delta_m}(x_m) \&_{01} h(\delta_1, .., \delta_m) \cdot (\delta_1, .., \delta_m)$

Это означает, что функция h получается из системы функций B. Поскольку система B содержит все функции одной переменной, принимающие произвольные два значения из E_k, то можно получить из системы B все функции, которые принимают любые два значения из $E_k = \{0,1, \dots, k-1\}$.

(2). Продолжение индукции. Предположим, что из системы B построены все функции, принимающие не более $l-1$ значений, где $l-1 < k$. Укажем процедуру для построения всех функций k-значной логики и принимающих l значений. Рассмотрим функцию $f(x_1, \dots, x_n)$. Из леммы 2 вытекает, что существуют n подмножеств $G_1, \dots, G_n$ в E_k таких, что $|G_i| \leq l-1$ при i=1,…,n, а на их декартовом произведении $G = G_1 \times \dots \times G_n$ функция f принимает l значений $b_0, b_1, \dots, b_{l-1}$. Обозначим наборы, на которых принимаются эти значения через

$$a^{(m)} = \left(a_1^{(m)}, \dots, a_n^{(m)}\right),$$

$$f\left(a^{(m)}\right) = b_m, \text{где } m = 0, \dots, l-1.$$

Далее укажем как из системы B можно построить произвольную функцию $h(x_1, \dots, x_m)$, которая принимает значения $b_0, \dots, b_{l-1}$. Положим $h(\delta_1, \dots, \delta_m) = b_i(\delta)$, где $\delta =$

$(\delta_1, \dots, \delta_m), \Psi_j(\delta_1, \dots, \delta_m) = a_j^{(i(r))}$. Поэтому $h(x_1, \dots, x_m) = f(\Psi_1(x_1, \dots, x_m), \dots, \Psi_n(x_1, \dots, x_m))$,
так как $f\left(\Psi_1(\delta_1, \dots, \delta_m), \dots, \Psi_n(\delta_1, \dots, \delta_m)\right) = f\left(a_1^{(i(\delta))}, \dots, a_n^{(i(\delta))}\right) = b_{i(\delta)}$.
Если уже построены все функции с заданными l значениями $b_0, \dots, b_{l-1}$ при $l < k$, то можно получить остальные функции с l значениями при помощи функций одной переменной, принимающих менее k значений.
Применение такой описанной выше процедуры индукцией по l в конце даёт случай $l = k$. Тогда все функции из P_k построены.

Следствие 1(Критерий Слупецкого).
Пусть система B функций из P_k содержит все функции одной переменной, где $k \geq 3$. Тогда следующие условия равносильны:
(1) Система полна,
(2) B содержит существенную функцию, принимающую все k значений.

Замечание 2.
Условие $k \geq 3$ в предыдущей теореме существенно. При $k = 2$ система функций $B = \{0,1, x, \neg x, x_1 + x_2\}$ неполная, т.к. B содержится в классе L линейных функций.
Для упрощения вычислений в формулировке теоремы /и последствия/ можно заменить условие «B содержит все функции одной переменной» на «B нерождает все функции одной переменной». Очевидно, что они тогда тоже будут выполняться. Последнее условие может быть установлено проще, чем первое. Для этого полезны тоже две теоремы.

Теорема 2(с.Пикар). Три функции
$f(x) = x - 1 \ (mod\ k)$,

$$g(x) = \begin{cases} x, \ \ 0 \leq x \leq k-3, \\ k = 1, \text{при } x = k-2 \\ k = 2, \text{при } x = k-1 \end{cases},$$

$$h(x) = \begin{cases} 1, \text{при } x = 0, \\ x, \text{при } x \neq - \end{cases}$$
Порождают все функции одной переменной из P_k , где $k \geq 3$.

Теорема 3. Система k функций.

$$f_i(x) = \begin{cases} i, \text{при } x = 0, \\ 0, \text{при } x = i, \\ x, \text{при } x \neq 0 \text{ и } x \neq i \end{cases},$$
где $i = 1, \dots, k-1$, вместе с функцией

$$h(x) = \begin{cases} 1, \text{при } x = 0, \\ x, \text{при } x \neq 0 \end{cases}$$
порождают все функции одной переменной из P_k , где $k \geq 3$.

Определение 2.
Функция из P_k называется функцией Шеффера, если она образует полную систему.

Теорема 4.
Пусть $f(x_1, \dots, x_n) -$ функция k-значной логики, где $k \geq 3$. Тогда следующие условия равносильны:
(1) $f -$является функцией Шеффера,
(2) f порождает все функции одной переменной, принимающие не более$k - 1$ значений.

Доказательство.$(1) \Rightarrow (2)$ очевидно
$(2) \Rightarrow (1)$. Из функции f в честном случае можно получить все постоянные $0,1, \dots, k-1$, то есть f принимает все k значений. Если $f -$ несущественная функция, то $f -$подстановка. В этом случае из неё можно получить лишь подстановки в результате композиций, а значит нельзя получать постоянных функций. Остается лишь тот вариант, что $f -$ существенная переменная. В силу теоремы 1 (критерия полноты) система $B = \{f(x_1, \dots, x_n)\}$ полна.

10. Лекция 10. Особенности k −значных логик.

Теорема 1(Янов). Для любого $k \geq 3$ в P_k имеется замкнутый класс, не обладающий базисом.
Доказательство.
Возьмем последовательность функций $f_0 = 0$,
$$f_i(x_1, \dots, x_i) = \begin{cases} 1, \text{при } x_1 = 2, \dots, x_i = 2 \\ 0, \text{в остальных случаях} \end{cases},$$
где $i = 1,2, \dots$. Рассмотрим множество M_k всех функций, которые получаются из $\{f_0, f_1, f_2, \dots\}$ путем переименования переменных, но без отожествления переменных. Очевидно, что класс M_k замкнут. Предположим, что в классе M_k существует базис. Тогда в базисе имеется функция g, которая получается из некоторой функции

$f_n \in M_k$ в результате переименования переменных. Среди таких g и f_n выберем пару с минимальным n_0. Представляются две возможности.

1.Имеется ещё одна функция g_2 в базисе. Ей соответствует некоторая функция f_{n_1} с $n_1 > n_0$. Но f_{n_0} можно получить из f_{n_1} с помощью отожествления переменных. Это противоречит определению базиса.

2. Имеется лишь одна функция g в базисе. Тогда f_n при $n > n_0$ нельзя получить g, так как $g(\dots, g, \dots) \equiv 0$. Опять получилось противоречие. Остается только то, что M_k не имеет базиса.

Теорема 2(Мучник).
Для любого $k \geq 3$ в P_k имеется замкнутый класс со счётным базисом.

Доказательство.
Зададим последовательность функций:

$$f_i(x_1, \dots, x_i) = \begin{cases} 1, \text{при } x_1 = \dots = x_{j-1} = x_{j+1} = \dots = x_i = 2, x_j = 1, j = 1, \dots, i \\ 0, \text{в остальных случаях,} \end{cases}$$

где $i = 2,3, \dots$. Рассмотрим замкнутый класс M_k, порожденный системой $\{f_2, f_3, f_4, \dots\}$. Проверим, что система $\{f_2, f_3, f_4, \dots\}$ является базисом в M_k. Достаточно доказать, что невозможно представление

$f_m = U_m[f_2, \dots, f_{m-1}, f_{m+1}, \dots]$,

где $U_m -$ формула, ни при каком $m = 2,3,4, \dots$.

Формула $U_m[f_2, \dots, f_{m-1}, f_{m+1}, \dots] = f_r(B_1[f_2, \dots, f_{m-1}, f_{m+1}, \dots], \dots,$

$B_r[f_2, \dots, f_{m-1}, f_{m+1}, \dots])$.

Разберём все возникающие возможности.

1. Имеются не менее двух формул B_i и B_j отличных от переменных среди $B_1, \dots, B_r$, где $r \geq 2$. Поэтому на i-м и j-м местах у функции f_r при любых значениях $x_1, \dots, x_m$ от B_i и B_j возможны лишь значения 0 и 1, следовательно, правая часть равна нулю. Последнее противоречит тому, что $f_m \not\equiv 0$.

2. Лишь одна формула B_s отлична от переменной. Тогда остальные формулы сводятся к переменным. Поскольку $r \geq 2$, то имеется формула $B_p = x_q$. На наборе $x_1 = 2, \dots, x_{q-1} = 2, x_{q+1} = 2, \dots, x_m = 2,$ формула B_s принимает значения либо 0, либо 1. Тогда на p-м и s-м местах у функции f_r стоят значения отличные от 2 при таком выборе значений переменных. Поэтому правая часть примет значение 0, а левая часть на данном наборе равна 1. Получилось противоречие.

3. Все формулы $B_1, \dots, B_r$ совпадают с символами переменных. Тогда $r > m$, следовательно, имеется некоторая переменная x_p, которая не менее двух раз входит в формулу. Для набора $x_1 = 2, \dots, x_{p-1} = 2, x_{p+1} = 2, \dots, x_m = 2, x_p = 1$ левая часть обращается в 1, а правая в 0. Значит, этот случай тоже невозможен.

Теорема 5.
Для любого $k \geq 3$ имеется континуум различных замкнутых классов в P_k.
Доказательство.
Множество P_k всех функций k-значной логики счётно, следовательно, число подмножеств в P_k равно континууму. Поэтому, число замкнутых классов в P_k не превосходит континуум.
Остается оценить снизу число замкнутых классов в P_k. Воспользуемся замкнутым классом M_k из доказательства предыдущей теоремы. Он имеет базис $\{f_2, f_3, \dots\}$. Для произвольной последовательности $\{m_1, m_2, \dots\}$ с $2 \leq m_1 < m_2 < \cdots$ введём класс $M_k(m_1, m_2, \dots)$, порождённый системой функций $\{f_{m_1}, f_{m_2}, \dots\}$. Тогда $M_k(m_1, m_2, \dots) \neq M_k(m_1', m_2', \dots)$, если $(m_1, m_2, \dots) \neq (m_1', m_2', \dots)$, следовательно, семейство $\{M_k(m_1, m_2, \dots) : (m_1, m_2, \dots)\}$ имеет мощность континуума.

Теорема.
Система полиномов по $mod\ k$ является полной в P_k тогда и только тогда, когда k −простое число.
Доказательство.
Сначала представим произвольную функцию $f(x_1, \dots, x_n)$ из P_k в виде $f(x_1, \dots, x_n) = \Sigma j_{\delta_1}(x_1) \dots j_{\delta_n}(x_n) \cdot f(\delta_1, \dots, \delta_n) \ \ (mod\ k)$. Остается разложить в виде полиномов функции $j_l(x)$, где $l = 0, \dots, k-1$. Из тождества $j_b(x) = j_0(x - b)$ вытекает , что достаточно разложить $j_0(x)$ в виде полинома.

1.Предположим, что $k = p -$ простое число. Согласно малой теореме Ферма $a^{p-1} \equiv 1\ (mod\ p)$ для любого целого числа a такого, что $1 \leq a \leq p-1$. Поэтому, $j_0(x) = 1 - x^{p-1}\quad (mod\ p)$, следовательно, система полиномов полна в P_k.

Полезно провести путь доказательства малой теоремы Ферма. Числа $r_1 = a \cdot 1, r_2 = a \cdot 2, \dots, r_{p-1} = a \cdot (p-1)$ не сравнимы по $mod\ p$, следовательно, $r_1, \dots, r_{p-1} \equiv (p-1)!\,(mod\ p)$ и $(p-1)! \equiv a^{p-1}(p-1)!\,(mod\ p)$ или $1 \equiv a^{p-1}(mod\ p)$.

Другой способ разложения функции одной переменной $g(x)$ в виде полинома основан на методе неопределенных коэффициентов: $g(x) = a_0 + a_1 x + \dots + a_{p-1}x^{p-1}$.

Тогда получается система линейных алгебраических уравнений:

$a_0 + a_1 \cdot 0' + a_2 \cdot 0^2 + \dots + a_{p-1} \cdot 0^{p-1} = g(0),$

$a_0 + a_1 \cdot 1' + a_2 \cdot 1^2 + \dots + a_{p-1} \cdot 1^{p-1} = g(1),$

...

$a_0 + a_1 \cdot (p-1)' + a_2 \cdot (p-1)^2 + \dots + a_{p-1} \cdot (p-1)^{p-1} = g(p-1),$

У этой системы определитель:

$$\Delta = \begin{vmatrix} 1 & 0 & 0 & \dots & 0 \\ 1 & 1 & 1^2 & \dots & 1^{p-1} \\ 1 & 2 & 2^2 & \dots & 2^{p-1} \\ \dots & \dots & \dots & \dots & \dots \\ 1 & p-1 & (p-1)^2 & \dots & (p-1)^{p-1} \end{vmatrix}$$

называется определителем Вандермонда. Он вычислен $\Delta = \Pi(j-i)$, где $0 \leq i < j \leq p-1$.

Поскольку p −простое число , то $\Delta \not\equiv 0(mod\ p)$. Тогда в целых числах решаются сравнения $a_i\Delta \equiv \Delta_i\ (mod\ p)$ для любого $i = 0, \dots, p-1$, где Δ_i – соответствующий определитель, получаемый из Δ заменой $(i+1)$-го столбца на столбец правых частей уравнений. $\begin{pmatrix} g(0) \\ g(1) \\ g(2) \\ \cdot \\ g(p-1) \end{pmatrix}$.

В силу правила Крамера получается решение системы линейных алгебраических уравнений. Значит, $g(x)$ раскладывается в виде полинома.

2. Предположим теперь, что K не есть простое число. Тогда K раскладывается в виде произведения целых чисел $k = k_1 k_2$, где $k > k_2 > 1$. Предположим, что $j_0(x) = b_0 + b_1 x + \cdots + b_s x^s (mod\ k)$. Положив $x = 0$, мы получим $b_0 = 1$. При $x = k$, должно выполняться равенство:

$0 = 1 + b_1 k_1 + \cdots + b_s k_1^s\ (mod\ k)$ или

$k - 1 = b_1 k_1 + \cdots + b_s k_1^s (mod\ k)$.

Следовательно, $k - 1$ должно делиться на k_1. Но делимость k и $k - 1$ на k_1 возможна лишь при $k_1 = 1$. Получилось противоречие. Значит, при $k \neq p$ функция $j_0(x)$ не может быть представлена в виде полинома по $mod\ k$.

Замечание 1.

Последняя теорема обобщается на случай, когда E_k можно снабдить сложением $\oplus$ и умножением $\times$, относительно которых E_k образует поле. В алгебре доказывается, что конечное поле, называемое также полем Галуа, существует тогда и только тогда, когда $k = p^m$, где p −простое число, а m − натуральное число $m = 1,2, \ldots$.

С точностью до изоморфизма конечное поле F_{pm} определяется однозначно. Относительно сложения конечное поле образует абелеву группу характеристики p, то есть для любого $a \in F_{pm}$ для p-кратной суммы $a_1 \oplus \ldots \oplus a_p = 0$, где 0 −нуль группы, $a_j = a$ для всех $j = 1, \ldots, p$. Эту группу можно задать в виде векторов $(b_1, \ldots, b_m) = b$, где $b_j \in \{0, \ldots, p-1\}$ для всех $j = 1, \ldots, m$, $b \oplus c = (b_1 \oplus c_1, \ldots, b_m \oplus c_m)$, где $b_j \oplus c_j$ −сложение по $mod\ p$. Ненулевые элементы $F_{pm} \backslash \{0\} = F_{pm}^x$ образуют мультипликативную циклическую группу.

Например, при $k = 2^2$ возьмем образующую α циклической группы $F_{2^2}^x$. Тогда $\alpha \neq 1, \alpha^3 = 1, \alpha^2 \oplus \alpha \oplus 1 = 0$. При $\alpha = 2$ получается $\alpha^2 = \ominus (\alpha \oplus 1) = 3$, следовательно, $2 \cdot 3 = \alpha \cdot\cdot \alpha^2 = \alpha^3 = 1$, $3 \cdot 3 = \alpha^2 \cdot \alpha^2 = \alpha^4 = \alpha = 2$. Таблицы сложения и умножения в F_{2^2} таковы:

$\oplus$	0	1	2	3
0	0	1	2	3
1	1	0	3	2
2	2	3	0	1
3	3	2	1	0

X	0	1	2	3
0	0	0	0	0
1	0	1	2	3
2	0	2	3	1
3	0	3	1	2

Итак, каждая функция из P_k при $k = p^m$ может быть записана в виде полинома над конечным полем F_{pm}.

11. Лекция 11. Рекурсивные функции. Формализация вычислимости.

Машина Тьюринга позволяет точно описать понятие вычислимости частичной функции $f: \Sigma^* \to \Sigma^+$, где Σ обозначает алфавит. В частности, класс всех вычислимых по Тьюрингу функций $f: \mathrm{N}^n \to \mathrm{N}$ совпадает с $Com(\mathrm{N}^n, \mathrm{N})$, где $Com(\mathrm{N}^n, \mathrm{N})$ обозначает класс всех вычислимых в интуитивном смысле n-местных частичных функций $f: \mathrm{N}^n \to \mathrm{N}$.

Другой подход состоит в том, чтобы описать f формулой, где $F -$ некоторое математическое выражение, $f(x_1, \dots, x_n) \simeq F$. Для этого можно использовать частично-рекурсивные функции. При этом частично-рекурсивная функция задается из некоторого определённого набора базисных функций с помощью операций подстановки, рекурсии и минимизации.

Определение 1.

Базисными называются следующие тотальные функции:

$S(x) = x + 1$

$0(x) = 0$

$I_m^n(x_1, \dots, x_n) = x_m$, где $1 \le m \le n$.

Такие функции называют также простейшими. Сами базисные функции вычислимы, и существуют машины Тьюринга для их вычисления.

Определение 2.

Если для n- местной (с $n \ge 1$) частичной функции h для любых $x_1, \dots, x_n$ из N выполняется условное равенство $h(x_1, \dots, x_n) \simeq f(g(x_1, \dots, x_n), \dots, g_k(x_1, \dots, x_n))$, где f- это k-местная функция, а $g_1, \dots, g_k$- это n-местные функции, то h получается в результате операции подстановки из f и $g_1, \dots, g_k$, т.е. функций $g_1, \dots, g_k$ в функцию f.

Очевидно, что если функция h получена подстановкой вычислимых функций $g_1, \dots, g_k$ в вычислимую функцию f, то h вычислима. Например, это можно осуществить по программе $v_1 = g_1(a_1, \dots, a_n); \dots; v_k = g_k(a_1, \dots, a_n); v = (f(v_1, \dots, v_k;\ return\ v,$ в которой также даны подпрограммы для вычисления $f, g_1, \dots, g_k$.

Если же функции $f, g_1, \dots, g_k$ всюду определены, то функция h тотальна.

Определение 3.

Пусть имеется n-местная функция f, $(n+1)$-местная функция h и $(n+2)$-местная функция g. Если для всех $x_1, \dots, x_n, y \in \mathrm{N}$ выполняются уравнения рекурсии $h(x_1, \dots, x_n, 0) \simeq f(x_1, \dots, x_n)$; $h(x_1, \dots, x_n, y+1) \simeq g(x_1, \dots, x_n, y, h(x_1, \dots, x_n, y))$, то частичная функция h получается с помощью операции рекурсии из f и g. В этом определении $n \geq 1$. Если же $n = 0$, то в качестве v-местной функции f служит некоторое натуральное число m, а определение функции h становится таким: $h(0) = m$, $h(y+1) \simeq g(y, h(y))$. Термин «рекурсия» (по латински recursio – возвращение) употребляется из-за того, что значение функции h в точке $(x_1, \dots, x_n, y+1)$ определяется через значение той же функции в предыдущей точке $(x_1, \dots, x_n, y)$.

Рассмотренная выше рекурсия называется также примитивной рекурсией.

Если функции f и g всюду определены, то получаемая рекурсией функция h тотальна.

Определение 4.

Если функция h из N^{n+1} в N^{n+1} получается из базисных функций с помощью конечного числа операций подстановки и рекурсии, то h называется примитивно-рекурсивной.

Из этих определений вытекает, что любая примитивно-рекурсивная функция вычислима и тотальна.

Пусть имеется теория первого порядка, содержащая предметные постоянные 0,1,2, ... и функциональные символы. Предположим, что t терм, а список переменных $x_1, \dots, x_n$ содержит все переменные входящие в t. Тогда можно однозначно задать функцию $f(x_1, \dots, x_n) = t$, которая для любых значений $K_1, \dots, K_n$ из $\{0,1,2, \dots\}$ переменных $x_1, \dots, x_n$ имеет значение $\mathrm{a}(K_1, \dots, K_n)$ равное значению терма t. Говорят, что терм t представляет функцию f.

Если функция f получается из функций $f_1, \dots, f_s$ и $I_m^n (n = 1,2, \dots; 1 \leq m \leq n)$ с помощью конечного числа подстановок, то f называется элементарной относительно $f_1, \dots, f_s$.

Предложение 1.

n-местная числовая функция f элементарна относительно $f_1, \dots, f_s$ тогда и только тогда, когда f представима термом, использующим только функциональные символы $f_1, \dots, f_s$ и переменные из списка $x_1, \dots, x_n$. При этом терм, который со-

держит лишь символы для примитивно-рекурсивных функций, называется примитивно-рекурсивным термом. Таким образом, если функция f представима примитивно-рекурсивным термом, то f – примитивно-рекурсивна.

12.Лекция 12. Примеры рекурсивных функций.

Примеры:

1. Постоянная функция $d_n(x) = n$ представляется термом $S\Big(\dots\big(S\big(0(x)\big)\dots\big)\Big)$, следовательно, она примитивно-рекурсивна, где S встречается n раз в терме. Тогда $d_n(I_1^K(x_1, \dots, x_k) -$ это K-местная постоянная функция представимая примитивно-рекурсивным термом.
2. Функция $h(x, y) = x + y$ примитивно-рекурсивна, так как $h(x, 0) = x + 0 = x = f(x)$, где $f(x) = x$; $h(x, y + 1) = x + (y + 1) = (x + y) + 1 = h(x, y) + 1 = g(x, y, h(x, y))$, где $g(x, y, z) = z + 1, g(x, y, z) = S\big(I_3^3(x, y, z)\big), f(x) = I_1^1(x)$.
3. Функция $h(x, y) = x \cdot y$ тоже примитивно-рекурсивна, так как $h(x, 0) = 0 = f(x)$, где $f(x) = 0(x)$, $h(x, y + 1) = x \cdot (y + 1) = x \cdot y + x = h(x, y) + x = g(x, y, h(x, y))$.
4. Пусть $p \geq 2$ натуральное число. Рассмотрим функцию $h(x) = p^x$. Тогда $h(0) = p^0 = 1 = d_1(x)$, где $d_1(x) = S(0(x))$, $h(y + 1) = p^{y+1} = p \cdot p^y = p \cdot h(y) = g\big(y, h(y)\big)$, где $g(x, y) = p \cdot y = d_p(x) \cdot I_2^2(x, y)$, следовательно, $h(x)$ есть примитивно-рекурсивная функция.
5. Возьмём функцию $h(x, y) = xy$. Тогда $h(x, 0) = x^o = 1 = d_1(x)$, $h(x, y + 1) = x^{y+1} = x^y \cdot x = h(x, y) \cdot x = g(x, y, h(x, y))$, где $g(x, y, z) = z \cdot x$. Поскольку g и d_1 примитивно-рекурсивны, то функция $h(x, y)$ тоже примитивно-рекурсивна.
6. Возьмём теперь функцию
   $Sg(x) = \begin{cases} 1, \text{при } x > 0 \\ 0, \text{при } x = 0 \end{cases}$.

Имеем $Sg(0) = 0, Sg(x + 1) = g(x, Sg(x))$, где $g(x, y) = 1$.

Для функции $\overline{Sg}(x) = \begin{cases} 1, \text{при } x = 0 \\ 0, \text{при } x > 0 \end{cases}$ получаем $\overline{Sg}(0) = 1$, $\overline{Sg}(x + 1) = g_1(x, \overline{Sg}(x))$, где $g_1(x, y) = 0$. Поскольку функции g и g_1 примитивно-рекурсивны, то функция $Sg(x)$ и $\overline{Sg}(x)$ тоже примитивно-рекурсивны.

7. Функция $p(x) = \begin{cases} x-1, \text{при } x > 0 \\ 0, \text{при } x = 0 \end{cases}$ удовлетворяет тождествам $p(0) = 0$, $p(x+1) = g(x, p(x))$, где $g(x,y) = x$, следовательно, функция $p(x)$ примитивно-рекурсивна.
8. Функция $x \div y = \begin{cases} x-y, \text{при } x \geq y \\ 0, \text{при } x < y \end{cases}$, примитивно-рекурсивна. Она выражается рекурсией из примитивно-рекурсивных функций I_1^1 и $g(y,z) = p(z)$, где $p(z) -$ функция из предыдущего примера.
9. Возьмём функцию $f(x,y) = |x-y|$. Для неё выполняется тождество $|x-y| = (x \div y) + (y \div x)$. Поскольку функции $h(x,y) = x + y$ и $(x \div y)$ примитивно-рекурсивны, то функция $f(x,y)$ тоже примитивно-рекурсивна.

13. **Лекция 13. Ограниченная сумма и ограниченное произведение**.

Предложение 2. Пусть

$$f(x_1, \dots, x_n) = \sum_{k=0}^{x_n} g(x_1, \dots, x_{n-1}, k) \quad ,$$

где g является примитивно-рекурсивной функцией. Тогда n-местная

функция f тоже примитивно-рекурсивна.

Доказательство. При k=0 выполняется равенство

$$f(x_1, \dots, x_{n-1}, 0) = g(x_1, \dots, x_{n-1}, 0) \quad .$$

При k=y+1 получим

$$f(x_1, \dots, x_{n-1}, y+1) = f(x_1, \dots, x_{n-1}, y) + g(x_1, \dots, x_{n-1}, y+1) \quad .$$

Следовательно, f получается из $g(x_1, \dots, x_{n-1}, 0)$ и

$$\mathrm{h}(x_1, \dots, x_{n-1}, y, z) = z + g(x_1, \dots, x_{n-1}, y+1) \quad .$$

Предложение 3. Предположим, что (n+1)-местная функция f задана

следующим образом:

$$f(x_1, \dots, x_{n-1}, y, z) \quad = \sum_{k=y}^{z} g(x_1, \dots, x_{n-1}, k)$$

при y≤z; $f(x_1, \dots, x_{n-1}, y, z) = 0$ при y>z, где g – примитивно-рекурсивная функция. Тогда функция f тоже примитивно-рекурсивна.

Доказательство. Из условий, наложенных на функцию f вытекает, что

$$f(x_1, \dots, x_{n-1}, y, z) = \left(\sum_{k=0}^{z} g(x_1, \dots, x_{n-1}, k) \div \sum_{k=0}^{y} g(x_1, \dots, x_{n-1}, k)\right)$$

$+g(x_1, \dots, x_{n-1}, y) \quad \overline{Sg}(y \div z).$

Из примитивной рекурсивности функции $\mathrm{p}(\mathrm{y}, \mathrm{z}) = y \div z$ и предыдущего предложения следует, что функция f является примитивно-рекурсивной.

Следствие 1. Пусть заданы n-местные примитивно-рекурсивные функции g, q, u, а функция F задана по формуле:

$$F(x_1, \dots, x_n, y_1, \dots, y_m) = \sum_{k=q(y_1,\dots,y_m)}^{u(y_1,\dots,y_m)} g(x_1, \dots, x_{n-1}, k),$$

если $q(x_1, \dots, x_n) \le u(x_1, \dots, x_n)$; $F(x_1, \dots, x_n) = 0$ в остальных случаях.

Тогда функция F тоже является примитивно-рекурсивной.

Доказательство. Для функции F выполняется формула

$$F(x_1, \dots, x_n, y_1, \dots, y_m) = f\big(x_1, \dots, x_n, q(y_1, \dots, y_m), u(y_1, \dots, y_m)\big),$$

где f – функция из предыдущего предложения. Поэтому F примитивно-рекурсивна.

Предложение 4. Предположим, что n-местная функция f задана с помощью произведения из примитивно-рекурсивной функции $g(x_1, \dots, x_n)$ по формуле:

$$f(x_1, \dots, x_n) = g(x_1, \dots, x_{n-1}, 0) \dots g(x_1, \dots, x_{n-1}, x_n).$$

Тогда функция f примитивно-рекурсивна.

Доказательство. Функция f задается из примитивно-рекурсивных функций $g(x_1, \dots, x_{n-1}, 0)$ и $\mathrm{h}(x_1, \dots, x_{n-1}, y, z) = z\, g(x_1, \dots, x_{n-1}, y+1)$ соотношениями $f(x_1, \dots, x_{n-1}, 0) = g(x_1, \dots, x_{n-1}, 0)$ и

$$f(x_1, \dots, x_{n-1}, y+1) = f(x_1, \dots, x_{n-1}, y)\ g(x_1, \dots, x_{n-1}, y+1).$$

Определение 5. Пусть имеется (n+1)-местная функция g и n-местная

функция h. Зададим функцию $f(x_1, \dots, x_n)$ равной y в том и только том случае, когда $g(x_1, \dots, x_{n-1}, 0), \dots, g(x_1, \dots, x_{n-1}, y-1)$ определены и не равны 0, причем $g(x_1, \dots, x_{n-1}, y) = 0$ и $y \le h(x_1, \dots, x_n)$.

Тогда функция f получена из g и h с помощью ограниченного оператора минимизации. Такой оператор также называют ограниченным μ-оператором.

Согласно этому определению функция f может оказаться не всюду определенной.

14. Лекция 14. Представления примитивно-рекурсивных функций.

Предложение 5. Пусть (n+1)-местная функция g и n-местная функция h

примитивно-рекурсивны, а тотальная функция f получена из g и h применением ограниченного μ-оператора. Тогда функция f примитивно-рекурсивна.

Доказательство. По условию данного предложения функция f тотальна.

Поэтому ее можно представить в виде

$$f(x_1, \dots, x_n) = \sum_{k=0}^{h(x_1,\dots,x_n)} Sg\ (g(x_1, \dots, x_{n-1}, 0) \dots g(x_1, \dots, x_{n-1}, k)) \quad .$$

Из примитивной рекурсивности функций h, g и Sg, предложения 4 и следствия 1 вытекает, что функция f примитивно-рекурсивна.

Замечание 1. Используют также обозначение

$$f(x_1, \dots, x_n) = \mu y_{\le h(x_1,\dots,x_n)}[g(x_1, \dots, x_n, y) = 0].$$

Если функции g и h тотальны, то f также тотальная функция. Для всяких

$x_1, \dots, x_n$ значение $f(x_1, \dots, x_n)$ равно наименьшему $y \le h(x_1, \dots, x_n)$, для которого $g(x_1, \dots, x_n, y) = 0$. Если такого y не существует, то

$$f(x_1, \dots, x_n) = h(x_1, \dots, x_n) + 1.$$

Примеры. 10. $f(x,y) = [\frac{x}{y}]$ – целая часть от деления x на y, где $\left[\frac{x}{0}\right] = 0$.

Из равенства $\left[\frac{x}{y}\right] = \sum_{k=0}^{x} \overline{Sg}((ky) \div x)$ следует, что функция $f(x,y)$

примитивно-рекурсивна.

11. $f(x,y) = rest(x,y)$ – остаток от деления x на y при

$y \neq 0$, $rest(x,0) = x$. Эта функция примитивно-рекурсивна, так как

$$rest(x,y) = x \div \left(y\left[\frac{x}{y}\right]\right).$$

12. $\tau(x)$ - количество делителей числа x, где τ(0)=0. Из представления

$\tau(x) = \sum_{k=0}^{x} \overline{Sg}(rest(x,k))$ вытекает, что $\tau(x)$ суть примитивно-рекурсивная функция.

13. $\pi(x)$ – количество простых делителей, не превосходящих x. Из записи

Этой функции в виде $\pi(x) = \sum_{k=0}^{x} \overline{Sg}(|\tau(\mathrm{k}) - 2|)$ мы получаем, что она при-

митивно-рекурсивна. В самом деле, для простого числа p имеются ровно два делителя 1 и p.

14. $p(x)$ – простое число с номером x в ряду упорядоченных по возрастанию простых чисел. В частности, $p(0) = 2, p(1) = 3, p(2) = 5$ и так

далее. Докажем, что $p(n) \le 2^{2^n}$ методом математической индукции. Основанием индукции служит $p(0) = 2$ при $n = 0$. Пусть неравенство доказано для всех $n \le m$. Тогда

$a = p(0)p(1) \dots p(m) + 1 \le 2^{2^0+2^1+\dots+2^m} + 1 < 2^{2^{m+1}}$.

Рассматриваемое число a>1 имеет некоторый простой делитель $p(k)$,

который отличен от $p(0), \dots, p(m)$, то есть $p(m+1) \le p(k) \le a$.

Итак, $p(m+1) \le 2^{2^{m+1}}$. С помощью последнего неравенства получаем

$$p(x) = \mu y_{\leq 2^{2^x}}[|\pi(y) - (x+1)| = 0].$$

Функция $h(x) = 2^{2^x}$ примитивно-рекурсивна. Из предложения 5 следует, что $p(x)$ является примитивно-рекурсивной функцией.

15. Пусть $ex(x,y)$ обозначает показатель степени числа $p(x)$ в разложении на простые множители числа y, где $ex(x,0) = 0$. Эта функция выражается через примитивно-рекурсивные функции

$$ex(x,y) = \mu z_{\leq y}\left[\overline{Sg}\left(rest\left(y,\left(p(x)\right)^{z+1}\right)\right)Sg(y) = 0\right].$$

Поэтому, $ex(x,y)$ является примитивно-рекурсивной функцией.

16. Рассмотрим целую часть квадратного корня из x, то есть $f(x) = [\sqrt{x}\,]$. Эту функцию можно представить в виде

$$\left[\sqrt{x}\,\right] = \mu z_{\leq x}[\overline{Sg}\,((z+1)^2 \div x) = 0],$$

следовательно, $f(x)$ примитивно-рекурсивна.

17. Для пар натуральных чисел x и y возьмём функцию

$$c(x,y) = \frac{(x+y)(x+y+1)}{2} + x.$$

Это канторовская функция, которая присваивает номер паре (x,y).

Она примитивно-рекурсивна, так как

$$c(x,y) = [\frac{(x+y)(x+y+1)}{2}] + x.$$

15. Лекция 15. Частично-рекурсивные и общерекурсивные функции.

Определение 6. Пусть задана некоторая (n+1) – местная частичная функция f. Опишем n-местную частичную функцию $g(x_1, \dots, x_n)$

равную y в том и только том случае, когда для всякого $z < y$ значение

$f(x_1, \dots, x_n, z)$ определено и на равно 0, а $f(x_1, \dots, x_n, y) = 0$.

Тогда говорят, что функция g получается в результате применения операции минимизации (или μ-оператора) из f и пишут

$$g(x_1, \dots, x_n) \simeq \mu y[f(x_1, \dots, x_n, y) = 0].$$

Определение 7. Если частичная функция $g: \boldsymbol{N}^{\mathrm{n}} \dashrightarrow \boldsymbol{N}$ может быть получена из базисных функций применением конечного числа подстановок, рекурсий и минимизаций, то она называется частично-рекурсивной. Если при этом функция f оказывается тотальной, то говорят, что она общерекурсивна.

Из определений получается, что любая примитивно-рекурсивная функция является общерекурсивной.

Примеры. 18. Если имеется n-местная функция g с пустой областью определения, то можно записать условное равенство

$$g(x_1, \dots, x_n) \simeq \mu y[s(x_1) + y = 0].$$

Следовательно, g частично-рекрсивна.

19. Пусть $f(x, y) \simeq$x-y определена лишь при $y \leq x$,

тогда $f(x, y) \simeq \mu z[|x - (z + y)| = 0]$, следовательно, f – суть частично-рекурсивная функция.

20. Рассмотрим частичную функцию $f(x, y) \simeq x^{1/\mathrm{y}}$, которая определена только при $z^{\mathrm{y}} = x$ для некоторого натурального z. Из условного равенства $f(x, y) \simeq \mu z[|x - z^{\mathrm{y}}| = 0]$ мы получаем, что функция f частично-рекурсивна.

21. Теперь возьмём частичную функцию $f(x, y) \simeq \frac{x}{y}$, которая определена, если y делит x. Тогда выполняется условное равенство

$$f(x, y) \simeq \mu z[|x - zy| = 0].$$

Поэтому f является частично-рекурсивной функцией.

Теорема 1. Функция частично-рекурсивна тогда и только тогда, когда она числовая и вычислима на машине Тьюринга.

Доказательство. В силу тезиса Тьюринга любая частично-рекурсивная функция вычислима. Более детально строятся машины Тьюринга, которые вычисляют базисные функции. Если $h = f(g_1, \dots, g_n)$ и имеются машины Тьюринга F, G_1,…, G_n, вычисляющие функции f, g_1,…, g_n, то по ним строится машина Тьюринга H, вычисляющая h. Из машин Тьюринга F и G,

вычисляющих n-местную функцию f и (n+2)-местную функцию g соответственно строится машина Тьюринга H, которая вычисляет (n+1)-местную функцию h, полученную рекурсией из f и g. Также из машины

Тьюринга F, вычисляющей (n+1)-местную функцию f, получают машину

G, которая вычисляет функцию $g \simeq \mu f$, полученную из f применением

оператора минимизации.

Докажем теперь обратное утверждение о том, что вычислимая по Тьюрингу числовая функция является частично-рекурсивной. Доказательство делится на несколько этапов. Сразу отметим, что отсюда будет следовать тезис Чёрча: любая числовая функция, которая вычислима в индуктивном смысле, также есть частично-рекурсивная функция. Сначала описывается техника сопоставляющая вычислениям на машине Тьюринга вычисления с помощью натуральных чисел.

Замечание 2. Техника арифметизации машин Тьюринга.

Рассмотрим машину Тьюринга, имеющую множество внутренних состояний $Q = \{q_0, \dots, q_m\}$ и ленточный алфавит

$\Sigma = \{a_1, \dots, a_p\}$, где $a_1 = \#, a_2 = 1, a_3 = \square$. С помощью символов 1 и □

можно записать любое натуральное число. Произвольное исходное данное

$< x_1, \dots, x_n >$ для вычисления n-местной числовой функции представляют словом $x_1 \square \dots \square x_n$. Рассмотрим функцию ν_Σ: $\Sigma^* \longrightarrow \boldsymbol{N}$, которая нумерует все слова в алфавите Σ, а Σ^* обозначает множество слов, составленных из букв принадлежащих алфавиту Σ. Для пустого слова Λ полагают $\nu_\Sigma(\Lambda) = 0$, а для

непустого слова $w = a_{i_s} \dots a_{i_1} a_{i_0}$ вычисляется значение

$\nu_\Sigma(w) = i_0 + i_1 p + \dots + i_s p^s$. Если в какой-либо момент времени внутреннее состояние машины M задаётся значением $q_k \in Q$, на ленте

написано слово $w \in \Sigma^*$, а остальные ячейки пустые, то есть заполнены символом #, головкой машины обозревается ячейка с символом в ней

$a_l \in \Sigma$, то слово w удобно записать в виде $w = Aa_l B$, в котром выделен обозреваемый символ a_l. Тогда говорят, что машина имеет конфигурацию

$Aq_k\, a_l\, B$, где 0≤k≤m, 0≤l≤n, A и B – некоторые слова в алфавите Σ. Допустимы также пустые слова A и B. Для конфигурации $t = Aq_k\, a_l\, B$ её номер удобно вычислять по формуле $\nu(t) = 2^{\nu_\Sigma(A)}3^{\mathrm{k}}5^{\mathrm{l}}7^{\nu_\Sigma(B)}$.

В частности, положим

$\delta^{\mathrm{n}}(x_1, \dots, x_n) = \nu_\Sigma(x_1 \square \dots \square x_n)$.

Предложение 6. Для всякого натурального числа 1≤n функция δ^{n}

примитивно-рекурсивна.

Доказательство. Воспользуемся методом математической индукции.

В качестве основания индукции возьмём n=1. Тогда

$$\delta^1(x) = \nu_\Sigma(1 \dots 1),$$

где 1 повторяется x раз, $\delta^1(0) = 0$, $\delta^1(x+1) = 2 + \delta^1(x)p$. То есть функция

δ вычисляется рекурсивно.

Предположение индукции: пусть δ^{n} примитивно-рекурсивна. Продолжение индукции: докажем, что $\delta^{\mathrm{n+1}}$ примитивно-рекурсивна.

Из равенства $\delta^{\mathrm{n+1}}(x_1, \dots, x_n, x_{n+1}) = \delta^{\mathrm{n}}(x_1, \dots, x_n)p^{1+x_{n+1}}$ следует, что

$\delta^{\mathrm{n+1}}$ примитивно-рекурсивна, так как она представлена примитивно-рекурсивным термом.

Теперь рассмотрим подробнее работу машины M, когда в качестве

исходного данного фигурирует слово $x_1 \square \dots \square x_n$.

Начальной конфигурацией машины является $q_1 \# X_1 \square X_2 \square \dots \square X_n$, которая имеет номер $\gamma^{\mathrm{n}}(X_1, \dots, X_n) = V(q_1 \# X_1 \square \dots \square X_n)$.

Предложение 7. Функция $\gamma^{\mathrm{n}}(X_1, \dots, X_n)$ является примитивно-рекурсивной для всякого n ≥ 1.

Доказательство. Функция δ^{n} примитивно-рекурсивная в силу Предложения 6. С другой стороны, выполняется равенство

$\gamma^{\mathrm{n}}(X_1, \dots, X_n) = 3 \cdot 5 \cdot 7^{\delta^n(X_1, \dots X_n)}$,

так как ν(A q_k a_l B) = $2^{\sqrt{\Sigma(A)}} * 3^K * 5^l * 7^{\sqrt{\Sigma(B)}}$

Поэтому функция γ^n тоже является примитивно-рекурсивной.

Теперь рассмотрим перемещение считающей головки машины Тьюринга: $\nabla \in$ {N, L, R} и команды вида q_i $a_j \mapsto q_k$ a_l ∇. В качестве номера для последней команды положим p_c^h(при h = $2^k \cdot 3^l \cdot 5^{s(\nabla)}$), где c = c(i, j) – канторовская функция, которая нумерует пары натуральных чисел, S(N) = 0, S(L)=1, S(R)=2. После этого номером машины M назовем произведение номеров команд машины M. Если у машины имеется конфигурация m = A$q_i a_j$B, то в результате выполнения одной из команд получается новая конфигурация m_M' = m', по правилам:

1) m' = m, если i=0;
2) если i≠0 и имеется команда $q_i a_j \mapsto q_k$ a_l N,то $m' = Aq_k a_l B$;
3) если i≠0, а команда имеет вид $q_i a_j \mapsto q_k$ a_l R, B = /\, то
m' = A a_l q_k #;

4) если i≠0, B ≠ /\, имеется команда $q_i a_j \mapsto q_k$ a_l R, то
m' = A a_l q_k B;

5) если i≠0, A = /\, выполняется команда $q_i a_j \mapsto q_k$ a_l L, то
m' = q_k # a_l B;

6) если i≠0, A ≠ /\, выполняется команда $q_i a_j \mapsto q_k$ a_l L,
A = A_1S, то m' = $A_1 q_k a_s a_l$B;

Таким образом, если на ленте написано слово Aa_jB, а считывающая голова машины находится в состоянии q_i и обозревает символ a_j, то m'=m'$_M$ обозначает конфигурацию, которая получается из исходной конфигурации m в результате выполнения одной из команд машины M.

Далее доказывается с использованием вспомогательных утверждений, что номер конфигурации m'$_M$ можно вычислить с помощью примитивно-рекурсивной функции, исходя из номера t машины M и номера X конфигурации m в предыдущий момент времени.

Предложение 8. Номер ν(m') конфигурации m' машины M можно вычислить с помощью пятиместной примитивно-рекурсивной функции ρ по начальной конфигурации M = A $q_i a_j$ B, номерам слов $u = \nu_\Sigma(A)$,

$\nu = \nu_\Sigma$ (B) и команде $q_i a_j \mapsto q_k a_l \nabla$,

S = S(∇) в виде $\rho(s, k, l, u, \nu) = \nu(m')$.

Доказательство. Зададим функцию ρ соотношениями =

$\rho(0,k,l,u,v) = 2^u \cdot 3^k \cdot 5^l \cdot 7^s$;

$\rho(1,k,l,u,v) = 2^{h(u)} \cdot 3^k \cdot 5^{sg(u)\cdot lt(u)+\neg(sg(u))} \cdot 7^{v+p^{\ln(v)}}$;

$\rho(2,k,l,u,v) = 2^{u\cdot p+l} \cdot 3^k \cdot 5^{\neg(sg(v))+sg(v)\times lh(v)} \cdot 7^{t(v)}$;

$\rho(s,k,l,u,v) = \neg(sg(s)) \cdot \rho(0,k,l,u,v) + \neg(sg)|s-1| \cdot \rho(1,k,l,u,v) + \neg(sg)|s-2| \cdot \rho(2,k,l,u,v)$,

где ln(x) обозначается длину слова с номером x, а остальные обозначения поясняются ниже. Функция ln(x) примитивно-рекурсивна, так как ln(x) = μz≤ x[¬(sg)(x÷α(z)) = 0], где α(n) обозначает количество всех слов длины, не превосходящей n, $\alpha(n) = \sum^{n}_{i=0} p^i$.

Функция α(n) примитивно-рекурсивна в силу последнего равенства, где p – число букв в алфавите ∑. При длине слова w равной n выполняются неравенства

$\alpha(n-1) \leq \nu_{\sum}(w) < \alpha(n)$

При этом в формулах выше для ρ номер последней буквы слова x обозначен lt(x), где lt(0) = 0. Эта функция lt(x) выражается в виде примитивно-рекурсивной функции lt(x) = b(x,0). В свою очередь функция b(x,n) задается следующим образом. Возьмем какое-либо слово $w = a_{js}\ldots a_{jo}$ с номером $\nu_{\sum}(w) = x$ и положим $b(x,n) = j_n$ для каждого n = 0,1...,s ; также b(x,n) = 0 при x=0 или n > s.

При x ≠ 0 выполняется равенство

$b(x,0) = sg(rest(x,p)) \cdot rest(x,p) + \overline{sg}\,(rest(x,p) \cdot p$;

$b(x,n+1) = sg(rest(\left[\frac{x}{p^{n+1}}\right] \div \left[\frac{b(x,n)}{p}\right], p) \cdot rest\left(\left[\frac{x}{p^{n+1}}\right] \div \left(\frac{b(x,n)}{p}\right), p\right] + \overline{sg}\,(rest(\left[\frac{x}{p^{n+1}}\right] \div \left[\frac{b(x,n)}{p}\right], p) \cdot p$

при n+1 ≤ S, следовательно, функция b(x,n) примитивно-рекурсивна.

Для непустого слова $W = W_1A_s$ с номером $\nu_{\sum}(W) = x$ через h(x) обозначим номер слова W_1, где h(0)=0. Тогда получим $h(x) = \left[\frac{x \div lt(x)}{p}\right]$, так как $\nu_{\sum}(W_1) = \left[\frac{x-s}{p}\right]$, следовательно, функция h(x) примитивно-рекурсивна.

Функция lh(x) обозначает номер первой буквы непустого слова с номером x, где lh(0)=0. Поскольку lh(x) = b(x,ln(x÷1)), то функция lh примитивно-рекурсивна.

Для непустого слова w = $a_s w_1$ с номером x функция t(x) полагается равной номеру слова w_1, где t(0)=0.

Представим эту функцию в виде t(x)= $\sum_{k=0}^{ln} b(x,k) \cdot p^k$, следовательно, она тоже примитивно-рекурсивна.

Таким образом, все функции, участвующие в представлении функции ρ примитивно-рекурсивны. Из представления ρ вытекает, что она тоже является примитивно-рекурсивной функцией.

Предложение 9. Имеется пятиместная функция σ такая, что они примитивно-рекурсивна и $\sigma(t,i,j,u,w) = \nu(m'_M)$, где t – номер машины M, m = $Aq_i a_j B$ – конфигурация машины M, u = $\nu_{\sum}$(A), W = $\nu_{\sum}$(B).

Доказательство. Из определений функции ex, номера машины t и функции c вытекает, что ex(c(i,j),t) – номер команды $q_i a_j$ |→ $q_k q_l \nabla$, где k = ex(0, ex(c(i,j),t)) ; l=ex(1,ex(c(i,j),t)) ; S(∇)=ex(2,ex(c(i,j),t)). Применение предложения 8 приводит к выражениям:

$$\sigma(t,0,j,u,w) = 2^u \cdot 3^0 \cdot 5^j \cdot 7^w \text{ и}$$

$$\sigma(t,i,j,u,w) = p(ex(2,ex(c(i,j),t))),$$

ex(0, ex(c(i,j),t)), ex(1, ex(c(i,j),t)), u,w) при i > 0.

Предложение 10.

Значение $\nu(m'_M)$ можно вычислить с помощью двухместной примитивно-рекурсивной функции τ, исходя из номера t машины M и номера x конфигурации m по формуле

$$\nu(m'_M) = \tau(t,x).$$

Доказательство. Поскольку $m = Ag_i a_j B$,

$\nu_\Sigma(A) = ex(0,x)$; $i = ex(1,x)$; $j = ex(2,x)$;

$\nu_\Sigma(B) = ex(3,x)$, то в силу предложения 9 выполняется формула:

$\tau(t,x) = \sigma\big(t, ex(1,x), ex(2,x), ex(0,x), ex(3,x)\big)$.

Предложение 11. Пусть $m_M^0 = m, \quad m_M^{n+1} = (m_M^n)'$,

t – номер машины M, x – номер конфигурации m, тогда существует трехместная примитивно-рекурсивная функция w для вычисления $\nu\left(m_M^y\right)$ по формуле $\nu\left(m_M^y\right) = w(t, x, y)$.

Доказательство. Это вытекает их предложения 10 и равенств $W(t, x, 0) = x$; $w(t, x, y+1) = \tau(t, w(t, x, y))$.

Предложение 12. Предположим, что машина Тьюринга M с номером t_0 чисто вычисляет n-местную частичную функцию f.

Тогда для всяких натуральных чисел $x_1, \ldots, x_n$ значение функции $f(x_1, \ldots, x_n)$ определено в том и только том случае, когда существует y такое, что $ex\left(1, w(t_0, \gamma^n(x_1, \ldots, x_n), y)\right) = 0$,
где γ^n – функция фигурирующая в предложении 7.

Доказательство. Поскольку машина Тьюринга M чисто вычисляет функцию f, то значение $f(x_1, \ldots, x_n)$ задается тогда и только тогда, когда машина Тьюринга M начинает работу с конфигурации $m = q_1 \# x_1 \square\, x_2 \square \ldots \square\, x_n$ и за конечное число шагов y переходит в заключительное состояние q_0 с конфигурацией $\# \ldots \# \, q_0 \, \# \, f(x_1, \ldots, x_n) \# \ldots \#$.

Согласно предложению 7 число $\gamma^n(x_1, \ldots, x_n)$ является номером конфигурации m, тогда как значение $w(t_0, \gamma^n(x_1, \ldots, x_n), y)$ – это номер заключительной конфигурации, как утверждает предложение 11.

Таким образом, $ex\left(1, w(t_0, \gamma^n(x_1, \ldots, x_n), y)\right) = 0$ (см. определение номера конфигурации $\nu(m)$ и функции $ex(m, x)$ выше).

Предложение 13. Если машина Тьюринга M с номером t_0 чисто вычисляет n-местную частичную функцию f, то

$$f(x_1, \ldots, x_n) \simeq \varepsilon_2\left(ex\left(3, w\left(t_0, \gamma^n(x_1, \ldots, x_n), h^{n+1}(t_0, x_1, \ldots, x_n)\right)\right)\right)$$

для любых натуральных чисел $x_1, \ldots, x_n$, где

$h^{n+1}(t_0, x_1, \ldots, x_n) \simeq \mu y[ex\left(l, w(t_0, \gamma^n(x_1, \ldots, x_n), y)\right) = 0]$, а

$\varepsilon_j(x)$ обозначает число вхождений буквы a_j в слово с номером x.

Доказательство. Функция $\varepsilon_j(x)$ примитивно-рекурсивна, так как $\varepsilon_j(x) = \sum_{k=0}^{\ln(x)\div 1} |b(x,k) - j|$,где функции b и ln те же что и в доказательстве предложения 8. Вычисление $f(x_1, \ldots, x_n)$ машиной Тьюринга M подразумевает, что начальной является конфигурация $q_1 \# x_1 \square x_2 \square \ldots \square x_n$, а заключительной $\# \ldots \# q_0 \# f(x_1, \ldots, x_n) \# \ldots \#$. При этом $\gamma^n(x_1, \ldots, x_n)$ равняется номеру начальной конфигурации, тогда как функция h^{n+1} задает номер шага при работе машины M на исходном данном $x_1 \square \ldots \square x_n$ на котором машина M переходит в заключительное состояние q_0. Это означает, что $h^{n+1}(t_0, x_1, \ldots, x_n)$ совпадает с количеством шагов в вычислении $f(x_1, \ldots, x_n)$ на машине Тьюринга M, причем $w(t_0, \gamma^n(x_1, \ldots, x_n), h^{n+1}(t_0, x_1, \ldots, x_n))$ – это номер заключительной конфигурации. Тогда $ex(3, w(t_0, \gamma^n(x_1, \ldots, x_n), h^{n+1}(t_0, x_1, \ldots, x_n)))$ является номером слова $f(x_1, \ldots, x_n) \# \ldots \#$. С другой стороны,
$f(x_1, \ldots, x_n)$ равняется количеству единиц в этом слове,

$$f(x_1, \ldots, x_n) = \varepsilon_2(ex(3, w(t_0, \gamma^n(x_1, \ldots, x_n), h^{n+1}(t_0, x_1, \ldots, x_n))))$$

Итак, если $f(x_1, \ldots, x_n)$ определено, то условное равенство утверждения данного предложения выполняется.

Предположим, что $f(x_1, \ldots, x_n)$ не определено , тогда работа машины M продолжается на $x_1 \square \ldots \square x_n$ и при любом y конфигурация с номером $w(t_0, \gamma^n(x_1, \ldots, x_n), y)$ не является заключительной и равенство $ex\big(1, w(t_0, \gamma^n(x_1, \ldots, x_n), y)\big) = 0$ не выполняется. То есть, выражение $h^{n+1}(t_0, x_1, \ldots, x_n)$ не определено.

Таким образом, в этом случае обе части условного равенства не определены.

Окончание доказательства теоремы 1.

Функция $\varepsilon_2(ex(3, w(t_0, \gamma^n(x_1, \ldots, x_n), h^{n+1}(t_0, x_1, \ldots, x_n))))$
частично-рекурсивная. Из последнего предложения вытекает, что каждая числовая функция, которая вычислима на машине Тьюринга, также частично-рекурсивна.

Замечание 3. Частично-рекурсивные функции довольно хорошо устроены: любую частично-рекурсивную функцию f можно записать с помощью одного оператора минимизации и одной подстановки, исходя из двух примитивно-рекурсивных функций. При этом одна примитивно-рекурсивная функция зависит

от f, а другая фиксирована. Это составляет содержание нижеследующей теоремы Клини. В ней фиксированной является примитивно-рекурсивная функция l.

16. Лекция 16. Представления частично-рекурсивных функций.

Теорема 2. Любая n-местная частично-рекурсивная функция f представима в виде

$$f(x_1, \dots, x_n) \simeq l(\mu z[F(x_1, \dots, x_n, z) = 0]),$$

где F – примитивно-рекурсивная функция, которая может зависеть от f, $l(n)$ обозначает левую компоненту пары с номером n.

Доказательство. Для пары натуральных чисел x и y ее номер $n = c(x, y)$ дается канторовской функцией, $l(n) = x,\ \ r(n) = y$, где $r(n)$ – обозначает правую компоненту пары (x, y). Тогда выполняется неравенство:

$$x + y + 1 \leq \frac{[\sqrt{8n+1}]+1}{2} < x + y + 2,$$

так как $2n = x^2 + 2xy + y^2 + 3x + y$,

$$8n + 1 = (2x + 2y + 1)^2 + 8x \qquad \text{и}$$
$$8n + 1 = (2x + 2y + 3)^2 - 8y - 8$$

Поэтому $x + y + 1 = [\frac{[\sqrt{8n+1}]+1}{2}]$,

следовательно,

$$l(n) = x = n \div \frac{1}{2}\left[\frac{[\sqrt{8n+1}]+1}{2}\right] \cdot \left[\frac{[\sqrt{8n+1}] \div 1}{2}\right],$$

так как $c(x, y) = \left[\frac{(x+y)(x+y+1)}{2}\right] + x$

Итак, функция $l(n)$ примитивно-рекурсивна. В силу теоремы 1 каждая частично-рекурсивная функция вычислима на машине Тьюринга.

Предполагаем, что t-номер машины Тьюринга M, которая чисто вычисляет n-местную функцию f. Для доказательства теоремы введем (n+2)-местную функцию

$S_{n,t}(X_1, \ldots, X_n, y, z) =$

$$\begin{cases} 0, \text{если M проводит конфигурацию } q_1 \neq X_1 \square X_2 \square \ldots \square X_n \\ \text{в конфигурацию } \#\ldots\#q_0\#y\#\ldots\# \text{ за } \leq z \text{ шагов,} \\ 1, \text{в противном случае} \end{cases}$$

Сначала установим, что эта функция $S_{n,t}$ примитивно-рекурсивна.

Лемма 1. Функция $S_{n,t}$ является примитивно-рекурсивной.

Доказательство. Сначала заметим, что $S_{n,t}(x_1, \ldots, x_n, y, z) = 0$ при $j \leq z$, где j – наименьшее натуральное число, такое, что $w(t, \gamma^n(x_1, \ldots, x_{n,j}))$ является номером конфигурации вида #...#q_0#y#...#, так как согласно предложению 11 число $w(t, \gamma^n(x_1, \ldots, x_{m,j}))$ будет номер конфигурации, в которую машина Тьюринга M перерабатывает исходную конфигурацию $q_1\#x_1\square x_2\square\ldots\square x_n$ за j шагов.

Для конфигурации вида #...#q0#y#...# с номером $w(t, \gamma^n(x1, \ldots, xn,j))$ выполняется равенство $ex(1, w(t\gamma^n(x_1, \ldots, x_n, j)) = 0$, а также $\varepsilon_2(ex(3, w(t, \gamma^n(x_1, \ldots, x_n j)) = y$–число единиц в рассматриваемой конфигурации. Отсюда вытекает, что $S_{n,t}(x_1, \ldots, x_n, y, z) = 0$ равносильно тому, что $\mu_{j\leq z}\,[ex(1, w\big(t\gamma^n(x_1, \ldots, x_n, j)\big) + |\varepsilon_2(ex(3, w(t, \gamma^n(x_1, \ldots, x_n j)) - y| = 0] \leq z$, что в свою очередь равносильно условию $\mu_{j\leq z}\,[ex(1, w\big(t\gamma^n(x_1, \ldots, x_n, j)\big) + |\varepsilon_2(ex(3, w(t, \gamma^n(x_1, \ldots, x_n j),3) - y| = 0] : z = 0$, отсюда получаем представление $S_{n,t}(x_1, \ldots, x_n, y, z) = \mu_{j\leq z}[ex(1, w\big(t\gamma^n(x_1, \ldots, x_n, j)\big) + |\varepsilon_2(ex(3, w(t, \gamma^n(x_1, \ldots, x_n j),3) - y| = 0] : z$, следовательно, $S_{n,t}$ является примитивно-рекурсивной функцией.

Окончательное доказательство теоремы 2.

Зададим теперь (n+3) – местную функцию $S_{n,t}(t_1, x_1, \ldots, x_n, y, z) = S_{n,t}(x_1, \ldots, x_n, y, z)$. В силу леммы 1 функция S_n примитивно-рекурсивна. Также определим еще (n+2) – местную функцию $T_n(t,x_1,\ldots, x_n, z) = S_n(t,x_1,\ldots,x_n,l(z),r(z))$, где z-номер пары (a,b) с l(z)=a, r(z)=b согласно нумерующей функции Кантора для пар неотрицательных целых чисел. Очевидно, что функция T_n тоже примитивно-рекурсивна. Тогда $T_n(t_1,x_1,\ldots x_n,z)=0$ равносильна тому, что вычисление функции f машиной Тьюринга M с номером t заканчивается за число шагов не превосходящего r(x) с результатом l(x). В противном случае получим $T_n(t_1,x_1,\ldots x_n,z)=1$. Если

t-фиксированный номер, то положим для простоты рассмотрения $F(x_1,\dots x_n,z)= T_n(t_1,x_1,\dots x_n,z)$

Предположим, что значение $f(x_1,\dots x_n)$ определено и $f(x_1,\dots x_n)=y$. Тогда вычисление на машине Тьюринга M заканчивается за некоторое конечное число K шагов, причем c(y,k) – наименьшее значение z, для которого $T_n\ (t,x_1,\dots x_n,z)=0$, следовательно, $F(x_1,\dots x_n,z)=0$ и $y=l(\mu z[F(x_1,\dots x_n,z)=0])$.

Пусть теперь значение $l(\mu z[F(x_1,\dots x_n,z)=0])$ определено и равняется y, следовательно, $F(x_1,\dots x_n,z)=0$, где z=c(y,k) для некоторого число k. Итак, в этом случае $T_n\ (t,x_1,\dots x_n,z)=0$ и вычисление $f(x_1,\dots x_n)$ на машине Тьюринга M оканчивается на k-м шаге с результатом y, то есть $f(x_1,\dots x_n)=y$. Таким образом, для функции F выполняется утверждение данной теоремы.

Определение 8. Для семейства F n-местных частичных числовых функций, если имеется (n+1) – местная функция u, которая удовлетворяет условиям (1) и (2):

(1) $f(x_1,\dots x_n)\cong u(j,x_1\dots,x_n)$ принадлежит F для всякого j;

(2) $\forall f(f\in F)\rightarrow\exists j\ \forall x_1\dots\forall x_n\ \ f((x_1,\dots x_n)\ )\cong u(j,x_1\dots,x_n)$,

то u называется универсальной функцией для семейства F.

Теорема 3. Если F_n – семейство всех n-местных частично-рекурсивных функций, где n-натуральное число, n=1,2…, то существует (n+1) – местная частично-рекурсивная функция u_n, которая универсальна для F_n.

Доказательство. Зададим частично-рекурсивную функцию $u_n(t,x_1\dots x_n) \cong l(\mu z[T_n(t,x_1,\dots x_n,z)=0])$. Если t фиксировано, то u_n как функция переменных $x_1,\dots,x_n$ задает n – местную частично-рекурсивную функцию, следовательно условие (1) из определения 8 выполнено. В силу теоремы 1 для произвольной частично-рекурсивной функции f существует некоторая машина Тьюринга M с номером t, которая вычисляет f. Из доказательства предыдущей теоремы вытекает, что выполняется условное равенство $f(x_1,\dots x_n)\cong u_n(t,x_1\dots,x_n)$, следовательно условие (2) из определения 8 тоже выполнено.

Зададим теперь (n+3) – местную функцию $S_n(t,x_1,\dots,x_n,y,z) = S_{n,t}(x_1,\dots,x_n,y,z)$. В силу леммы 1 функция S_n примитивно-рекурсивна. Также определим еще (n+2) – местную функцию $T_n(t,x_1,\dots,x_n,z) = S_n\big(t,x_1,\dots,x_n,l(z),r(z)\big)$, где z –номер пары (a,b) с $l(z)=a, r(z)=b$ согласно нумерующей функции Кантора для пар неотрицательных чисел. Очевидно, что

функция T_n тоже примитивно-рекурсивна. Тогда $T_n(t, x_1, \dots, x_n, z) = 0$ равносильна тому, что вычисление функции f машиной Тьюринга M с номером t заканчивается за число шагов не превосходящего r(x) с результатом l(x). В противном случае получим $T_n(t, x_1, \dots, x_n, z) = 1$. Если t-фиксированный номер, то положим для простоты рассмотрения F($x_1, \dots, x_n, z$)= $T_n(t, x_1, \dots, x_n, z)$

Предположим, что значение f($x_1, \dots, x_n$) определено и f($x_1, \dots, x_n$) $= y$. Тогда вычисление на машине Тьюринга M заканчивается за некоторое конечное число k шагов, причем c(y,k) – наименьшее значение z, для которого $T_n(t, x_1, \dots, x_n, z) = 0$, следовательно, F($x_1, \dots, x_n, z$)=0 и $y = l(\mu z[F(x_1, \dots, x_n, z) = 0])$.

Получается условное равенство $f(x_1, \dots, x_n) = u_n(t, x_1, \dots, x_n)$, следовательно условие (2) из определения 8 тоже выполнено.

Литература.

1) Мальцев А.И. Алгоритмы и рекурсивные функции. - М.: Наука, 1986, - 368 с.
2) Крупский В.Н., Плиско В.Е. Математическая логика и теория алгоритмов. - М.: Академия, 2013. – 416 с.
3) Мендельсон Э. Введение в математическую логику. - М.: Наука, 1984. – 320 с.
4) Яблонский С.В. Введение в дискретную математику. - М.: Высшая Школа, 2002. – 384 с.
5) Гаврилов Г.П, Сапоженко А.А. Задачи и упражнения по дискретной математике. - М.: Наука, 2006. – 416 с.
6) Лавров И.А., Максимова Л.Л. Задачи по теории множеств, математической логике и теории алгоритмов. – М.: Наука, 1975. – 240 с.
7) Справочная книга по математической логике. Часть III. Теория рекурсии / под ред. Дж. Барвайса. – М.: Наука, 1982. – 360 с.
8) Шенфилд Дж. Математическая логика. – М.: Наука, 1975. – 528 с.
9) Кормен Т.Х., Лейзерсон Ч.И., Ривест Р.Л., Штайн К. Алгоритмы: построение и анализ. - М.: «Вильямс», 2019. - 1328 с.