233 - 282

ВЫЧИСЛИМОСТЬ В ПРОИЗВОЛЬНЫХ ОБЛАСТЯХ И БАЗИСАХ

А.П.Ершов
Вычислительный центр, Новосибирск 630090

Резюме. В этой описательной статье мы обсуждаем одно из главных понятий математки и вычислительной науки - понятие эффективной вычислимости. Демострируя разнообразие определений вычислимости, мы в то же время обнаруживаем, что эти определения не объясняют нам, почему они оказываются эквивалентными друг другу. Размышляя над тем, как можно строго поставить этот вопрос, мы приходим к понятию абстрактной вычислимости в произвольных областях и базисах. Анализируя различные работы, посвященные абстрактной вычислимости, мы высказываем точку зрения, что понятие относительной вычислимости и истории вычислений представляются более первичными, нежели понятие программы и ее свойства композиционной замкнутости. В результате мы приходим к определению вычислимости, которое строго разделяет ее комбинаторный и вычислительный аспекты. Это определение является также попыткой синтеза взглядов на вычислимость, сложившихся как в математической логике, так и в теоретическом программировании.

Введение

Распространение ЭВМ и программирования делают математическую логику прикладной наукой. Каста математических логиков - жрецов хранителей священного огня оснований математики к своему изумлению оказываются лицом к лицу с нашествием орд программистов, чья

233-283

математическая культура близка к варварству, но которые, тем не менее, вдохновляемые своими пророками структурного программирования, стремятся зажечь факел от животворного огня и утащить его к себе, чтобы осветить углы своих подразделений, забитых лампами, инструкциями по ОС/360, фортрановскими бланками и прочими символами идолопоклонства. Тем не менее, для того чтобы установить союз столь разных культур, каждой стороне - логикам и программистам - необходимо преодолеть свои комплексы неполноценности, выработать способность к взаимопониманию, а главное - убедиться в своей интересности для другой стороны.

Это описательная статься представляет собой персональную реакцию на сложившуюся ситуацию. Когда автор был аспирантом Московского университета в середине 50-х годов, он делил свое время между изучением известной книги С.К.Клини "Введение в метаматематику" (Клини, 1952) и разработкой трансляторов для машин БЭСМ и "Стрела". Чтение книги заразило автора изумлением, с которым Клини (§ 62) говорил о необыкновенной устойчивости понятия вычислимости перед лицом самых разных предпосылок и способов его определения. С другой стороны, работа над трансляторами, над методологией программирования, над тем, что стало впоследствии теорией схем программ, - все это постоянно наталкивало автора на мысль, что в программировании надо уметь различать комбинаторную и собственно вычислительную стороны процесса вычислений. Мы то манипулируем с элементарными командами как с символами, то пускаем их в ход как окончательные орудия обработки информации. Ощущалось, что понимание взаимодействия этих двух сторон вычислений является фундаментальной проблемой, требующей более глубокого раскрытия сущности вычислимости.

233-284

Прошло однако почти двадцать пять лет, прежде чем для автора стало ясным содержание возможной дискуссии. За это время теория вычислимости получила большое развитие, хотя, как не без сожаления констатировал Крайзел, 1969, вопрос о сущности вычислимости был скорее периферийной, нежели центральной проблемой. За это же время возникло теоретическое программирование, в котором сложился свой круг понятий и свои модели вычислений (см., например, Манна, 1974; Ершов, 1977; Котов, 1978).

Мы проанализируем обе линии развития как в теории вычислимости, так и в теоретическом программировании. Этому анализу мы предпошлем сводку тех основных определений вычислимости, которые возникли в связи с главной моделью в конструктивной математике – (модель) вычислимых арифметических функций. Мы закончим наш анализ определением вычислимой функции, которое, как мы надеемся, обладает следующими достоинствами:

– оно определяет вычислимость в любой предметной области и любой системе базовых операций;

– оно четко разделяет комбинаторную и "исполнительную" стороны вычислимости;

– оно не опирается на какие бы то ни было специфические синтаксис программ и механизм их выполнения.

В заключение мы изложим ряд аргументов в пользу такого определения и некоторые построения теории вычислимости, основанной на этом определении. Существенным будет синтетический характер этих аргументов, взятых как из логики, так и из программирования.

В обзорной части работы автор ссылается, по возможности, на оригинальные работы. Однако, в ряде случаев реальными источниками

233-285

были обобщающие трактаты, из которых нужно, прежде всего, назвать уже упомянутую книгу Клини, 1952 и недавно изданный компендиум по математической логике Барвайз, 1977.

Часть I. ВЫЧИСЛИМОСТЬ АРИФМЕТИЧЕСКИХ ФУНКЦИЙ

Везде ниже мы будем рассматривать частичные однозначные функции, берущие свои аргументы и принимающие значения на множестве N, т.е. на натуральном ряде, включающем нуль $\{0,1,2,\ldots\}$. Эти функции, образующие пространство A, мы будем называть арифметическими. Говоря об арифметических функциях вообще, мы не будем себя ограничивать абстракциями конструктивной математики. В классе A мы с помощью различных определений будем выделять подклассы арифметических функций, которые будут называться "вычислимыми функциями по XX", где XX – имя одного из авторов данного определения. В дальнейшем окажется, что все эти подклассы совпадают, образуя класс вычислимых арифметических функций C. В классе C выделяется подкласс всюду определенных, или тотальных, вычислимых функций. Какой-либо из классов "вычислимых функций по XX" может иногда именоваться особым термином, получившим распространение в литературе.

С понятием вычислимой функции тесно связаны понятия эффективно порождаемых (перечислимых) и эффективно выделяемых (разрешимых) множеств. Множество называется перечислимым, если оно является областью значений тотальной вычислимой функции. Множество называется разрешимым, если его характеристическая функция является тотальной вычислимой функцией. Предикат называется вычислимым, если его область истинности является разрешимым множеством, и полувычислимым, если область его истинности перечислима.

233-286

В некоторых теориях понятие перечислимого множества является первичным. В таких случаях понятия вычислимой функции и разрешимого множества вводятся путем следующих определений. Функция называется вычислимой, если ее график перечислим. Множество называется разрешимым, если перечислимы как оно, так и его дополнение.

Представляется уместным расклассифицировать определения вычислимости по четырем классам: логические, функциональные, алгоритмические и арифметические определения. Логические определения выделяют в формальных логических системах класс вычислимых функций с помощью некоторых формул, обладающих определенными дедуктивными свойствами. Функциональные определения выделяют класс вычислимых функций в функциональном пространстве с помощью замыкания некоторых операций. Алгоритмические определения задают вычислимые функции описанием алгоритма, вычисляющего значение функции по указанному аргументу и по заданной программе. Наконец, арифметическое определение задает перечислимые множества как множества, возникающие при решении простейших арифметических уравнений.

Логические определения

Изобразимость по Гёделю. Рассматривается арифметическое исчисление предикатов S, например, по Клини, 1952, гл. IУ, содержащее одну предметную константу 0, один предикатный знак =, три функциональных знака $'$, $+$, $\cdot$ и в качестве нелогических аксиом аксиомы Пеано и рекурсивные определения для сложения + и умножения $\cdot$. Исчисление содержит счетное множество предметных переменных (их метаобозначения $x, y, x_1, y_1, \ldots$).

Функция $\varphi(x_1, \ldots x_n)$ изобразима в S, если имеется формула $P(x_1, \ldots, x_n, y)$, такая, что

$$\varphi(x_1, \ldots, x_n)=y \Leftrightarrow \vdash P(cx_1, \ldots, cx_n, cy),$$

где $\underline{cx}$ – терм, изображающий натуральное число x (можно всегда считать, что он имеет вид $0^{\overbrace{'\dots'}^{x\text{ раз}}}$). Множество арифметических функций, изобразимых в арифметическом исчислении предикатов, образует класс вычислимых функций (по Гёделю, 1936, формулируется по Клини, 1952, § 59,62).

Рекурсивная определимость по Эрбрану и Гёделю. Рассматривается формальное исчисление уравнений над счетными множествами переменных для натуральных чисел $x, y, x_1, y_1, \dots$ и функциональных букв $f, g, h, f_1, g_1, h_1, \dots$, содержащее одну предметную константу 0 и один функциональный символ $'$, а также разделители $=, (,)$ и $,$. Термами являются 0, переменные, τ', где τ – терм, и $f(\tau_1, \dots, \tau_n)$, где f – функциональная буква, а $\tau_1, \dots, \tau_n$ $(n \geqslant 0)$ – термы. Уравнение имеет вид $\tau = \sigma$, где τ и σ – термы. Система уравнений – это конечная последовательность уравнений $E = \{e_0, \dots, e_s\}$. Самая левая функциональная буква в последнем уравнении e_s называется главной буквой. Даны два правила вывода.

R1 (конкретизация): $e(y) \vdash e(c)$, где $e(y)$ – уравнение, содержащее переменную y, а c – любая цифра, т.е. терм 0 или $0^{'\dots'}$;

R2 (замена): $\tau = \sigma(h(c_1, \dots, c_p)), h(c_1, \dots, c_p) = c \vdash \tau = \sigma(c)$, где $\sigma(h(c_1, \dots, c_p))$ – терм, содержащий вхождение терма $h(c_1, \dots, c_p)$, в котором h – функциональная буква, $c_1, \dots, c_p, c$ – цифры.

Функция $\varphi(x_1, \dots, x_n)$ рекурсивно определима, если имеется система уравнений E, такая, что

$$\varphi(x_1, \dots, x_n) = y \Leftrightarrow E \vdash f(cx_1, \dots, cx_n) = cy,$$

где f – главная буква в E, cx – цифра, изображающая число x. Множество рекурсивно определимых арифметических функций образует класс вычислимых функций (по Эрбрану, 1931, Гёделю, 1934, сформулировано по Клини, 1952, §55).

λ-определимые функции по Чёрчу. Над счетным алфавитом переменных V строится множество термов T: а) $V \subset T$; б) $x \in V$, $t \in T \Rightarrow (\lambda x t) \in T$ (рассмотрение терма t как функции от x); в) $t \in T, u \in T \Rightarrow (tu) \in T$ (применение терма-функции t к терму-аргументу u). На термов строится множество формул F: а) $s, t \in V \Rightarrow s \to t \in F$ (s редуцируется к t); б) $s, t \in V \Rightarrow s = t \in F$ (s равно t). Для сокращения числа скобок используются следующие упрощения: (1) $t_1 t_2 \dots t_n \equiv (\dots(t_1 t_2)\dots t_n)$ и (2) $\lambda x_1 \dots x_n . t \equiv (\lambda x_1 (\lambda x_2 \dots (\lambda x_n t)\dots))$. Последнее сокращение одновременно подсказывает, как в языке "вычисляются" функции многих переменных: $f x_1 x_2 \dots x_n$ вычисляется так, что применяется f к x_1, затем $f x_1$ применяется к x_2 и т.д.

Над формулами строится λ-исчисление со следующими аксиомами и правилами вывода.

I. 1. $(\lambda x . s) t \to s[x/t]$ (правая часть редукции означает терм s, в котором все свободные вхождения x заменены на терм t);

2. $s \to s$; 3. $s \to t, t \to u \vdash s \to u$;

4. (a) $s \to t \vdash us \to ut$; (в) $s \to t \vdash su \to tu$; (с) $s \to t \vdash \lambda x . s \to \lambda x . t$;

II. 1. $s \to t \vdash s = t$; 2. $s = t \vdash t = s$; 3. $s = t, t = u \vdash s = u$;

4. (a) $s = t \vdash us = ut$; (в) $s = t \vdash su = tu$; (с) $s = t \vdash \lambda x . s = \lambda x . t$.

Натуральные числа можно изображать в виде термов λ-исчисления. Вот один из способов ($\underline{n}$ – изображение числа n): $\underline{n} = \lambda f x . f^n x$, где $f^0 x = x$, $f^{n+1} x = f(f^n x)$. Арифметическая функция $\varphi(x_1, \dots, x_n)$ λ-определима, если существует такой замкнутый (т.е. без свободных переменных) терм f, что

$$\varphi(x_1, \dots, x_n) = y \Leftrightarrow \vdash f \underline{x_1} \dots \underline{x_n} = \underline{y} .$$

Множество λ-определимых функций образует класс вычислимых функций (по Чёрчу, 1936, 1941, сформулировано по Барендрегту, 1977).

233–229

Функциональные определения.

Частичная рекурсивность по Клини. Берется счетный базис арифметических функций:

I) функции счета $S(x)=x'$;

II) функции-константы $C_c^n(x_1,\dots,x_n)=c$, где c – натуральное число;

III) функции выбора аргумента $U_i^n(x_1,\dots,x_n)=x_i$.

Далее, рассматриваются три типа операторов, т.е. схем определения новых функций через данные:

IV) оператор подстановки $\varphi=S_m^n(\psi,\chi_1,\dots,\chi_m)$, для которого

$$\varphi(x_1,\dots,x_n)=\psi(\chi_1(x_1,\dots,x_n),\dots,\chi_m(x_1,\dots,x_n));$$

V) оператор примитивной рекурсии $\varphi=R^n(\psi,\chi)$, для которого

$$\varphi(0,x_2,\dots,x_n)=\psi(x_2,\dots,x_n),$$

$$\varphi(y',x_2,\dots,x_n)=\chi(y,\varphi(y,x_2,\dots,x_n),x_2,\dots,x_n);$$

VI) μ-оператор, или оператор упорядоченного поиска $\varphi=M^n(\chi)$, для которого

$$\varphi(x_1,\dots,x_n)=\mu y[\chi(x_1,\dots,x_n,y)=0],$$

т.е. наименьшее y, для которого $\chi(x_1,\dots,x_n,y)=0$ (и для всех $y'<y$ $\chi(x_1,\dots,x_n,y')\neq 0$), и неопределено, если такого y не существует.

Указанные операторы позволяют рассматривать замыкания базисных функций этими операторами и их комбинациями. При этом интересно выделить три класса функций:

а) замыкание оператором подстановки создает класс просто вычислимых функций;

б) замыкание операторами подстановки и примитивной рекурсии создает класс примитивно рекурсивных функций (Гёдель, 1931);

в) замыкание всеми тремя операторами создает класс частично рекурсивных функций, образующих класс вычислимых функций (по Клини, 1938, сформулировано по Клини, 1952, §63).

233-290

Наименьшие неподвижные точки рекурсивных программ по Маккарти.

Рассматривается формальный язык функциональных уравнений, содержащий счетные множества переменных для натуральных чисел $x, y, x_1, y_1, \ldots$ и функциональных букв $f, g, h, f_1, g_1, h_1, \ldots$, одну предметную константу 0, один функциональный символ τ, один предикатный символ = (равенство), а также разделители $\Leftarrow$, (,) и , . Каждой функциональной букве f сопоставляется ее арность: целое число $\varphi(f) \geqslant 0$. Именами функций являются выражения $f(x_1, \ldots, x_{\varphi(f)})$, где f – функциональная буква, а $x_1, \ldots, x_{\varphi(f)}$ – разные переменные. Функциональными термами являются 0, переменные, τ', где τ – функциональный терм, и $f(\tau_1, \ldots, \tau_{\varphi(f)})$, где f – функциональная буква, а $\tau_1, \ldots, \tau_{\varphi(f)}$ – функциональные термы. Предикатные термы имеют вид $x=y$, где x и y – переменные. Условные термы имеют вид $(p|\tau|\sigma)$, где p – предикатный терм, а τ и σ – функциональные или условные термы. Значение val условного терма определяется по правилу разбора случаев:

$$val\,(p|\tau|\sigma) = \begin{cases} val\,\tau, & \text{если } p; \\ val\,\sigma, & \text{если } \neg p. \end{cases}$$

Уравнением называется выражение вида $f(x_1, \ldots, x_{\varphi(f)}) \Leftarrow t$, где t – функциональный или условный терм с переменными, принадлежащими множеству $\{x_1, \ldots, x_{\varphi(f)}\}$. Система уравнений $e_1, \ldots, e_n$ $(n > 0)$ называется совместной, если она не содержит уравнений с одинаковыми функциональными буквами и для любой функциональной буквы, входящей в правую часть какого-либо уравнения, есть уравнение с именем, содержащим эту букву. Рекурсивной программой называется совместная система уравнений, в которой выделено одно главное уравнение. Неподвижной точкой системы совместных уравнений $e_1, \ldots, e_n$ называется совокупность частичных функций $\varphi_1, \ldots, \varphi_n$,

обращающих систему уравнений в тождественные равенства (если заменить $\Leftarrow$ на $=$). Существует эффективное доказательство существования для любой совместной системы уравнений наименьшей неподвижной точки, т.е. такой совокупности функций $\varphi_1^*,\dots,\varphi_n^*$, что для любой другой неподвижной точки $\varphi_1,\dots,\varphi_n$ имеет место $\varphi_i^* \subseteq \varphi_i$ ($i=1,\dots,n$). Наименьшей неподвижной точкой рекурсивной программы является таковая для главного уравнения. Класс наименьших неподвижных точек рекурсивных программ образует класс вычислимых функций (по Маккарти, 1961 и Манне, 1974, гл. IV).

Примечание. Рассмотрим уравнение

$$f(x_1,\dots,x_n) \Leftarrow t(x_1,\dots,x_n; f, g_1,\dots,g_k),$$ где

$g_1,\dots,g_k$ – другие функциональные буквы, входящие в t. Сопоставим этим буквам некоторые арифметические функции $\lambda_1,\dots,\lambda_k$ и рассмотрим наименьшую неподвижную точку $\varphi(x_1,\dots,x_n)$ получающегося уравнения относительно f. Тогда выражение t можно рассматривать как оператор $\varphi = \mu b^n[t, \lambda_1,\dots,\lambda_k]$ взятия неподвижной точки. Замыкание пустого множества функций оператором взятия наименьшей неподвижной точки также образует класс вычислимых функций по Маккарти.

Алгоритмические определения

Машины Тьюринга. Рассматривается язык программ для машин Тьюринга. Машина Тьюринга работает с бесконечной в обе стороны лентой, разделенной на клетки, в каждой из которых может находиться или быть записанным пустой символ или "единица". Машина может находиться в одном из конечного числа состояний $q_0, q_1,\dots,q_k$, одно из которых q_0 является пассивным, заключительным, а остальные – активными. Одно из активных состояний q_1 считается начальным. В любом состоянии машина обозревает некоторую клетку ленты.

233-292

II.

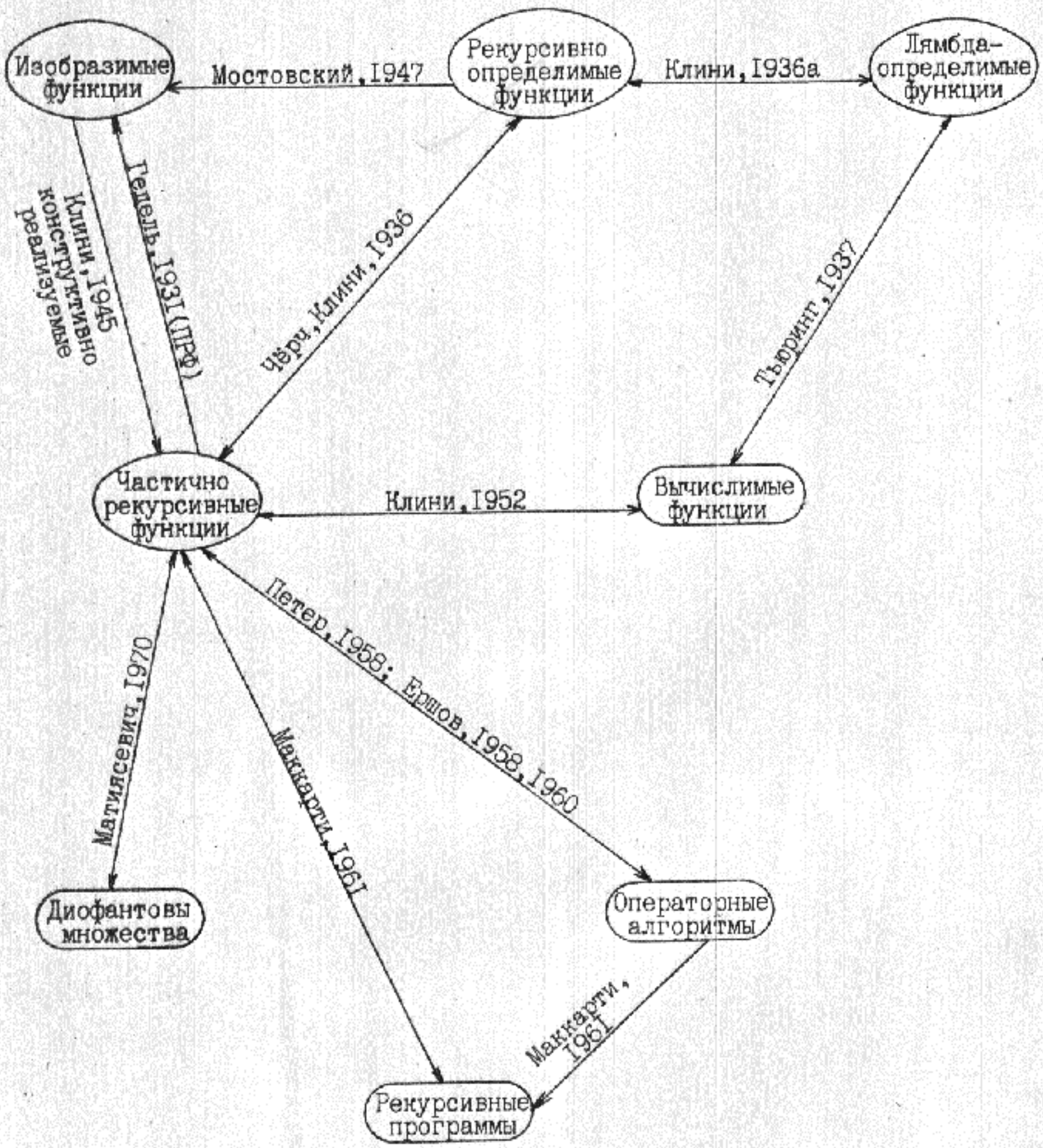
Машина может выполнить следующие операции: определить содержимое обозреваемой клетки, записать в клетку пустой символ или единицу, сдвинуться вдоль ленты на одну клетку вправо или влево или остаться на месте, перейти из одного состояния в другое (в том числе в то же самое). Программа для машины Тьюринга с k активными состояниями имеет вид матрицы порядка $k \times 2$. В позиции матрицы (i,j) указывается, что делает машина, находясь в i-м состоянии и обнаруживая в обозреваемой клетке j-й символ ($j=1$ пусто, $j=2$ единица). Действие указывается набором $\alpha\beta\gamma$, где α – символ, который пишется на обозреваемую клетку, β – характер движения вдоль ленты (влево, на месте, вправо), γ – состояние, в которое переходит машина.

Для каждой программы указывается, от какого числа n аргументов зависит функция, вычисляемая данной машиной ($n \geqslant 0$). Задание набора аргументов $x_1, \ldots, x_n$ на ленте выглядит так: x_1+1 единиц, пусто, x_2+1 единиц, пусто и т.д. до x_n+1 единиц. По обе стороны от части, занимаемой аргументами, лента предполагается пустой. Машина, находясь в начальном состоянии, обозревает крайнюю слева единицу в записи аргументов, после чего начинается работа по программе до тех пор, пока машина не перейдет в конечное состояние. Если при этом на ленте останется блок из $y+1$ единиц и машина будет в заключительном состоянии обозревать крайнюю слева единицу, то в этом и только в этом случае y будет считаться значением функции, задаваемой данной машиной Тьюринга (более точно, ее программой). Множество частичных арифметических функций, задаваемых машинами Тьюринга, образует класс вычислимых функций (по Тьюрингу, 1936, сформулировано по Клини, 1952, §67).

233-213

Операторные алгоритмы по Ершову. Рассматривается язык программ для операторных алгоритмов, содержащий счетное множество переменных для натуральных чисел $x, y, x_1, y_1, \ldots$, одну предметную константу 0, один функциональный символ $'$, один предикатный символ = (равенство), а также разделитель := (знак присваивания). Термами являются 0, переменные и τ', где τ – терм. Присваивание имеет вид $x := \tau$, где x – переменная и τ – терм. Условие имеет вид $x=y$, где x и y – переменные. Графом переходов является ориентированный граф с выделенными входной и выходной вершинами. Все вершины (кроме выходной) в графе делятся на две категории – преобразователи, имеющие ровно одного преемника, и распознаватели, имеющие ровно двух преемников (плюс-преемник и минус-преемник). Операторным алгоритмом $A(x_1, \ldots, x_n, y)$ называется граф переходов, в котором каждому преобразователю сопоставлено некоторое присваивание и каждому распознавателю – некоторое условие, а выходной вершине – переменная y. Выполнение операторного алгоритма, состоящее в вычислении значения y как некоторой функции $\varphi(x_1, \ldots, x_n)$ состоит в следующем. Все переменные, входящие в программу A, образуют память этой программы. Задаются начальные значения $x_1, \ldots, x_n$ переменных $x_1, \ldots, x_n$ и управление передается начальной вершине графа переходов. Если управление передано преобразователю с присваиванием $x := \tau$, то, если аргумент терма τ определен, значение этого терма присваивается переменной x и управление передается единственному преемнику преобразователя. Если управление передается распознавателю с условием $x=y$, то в случае заданности x и y управление передается плюс- или минус-преемнику в зависимости от истинности или ложности предиката $x=y$. Если управление передано на выход с переменной y и y имеет значение y, то оно берется в качестве значения функции

233-294



Фиг.I. Схема связи определений арифметической вычислимости

233-295

$\varphi(x_1,\dots,x_n)$. Множество функций, вычисляемых операторными алгоритмами, образует класс вычислимых функций (по Ершову, 1959).

Арифметическое определение

Арифметические определения задают перечислимые множества, (выявляющие при рассмотрении) (системы уравнений, построенных из термов в некотором базисе простых арифметических функций и содержащих неотрицательные константы, параметры и неизвестные. В качестве множеств, представимых уравнениями, рассматриваются множестве значений параметров, при которых уравнения разрешимы в целых (неотрицательных, положительных) числах.

Диофантова вычислимость по Матиясевичу. Рассматриваются обычные полиномы с целыми коэффициентами и переменными $y_0, y_1, \dots, y_n$, $x_1, \dots, x_l$, пробегающими натуральное значение. Множество $M = \{\mu_0, \mu_1, \dots, \mu_n\}$ является диофантовым, если существует полином $P(y_0, y_1, \dots, y_n, x_1, \dots, x_l)$, такой, что $\mu_0, \mu_1, \dots, \mu_n \in M$ тогда и только тогда, когда уравнение

$$P(\mu_0, \mu_1, \dots, \mu_n, x_1, \dots, x_l) = 0$$

имеет решение. Множество функций, графики которых являются диофантовыми множествами, образует класс вычислимых функций (по Матиясевичу, 1970).

Общая теория вычислимости

(Устойчивость) Одним из самых замечательных и удивительных достижений современной математики является постепенное обнаружение того факта, что все эти определения вычислимости, а также ряд других им родственных*) задают один и тот же класс $\mathcal{C}$ арифметических функций (Фиг.I), которые мы и будем называть вычислимыми (неважно на ос-

*) Следует упомянуть среди алгоритмических определений машины Поста, 1936 и нормальные алгоритмы Маркова, 1951.

233-296

нове какого определения). Мало того, каждая теория вычислимых функций и/или перечислимых множеств, строящаяся над каждым определением, обнаруживала в себе ряд фундаментальных свойств, выглядящих по сути своей сходно с построениями "параллельной" теории. Естественно, что со временем эти сходства стали выскакивать сознательно, как бы желая, чтобы каждый новый вариант теории вычислимости был бы похож на остальные. Однако к моменту наступления такого периода насыщения теория вычислимости накопила некоторую систему фундаментальных фактов и закономерностей, проливающих свет на сущность вычислимости, а также позволяющих работать с этим понятием в других областях математики и ее приложений.

Не претендуя на полноту, дадим краткий очерк общей теории вычислимости, имея в виду ее так называемую элементарную часть. Существенно более полный обзор теории вычислимости и ее применений можно найти у Успенского и Семенова, 1981.

(Универсальный алгоритм.)
Для каждой вычислимой функции (перечислимого множества) существует конечный источник (программа) исчерпывающей информации об этой функции (множестве), являющийся эффективно определимой конструкцией некоторого формального языка. Существует процедура, удовлетворяющая неформальным критериям эффективности (кодировка), взаимно-однозначно отображающая множество программ во множество натуральных чисел. Это отображение может быть распространено на весь натуральный ряд (нумерация). Существует универсальная процедура, удовлетворяющая неформальным критериям эффективности, гарантирующая для любой программы функции φ и любого набора $x_1,\dots,x_n$ аргументов получение значения $\varphi(x_1,\dots,x_n)$ при условии, что $\varphi(x_1,\dots,x_n)$ определена, или установить, что набор $x_1,\dots,x_n$, y

233-297

принадлежит графику функции φ. Эффективность этой процедуры находит свое подтверждение в том, что при подходящей нумерации универсальная функция $U^n(z, x_1, \ldots, x_n)$, т.е. такая, что для любого номера n_φ вычислимой функции φ от n аргументов

$$U^n(n_\varphi, x_1, \ldots, x_n) = \varphi(x_1, \ldots, x_n),$$

сама оказывается вычислимой функцией.

Относительная вычислимость.

Всем определениям вычислимости с той или иной степенью естественности может быть придан индуктивный характер, где в качестве базиса индукции постулируется вычислимость некоторых элементарных арифметических функций. Определение класса вычислимых функций может быть релятивизировано к конечному набору $\psi_1, \ldots, \psi_\ell$ произвольных арифметических функций (оракулов), значения которых находятся, вообще говоря, внешним по отношению к теории вычислимости способом. Класс функций, вычислимых относительно набора $\psi_1, \ldots, \psi_\ell$, называется вычислимым замыканием функций $\psi_1, \ldots, \psi_\ell$.

Если в формализме относительной вычислимости рассмотреть некоторую программу $P(\psi_1, \ldots, \psi_\ell)$, то при разных исходных функциях $\psi_1, \ldots, \psi_\ell$ мы будем получать либо разные функции, задаваемые этой программой, либо разные значения (если зафиксированы числовые аргументы программы). Таким образом с каждой программой можно связать вычислимый оператор или вычислимый функционал.

Замкнутость.

Очень важным свойством класса $\mathfrak{C}$ в значительной степени объясняющим его устойчивость, являются его замкнутость по отношению к самым разнообразным операциям над функциями (и теоретико-множественным операциям над их графиком). Отметим сначала свойства композиционной замкнутости: замкнутость относительно суперпозиции, связывания аргументов константой или ограниченным квантором, перестановки и отождествления аргументов, итерации и разветвления (в

233-298

смысле языков программирования). Для множеств аналогичная замкнутость имеет место относительно операций объединения, пересечения, прямого произведения и проектирования. Далее, вычислимые замыкания любых наборов вычислимых функций принадлежат классу $\mathfrak{C}$, в частности, замыкания относительно любого вычислимого оператора.

Для каждого вычислимого оператора $\underline{R}(\psi)$ можно определить его наименьшую неподвижную точку, которая также оказывается вычислимой функцией (1-я теорема о рекурсии). Наконец, класс $\mathfrak{C}$ замкнут относительно диагонализации, т.е. для любой нумерации $\{\varphi\}$ вычислимых функций и аргументов функция $\varphi^*(x_1,\ldots,x_n)=\varphi_{x_1}(x_1,x_2,\ldots,x_n)$ вычислима.

Вычислимость универсальных функций также является выражением определенных свойств замкнутости класса $\mathfrak{C}$.

Важные примеры.
Объем класса $\mathfrak{C}$ устанавливается построением конкретных функций и множеств, находящихся за его пределами. Наиболее принципиальными конструкциями являются: невычислимая функция; вычислимая функция, не продолжаемая до всюду определенной вычислимой функции (для множеств: неперечислимое множество; перечислимое, но не разрешимое множество). В основе этих конструкций обычно лежит универсальная функция.

Программы.
Значительное место в общей теории вычислимости занимает изучение программ вычислимых функций. Важно заметить, что все "языки программирования" создают избыточный запас изобразительных средств: каждая вычислимая функция имеет бесконечное множество задающих ее программ. Мало того, программы, задающие функции, настолько разнообразны, что для любой тотальной вычислимой функции $f(x)$ можно найти такую точку ее графика $(n,f(n))$, что указанные числа, трактуемые как программы, будут задавать одну и ту же вычислимую функ-

233-299

цию (2-я теорема о рекурсии). Наряду с этим имеет место принципиальный и на первый взгляд парадоксальный факт: программа, задающая для универсального алгоритма исчерпывающую информацию для получения значения функции по заданным аргументам, в то же время не говорит ничего об общих свойствах функции: каково бы ни было нетривиальное свойство вычислимой функции (т.е. которым обладают не все функции из $\mathfrak{C}$), не существует алгоритма, который по программе обнаруживал, обладает ли функция, вычислимая по этой программе, указанным свойством. В частности, по программе нельзя определить, задает ли она тотальную функцию; для двух программ нельзя определить, задают ли они одну и ту же функцию, и т.п.

(нормальные формы) (Естественной реакцией на такую) (неинформативность всякого описания программы) является выделение классов вычислимых функций и различных "нормальных форм" для программ. Классы функций обычно выделяются с помощью тех или иных структурных ограничений на задающие их программы, однако впоследствии эта классификация иногда подтверждается установлением внутренних свойств класса, например, по скорости роста и т.п.

В первом приближении рассматриваются следующие классы: класс $\mathfrak{C}$ (синоним – частично рекурсивные функции), тотальные вычислимые функции (синоним – общерекурсивные функции), примитивно рекурсивные функции, различные субрекурсивные классы, прямо вычислимые функции. Среди субрекурсивных классов выделяются такие, которые, будучи как можно более узкими, сохраняют способность порождать любое перечислимое множество.

Первым источником нормальных форм является программа универсальной функции, которая фиксирует в своей структуре тот запас действий и минимум организации вычислений, которые обеспечивают выполнение любой программы. Рисунок нормальной формы сильно зависит от конкрет-

233-300

ного способа программирования, однако для многих из них характерно разделение "комбинаторной" и "поисковой" стадий выполнения алгоритма с возможной концентрацией последней в одной части программы. Слова "поисковый" и "комбинаторный" – это типичные примеры математических эпитетов, которые никогда не определяются, используясь главным образом в предисловиях и в комментариях, но в которых зачастую сидит вся суть математического рассуждения. Мы понимаем под комбинаторной стадией манипуляции, связанные с элементами конечного множества, причем объем этих манипуляций имеет достоверную верхнюю оценку, известную до начала стадии. Поисковая стадия связана с обозреванием элементов бесконечного множества, причем как без гарантии того, что нужный нам элемент у принадлежит множеству, так и без оценки числа шагов, которые надо сделать, прежде чем добраться до цели.

В гёделевых изображениях вычислимых функций существует целая иерархия нормальных форм в виде общерекурсивных предикатов, представленных разными комбинациями кванторов (иерархия Клини). В частично рекурсивных функциях существует нормальная форма с единственным вхождением оператора поиска. Для рекурсивных программ аналогом является нормальная форма с одним рекурсивным уравнением, а для операторных алгоритмов – с одним оператором цикла. Диофантовы множества сами по себе образуют нормальную форму, в которой комбинаторным шагом является вычисление полинома, а поиск заключается в переборе параметров и неизвестных.

Программные процессоры.

Существенное место в классе вычислимых функций занимают некоторые конкретные функции, смысл которых состоит в том, что они работают с программами, т.е. с номерами вычислимых функций. Программистам естественно называть такие функции программными процессорами.

233-301

Самым первым из них следует назвать универсальную функцию - интерпретатор языка программирования, запрограммированный в нем самом. Не менее фундаментальную роль играет так называемая $s-m-n$ - функция, или смешанный вычислитель $s_n^m(p^{m+n}, x_1, \ldots, x_m)$, который по программе p, задающей функцию $n+m$ переменных $\varphi(x_1, \ldots, x_n, x_{n+1}, \ldots, x_{n+m})$, и по заданным аргументам $x_1, \ldots, x_n$ вычисляет (генерирует) программу $p_{x_1, \ldots, x_n}$, задающую функцию $\psi(x_{n+1}, \ldots, x_{n+m}) = \varphi(x_1, \ldots, x_n, x_{n+1}, \ldots, x_{n+m})$. Важную роль играют "трансляторы" - функции, переводящие образы функций из одной нумерации в другую. К ним примыкают перестановки - тотальные функции, переводящие взаимно однозначно $\mathcal{N}$ в $\mathcal{N}$. С их помощью выделяются инвариантные свойства функций и множеств, т.е. такие, которые сохраняются при всевозможных перестановках координатных множеств. Ни одна развитая теория вычислимости не обходится без библиотеки стандартных процедур, связывающих функции и множества: тотальные функции, перечисляющие график вычислимой функции, частичные функции, представляющие перечислимое множество своей областью определения, и т.п. Общим для всех этих служебных функций является стремление представить их в некоторой стандартной форме или отнести их к как можно более простому классу. Для тотальных функций стандартным является класс примитивно рекурсивных функций, а также всюду, где возможно, суб-рекурсивные классы. Для ~~нумерений~~ нумераций нужно уметь сворачивать взаимно однозначно последовательности чисел в одно число; базовой конструкцией является функция пересчета пар, фактически представляющая собой семейство трех функций $z = \mathfrak{w}(x,y)$, $x = \sigma_1(z)$ и $y = \sigma_2(z)$, удовлетворяющих следующим тождествам

$$\forall z: z \equiv \mathfrak{w}(\sigma_1(z), \sigma_2(z)); \quad \forall x,y: x \equiv \sigma_1(\mathfrak{w}(x,y)) \;\&\; y \equiv \sigma_2(\mathfrak{w}(x,y)).$$

Сложность

Последние годы с общей теорией вычислимости начинает сливаться теория сложности вычисления функций. В ее основе лежат абстракции разных форм физической реализации вычислений во времени и в пространстве, а также оценки объема той информации, которую надо задать в программе. Теория сложности рассматривает обычно только тотальные функции и является принципиально релятивизированной теорией, исчисляя сложность вычислений по отношению к фиксированному набору элементарных средств обработки и хранения информации (функциональный базис). Исходным понятием в теории сложности является протокол, или история вычислений, или просто вычисление: организованная совокупность элементарных шагов, реализующих функциональную зависимость, т.е. приводящая от $x_1,\dots,x_n$ к $\varphi(x_1,\dots,x_n)$. Организованность выглядит как (частичный) порядок на элементарных шагах, отражающий логические и информационные связи между ними. Этот порядок отчасти индуцируется программой, отчасти – универсальным алгоритмом исполнения.

На протоколах задается некоторая мера, отражающая затраты времени и пространства на выполнение вычисления согласно данному протоколу. Затем вводится некоторый способ "интегрирования" мер сложности отдельных вычислений, позволяющий сопоставить некоторую сложность данной программе. Навешивание кванторов на сложности множества функционально эквивалентных программ дает оценки сложности решения задачи в данной вычислительной модели. Квантор всеобщности дает нижнюю оценку (какова ни была бы программа, она будет не проще чем...); квантор существования дает верхнюю оценку (существует программа со сложностью не более...). Поскольку затраты ресурсов решительно зависят от объема входной информации, сложность обычно указывается как некоторая функция от параметра, характеризующего объем.

233 – 303

Вклад разных моделей в общую теорию

Формирование общей теории вычислимости и ее применение обогатили математику серией принципиальных достижений. Некоторые модели вычислений играют важную роль в информатике и программировании. Ряд понятий и утверждений теории вычислимости имеет глубокую методологическую и философскую интерпретацию. Не претендуя ни на полноту, ни на глубину анализа охарактеризуем вклад разных моделей вычислимости в общую теорию и ее приложения.

Классические модели. Изобразимость некоторых конкретных вычислимых функций, охватываемых классом примитивно рекурсивных функций, в логическом исчислении формальной арифметики позволила Гёделю, 1931 установить свой знаменитый результат о неполноте формальных теорий. Очень важную роль для теории вычислимости и математики вообще сыграла концепция гёделевой нумерации, т.е. отображения конструкций формального языка в предметную область, в данном случае – в $\mathcal{N}$. Выделенный при этом класс примитивно рекурсивных функций на долгие годы стал основной "системой программирования" разных служебных функций теории вычислимости и был подвергнут детальному изучению, в частности, расчленен на иерархию субрекурсивных классов, задаваемую рисунком формального представления и градуируемую шкалой оценок скорости роста (Гжегорчик, 1953).

Понятие рекурсивной функции по Эрбрану позволило нащупать объем класса тотальных вычислимых (общерекурсивных) функций, распространить на них понятие изобразимости (Мостовский, 1947), установить первую эквивалентность с независимым определением вычислимости (λ-определимостью) (Клини, 1936а), дать первые формулировки теорем о нормальной форме (Клини, 1936) и о нумерации (Клини, 1943). Уже в последние годы в виде так наз. базовых трансформаций (rewriting

233-304

rules.) этот подход к определению вычислимости вошел в программирование под названием трансформационного подхода (см., например, Дарлингтон, 1978).

Теория λ-определимости менее распространена, нежели другие модели вычислимости, если не считать λ-нотации Черча, ставшей одним из фундаментальных обозначений в математике. В то же время эта модель сыграла большую роль в становлении теории вычислимости. Будучи весьма абстрактной моделью, λ-исчисление интуитивно выделило в качестве исходных понятий наиболее важные универсальные конструкции класса вычислимых функций: отображение функций в предметную область (в виде способности одного и того же терма употребляться и как функция, и как аргумент), взятие значения функции от заданного аргумента с помощью универсальной операции (операция (tu) применения t к u), $s-m-n$-функция (способ вычисления функции нескольких аргументов путем поочередного применения). Необходимость экспликации этих понятий в конкретных вычислительных моделях была, возможно, толчком к нахождению соответствующих конструкций в теории рекурсивных функций. Аналогично, теорема о неподвижной точке в λ-исчислении привела к нахождению теорем о рекурсии.

Изучение свойств λ-исчисления позволило впервые доказать алгоритмическую неразрешимость конкретной проблемы, состоящей в следующем. Терм в λ-исчислении называется вызовом функции, если он имеет вид $(\lambda x.f)t$. Терм является нормальной формой, если он не содержит вызовов функции. Терм t имеет нормальную форму u, если $\vdash t \to u$. Например, терм $(\lambda x.xx)y$ имеет нормальную форму yy, а терм $(\lambda x.xx)(\lambda x.xx)$ нормальной формы не имеет (действительно, если редуцировать этот вызов, то он перейдет сам в себя). Проблема, неразрешимость которой была установлена Чёрчем, 1936, состоит в уста-

233-305

новлении для любого терма, имеет ли он нормальную форму.

Совпадение λ-определимости с общерекурсивностью (Клини,1936а) позволило Чёрчу сформулировать свой тезис о том, что общерекурсивные (и λ-определимые) функции отвечают и полностью охватывают интуитивное понятие вычислимой функции.

Мы еще будем говорить о роли λ-исчисления в абстрактной вычислимости, а пока заметим, что внимание к этой модели возросло в связи с развитием программирования. Язык Лисп (Маккарти, 1960) почти буквально содержит в себе формализм λ-исчисления; Ландин, 1965 указал на связь конструкций языка Алгол 60 с λ-исчислением; совсем же недавно Бекус, 1978 выдвинул функциональное, или аппликативное программирование как альтернативу сложившемуся стилю составления детерминированных императивных программ.

Машины Тьюринга оказались в высшей степени убедительной моделью, демонстрирующей механический характер вычислений, однозначно предписываемых программой и состоящих из элементарных шагов, каждый из которых обладает свойством непосредственной очевидности. В машинах Тьюринга работа с количествами полностью сводится к работе со знаками. Для машин Тьюринга был построен универсальный алгоритм, была доказана алгоритмическая неразрешимость проблемы остановки и введено понятие относительной вычислимости (Тьюринг, 1936). Характеризация λ-определимых функций машинами Тьюринга и наоборот оказались весьма веским подтверждением тезиса Черча. В дальнейшем машины Тьюринга благодаря крайней элементарности своих шагов стали популярной моделью в работах по сложности алгоритмов (см., например, Трахтенброт, 1967). Введение различных ограничений на манипулирование с лентой привело к развитой классификации машин Тьюринга, получивших в области субрекурсивных функций название автоматов и

образовавших свою развитую теорию.

Понятие частично рекурсивной функции позволило осознать свойства замкнутости класса вычислимых функций, в частности, по отношению к диагонализации, получить в окончательной форме теорему об универсальной функции (Клини, 1943), s-m-n-теорему, первую (Клини, 1952, §66) и вторую (Клини, 1938) теоремы о рекурсии, установить в общей форме нераспознаваемость нетривиальных свойств вычислимых функций по программам (Райс, 1953), полностью разработать понятие относительной вычислимости и ввести на его основе вычислимые функционалы и операторы (Клини, 1952, ч.III).

Программистские модели. Операторные алгоритмы и рекурсивные программы возникли в теоретическом программировании и до сих пор являются моделью вычислений, периферийной для общей теории вычислимости. Их роль возрастет, когда мы продолжим их обсуждение в разделе абстрактной вычислимости. Однако уже сейчас можно заметить, что наличие механизма произвольного доступа к памяти (по адресу или индексу) сделало некоторый подкласс операторных алгоритмов одной из основных вычислительных моделей в теории сложности (Ахо и др., 1976). Выделение в качестве отдельной конструкции графа переходов позволило провести классификацию программ по структуре графа переходов (бесцикловые, структурированные программы), в частности, обнаружить, что уже весьма регулярной и простой структуры достаточно, чтобы запрограммировать любую функцию (Петер, 1958; Бём, Якопини, 1966), а также подтвердить иерархию субрекурсивных функций (Констебль, Бородин, 1972). Аналогом теоремы о нормальной форме для рекурсивных функций здесь служит теорема о нормальной форме с единственным оператором цикла (Харел, 1980). Поскольку правила программирования операторных алгоритмов задани относительно

базиса элементарных предикатов и операций, они позволяют изучать критерии алгоритмической полноты базиса, т.е. возможности задать операторными алгоритмами любую вычислимую функцию (Непомнящий,1972).

Рекурсивные программы позволили охарактеризовать класс вычислимых функций как неподвижные точки уравнений, в которых допускаются прямые вычисления и простейшие определения разбором случаев. Это сочетание конкретной программной конструкции со столь абстрактным определением результата вычислений оказалось очень удобным методологическим средством для описания смысла программ, т.е. для описания вычисляемой программой функции некоторыми средствами, отличными от универсального алгоритма. Соответствующая техника под названием денотационной семантики получила распространение в теоретическом программировании (Манна, 1974).

Операторные алгоритмы и рекурсивные программы стали в последние годы доминирующими моделями в теории доказательств свойств программ. Как известно, изобразимость вычислимой функции φ в логическом языке формальной арифметики позволяет средствами логического вывода доказывать принадлежность отдельных точек графику функции φ, но не даст никакого подхода к доказательству общих свойств произвольной вычислимой функции. По программе примитивно рекурсивной функции в ее определении по Гёделю можно извлечь доказательство теоремы существования (Клини, 1952, §49), но этого уже нельзя сделать для общерекурсивной функции, поскольку ее программа неотличима от программ частично рекурсивных функций, для которых справедлива уже упомянутая теорема Райса о нераспознаваемости нетривиальных функциональных свойств.

Теория доказательств свойств программ под воздействием практических потребностей стремится разными методами преодолевать этот

233- Зар

барьер неразрешимости. Один из методов - работа с так называемыми аннотированными программами. Суть его состоит во введении своеобразного программно-логического исчисления, основанного на языке, объединяющем формулы исчисления предикатов и программные конструкции. Аннотация - это логическая формула (утверждение), отнесенная некоторой точке программы. Аксиомы программно-логического исчисления позволяют провести индуктивное рассуждение, управляемое структурой программы, и которое, отправляясь от аннотаций, как от посылок, выводит некоторое общее утверждение о программе, например, тотальность вычисляемой ею функции или ее равенство другой, независимо определенной функции (задача о правильности программы). Успех дела состоит в искусстве выбора аннотаций. Это, естественно, творческая задача, но ее решению часто способствует то или иное априорное знание о программируемой задаче. Используемые при этом смешанные программно-логические исчисления образуют то, что называется логической семантикой языка программирования (Флойд, 1967; Хоар, 1967; Берталл, 1969).

Ситуация выглядит по-иному при взгляде на связь между логикой и программированием с другой стороны. В ряде случаев априорная информация о задаче, выраженная в некотором языке спецификаций, дает возможность получить из этой спецификации программу вычислений путем систематического синтеза программы. В основе этой возможности лежит фундаментальный факт извлекаемости рекурсивного описания вычислимой функции по доказательству теоремы существования в конструктивной логике (Клини, 1945; 1952, §82). Построение аналогичных синтезаторов для других вычислительных моделей с дополнительными требованиями удовлетворения разным критериям эффективности составляет один из центральных разделов теоретического

233-309

программирования.

Недавно выяснилось, что программные процессоры типа μ-π-ρ-функции, или смешанного вычислителя, играют фундаментальную роль в трансляции и других прикладных вопросах программирования (Ершов, 1977). Это привело к задаче строгого описания этих программных процессоров в моделях операторных алгоритмов и рекурсивных программ (Ершов, 1978, 1980).

Диофантовы множества. Характеризация перечислимых множеств и вычислимых функций диофантовыми множествами была найдена совсем недавно (Матиясевич, 1970). Изложение соответствующей теории еще в значительной мере отражает путь ее становления (Матиясевич, 1972). Движущей силой в развитии вопроса был вызов Гильберта, сформулировавшего в качестве своей 10-й проблемы задачу поиска алгоритма нахождения целочисленных решений полиномиальных уравнений с целыми коэффициентами (диофантовы уравнения). Огромные и безуспешные усилия, затраченные на попытки позитивного решения этой проблемы, в сочетании с накапливающимися к этому времени примерами алгоритмической неразрешимости приучали к мысли видеть в 10-й проблеме Гильберта просто еще один пример неразрешимой массовой проблемы, однако гипотеза Дэвиса, 1953 о том, что эта проблема может быть неразрешима из-за гораздо более фундаментальной причины, а именно, из-за диофантовости перечислимых множеств, сразу придала этой проблеме дополнительную глубину.

Не отвлекаясь на переизложение истории окончательного установления диофантовости перечислимых множеств, заметим лишь, что построение теории вычислимости в рамках этой модели еще продолжается (Дэвис и др., 1976). Найден универсальный многочлен, т.е. обладающий параметром, варьирование которого позволяет получать любые

перечислимые множества фиксированной размерности. Степень и число переменных полинома оказались удобными параметрами, характеризующими сложность представления его диофантова множества. С привязкой к этим параметрам был установлен ряд теорем редукции (например, что достаточно рассматривать диофантовы уравнения степени не более 4), а также получены оценки сложности конкретных множеств. Пожалуй, наиболее загадочной теоремой о нормальной форме является представление любого перечислимого множества из $\mathcal{N}$ в виде положительных значений некоторого полинома. Это значит, что полиномы обладают определенной универсальной способностью кодирования информации. Эксплуатация этих способностей доступна пока что лишь нескольким экспертам и они время от времени поражают воображение остальных математиков магическими формулами многочленов, вычисляющих, например, все простые числа или - как этот простенький полином, найденный Джоунзом (цит. по Манину, 1980, гл.2, §8),

$$2a^4b + a^3b^2 - 2a^2b^3 - a^5 - ab^4 + 2a$$

- все числа Фибоначчи и только их.

Часть II. АБСТРАКТНАЯ ВЫЧИСЛИМОСТЬ

Преданализ

Знакомство с теорией вычислимости во всем разнообразии образующих их вычислительных моделей оставляет смешанное впечатление, даже если оставаться строго в рамках арифметических функций. Естественно, первое, к чему нас приучают, - это "железная" круговая порука взаимной сводимости. Далее мы можем усмотреть контуры - но не более - инвариантной теории, утверждения которой можно выразить в любой вычислительной модели и доказать в любом языке программирования. Для того, чтобы воплотить контуры в реальность, мы имеем, грубо говоря, следующие альтернативы: (I) провести все построения

в данной модели, используя понятия и стиль рассуждений и конструкций, характерный для ее языка программирования, или (2) "украсть" результат из другой теории, применив двакды трансляцию из одной модели в другую. Можно также предаться оппортунизму и заниматься то заимствованиями, то собственными построениями в зависимости от того, что "удобнее".

В каком-то смысле, никакой из этих способов не хорош. Первый подход превращает науку в монологи глухих, оставляя каждого исследователя наедине со спецификой выбранной модели и ее языка программирования, как правило, для чего-то весьма неудобного и искусственного. Второй подход заведомо сужает взгляд на теорию вычислимости, ставит в особое положение какую-то одну специфическую версию этой теории, подрывает интерес к поискам независимых характеризаций вычислимых функций и перечислимых множеств*). Третий подход, обеспечивая сосуществование и взаимную дополнительность разных моделей, представляется самым просвещенным, но он уводит от ответа на простой вопрос: почему? Почему все вычислительные модели создают один и тот же класс вычислимых функций? С одной стороны, понятие вычислимости демонстрирует непоколебимую устойчивость, иссл по мнению многих математиков абсолютный характер (см., например, Роджерс, 1967, §1.6), а, с другой стороны, сущность вычислимости ускользает от нас под покровом столь непохожих друг на друга одежд конкретных вычислительных моделей и их языков программирования.

---

*) Автору запомнилась сценка из работы 3-го съезда советских математиков в 1956 г. Докладчик, рассказав об одном из уточнений понятия алгоритма, завершил изложение результата эквивалентности этого уточнения к другим понятиям алгоритма бодрой фразой: "это, таким образом, лишний раз подтверждает тезис Чёрча," в ответ на что из зала раздалось: "В каком смысле лишний?".

233-312

Естественно, автор далеко не первый, кто ставит этот вопрос, и нам предстоит рассмотреть серию очень интересных работ, в той или иной мере пытающихся на этот вопрос ответить. Однако представляется полезным немного порассуждать заранее по поводу того, как можно подходить к ответу на вопрос о сущности вычислимости, какой парадигмой при этом пользоваться. Здесь нам придется преодолевать традиционный замкнутый круг развития теории: говорить наперед о том, чего еще не знаем; мы разомкнем этот круг, развернув его в три витка анализа - предобзорный, постобзорный и заключительный.

Сущность вычислимости - это нечто, находящееся в равном положении по отношению к любой конкретной вычислительной модели. Идеальной является некоторая абстрактная теория, по отношению к которой любая конкретная вычислительная модель является моделью в смысле предикатной теории доказательств. Выполнимость аксиом абстрактной теории проверяется в конкретном языке программирования данной модели. Уже на этом уровне возникает много вопросов: от каких свойств конкретных моделей следует абстрагироваться, какие объекты следует сделать параметрами абстрактной теории, какими свойствами их следует наделить аксиоматически.

Уже в начале размышлений по поводу этих вопросов возникает вопрос следующего уровня: можно ли в принципе построить абстрактную теорию вычислимости, оставаясь в пределах натурального ряда $\mathcal{N}$ и пространства арифметических функций $\mathbb{A}$? Забегая несколько вперед, заметим, что это один из самых противоречивых вопросов теории вычислимости. Некоторые современные пифагорейцы считают, что все научное как раз и сконцентрировано в $\mathcal{N}$, $\mathbb{A}$, $\mathbb{C}$ и нумерациями остальных теорий можно получить все утверждения, интересные для математики. Этому противопоставляется такой взгляд, что на практике

233-313

вычисления встречаются в самых разных предметных областях и с самым разным запасом элементарных вычислительных средств и поэтому мы обязаны уметь развивать теорию вычислимости в любых областях.

Заметим, что вопрос о выходе за пределы $\mathcal{N}$, A и $\mathcal{C}$ может иметь еще и чисто методологическое значение, поскольку сущности, лежащие в основе понятия вычислимости, слишком плотно упакованы или просто слиты в понятии натурального числа. Выход не на более абстрактные объекты позволит разделить и, стало быть, эксплицировать эти сущности.

Анализ сущностей - это в высшей степени тонкое дело, поэтому мы только наметим некоторые особенности натуральных чисел и их использования в теории вычислимости.

I. Натуральное число является простейшим конструктивным объектом с одной константой 0 (нуль) и единственным конструктом ' (счет).

2. Для конструктивных объектов очень важной является связь между объектом как некоторой абстрактной индивидуальностью ("количеством" для числа) и его знаковым выражением, или представлением. Для объекта $x \in D$ и его представления $\underset{\sim}{c}x \in W$ (W - пространство символических выражений) необходимы две функции $æ: D \to W$ и $\delta: W \to D$ со свойствами $æ(x) = \underset{\sim}{c}x$, $\delta(\underset{\sim}{c}x) = x$, $x_1 = x_2 \Leftrightarrow \underset{\sim}{c}x_1 = \underset{\sim}{c}x_2$. Для натуральных чисел связь между количеством и его представлением с помощью операции счета предельно симметрична и проста: $n \leftrightarrow 0\overbrace{'\ldots'}^{n\text{ раз}}$ (заметим, что это одно из самых "коварных" для анализа свойств натурального числа).

3. Любое множество из $\mathcal{N}$ линейно упорядочено. Любое множество имеет минимальный элемент; любое конечное множество имеет максимальный элемент (это также очень сильное свойство, выводящее многие алгоритмические операции над числами, особенно поиск).

233-314

4. Основные арифметические операции сложения и умножения, используемые в теории вычислимости для функций кодирования, образуют на целых числах кольцо, обладающее богатым запасом тождеств. Эти тождества иногда используются случайным образом, иногда в комбинаторной (знаковой) части вычислений, иногда во время работы с количествами, затемняя тем самым логическую структуру вычислительного процесса.

При выходе за пределы $\mathcal{N}$, $\mathcal{A}$ и $\mathcal{C}$ возникает еще одна альтернатива: говоря о вычислимости в произвольных областях можно ли позволить себе опираться на теорию вычислимости в $\mathcal{N}$? Для пифагорцев этот вопрос не возникает, поскольку они выражают произвольную вычислимость в терминах арифметической вычислимости. Автору, однако, представляется, что абстрактная теория вычислимости должна строиться без явных или косвенных ссылок на концепции арифметической вычислимости.

Рассуждая об абстрактном определении класса вычислимых функций над произвольными областями, с самого начала можно выделить два альтернативных подхода. Их можно называть вычислимостью в большом и вычислимостью в малом. Первый подход, рассматривая абстрактные функциональные пространства, выделяет класс вычислимых функций (над неуказанными областями) в целом, постулируя свойства замкнутости и существование определенных универсальных функций. Истоки этого подхода можно усмотреть в абстрактных системах Чёрча, 1941 и Карри, 1930. Второй подход более конструктивен: постулируя некоторые свойства предметной области D, он связывает с ней набор (базис) "элементарных" операций, т.е. предикатов $\Pi = \{\pi_1, \ldots, \pi_m\}$ и функций $\Phi = \{\varphi_1, \ldots, \varphi_n\}$, превращая тем самым D в алгебраическую систему $\langle D, \Pi, \Phi \rangle$. Константы достаточно рассматривать как нульместные операции. Если необходимо, операции также наделяются некоторыми

233-315

свойствами. Класс вычислимых функций, грубо говоря, задается с помощью некоторой системы программирования в репертуаре команд (П),(Ф). Сразу заметим, что в такой подход вкладываются без особых трудностей все вычислительные модели арифметической вычислимости с базовой алгебраической системой $\langle N, =, 0, ' \rangle$

В заключение заметим, что следует различать понятие абстрактной и обобщенной вычислимости. Не вдаваясь в детали, мы проводим водораздел между ними в зависимости от тех или иных абстракций бесконечного. В нашем понимании абстрактная вычислимость, рассматривая функции в произвольных областях и базисах, остается в пределах абстракции потенциальной бесконечности и строгого понимания конечного. Мы приходим к обобщенной вычислимости, если допускаем ординалы и теории высших порядков. Вопросы обобщенной вычислимости в нашем обзоре не рассматриваются, хотя требования обобщений такого рода во многих работах сильно влияли на способы построения абстрактной теории. Более широкий и богатый идеями анализ обобщенной и абстрактной вычислимости был проделан Крайзелом, 1969. Автор позволил себе использовать в обзорной части некоторые выдержки из другой своей работы, посвященной этому же вопросу (Ершов, 1981).

АБСТРАКТНАЯ ВЫЧИСЛИМОСТЬ В БОЛЬШОМ

Униформно вычислительные структуры по Вагнеру. Вагнер, 1963, 1968, рассматривает в качестве исходной предметной области абстрактное множество U. Он постулирует существование гёделизации, т.е. однозначного отображения $G: U \to U^U$, образ которого задает некоторый запас унарных функций типа $U \to U$. Обозначая $G(u) = [u]$, Вагнер определяет n-местные функции по схеме Шенфинкеля, 1922:

$$[u]_n(x_1, \ldots, x_n) = [[u]_{n-1}(x_1, \ldots, x_{n-1})](x_n) \text{ и } [u]_1 = [u]$$

и определяет для заданной пары $I = \langle U, G \rangle$ функциональное простран-

ство $F(I)=\{[u]_n \mid u\in U, n\geq 1\}$.

Пара $\langle U,G\rangle$ называется униформно рефлексивной структурой (УРС), если в U можно указать три разных элемента $*$ (индекс неопределенной функции), α (функция смешивания – один из комбинаторов Карри, 1930, также восходящий к Шенфинкелю, 1922) и ψ (хорошо известная программистам конструкция если-то-иначе), удовлетворяющие следующим аксиомам:

1. $[u](*) = * = [*](u)$;

2. $[\alpha](f,g) \neq *$,

   $[[\alpha](f,g)](x) = [f](x,[g](x))$;

3. $[[\psi](c,b,a)](x) = \begin{cases} a, \text{если } x=c, \\ b, \text{если } x\neq c. \end{cases}$

Вагнер показывает, что из этих аксиом вытекает существование многих "стандартных" вычислимых функций (константы, тождественные функции, функции выбора аргумента), наличие мощных теорем замыкания и другие свойства, обычно адресуемые вычислимым функциям и их классам. Показывается далее, что в $\mathcal{N}$ существуют рекурсивные конструкции, которые обеспечивают на нем выполнение аксиом УРС. Возникающий на этой структуре класс функций в точности совпадает с частично рекурсивными функциями. Вагнер далее показывает, что в произвольной УРС можно естественным образом задать 1-1-образ натурального ряда (силинтер). Если потребовать, чтобы силинтер $\mathcal{N}$ как подмножество в U имел бы в $F(I)$ всюду определенную характеристическую функцию, то тогда $F(I)$, рассматриваемое на силинтере, совпадает с $\mathcal{C}$ в некотором естественном изоморфизме.

Базисные теории рекурсивных функций. Униформно рефлексивные структуры $I=\langle U, G\rangle$ постулируют наличие некоторой геделизации G. Все свойства класса $F(I)$ доказываются (хотя и равномерно) относительно уже заданной G. Строит, 1968 и Фридман, 1969, рассматривая

233-317

пространство функций $F=\{f\}\ f:D^n\to D\ (n\geq 0)$ отдельно от предметной области D, выписывают дополнительные аксиомы, которым должно удовлетворить F, чтобы на D можно было бы задать некоторую УРС, образующую то, что эти авторы называют базисной теорией рекурсивных функций (BRFT).

Аксиоматика Стронга:

(1) F содержит константные функции для любого элемента из D и функции выбора аргумента от любого числа аргументов;

(2) F содержит характеристическую функцию проверки равенства константе;

(3) F замкнуто относительно подстановки;

(4) Для каждого $m>0$ в F имеется универсальная функция для функций из F от m аргументов;

(5) Для каждых $m,n>0$ в F имеется $s-m-n$ -функция (универсальная функция для смешанных вычислений).

Аксиоматика Фридмана:

(1) D имеет не менее двух элементов;

(2) F содержит не более чем двуместные функции над D;

(3) F замкнуто относительно подстановки;

(4) F содержит тождественную функцию и функции пересчета пар;

(5) F содержит все одноместные функции-константы;

(6) F содержит характеристическую функцию равенства;

(7) F содержит универсальную функцию для одноместных функций.

В последней аксиоматике отсутствует функция смешанного вычисления, а ее отсутствие компенсируется тонкими отличиями в других аксиомах. Фридман далее доказывает, что аксиомы BRFT обладают свойством категоричности (относительно фиксированной D): любые две гёделизации, удовлетворяющие аксиомам BRFT, создают изоморфные классы функций.

233-318

Итеративные комбинаторные пространства. Свердлов, 1976, 1980 рассматривает абстрактные функциональные пространства в виде частично упорядоченных полугрупп, т.е. содержащих тождественную функцию, замкнутых относительно композиции и с отношением частичного порядка: $f \leq g \Leftrightarrow$ график $f \subseteq$ график g. Функциональное пространство $\mathcal{F}$ называется комбинаторным итеративным пространством, если оно:

1) содержит константные функции, образующие множество C, содержащее два выделенных элемента (истина и ложь);

2) на $\mathcal{F}$ задана операция пересчета пар, при этом функции взятия элементов пары принадлежат $\mathcal{F}$;

3) C замкнуто относительно пересчета пар;

4) на $\mathcal{F}$ задана операция если-то-иначе;

5) на $\mathcal{F}$ задана операция взятия неподвижной точки уравнения $\psi =$ если χ то I иначе $\psi\varphi$, где I – тождественная функция (итерация).

Если Φ – конечный набор элементов из $\mathcal{F}$, то любой элемент из $\mathcal{F}$, который получается из $\Phi \cup \{$тождественная функция, взятие элементов пары, истина, ложь$\}$ с помощью применения конечного числа операций, заданных на $\mathcal{F}$, называется функцией, рекурсивной в Φ. Для этих функций можно построить теорию рекурсии, включающую аналоги теорем о нормальной форме, 1-й и 2-й теоремам о рекурсии и теореме о нумерации.

Вычислительные теории. Московакис, 1969б предложил новый подход к аксиоматизации вычислимости, использующий понятие вычисления. Он требует, чтобы предметная область D была бы "вычислительной областью", т.е. содержала бы в себе множество C программ функций (индексов), в свою очередь содержащее в себе образ $\mathcal{N}$ на-

233 - 319

турального ряда $\mathcal{N}$, а на C был бы задан пересчет пар. Затем Мосховакис вводит центральное понятие вычисления как набора элементов из D вида $(e,x_1,\ldots,x_n,y)$, где $e\in C$, а набор объявляет, что программа e вычисляет y по $x_1,\ldots,x_n$. Для некоторого множества Θ функция $f(x_1,\ldots,x_n)$ (называется) Θ-предвычислимой если существует такое $e\in C$, что для любой точки $x_1,\ldots,x_n,y$ графика f вычисление $(e,x_1,\ldots,x_n,y)$ принадлежит Θ. Благодаря постулированному свойству рефлексивности тотальным функционалам и операторам можно сопоставить функции и, стало быть, перенести на них понятие Θ-предвычислимости. Множество вычислений Θ называется предвычислительной теорией, если следующие функции, функционалы и операторы оказываются Θ-предвычислимыми:

1) константы, тождественная функция и функция "счета" в $\mathcal{N}$,

2) характеристическая функция множеств C и $\mathcal{N}$,

3) функции пересчета пар на C,

4) подстановка и примитивная рекурсия по $\mathcal{N}$,

5) универсальные функции вычисления, частичного вычисления ($s-m-n$-функция) и вычисления с перестановкой аргументов.

Для того, чтобы охарактеризовать класс вычислимых функций, Мосховакис постулирует, что каждое вычисление обладает длиной, т.е. ординалом (конечным или бесконечным). При переходе от предвычислительной теории к вычислительной возникает различие между функциями и функционалами. Предвычислимая функция является вычислимой всегда, а предвычислимый функционал Γ объявляется вычислимым, только если длина вычисления Γ, грубо говоря, не меньше длин вычисления функций, индексы которых являются аргументами вычисления функционала. Кроме того, чтобы предвычислительная теория была бы вычислительной, требуется (— по определению —) чтобы длина вычисления по уни-

версальной функции была бы, как скажут программисты, меньше длины частичесго вычисления и последующего вычисления по остаточной программе (ср. Ершов, 1980).

Московакис показал, что вычисления на машинах Тьюринга образуют вычислительную теорию и установил справедливость I-й теоремы о рекурсии для вычислительных теорий.

Фенстад, 1974, рассматривая множество вычислений $\Theta = \{e \mathfrak{S} y\}$, где $\mathfrak{S}$ - набор аргументов, вводит вместо длины вычисления транзитивное отношение "быть подвычислением", т.е. $(e \mathfrak{S} y) < (e' \mathfrak{S}' y')$ значит, что вычисление y из $\mathfrak{S}$ по программе e является частью вычисления y из $\mathfrak{S}'$ по программе e. Множество вычислений с указанным отношением $\langle \Theta, < \rangle$ называется вычислительной структурой, если для любого вычисления множество его подвычислений конечно. Вычислительная структура объявляется вычислительной теорией, если она удовлетворяет определениям Московакиса, 1969б, где вместо неравенств между длинами вычислений постулируются соответствующие отношения быть подвычислением.

Фенстад, 1978 строит множество "рекурсивных" функций в абстрактном функциональном пространстве в виде замыкания $R_\omega(\Phi)$, где Φ - базовое (опорное) множество функций, как наименьшее множество функций, замкнутое относительно операторов композиции, взятия неподвижной точки, содержащее правило определения функции если-то-иначе и содержащее следующие "комбинаторы" - термин, восходящий к Карри, 1930: $I(f)=f$, $K(f,g)=f$ и $S(f,g,h)=f(h)(g(h))$. Он показывает, что $R_\omega(\Phi)$ представимо в виде вычислительной теории.

ВЫЧИСЛИМОСТЬ НАД АЛГЕБРАИЧЕСКИМИ СИСТЕМАМИ

Ниже мы будем использовать одни и те же обозначения для абстрактных алгебраических систем $A = \langle D, C, \Phi, \Pi, R \rangle$, где $R : \Phi \cup \Pi \to \mathcal{N}$.

Здесь R – тип алгебраической системы, $\Phi = \{\varphi_1, \ldots, \varphi_m\}$ $(m \geq 0)$ – функциональные символы, $\Pi = \{\pi_1, \ldots, \pi_n\}$ $(n \geq 0)$ – предикатные символы, образующие в совокупности сигнатуру системы A; D – носитель (предметная область), C – константы носителя. При этом C и/или Φ могут отсутствовать. Систему без Φ будем иногда называть реляционной системой, оставляя слово модель для теоретико-модельного употребления. Обозначение типа, т.е. R, будет для краткости, как правило, опускаться. Суперпозиции операций из Φ с аргументами-константами и переменными будут называться функциональными термами. Предикатный символ с функциональными термами в качестве аргументов называется предикатным термом. В некоторых случаях отдельным операциям и предикатам будет придан конкретный смысл, например, тождественная функция или предикат равенства. Саму систему мы будем иногда называть базовой для создаваемого ей класса функций. Поскольку большинство определений абстрактной вычислимости имеют прототипы среди арифметических функций, группировка разных определений будет аналогичной.

Нумерационная вычислимость. Идея определения абстрактной вычислимости, непосредственно опираясь на арифметические вычислимые функции, принадлежит Мальцеву, 1961. Рассматривая произвольную алгебраическую систему $\langle D, \Phi, \Pi \rangle$, Мальцев постулирует существование для D нумерации α, т.е. отображения $\alpha : \mathcal{N} \to D$, которое позволяет связать произвольную функцию $f : D^r \to D$ с некоторой арифметической $F : \mathcal{N}^r \to \mathcal{N}$ с помощью соотношения

$$f(\alpha x_1, \ldots, \alpha x_r) = \alpha F(x_1, \ldots, x_r).$$

Наличие нумерации позволяет переносить на функции f свойства вычислимости их прообразов F. Эту идею развил Лакомб, 1969, рассматривая абстрактную вычислимость в реляционных системах с

233-322

равенством $\langle D, \Pi, = \rangle$, где D – нумерованный носитель. Предикат $P(x_1,\ldots,x_r)$ абстрактно вычислим в Π, если он рекурсивен для любой нумерации D. Для целей нашего анализа важно подчеркнуть, что Лакомб получил чисто символическую характеризацию вычислимого (посредством) предиката $P(x_1,\ldots,x_r)$ примитивно рекурсивного множества булевых формул в алфавите $\Pi \cup \{P, =$, предметные переменные $x_1,\ldots,x_r$, конечный набор предметных констант$\}$. Корректность определения абстрактной вычислимости по Лакомбу подтверждается тем, что в $\langle N, 0, =, ' \rangle$ его вычислимость совпадает с обычной относительной вычислимостью.

Инвариантная определимость является обобщением изобразимости в формальной арифметике. Ее применение к определению абстрактной рекурсивности было предложено Крейзелом, 1963. Более конкретные определения были даны Фрассе, 1959. Рассматривается (для простоты) реляционная система $\langle D, \Pi \rangle$. В язык исчисления предикатов вводятся в качестве предикатных символов $\pi_1,\ldots,\pi_n$, символ определяемого предиката P, а также индивидуальные константы $\underline{x}$ для каждого элемента $x \in D$. Пусть $\Phi(P)$ – формула исчисления предикатов и пусть $C_\Phi(D,\Pi)$ – класс всех моделей формулы Φ, получаемых всевозможными расширениями базовой системы с сохранением носителя. Тогда предикат $P(x_1,\ldots,x_n)$, который сохраняет свои значения на каждом расширении из $C_\Phi(D,\Pi)$, есть предикат, инвариантно определимый посредством $\Phi(P)$. Поскольку в реляционной системе $\langle N, \Pi, = \rangle$ инвариантная определимость совпадает с относительной вычислимостью, естественно считать ее определением вычислимости в $\langle D, \Pi \rangle$.

Полнота исчисления предикатов позволяет дать инвариантной определимости следующую эквивалентную и чисто символическую форму. С каждым τ-местным предикатным символом π связывается счет-

ное множество формул $\Delta(\mathfrak{T})$ вида $\mathfrak{r}(cx_1,\ldots,cx_\ell)$ или $\neg\mathfrak{r}(cx_1,\ldots,cx_\ell)$ в зависимости от истинности или ложности $\mathfrak{r}(x_1,\ldots,x_\ell)$. Пусть $\Delta(\Pi)=\Delta(\mathfrak{r}_1)\cup\ldots\cup\Delta(\mathfrak{r}_n)$. Это множество называется диаграммой алгебраической системы $\langle D,\Pi\rangle$. Тогда для любого вычислимого в $\langle D,\Pi\rangle$ предиката $P(x_1,\ldots,x_n)$ справедливо

$$P(x_1,\ldots,x_\ell)\Leftrightarrow\Delta(\Pi)\cup\{\Phi(P)\}\vdash P(cx_1,\ldots,cx_\ell)$$

и

$$\neg P(x_1,\ldots,x_\ell)\Leftrightarrow\Delta(\Pi)\cup\{\Phi(P)\}\vdash\neg P(cx_1,\ldots,cx_\ell).$$

Простая и поисковая вычислимости. Московакис, 1969 приходит к абстрактной вычислимости в произвольном множестве D путем обобщения определения частично рекурсивных функций по Клини. Расширяя D выделенным элементом 0, он постулирует существование функции пересчета пар $z=(x,y)$, отображающей $D^*\times D^*\to D^*$ (D^* - замыкание $D\cup\{0\}$ относительно (x,y)), а также функций разбора пары. Предполагается далее существование предикатов равенства элементу 0 и принадлежности элемента из D^* множеству D. После этого в D^* выделяется образ $\mathcal{N}$ натурального ряда (сплинтер) 0,1,2,... в виде объектов 0, (0,0), ((0,0),0) и т.д. с образом функции счета в виде $n+1\sim(n,0)$. С использованием этого сплинтера Московакис описывает класс "примитивно-рекурсивных" функций со значениями и аргументами из D^* относительно некоторого списка $(\Phi)=\{\varphi_1,\ldots,\varphi_m\}$ произвольных функций над D^*. Характерной особенностью построения является то, что (Φ) может содержать многозначные функции и что построение класса функций сопровождается построением их программ-индексов - последовательностей натуральных чисел, сохраняющих информацию о строении функции и упакованных с помощью пересчета пар в объекты из $\mathcal{N}$. Показывается, что так определенные "примитивно-рекурсивные" функции относительно пустого (Φ) модели-

руют арифметические примитивно рекурсивные функции.

Московакис приходит к "частично рекурсивным" функциям, постулируя вычислимость универсальной функции над D^*. При этом, однако, обеспечивается, что эта функция определена лишь в том случае, если первый аргумент этой функции оказывается индексом, построенным в соответствии с заданными индуктивными правилами. Необходимость проводить "синтаксический анализ" индекса требует допущения операций поиска (аналогичным поиску в μ-операторе для арифметических функций) во множестве, которое может быть как упорядоченным, так и нет. В первом случае получается класс PC, или так называемая простая вычислимость, а во втором случае - класс SC, или поисковая вычислимость.

Для так определенных классов вычислимых функций Московакис строит элементарную теорию, включающую теоремы о нормальной форме, теоремы о рекурсии, теоремы о замкнутости и т.п.

Скордев, 1980 находит алгебраизированную характеризацию простой и поисковой вычислимостей для случая бинарных отношений в $D^* \times D^*$, т.е. многозначных функций одного аргумента, которая более четко устанавливает связь с частично рекурсивными функциями, представленными в виде алгебры Робинсона. Пусть $\mathcal{F}$ - множество всех бинарных отношений в $D^* \times D^*$. Для любых двух бинарных отношений φ и ψ определяются три базовых операции: обычная композиция, сочетание $\{(p,s): \exists q \exists r((p,q) \in \varphi \& (p,r) \in \psi \& (q,r) = s\}$ и итерация (грубо говоря, композиция φ с самим о собой, пока ψ, рассматриваемая как функция, не окажется равной нулю на очередном звене возникающей цепочки значений). Пусть, далее, даны произвольные бинарные отношения $\varphi_1, \ldots, \varphi_m$ и три базовых отношения $L =$ "p есть первый элемент пары q", $R =$ "p есть второй элемент

233 – 325

пары φ" и "максимальное" отношение $U = D^* \times D^*$. Тогда замыкание $\varphi_1, \ldots, \varphi_m, L$ и R тремя указанными операциями дает $\mathcal{F}_0$PC, а то же замыкание $\varphi_1, \ldots, \varphi_m, L, R$ и U дает $\mathcal{F}_0$SC .

Тьюринговы вычисления в произвольной области. Фридман, 1969а описал модель вычислений T, которая обобщает понятие машины Тьюринга на произвольную алгебраическую систему $\langle D, C, \Phi, \Pi \rangle$. Обобщение носит весьма прямолинейный характер: на одну клетку ленты помещается один элемент из D или вспомогательный символ. Все функции, вычисляемые T-машинами, называются T-рекурсивными. Фридман вводит важную характеризацию T-рекурсивных функций с помощью модели Д так называемых эффективных определений, которая обобщает определение функции разбором случаев на бесконечное, но перечислимое (в смысле арифметической перечислимости, подразумевающей некоторую нумерацию) множество случаев. Отдельным случаем у Фридмана является "клауза" вида $p \to t$, где p – либо пусто, либо непротиворечивая конъюнкция предикатных термов или их отрицаний, а t – функциональный терм. Отсутствие p означает тождественную истину. Роль эффективного определения играет перечислимое множество клауз, такое, что конъюнкция условий двух разных клауз всегда противоречива. Таким образом одна клауза "отвечает" за одну точку графика функции: для того, чтобы вычислить $\varphi(x_1, \ldots, x_n)$, перечисляем клаузы, пока не найдем такую $p \to t$, что $p(x_1, \ldots, x_n)$. Тогда $\varphi(x_1, \ldots, x_n) = t(x_1, \ldots, x_n)$. Фридман показывает, что каждое вычисление в T-модели создает протокол, совокупность которых образует эффективное определение. Обратное доказательство Д→T апеллирует к тезису Чёрча (в применении к модели T) и к перечислимости эффективного определения.

Фридман изучает, каким дополнительным свойствам должны удов-

летворять Т-рекурсивные функции для выполнения основных теорем элементарной общей теории вычислимости. Этих свойств оказывается три:

(I) Т-рекурсивность предиката равенства,

(2) Т-рекурсивность пересчета пар,

(3) конечная порождаемость $\mathcal{D}$ с помощью Т-рекурсивных тотальных функций.

В дальнейшем он показывает, что (1)& (3)⇒(2).

Операторные алгоритмы и рекурсивные программы. Эти модели с самого начала были введены для определения вычислимости в любой алгебраической системе (Ершов, 1960; Маккарти, 1963). В операторных алгоритмах класс вычислимых функций над алгебраической системой $\langle \mathcal{D}, C, \Phi, \Pi \rangle$ – это все функции, программируемые в "системе команд" (Φ, Π). В рекурсивных программах класс вычислимых функций – это наименьшие неподвижные точки рекурсивных программ, строящихся из условных термов в сигнатуре базовой системы. В операторных алгоритмах рассматриваются классы программ, включающих некоторые действия в натуральном ряду. Наиболее частые варианты – это схемы со счетчиками, т.е. с операциями $x+1$, $x-1$ и предикатом $x=0$. Счетчики используются либо для кодирования, либо в программах с массивами – односторонними лентами с прямым доступом в любую клетку по ее номеру.

Общая теория вычислимости в этих моделях практически не развивалась, однако в интересах теоретического программирования в этих моделях внимательно изучались инвариантные, или абстрактные, свойства программ, т.е. такие, которые выполняются в любой алгебраической системе с данной сигнатурой. Такие свойства естественно приписывать не функциям, вычисляемым программам, а самим прог-

233-327

раммам, или, более точно, их схемам, т.е. синтаксическим конструкциям, выражающим программы в соответствующем языке программирования.

Для определенности остановимся на схемах операторных алгоритмов, (эти схемы, впервые рассмотренные Криницким 1959, 1970, В литературе получили впоследствии название стандартных схем или граф-схем с памятью. Распространим их конструкцию на случай произвольных систем $\langle D, C, \Phi, \Pi \rangle$. Роль логических условий играют произвольные предикатные термы и их булевы комбинации; операторы присваивания имеют вид $y := t$, где t – функциональные термы с операциями из Φ. Важнейшим отношением для схем программ является отношение функциональной эквивалентности: две схемы S_1 и S_2 в сигнатуре Φ, Π функционально эквивалентны, если они вычисляют одну и ту же функцию для любой алгебраической системы с данной сигнатурой. Для некоторых сигнатур и классов схем это отношение оказывается разрешимым. Янов, 1957 и Ратледж, 1964 установили это для сигнатур с унарными операциями и схем, использующих только одну переменную и предикатные термы без функциональных символов. Для произвольных сигнатур это отношение разрешимо только для схем без циклов или непосредственно к ним сводящихся (Криницкий, 1959; Игараши, 1968). В общем же случае отношение функциональной эквивалентности оказалось неразрешимым (Патерсон, 1968; Летичевский, 1969).

С другой стороны изучение схем программ позволило сделать одно весьма интересное наблюдение. Придадим в схеме программы предикатным символам произвольные (истинностные) значения. В этом случае из текста программы непосредственно извлекается конкретная вычислительная последовательность, протокол, или прокрутка, как

233 - 328

говорят программисты. При этом с непосредственной очевидностью обнаруживаются три альтернативы: либо этот протокол значим, т.е. по нему в некоторой конкретной алгебраической системе может быть проведено вычисление, либо он логически противоречив, либо он бесконечен. Такие протоколы можно с той или иной степенью детальности свернуть в один или несколько термов в сигнатуре (Φ,Π) с использованием употребленных в программе символов констант и переменных. Заметим, что эти термы не содержат констант, обозначающих значения входных переменных, но содержат приписанные предикатным символам значения. Заметим, что в общем случае значения приписываются не вхождениям предикатных символов в программу, а вхождениям их в протокол. Таким образом у нас появляется порождающий процесс, управляемый выбором значений предикатов, результатом которого является бесконечное множество термов в сигнатуре (Φ,Π). Это множество назовем формальной разверткой схемы программы. Слово "формальный" подчеркивает то, что при его построении не используется какая бы то ни было интерпретация функциональных и предикатных символов. Чисто символический характер формальной развертки позволяет говорить о языке, воспринимающем это множество.

Некоторые классы формальных разверток продемонстрировали два принципиально важных свойства:

(1) Они полностью характеризуют функциональное назначение программы, т.е.

а) при заданной интерпретации операций и заданных значениях входных переменных и при порождении формальных протоколов из формальной развертки однозначно выхватывается протокол, вычисляющий нужное значение функции;

б) схемы, имеющие одинаковые формальные развертки, оказываются функционально эквивалентными.

Такие формальные развертки удобно называть детерминантами схем программ.

(2) Некоторые детерминанты воспринимаются суперрекурсивными языками, существенно более простыми, нежели языки, воспринимаемыми перечислимые множества.

Первый пример такого детерминанта привел Янов, 1957 для вышеуказанного подкласса схем, получивших название схем Янова. Для его сигнатуры $\Phi = \{A_1, \ldots, A_n\}$ и $\Pi = \{p_1, \ldots, p_m\}$ детерминант был образован последовательностями вида

$$(\sigma_1^{(a)} \ldots \sigma_m^{(a)}) A_{i_a} \ldots (\sigma_1^{(k)} \ldots \sigma_m^{(k)}) A_{i_k} \ldots$$

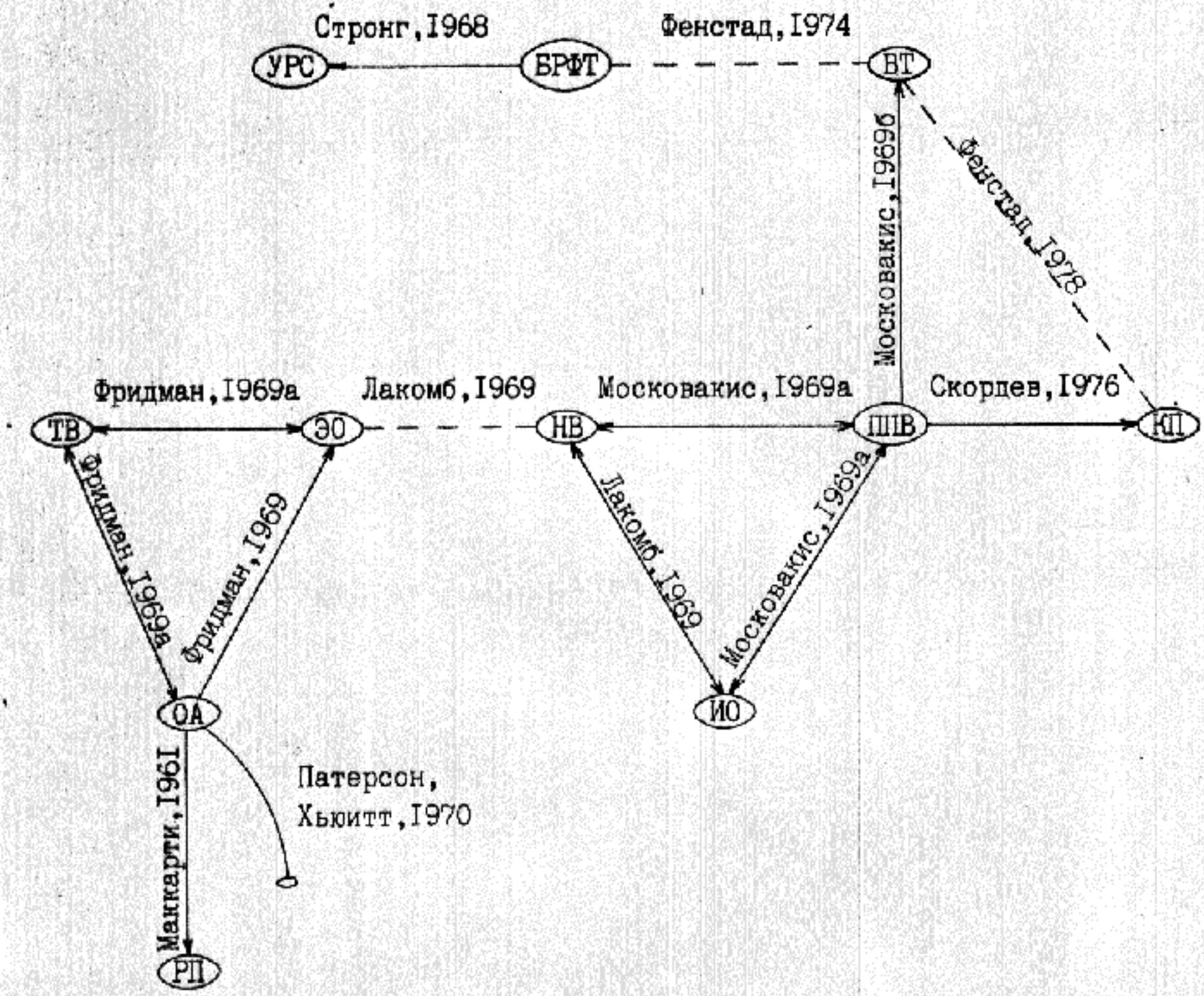
где σ_j – это p_j или $\neg p_j$, и воспринимался конечным автоматом. Это, в частности, означало, что свойство схем Янова иметь одинаковый детерминант оказывается разрешимым, поскольку разрешима эквивалентность двух конечных автоматов.

Детерминант для произвольных схем программ был построен Иткиным, 1972. Его протоколы строились следующим образом: бралося произвольный путь от входа до выхода из схемы программы. Каждому вхождению предикатного символа в этот путь присваивалось значение, предписанное данным путем (входная вершина формально считалась предикатным символом, аргументами которого были все переменные программы). В аргументы каждого предикатного символа ставилось его "термальное значение" – функциональный терм с аргументами из констант и входных переменных, полученный путем замыкания подстановок, вызываемых операторами присваивания, попавшими на выбранный путь. Этот детерминант воспринимается двухленточным конечным автоматом и, стало быть, тоже разрешим.

ПОСТАНАЛИЗ

Прежде всего заметим, что в работах по абстрактной и обобщенной вычислимости, равно как и в теоретико-программных работах,

233-330



Фиг.2. Схема связи определений абстрактной вычислимости

УРС - униформные рефлексивные структуры (Вагнер, I963, I969)
БРФТ - ~~базовые~~ записные теории рекурсивных функций (Стронг, I968; Фридман, I969)
ВТ - Вычислительные теории (Московакис, I969б; Фенстад, I974, I978)
КП - комбинаторные пространства (Скордев, I974, I980)
ППВ - простая и поисковая вычислимости (Московакис, I969)
ИО - инвариантная определимость (Фрассе, I959; Крайзел, I963; Лакомб, I969; Московакис, I969а)
НВ - нумерационная вычислимость (Мальцев, I96I; Лакомб, I969)
ТВ - Тьюринговы вычисления (Фридман, I969а)
ЭО - эффективная определимость (Фридман, I969а)
ОА - операторные алгоритмы (Ершов, I958, I960; Криницкий, I959; Патерсон, I968)
РП - рекурсивные программы (Маккарти, I96I; Манна, I974)

233 – 331

содержится гораздо больше идей и тонкостей, нежели те, которые автор в состоянии был усмотреть и обсудить. Приходится сожалеть, что такой интересный материал рассеян, в основном, по трудам конференций и журналам и в целом с трудом доступен одному читателю. Поэтому, и критика и апология, которые содержатся в этом разделе, будут неизбежно поверхностными.

Объем класса абстрактно вычислимых функций. На фиг. 2, аналогично 1-й части статьи, показана схема эквивалентностей и сводимостей разных определений абстрактной вычислимости. Автор не уверен, что схема исчерпывающа. Сделаем к ней некоторые замечания.

Определения абстрактной вычислимости имеют разные формальные параметры, создавая конструкции разной общности и часто при несовпадающих предпосылках. Некоторые эквивалентности доказываются при дополнительных условиях.

УРС и БРФТ – это системы разной степени общности. В УРС нумерация задана изначально, в БРФТ ее построение обеспечивается более подробными аксиомами. Связь ВТ с БРФТ лишь только обозначена Фенстадом, 1974. Не исключено, однако, что такая связь изучается в книге Фенстада, 1978а, которая была автору недоступна. От себя добавим, что должен существовать сравнительно прямой способ вложения ПВ в БРФТ в силу явного построения индексов и постулирования вычислимости универсальной функции. Вариант вычислительной теории, предложенный Фенстадом, 1978, сближается с КП.

Думается, что треугольники сводимостей НВ–ПВ–ИО и ТВ–(ОА, РП)–ЭО достоверно и интуитивно убедительно формируют объем класса функций, вычислимых в некоторой алгебраической системе $\langle D, C, \Phi, \Pi, R \rangle$. При связи ПВ–ИО и ПВ–НВ Московакис, 1969а включает со стороны ПВ в сигнатуру Π предикат равенства. Во

всех трех моделях НПВ, НО и НЗ определение абстрактной вычислимости может быть релятивизировано к любому набору констант C из D. При изучении сводимостей оказывается весьма удобной наличие для НВ-вычислимых функций примитивно рекурсивного детерминанта из булевых формул (мы можем использовать этот термин, т.к. эта характеризация, установленная Лакомбом вполне соответствует определению детерминанта).

Эффективные определения Фридмана, 1969а - это просто задание абстрактно вычислимой функции ее перечислимым детерминантом. Связи ТВ→ЭО и ОА→ЭО устанавливаются простой демонстрацией того факта, что протоколы вычислений в ТВ и ОА непосредственно транслируются в перечислимый детерминант, образованный клаузами Фридмана. При доказательстве ЭО→ТВ Фридман апеллирует к тезису Чёрча (в применении к модели Т) и к перечислимости ЭО-детерминанта. В изучении ОА Фридман, 1969а использует свой вариант модели ОА и отмечает, что программисты обратили его внимание на связь его модели с моделью граф-схем с памятью в варианте Патерсона, 1968. В связи ТВ→ОА Фридман использует схемы программ со счетчиками, поскольку ОА-программы могут использовать только ограниченное, независимое от вычислений, число ячеек памяти, а также постулирует наличие предиката равенства. Счетчики моделируют перебор и вычисление по геделевым номерам клауз детерминанта. ОА и РП без дополнительных требований к сигнатуре обладают разной силой вычислимости: ОА сводится к РП в той же сигнатуре, однако, как показали Патерсон и Хьюитт, 1970 существует РП, для которой нет эквивалентного ОА в той же сигнатуре (на самом деле доказывается отсутствие ОА с тем же детерминантом).

233-333

Связь ЭО и НВ аккуратно не прослежена, но трансляция Клини Фридмана в булевый детерминант Лакомба при наличии предиката равенства очевидных трудностей не обнаруживает.

Вложимость арифметических моделей. Практически во всех моделях абстрактной вычислимости демонстрируется буквальная вложимость той или иной модели арифметической вычислимости, но не каждой. Мы ограничимся констатациями, хотя более глубокий анализ был бы весьма интересен.

Изобразимость по Гёделю находит свое прямое обобщение в инвариантной определимости.

Рекурсивная определимость по Эрбрану, Гедель допускает тривиальное обобщение на систему вида $\langle D, \Phi, = \rangle$, если ввести в исчисление равенств в виде аксиом счетное множество равенств, представляющее диаграмму системы $\langle D, \Phi, = \rangle$, т.е. равенств вида $\varphi_i(c\underset{\sim}{x}_1, \ldots, c\underset{\sim}{x}_{\eta_i}) = c\underset{\sim}{y}$, выражающих информацию о том, что $\varphi_i(\underset{\sim}{x}_1, \ldots, \underset{\sim}{x}_{\eta_i}) = \underset{\sim}{y}$. В литературе по программированию (например, Дарлингтон, 1978) рассматривается исчисление равенств для произвольной системы $\langle D, \Phi, \Pi, = \rangle$, в которой рекурсивное описание записывается в синтаксисе условных термов Маккарти, 1961, а правила вывода называются "переписываниями". Теория этой модели пока небогата: в близких формализмах доказана теорема о неподвижной точке (Манна, 1974), схема смешанных вычислений, или аналог $s-m-n$-теоремы (Ершов, 1978).

Лямбда-определимость по Черчу сама по себе является абстрактной моделью, в которую вкладывается арифметическая вычислимость путем отождествлений выделенных термов с элементами системы $\langle N, 0, =, ' \rangle$. Автор не нашел, однако, аналогичного прямого вложения произвольных систем в тот или иной вариант λ-исчисления.

233-334

Следует, однако, отметить его очевидную связь с теориями абстрактной вычислимости в большом (УРС, БРЭТ). Стронг, 1968 рассматривает свою теорию как вариант комбинаторной логики Карри, 1930, Карри и Фейс, 1958.

Частично рекурсивные функции по Клини непосредственно обобщаются в простую и поисковую вычислимости.

Машины Тьюринга со временем были обобщены на теорию вычислимости в алфавитных функциях (ср. Марков, 1951). Обобщение на Т-модель Фридмана, 1969а следует признать весьма прямолинейным.

Операторные алгоритмы и рекурсивные программы с самого начала были определены для произвольной алгебраической системы.

Пока что нет никакого подхода к тому, чтобы определить какой-то аналог диофантовых множеств для произвольных предметных областей. В то же время такое обобщение было бы весьма интересно, т.к. диофантовы множества в очень чистой форме отделяют три основных стороны эффективных вычислений: комбинаторный перебор (аргументов и параметров многочлена), прямое вычисление (значение многочлена) и отождествление (проверка на равенство). Кроме того, такое обобщение позволит, по-видимому, лучше разобраться в кодирующих свойствах многочленов.

Несколько слов о полноте воспроизведения общей теории вычислимости. В той или иной модели найдены разные формулировки всех основных теорем, особенно в БРЭТ, КП, ПШВ и ТВ. Производит большое впечатление доказательная сила аксиом УРС. В то же время бросается в глаза некоторая произвольность исходных посылок в построении абстрактной теории вычислимости. В теории, если можно так выразиться, преобладает наблюдательный подход.

233-335

Абстрагированность от арифметической вычислимости. Анализ этого обстоятельства обнажает самые чувствительные места большинства работ по абстрактной вычислимости. Многие авторы (например, Крейзел, 1969; Фридман, 1969а; Грильо, 1974) обращают внимание на большую трудность отстраниться от явной или скрытой опоры на арифметическую вычислимость. Наблюдая цитированные работы, можно найти два главных употребления арифметической вычислимости. Одно из них – это выделение в предметной области образа натурального ряда (сплинтера) как средства обеспечения свойств рефлексивности (нумеруемости) или как средства пересчета предметной области. Второе – это определение абстрактно рекурсивных функций с использованием понятия перечислимого множества (см. перечислимость детерминантов у Лакомба в НВ и Фридмана в ЭО). Оба использования, по мнению автора, скрывают сущности. Первое уводит в сторону от выяснения абстрактных условий кодируемости программ и представления количеств в символической форме. Второе – вообще отбрасывает поиски сущности вычислимости на исходные позиции.

Если не говорить о теории абстрактной вычислимости в большом (УРС, БРФТ), наиболее фундаментальным подходом представляется инвариантная определимость. Правда, из соображений скорее философского, нежели технического характера, этой точке зрения может быть противопоставлено предпочтение какой-нибудь чисто алгоритмической модели типа ОА или ТВ (см., например, Цейтин, 1981). В любом варианте, однако, остается пока что открытая проблема обеспечения рефлексивности или – более общо – придания предметной области свойства быть носителем информации, свойства, выраженного в терминах сигнатуры базовой системы.

233-336

Абстрагированность от синтаксиса программ. До последнего времени эта задача в ее математическом выражении, похоже, даже не возникала. Как ни странно, впервые обратили внимание на внепрограммное представление программных сущностей сами программисты. В теоретическом программировании постепенно созрел взгляд на программу как на упаковку множества вычислений, возникающих при исполнении программы для разных ее входных данных. Это множество вычислений оказывалось более устойчивым объектом, нежели разные отличья программы. В то же время некоторые свойства программы находили в этом множестве более явное выражение, нежели в ее конкретном тексте. Так появились понятия разных программных инвариантов, в том числе и уже упомянутые детерминанты. В рассмотренных статьях по абстрактной вычислимости мы находим аналогичные конструкции только в двух моделях: клаузы Фридмана в ЭО и булевы характеризации Лекомба в НВ. Естественно, не надо забывать и о перечислении графика вычислимой функции элементарными функциями по Кальмару, 1943.

Другую форму абстрагируемости от программ мы находим в вычислительных теориях. Эта абстракция имеет свои преимущества, однако она не позволяет говорить об индивидуальных функциях.

Еще один подход к абстрактной вычислимости. После всего сказанного нам очень легко выразить основную идею: взять понятие детерминанта за определение вычислимой функции. Вторая часть основной идеи – строить детерминант для сигнатуры, а не для конкретной алгебраической системы, т.е. не привлекая интерпретации операционных символов. Дадим более точную схему определения.

Пусть дана алгебраическая система $A=\langle D, C, \Phi, \Pi, R\rangle$. Пусть $X_n=\{x_1,\dots,x_n\}$ – символы переменных. Определим множество T_A^n

вычислительных термов ($n \geq 0$). Любая константа $c \in C$ и любая переменная $x_i \in X_n$ – это вычислительный терм. Если φ_i – функциональный символ и $R(\varphi_i)=r$ и $t_1,\dots,t_r$ – вычислительные термы, то $\varphi_i(t_1,\dots,t_r)$ – вычислительный терм. Если π_j – предикатный символ и $R(\pi_j)=s$ и $t_1,\dots,t_s$ – вычислительные термы, то $\pi_j(t_1,\dots,t_s)$ – предикатный терм. Если p – предикатний терм и t – вычислительний терм, то $(p:t)$ и $(\neg p:t)$ – вычислительные термы. Определим для вычислительных термов функцию оценки val. Если x значение константы cx или переменной x, то $val\ cx=x$, $val\ x=x$. Далее, $val\ \varphi_i(t_1,\dots,t_r)=\varphi_i(val\ t_1,\dots,val\ t_r)$; $val\ \pi_j(t_1,\dots,t_s)=\pi_j(val\ t_1,\dots,val\ t_s)$. Наконец,

$val\ (p:t)=$ **если** $val\ p$ **то** $val\ t$ **иначе** не определен,

$val\ (\neg p:t)=$ **если** p **то** не определен **иначе** $val\ t$

Значение любого из термов не определено, если какой-либо из его аргументов не определен.

Схема определения. Функция $f: D^n \to D$ называется вычислимой в A, если существует конечно порожденное множество $Det_f \subseteq T_A^n$ (детерминант функции f), такое, что

(1) $\forall (x_1,\dots,x_n,y) \in f\ \exists t(x_1,\dots,x_n) \in Det_f : val\ t\ (x_1,\dots,x_n)=y$;

(2) $\forall (x_1,\dots,x_n,y) \forall t(x_1,\dots,x_n) \in Det_f : val\ t\ (x_1,\dots,x_n)=y \Rightarrow (x_1,\dots,x_n,y) \in f$.

Заметим, что это определение не исключает многозначных функций.

Естественно, главный пробел в этой схеме определения – это неуказанность способа порождения детерминанта. Опираясь на результаты из теории схем программ, мы надеемся, что это может быть какой-либо автоматный язык, благодаря чисто комбинаторному способу порождения детерминантов. Напомним, что для ОА детерминантом может быть конечний двухленточний автомат. Для рекурсивных прог-

233-338

рами Розен, 1975 показал, что детерминант, весьма похожий на тот, описывается контекстно-свободным языком. Этот результат, однако еще не может считаться окончательным: [illegible] КС-грамматики имеет неразрешимую проблему эквивалентности, а разрешимость детерминантов рекурсивных программ - открытая проблема с шансами на успех (см. последний результат Сабельфельда, 1981).

Заметим, что это определение чисто синтаксическое. Оно задает пучек вычислимых функций, которые для разных интерпретаций операционных символов будут иметь один и тот же детерминант, а рисунок графиков будет помимо конкретных значений отличаться еще и различием областей определения. Даже в рамках одной интерпретации каждый детерминант может воплощаться в бесконечном разнообразии программ.

Самым твердым орешком для такого рода определения будет реализация свойств рефлексивности. До сих пор не ясно, можно ли будет найти схемное доказательство теоремы о вычислимости универсальной функции. Один из возможных подходов к этому - это использование понятия универсальных грамматик в теории формальных языков.

Другой подход - это аксиоматизация предметной области как множества конструктивных объектов. Такая аксиоматика по-видимому создаст на носителе некоторую присущую ему систему и отношений и конструкторов, образующих некоторую систему операций, которая через то или иное определение вычислимости будет создавать на данном носителе некоторый "абсолютный" класс вычислимых функций, который уже потом можно будет релятивизировать к любой системе с данным носителем.

Такой подход, однако, оставляет в стороне случай произвольных областей, для которых требование их конструктивности будет избыточным. В таком случае операции базовой сигнатуры становятся для

133 – 339

нас оракулами по существу, и нам не остается ничего другого, как чисто эвристически попробовать выразить их кодирующие свойства.

Автору представляется весьма интересным изучить вопрос порождения детерминантов из инвариантных определений вычислимых функций логическими средствами.

ЗАКЛЮЧЕНИЕ

Абстрактная теория вычислимости представляет интерес по многим показателям. Мы не будем повторять глубокий анализ Крайзела, 1969 и лишь добавим к его аргументам, что одно из серьезных употреблений такой теории будет лежать в программировании, где релятивизация свойств программы к заданной "системе команд" является одним из основных принципов. Другим принципом системного программирования является как можно более полное изучение схемных, т.е. инвариантных свойств программы.

Мы уже отмечали, что объем понятия абстрактно вычислимой функции уже нащупан. Однако логический анализ предпосылок абстрактной вычислимости может быть продолжен. Абстрактная теория вычислимости может сыграть свою роль в усилении позитивной стороны теории. Действительно, не так уж интересно разыскивать абстрактное обобщение неразрешимой арифметической задачи, хотя и здесь возможен познавательный прогресс за счет разделения сущностей. Другой, совершенно не затронутой стороной теории является разработка методов синтеза программ из разных форм априорного знания о задаче в разных предметных областях. Эта сторона проблемы, однако, требует своего анализа.

233–340

Литература

Ахо и др.

1976 Aho A.V., Hopcroft J.E., Ullman J.D. The design and analysis of computer algorithms. - Reading: Addison-Wesley, 1976. (Русский пер. под ред. Ю.В.Матиясевича: Ахо А., Хопкрофт Дж., Ульман Дж. Построение и анализ вычислительных алгоритмов. – М.: Мир, 1979, 536 с.).

Барвайз

1977 Barwise J. (Ed) Handbook of mathematical logic. Amsterdam: North-Holland, 1977, 1165 p.

Барендрегт

1977 Barendregt H. The type free lambda calculus. -In: J.Barwise (Ed) Handbook of mathematical logic. Amsterdam: North-holland, 1977, p. 1091-1132.

Бёрсталл

1969 Burstall R. Proving properties of programs by structural induction. - Comp. Journ., 12, 1969, p. 41-48.

Бём, Якопини

1966 Böhm C., Jacopini G. Flow diagrams, Turing machines and languages with only two formulation rules. - Comm. ACM, 9, 1966, no.5, p. 366-371.

Блюм

1967 Blum M. A machine-independent theory of the complexity of recursive functions. - Journ. ACM, 1967, v.14, p. 322-336.

Бэкус

1978 Backus J. Can programming be liberated from the von Neumann style? A functional style and its algebra of programs. - Comm. ACM, 21, 1978, no.8, p. 613-641.

Вагнер

1963 Wagner E.G. Uniformly reflexive structures: towards an abstract theory of computability. - Doctoral dissertation, Columbia University, 1963.

1969 Wagner E.G. Uniformly reflexive structures. - Information sciences, 1, 1969, p. 343-362.

Гёдель

1931 Gödel K. Über formal unentscheidbare Sätze der Principia mathematica und verwandter Systeme I. - Monatshefte für Math. und Phys., 38, s. 173-198. (English translation: Gödel K. On formally undecidable propositions of Principia mathematica and related systems I. -In: J. van Heijenoort (Ed) "From Frege to Gödel". Cambridge: Harward Univ. Press, 1967, p. 596-616.)

1934 Gödel K. On undecidable propositions of formal mathematical systems. Lecture notes by S.C.Kleene and J.B.Rosser. - Princeton, N.J.: The Inst. for Adv. Study, 1934, 30 p.

1936 Gödel K. Über die Länge von Beweisen. - Ergebnisse eines math. Koll. Heft 7, 1934-1935, S. 23-24.

1963 Gödel K. A letter to J. van Heijenoort dated 23 April 1963. -In: J. van Heijenoort (Ed) From Frege to Gödel. Cambridge, Mass.: Harward Univ. Press, 1967, p. 619.

Гриллъо

1974 Grilliot T.J. Dissecting abstract recursion. -In: Fenstad J.E., Hinman P.G. (Eds) Generalized recursion theory. Amsterdam: North-Holland, 1974, p. 405-420.

Гжегорчик

1953 Grzegorczyk A. Some classes of recursive functions. - Rozprawy Matematiçzne, IV, Warszawa, 1953. (Русский перевод: Гжегорчик А. Некоторые классы рекурсивных

функций. – В сб. "Проблемы математической логики" под ред. В.А.Кознидзеи и А.А.Кучнина. М.: "Мир", 1970, с. 9-49).

Дармингтон

1978 Darlington J. A synthesis of several sorting algorithms. – Acta Informatica, 1978, 11, p. 1-30.

Дейвис

1953 Davis M. Arithmetical problems and recursively enumerable predicates. – Journ. Symb. Logic, 18, 1953, no.1, p.33-41. (Русск. перевод: Дейвис М. Арифметические проблемы и рекурсивно перечислимые предикаты. – В сб. "Математика", 8, 1964, № 5, 15-22).

1973 Davis M. Speed-up theorems and Diophantine equations. -In: Courant Computer Science Symposium 7: Computational complexity. New York, 1973, p. 87-95.

Дейвис и др.

1976 Davis M., Matijasevič Yu., Robinson J. Hilbert's tenth problem. Diophantine equations: positive aspects of a negative solution. -In: Proceedings of Symposia in Pure Math., 28. Providence: Amer. Math. Soc., 1976, p. 323-378.

Джоунз

1975 Jones J.P. Diophantine representation of the Fibonacci numbers. (Fibonacci Quarterly, 13, 1975, p. 84-88.

Ершов

1958 Ершов А.П. Об операторных алгоритмах. – ДАН СССР, 122, 1958, № 6, с. 967-970.

1960 Ершов А.П. Операторные алгоритмы I. – В кн.: Проблемы кибернетики. Вып. 3. М.: Физматгиз, 1960, с. 5-48.

1977 Ершов А.П. Введение в теоретическое программирование (беседы о методе). – М.: Наука, 1977, 288 с.

1977a Ershov A.P. On the essence of compilation. -In: E.J.Neuhold (Ed) Formal description of programming concepts. Amsterdam: North-Holland, 1977, p. 391-420. (Русский вариант: А.П.Ершов. О сущности трансляции. – Программирование, 1977, № 5, с. 21-39).

1978 Ershov A.P. Mixed computation in the class of recursive program schemata. - Acta Cybernetica, 4, 1978, Fasc. 1, p. 19-23. (Русск. перевод: Ершов А.П. Смешанные вычисления в классе рекурсивных схем программ. – ДАН СССР, 245, 1979, № 5, с. 1041-1044).

1980 Ершов А.П. Смешанные вычисления: потенциальные применения и проблемы исследования. – В кн.: Всесоюзная конф. "Методы матем. логики в проблемах иск.интелл. и систематич. программирование". Часть 2. Паланга, 3-5 сент.1980г. Вильнюс: Ин-т матем. и киберн. АН ЛитССР, 1980, с.26-55.

1981 Ershov A.P. Abstract computability on algebraic structures. -In: A.P.Ershov, D.Knuth (Eds) Urgench'79 papers. Lecture Notes in Comp. Sci. Berlin: Springer, 1981 (in print).

Игараши

1968 Igarashi S. On the equivalence of programs represented by Algol-like statements. - Reprint Comp. Centre Univ. Tokyo, 1, 1968, no.1.

Иткин

1972 Иткин В.Э.Логико-термальная эквивалентность схем программ.– Кибернетика, 1972, № 1, с. 5-27.

Калмар

1943 Kalmár L. Egyszerü példa eldönthetlen aritmetikai problémára (Ein einfaches Beispiel für ein unentscheidbares arithmetisches Problem.) Matematikai és fizikai lapok, 50, 1943, s. 1-23.

Карри

1930 Curry H.B. Grundlagen der kombinatorischen Logic. - Amer. journ. of math., 52, 1930, p. 509-536, 789-834.

Карри и Фейс

1958 Curry H.B., Feys R. Combinatory logic. - Amsterdam: North-Holland, 1958.

Клини

1936 Kleene S.C. General recursive functions of natural numbers. - Math. Ann., 1936, 112, p. 727-742.

1936a Kleene S.C. λ-definability and recursiveness. - Duke mathematical journal, 1936, 2, p. 340-353.

1938 Kleene S.C. On notation for ordinal numbers. - Journ. symb. logic, 3, 1938, p. 150-155.

1943 Kleene S.C. Recursive predicates and quantifiers. - Trans. Amer. Math. Soc., 1943, 53, p. 41-73.

1945 Kleene S.C. On the interpretation of intuitionistic number theory. - Journ. symb. logic, 10, 1945, p.109-124.

1952 Kleene S.C. Introduction to metamathematics. - New York: Van Nostrand, 1952. (пер. с английского под ред. В.А.Успенского: Клини С.К. Введение в метаматематику. – М: Изд-во иностр. лит., 1957, 526 с.)

Констебл, Бородин

1972 Constable R.L., Borodin A.B. Subrecursive programming languages, Part I: efficiency and program structure. - Journ. ACM, 19, 1972, no.3, p. 526-568.

233-345

Котов

1978 Котов В.Е. Введение в теорию схем программ. – Новосибирск: Наука. Сибирское отделение, 1978, 256 с.

Крайзел

1963 Kreisel G. Model theoretic invariants: application to recursive and hyperarithmetic operations. -In: The theory of models. Proceedings of the 1963 International Symposium at Berkeley. Amsterdam: North-Holland, 1965, p. 190-205.

1969 Kreisel G. Some reasons for generalizing recursion theory. -In: Gandy R.O., Yates C.E.M. (Eds) Logic Colloquium'69. Amsterdam: North-Holland, 1971, p. 139-198.

Криницкий

1959 Криницкий Н.А. Равносильные преобразования логических схем. Автореферат диссертации. – М.: Московский государственный университет, 1959.

1970 Криницкий Н.А. Равносильные преобразования алгоритмов и программирование. – М.: Советское радио, 1970.

Лакомб

1969 Lacombe D. Recursion theoretic structures for relational systems. -In: Gandy R.O., Yates C.E.M. (Eds) Logic Colloquium'69. Amsterdam: North-Holland, 1971, p. 3-17.

Летичевский

1970 Летичевский А.А. Функциональная эквивалентность дискретных преобразователей II.– Кибернетика (Киев),1970,№2.

Ландин

1965 Landin P. A correspondence between Algol 60 and Church's lambda calculus. – Comm. ACM, 8, 1965, no.3, p. 158-165.

233-346

Маккарти

1960 McCarthy J. Recursive functions of symbolic expressions and their computation by machine, p.1. - Comm. ACM, 3, 1960, no.4, p. 184-195.

1961 McCarthy J. Towards a mathematical theory of computation. -In: Proc. IFIP Congress 1961. Amsterdam: North-Holland, 1962, p. 21-28.

Мальцев

1961 Мальцев А.И. Конструктивные алгебры. I - Усп. матем. наук, 16, 1961, вып. 3, с. 3-60.

Манин

1980 Манин Ю.И. Вычислимое и невычислимое. - М.: Советское радио, 1980.

Манна

1974 Manna Z. Mathematical theory of computation. - New York: McGraw-Hill, 1974, 448 p. (Русск. перевод гл.5: Манна З. Теория неподвижной точки для программ. - Кибернетический сб., под ред. О.Б.Лупанова. - М.: Мир, 1978, вып. 14).

Марков

1951 Марков А.А. Теория алгоритмов. - В кн.: Труди матем. ин-та АН СССР, 38, 1951, с. 176-189.

Матиясевич

1970 Матиясевич Ю. Диофантовость перечислимых множеств. - ДАН СССР, 191, 1970, с. 279-282.

1972 Матиясевич Ю.В. Диофантовы множества. - Усп. матем. наук. 27, 1972, вып. 5, с. 185-222.

Москоракис

1969 Moschovakis Y.N. Abstract first order computability. - Trans. Amer. Math. Soc., 138, 1969, p. 427-464.

1969a Moschovakis Y.N. Abstract computability and invariant definability. - Journ. symb. logic, 34, 1969, no.4, p. 605-633.

1969b Moschovakis Y.N. Axioms for computation theories. -In: Gandy R.C., Yates C.M.E. (Eds) Logic Colloquium'69. Amsterdam: North-Holland, 1971, p. 199-256.

Мостовский

1947 Mostowski A. On definable sets of positive integers. - Fundamenta mathematicae, 1947, 34, p. 81-112.

Непомнящий

1972 Непомнящий В.А. Критерии алгоритмической полноты систем операций. - В кн.: Теория программирования. Новосибирск: Вычислительный центр СО АН СССР, 1972, т.I, с. 267-279.

Патерсон

1968 Paterson M. Program schemata. -In: Machine intelligence, 3. Edinburgh: Edinburgh University Press, 1968.

Патерсон, Хьюитт

1970 Paterson M.S., Hewitt C.E. Comparative schematology. -In: Records of Project MAC conf. on concur. syst. and parall. comp. N.Y.: ACM, 1970, p. 119-128. (Русск. пер. Патерсон М., Хьюитт К. Сравнительная схематология (пер. М.Б.Трахтенброта) - Киб.сб., вып. 13, М.: Мир, 1976, с. 62-72).

Петер

1958 Peter R. Graphschemata und rekursive Funktionen. - Dialectica, 12, 1958, no.3-4, S. 373-393 (RZhMat, 1959, 9682).

Пост

1936 Post E. Finite combinatorial processes-formulation I. - Journ. symb. logic, 1, 1936, p. 103-105.

Райс

1953 Rice H.G. Classes of recursively enumerable sets and their decision problems. - Trans. Amer. Math. Soc., 74, 1953, p. 358-366.

Ратледж

1964 Rutledge J.D. On Ianov's program schemata. - Journ. ACM, 11, 1964, p. 1-19.

Роджерс

1967 Rogers H., Jr. Theory of recursive functions and effective computability. New York a.o.: McGraw-Hill, 1967.

(Перевод с англ. под ред. В.А.Успенского: Роджерс Х. Теория рекурсивных функций и эффективная вычислимость - М.: Мир, 1972, 624 с.).

Розен

1975 Rosen B.K. Program equivalence and context-free grammars. - Journ. Comp. Syst. Sci., 9, 1975, p. 358-374.

Сабельфельд

1981 Sabelfeld V.K. The tree equivalence of linear recursive schemata is polynomial-time decidable. - Information processing letters (to appear).

Скордев

1976 Skordev D. Recursion theory on iterative combinatory spaces. - Bull. Acad. Polon. Sci., Ser. Sci. Math. Astr. Phys., 24, 1976, p. 23-31.

1980 Скордев Д. Комбинаторные пространства и рекурсивность в них. - София: Изд-во Болгарской АН, 1980, 455 с.

Трахтенброт

1967 Трахтенброт Б.А. Сложность алгоритмов и вычислений. - Новосибирск: Новосибирский гос. ун-тет, 1967, 258 с.

Тюринг

1936 Turing A.M. On computable numbers, with an application to the Entscheidungsproblem. - Proc. London Math. Soc., ser. 2, 1936, v.42, p. 230-265; 1937, v.43, p. 544-546.

1937 Turing A.M. Computability and λ-definability. - J. symb. logic, 1937, v.2, p. 153-163.

Успенский, Семенов

1981 Uspensky V.A., Semenov A.L. What are the gains of the theory of algorithms: basic developments connected with the concept of algorithm and with its application in mathematics. -In: A.P.Ershov, D.Knuth (Eds) Urgench'79 papers. Lecture notes in Comp. Sci. Berlin: Springer, 1981 (in print).

Фенстад

1974 Fenstad J.E. On axiomatizing recursion theory. -In: Fenstad J.E., Hinman P.G. (Eds) Generalized recursion theory. Amsterdam: North-Holland, 1974, p. 385-404.

1978 Fenstad J.E. On the formulation of general recursion theory: computation versus inductive definability. -In: Fenstad J.E. et al. (Eds) Generalized recursion theory II. Amsterdam: North-Holland, 1978, p. 99-110.

1978a Fenstad J.E. Recursion theory: an axiomatic approach. - Berlin: Springer Verlag, 1978(?).

Флойд

1967 Floyd R.W. Assigning meanings to programs. -In: J.T. Swartz (Ed) Mathematical aspects of computer science.

Proc. Symposia in Applied Math., 19. Providence (R.I.): American Math. Soc., 1967, p. 19-32.

Фрассе

1959 Fraissé R. Une notion de récursivite relative. -In: Infinistic methods. Proc. Symp. on found. of math., Warsaw, 1959. Oxford: Pergamon Press, 1961, p. 323-328.

Фридман

1969 Friedman H. Axiomatic recursive function theory. -In: Gandy R.O., Yates C.M.E. (Eds) Logic Colloquium'69. Amsterdam: North-Holland, 1971, p. 113-138.

1969a Friedman H. Algorithmic procedures, generalized Turing algorithms and elementary recursion theories. -In: Gandy R.O., Yates C.M.E. (Eds) Logic Colloquium'69.

Харел

1980 Harel D. On folk theorems. - Comm. ACM, 23, 1980, no.7, p. 379-389.

Хоар

1967 Hoare C.A.R. An axiomatic basis for computer programming. - Comm. ACM, 12, 1967, p. 576-580.

Цейтин

1981 Tseitin G.S. From logicism to proceduralism. An autobiographical account. -In: A.P.Ershov, D.Knuth (Eds) Urgench'79 papers. Berlin: Springer Verlag, 1981 (to be published).

Чёрч

1936 Church A. An unsolvable problem of elementary number theory. - Amer. journ. of math., 58, 1936, p. 345-363.

1941 Church A. The calculi of lambda-conversion. - Annals of Math. stud., no.6. Princeton, N.J.: Princeton Univ. Press, 1941, ii + 77 p.

233-351

Шёнфинкель

1924 Schönfinkel M. Über die Bausteine der mathematischen Logik. - Math. Ann., 92, 1924, S. 305-316.

Эрбран

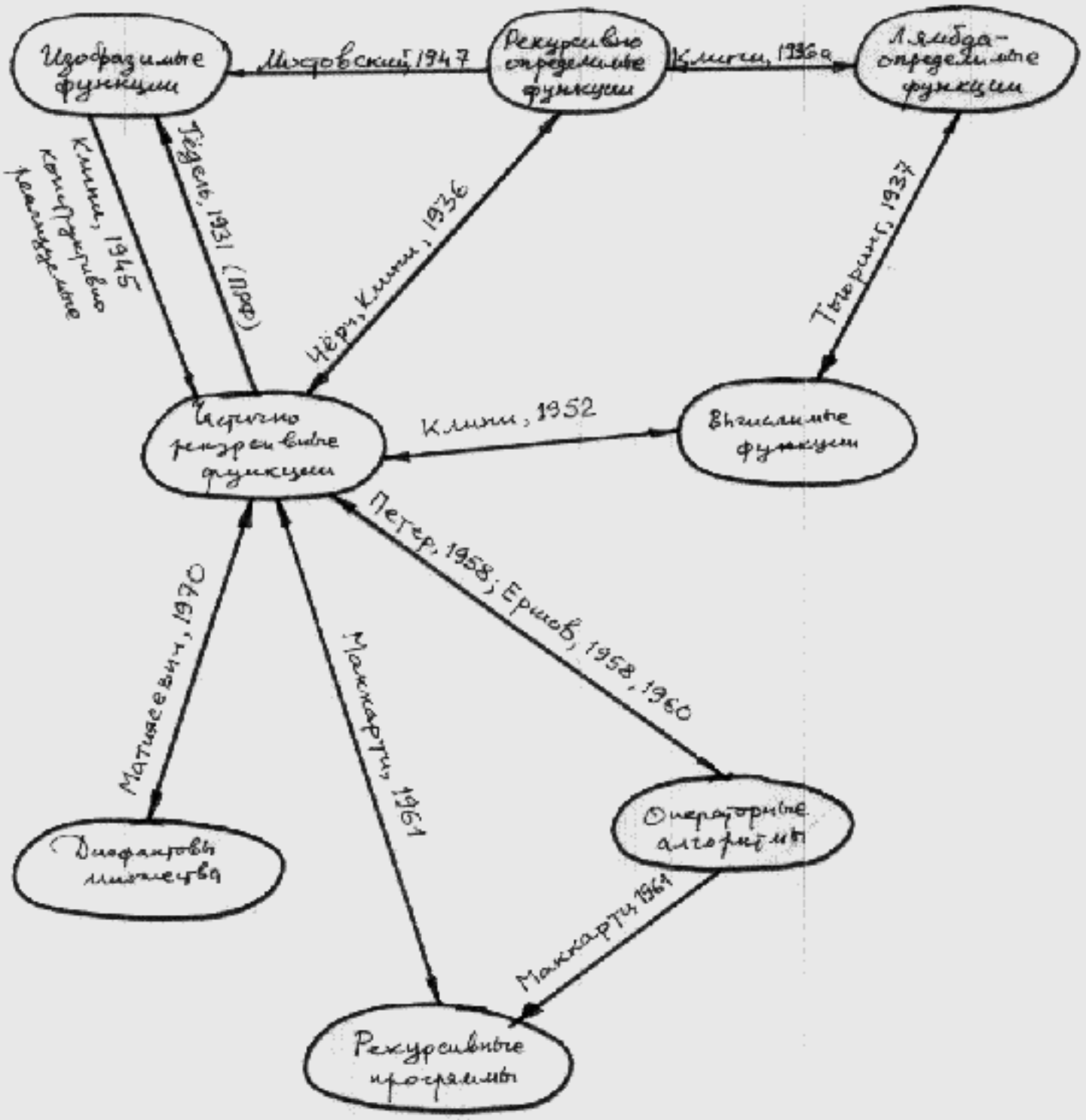
1931 Эрбран Дж. Неопубликованное письмо К.Гёделю. Воспроизведено Гёделем, 1934 с комментариями Гёделя, 1963.

Янов

1957 Янов Ю.И. О равносильности и преобразованиях схем программ. - ДАН СССР, 113, 1957, № 1, с. 39-42.

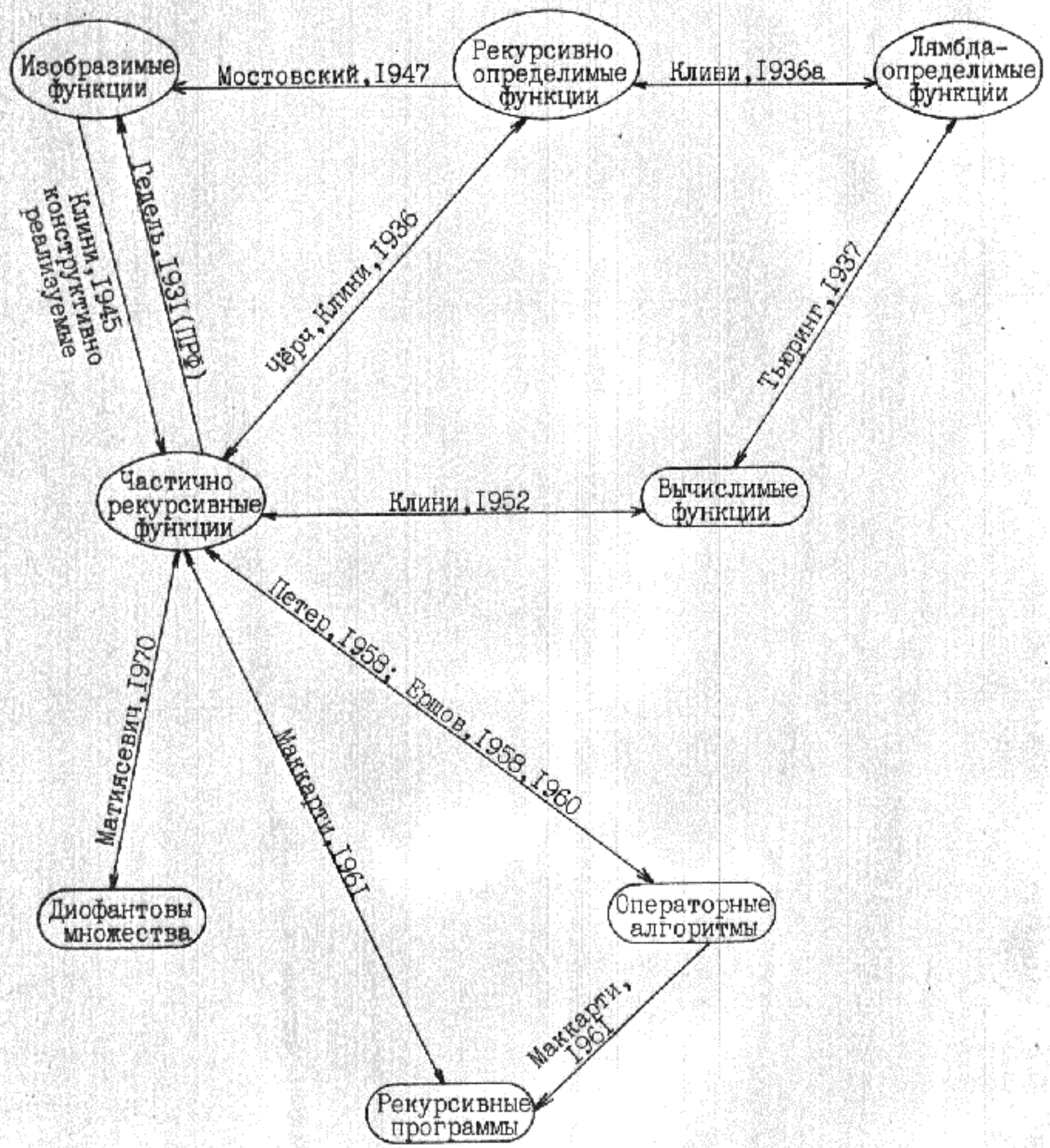
Дома 20.8.81

133-352

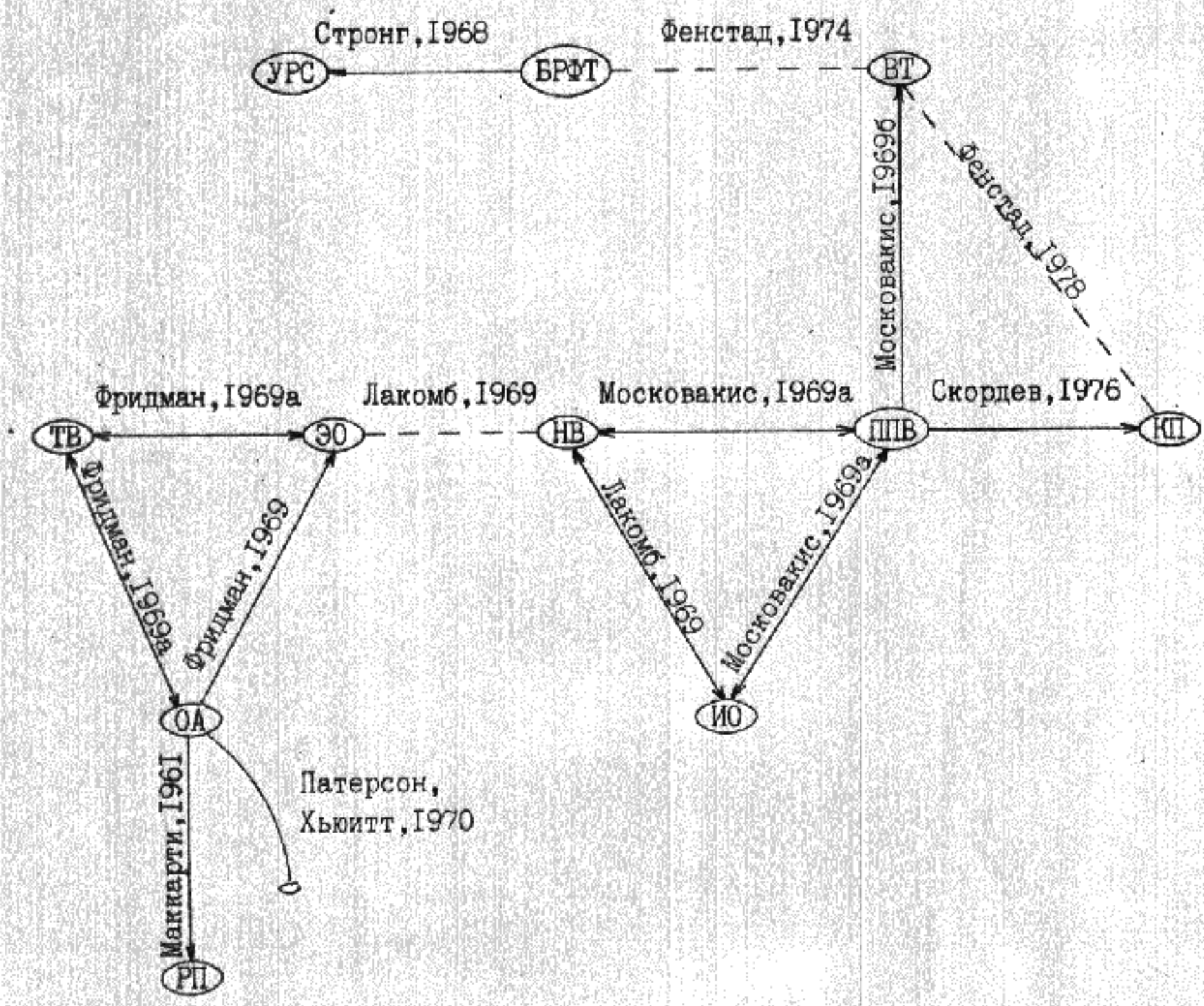


Фиг. 1. Схема связи определений арифметической вычислимости

К стр. 13

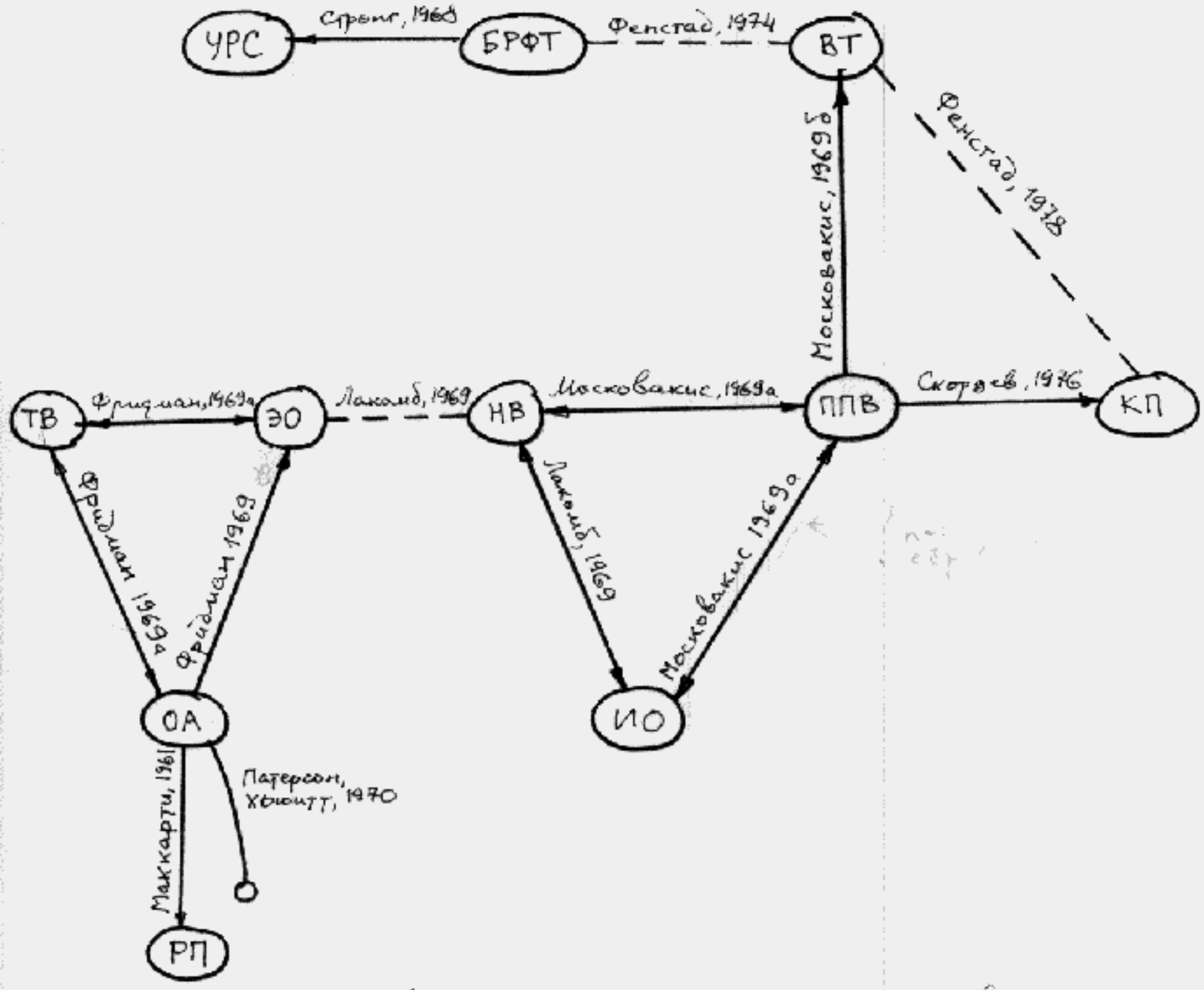


Фиг.I. Схема связи определений арифметической вычислимости



Фиг.2. Схема связи определений абстрактной вычислимости

УРС - униформные рефлексивные структуры (Вагнер, I963, I969)
БРФТ - базовые теории рекурсивных функций (Стронг, I968; Фридман, I969)
ВТ - Вычислительные теории (Московаким, I969б; Фенстад, I974, I978)
КП - комбинаторные пространства (Скордев, I974, I980)
ППВ - простая и поисковая вычислимости (Московакис, I969)
ИО - инвариантная определимость (Фрассе, I959; Крайзел, I963; Лакомб, I969; Московакис, I969а)
НВ - нумерационная вычислимость (Мальцев, I96I; Лакомб, I969)
ТВ - Тьюринговы вычисления (Фридман, I969а)
ЭО - эффективная определимость (Фридман, I969а)
ОА - операторные алгоритмы (Ершов, I958, I960; Криницкий, I959; Патерсон, I968)
РП - рекурсивные программы (Маккарти, I96I; Манна, I974)



Фиг 2 Схема связи определений абстрактной вычислимости

УРС – униформные рефлексивные структуры (Вагнер, 1963, 1969)
БРФТ – базовые теории рекурсивных функций (Стронг, 1968, Фридман, 1969)
ВТ – вычислительные теории (Московакис, 1969б; Фенстад, 1974, 1978)
КП – комбинаторные пространства (Скордев, 1974, 1980)
ППВ – простая и поисковая вычислимости (Московакис, 1969)
ИО – инвариантная определимость (Фрасне, 1959; Крайзел, 1963; Лакомб 1969, Московакис, 1969а)

НВ – нумерационная вычислимость (Мальцев, 1961; Лакомб, 1969)
ТВ – Тьюринговы вычисления (Фридман, 1969а)
ЭО – эффективная определимость (Фридман, 1969а)
ОА – операторные алгоритмы (Ершов, 1958, 1960; Кристиански, 1959; Патерсон, 1968)
РП – рекурсивные программы (Маккарти, 1961; Манна, 1974).

К стр. 48