FOREWORD TO THE ARTICLE BY A. P. ERSHOV „COMPUTABILITY IN ARBITRARY DOMAINS AND BASES“

In this issue, in the section “Turning over the old pages”, the editorial board offers readers the article by A. P. Ershov “Computability in arbitrary fields and bases”.

Reading this article many years ago, I was struck at that time, and now I am still amazed at how A. P. Ershov was able to specifically, not abstractly, foresee that due to the development of programming, the mathematical logic will be an applied discipline.

For programmers-encoders, this article may seem to be uninteresting and unnecessary. Those, who want to understand, even if not completely, may wish to better understand the basis of the modern programming, to understand and realize the connections of set theory, theory of algorithms and mathematical logic and their role in the development of programming in the coming decades.

As an epigraph to this publication, an extract from the same article begs:

The spread of computers and programming make mathematical logic an applied science. The caste of mathematical logicians, the priest-keepers of the sacred fire of the foundations of mathematics, to their amazement finds themselves face to face with the invasion of hordes of programmers, whose mathematical culture is close to barbarism, but who, nevertheless, inspired by their prophets of structural programming, seek to light a torch from the life-giving fire and drag it to them in order to illuminate the corners of their units clogged with lamps, OS / 360 instructions, Fortran forms and other symbols of idolatry. Nevertheless, in order to establish a union of such different cultures, each side - logicians and programmers - needs to overcome their inferiority complexes, to develop a capacity for mutual understanding and, what is more important, to become convinced in the interest to the other side.

The text is technically somewhat out of date, for example, the symbols of idolatry among programmers are now different, but the thoughts expressed are true today. Indeed, the era of programming, based largely on the mathematical logic is beginning. Logic is slowly but surely becoming an applied science, and programming systems in the mass-scale programming are gradually beginning to be replaced by systems for the automatic construction of algorithms and programs on the bases of the axiomatic description of a subject domain. There appear systems, in which an algorithm and a program are taken from the axiomatic description of the subject area, see, for example, Charm and LuNA systems (Victor Malyshkin. Active Knowledge, LuNA and Literacy for Oncoming Centuries // Springer, LNCS, Vol. 9465, P. 292 –303). Nowadays the students, who are trying to minimize their knowledge of mathematics, may have problems in finding a job in the future: they simply will not be able to understand the formulation of problems and how to solve them.

V. E. Malyshkin

ПРЕДИСЛОВИЕ К СТАТЬЕ А. П. ЕРШОВА „ВЫЧИСЛИМОСТЬ В ПРОИЗВОЛЬНЫХ ОБЛАСТЯХ И БАЗИСАХ“

В настоящем номере, в разделе „Листая старые страницы“ редколлегия предлагает вниманию читателей статью А. П. Ершова „Вычислимость в произвольных областях и базисах“.

Читая много лет назад эту статью, я был поражен тогда, и сейчас еще удивляюсь тому, как сумел А. П. Ершов конкретно, а не только абстрактно, предвидеть, что развитие программирования сделает математическую логику прикладной дисциплиной.

Программистам-кодировщикам статья эта может показаться неинтересной и не нужной. Тем же, кто захочет разобраться, пусть даже не полностью, можно пожелать лучше понять базис современного программирования, понять и осознать связи теории множеств, теории алгоритмов и математической логики и их роль в развитии программирования в ближайшие десятилетия.

В качестве эпиграфа к этой публикации напрашивается отрывок из этой же статьи:

> *Распространение ЭВМ и программирования делают математическую логику прикладной наукой. Каста математических логиков — жрецов-хранителей священного огня оснований математики, к своему изумлению оказывается лицом к лицу с нашествием орд программистов, чья 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 — увериться в своей интересности для другой стороны.*

Текст технически несколько устарел, например, символы идолопоклонства у программистов теперь другие, но высказанные мысли верны и сегодня. Действительно, наступает эра программирования, основанного в значительной степени на математической логике. Логика медленно, но верно становится прикладной наукой, а системы программирования в массовом программировании постепенно начинают заменяться системами автоматического конструирования программ на основе аксиоматического описания предметной области. Появляются системы, в которых алгоритм и программа извлекаются из аксиоматического описания предметной области, см., например, системы Charm и LuNA (Victor Malyshkin. *Active Knowledge, LuNA and Literacy for Oncoming Centuries* // Springer, LNCS, Vol. 9465, P. 292–303). Нынешние студенты, которые сейчас стараются минимизировать свои знания математики, в будущем могут испытывать проблемы при поиске работы: просто не будут понимать формулировки задач и как их решать.

В. Э. Малышкин

ВЫЧИСЛИМОСТЬ В ПРОИЗВОЛЬНЫХ ОБЛАСТЯХ И БАЗИСАХ

А. П. Ершов

Вычислительный центр, Новосибирск, 630090

DOI: 10.24411/2073-0667-2019-00012

В этой описательной статье мы обсуждаем одно из главных понятий математики и вычислительной науки — понятие эффективной вычислимости. Демонстрируя 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 понятия относительной вычислимости и истории вычислений представляются более первичными, нежели понятие программы и ее свойства композиции-охватываемости. 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 делает математическую логику прикладной наукой. Каста математических логиков — жрецов-хранителей священного огня оснований математики — к своему изумлению, оказывается лицом к лицу с нашествием орд программистов, чья 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 — увериться в своей интересности для другой стороны.

Эта описательная статья 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 математику“ (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 важных 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 взаимодей-

ствия этих двух сторон вычислений является фундаментальной проблемой, требующей более глубокого раскрытия сущности вычислимости.

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 Крайел, 1959, 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 — моделью 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 заключении 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 были обобщающие трактаты, из которых нужно, прежде всего, назвать уже упомянутую книгу Клини, 1952, и недавно изданный компендиум по математической логике: Барвайз, 1977.

Часть 1. Вычислимость арифметических функций. Везде ниже мы будем рассматривать частичные однозначные функции, берущие свои аргументы и принимающие значения на множестве $\mathbf{N}$, то есть на натуральном ряде, включающем нуль $\{0, 1, 2, \dots\}$. Эти функции, образующие пространство $\mathbf{A}$, мы будем называть арифметическими. Говоря об арифметических функциях вообще, мы не будем себя ограничивать абстракциями конструктивной математики. В классе $\mathbf{A}$ 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 $\mathbf{C}$. В классе $\mathbf{C}$ выделяется подкласс всюду определенных как 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 или пусто или 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.

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 Геделю. Рассматривается арифметическое исчисление предикатов S, например, по Клини, 1952, гл. IV, 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

$$\varphi(x_1, \ldots, x_n) = y \Leftrightarrow 1 - P(cx_1, \ldots, cx_n, cy),$$

где cx- терм, изображающий натуральное число x (можно всегда считать, что он имеет вид $0\underbrace{/\ldots/}_{x}$ раз. Множество арифметических функций, изображаемых в арифметическом исчислении предикатов, образует класс вычислимых функций (по Геделю, 1936, формулируется по Клини, 1952, §59, 62).

Рекурсивная определимость по Эрбрану и Геделю. Рассматривается формальное исчисление уравнений над счетными множествами переменных для натуральных чисел x, y, x_1, y_1, ... и функциональных букв f, g, h, f_1, g_1, h_1, ..., содержащее одну предметную константу 0 и один функциональный символ ‘, а также разделители $=$, (,) и ,. Термами являются 0, переменные, r', где r — терм, и $f(r_1, \ldots, r_n)$, где f — функциональная буква, а $r_1, \ldots, r_n$ $(n \geqslant 0)$ — термы. Уравнение имеет вид $r = s$, где и r, и s — термы. Система уравнений — это конечная последовательность уравнений $E = \{e_0, \ldots, e_s\}$. Самая левая функциональная буква в последнем уравнении e_s называется главной буквой. Даны два правила вывода.

R_1 (конкретизация): $e(y)\ |\text{-}e(c)$, где $e(y)$ — уравнение, содержащее переменную y, а c — любая цифра, т. е. терм 0 или $0^{/\cdots/}$;

R_2 (замена): $r = s(h(c_1, \ldots, c_p)), h(c_1, \ldots, c_p) = c| - r = s(c)$, где $s(h(c_1, \ldots, c_p))$ — терм, содержащий вхождение терма $h(c_1, \ldots, c_p)$, в котором h — функциональная буква, $c_1, \ldots, c_pc$ — цифры.

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

$$\varphi(x_1, \ldots, x_n) = y \Leftrightarrow E| - f(cx_1, \ldots, cx_n) = cy,$$

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

λ-определимые функции по Черчу. Над счетным алфавитом переменных V строится множество термов T: а) $V{\in}T$; б) $x{\in}V$, $t{\in}T{\Rightarrow}(\lambda xt){\in}T$(рассмотрение терма t как функции от x); в) $t{\in}T$(применение терма-функции t к терму-аргументу u). Из 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) $t_1t_2\ldots t_n{\equiv}$ $(\ldots(t_1t_2)\ldots t_n)$ и (2) $\lambda x_1\ldots x_n.t{\equiv}{\equiv}$ $(\lambda x_1(\lambda x_2\ldots(\lambda x_n t)\ldots))$. Последнее сокращение одновременно подсказывает, как в языке „вычисляются“ функции многих переменных: $fx_1x_2\ldots x_n$ вычисляются так, что применяется f к x_1, затем fx_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$|-$s$\to$u; 4. (a) s $\to$ t $|-$ us $\to$ ut; (b) s $\to$ t $|-$ su $\to$ tu; (c) s $\to$ t $|-$ λx.s $\to$ λx.t;

II. 1. s $\to$ t $|-$ s = t; 2. s $\to$ t $|-$ t = s; 3. s = t, t =u $|-$ s = u; 4. (a) s =t $|-$ us = ut; (b) s =t $|-$ su = tu; (c) s=t $|-$ λx.s=λx.t.

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

$$\varphi(x_1,\ldots,x_n) \Leftrightarrow |- \mathbf{fx_1},\ldots,\mathbf{x_n} = \mathbf{y}.$$

Множество λ-определимых функций образует класс вычислимых функций по Черчу (Черч, 1936, 1941); сформулировано по Барендрегту, (Барендрегт, 1977).

Функциональные определения.

Частичная рекурсивность по Клини. Берется счетный базис арифметических функций:

1) функции счета S(x)=';

2) функции-константы $C^n(x_1, \ldots, x_n) = c$, где c— натуральное число;

3) функции выбора аргумента $\cup_i{}^n(x_1, \ldots, x_n){=}x_i$. Далее, рассматриваются три типа операторов, т. е. схем определения новых функций через данные:

4) оператор подстановки $\varphi = S^n_m(\psi, \chi_1, \ldots, \chi_m)$, для которого

$$\varphi(x_1,\ldots,x_n) = \psi(\chi_1(x_1,\ldots,x_n),\ldots,\psi_m(x_1,\ldots,x_n));$$

5)оператор примитивной рекурсии $\varphi = \mathbf{R}^n(\psi\chi)$, для которого

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

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

6)μ-оператор, или оператор упорядоченного поиска $\varphi = \mathbf{M}^n(\chi)$, для которого

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

т. е. $\varphi(x_1, \ldots, x_n)$ равно наименьшему y, для которого $\chi(x_1, \ldots, x_n, y){=}0$, и для всех $y'{<}y\lambda(x_1, \ldots, x_n, y){\neq}0$, и неопределенно, 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 прямо вычислимых функций;

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

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

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, ... и функциональных букв f, g, h, f_1, g_1, h_1. одну предметную константу 0, один функциональный символ $'$, один предикатный символ $=$ (равенство), а также разделители $\Leftarrow$ (,) и , . Каждой функциональной букве f сопоставляется ее арность: целое число $\rho(f)\geqslant 0$. Именами функций являются выражения $f(x_1, \ldots, x_{\rho(f)})$, где f— функциональная буква, а $x_1, \ldots, x_{\rho(f)}$ — разные переменные. Функциональными термами являются 0, переменные, r', где r — функциональный терм, и $f(r_1, \ldots, r_{\rho(f)})$, где f — функциональная буква, а $r_1, \ldots, r_{\rho(f)}$ — функциональные термы. Предикатные термы имеют вид $x = y$, где x и y — переменные. Условные термы имеют вид $(p|r|s)$, где — предикатный терм, а r и s — функциональные или условные термы. Значение **val** условного терма определяется по правилу разбора случаев:

$$\mathbf{val}(p|r|s) = \begin{cases} \text{val } r, & \text{если } p \\ \text{var } s, & \text{если } \neg p \end{cases}$$

Уравнением называется выражение вида $f(x_1, \ldots, x_{\rho(f)}) \Leftarrow t$, где t — функциональный или условный терм с переменными, принадлежащими множеству $\{x_1, \ldots, x_{\rho(f)}\}$. Система уравнений $\mathrm{e}_1, \ldots, \mathrm{e}_n$ $(n>0)$ 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 $\mathrm{e}_1, \ldots, \mathrm{e}_n$ 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 $\varphi_1{}^*, \ldots, \varphi_n{}^*$, что для любой другой неподвижной точки $\varphi_1, \ldots, \varphi_n$ имеет место $\varphi_i{}^* \subseteq \varphi_i (i=1, \ldots, n)$. Наименьшей неподвижной точкой рекурсивной программы является таковая для главного уравнения. Класс наименьших неподвижных точек рекурсивных, программ образует класс вычислимыхфункций по Маккарти (Маккарти, 1961) и Манне (Манна, 1974, гл. IV).

Примечание. Рассмотрим уравнение

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

где $\mathrm{g}_1, \ldots, \mathrm{g}_k$ — другие функциональные буквы, входящие в t. Сопоставим этим буквам некоторые арифметические функции $\lambda_1, \ldots, \lambda_k$ и рассмотрим наименьшую неподвижную точку $\varphi(\mathrm{x}_1, \ldots, \mathrm{x}_n)$ получившегося уравнения относительно f. Тогда выражение t можно рассматривать как оператор $\varphi = \mathrm{Lub}^n[\mathrm{t}, \lambda_1, \ldots, \lambda_k]$ 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, ..., q_k, одно из которых q_0 является пассивным, заключительным, а остальные — активными. Одно из активных состояний q_1считается начальным. В любом состоянии машина обозревает некоторую клетку ленты. Машина может выполнять 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, ..., 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 (Тьюринг, 1936); сформулировано по Клини (Клини, 1952, § 67).

Операторные алгоритмы по Ершову. Рассматривается язык программ для операторных алгоритмов, содержащий счетное множество переменных для натуральных чисел x, y, x_1, y_1,..., одну предметную константу 0, один функциональный символ ', один предикатный символ = (равенство), а также разделитель := (знак присваивания). Термами являются 0, переменные и r', где r — терм. Присваивание имеет вид x:= r, где x — переменная и r — 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, ..., x_n переменных $x_1, \ldots, x_n$, и управление передается начальной вершине графа переходов. Если управление

передано преобразователю с присваиванием x:=r и аргумент терма r определен, то 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 $\varphi(x_1, \ldots, x_n)$. Множество функций, вычисляемых операторными алгоритмами, образует класс вычислимых функций по Ершову (Ершов, 1958).

Арифметическое определение.

Арифметические определения задают перечислимые множества, возникающие 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 множества значений параметров, при которых уравнения разрешимы в целых (неотрицательных, положительных) числах.

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

$$p(\mu_0, \mu_1, \ldots, \mu_n, x_1, \ldots, x_l) = 0$$

имеет решение. Множество функций, графики которых являются диофантовыми множествами, образует класс вычислимых функций по Матиясевичу (Матиясевич, 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 **C** арифметических функций (рис. 1), которые мы и будем называть вычислимыми (неважно, на основе 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 это сходство стали выискивать 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 В. А. Успенского и А. Л. Семенова (Успенский, Семенов, 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 $x_1, \ldots, x_n$ аргументов получение значения $\varphi(x_1, \ldots, x_n)$ при условии, что это значение $\varphi(x_1, \ldots, x_n)$ определено. 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

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

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 $\psi_1, \ldots, \psi_l$ произвольных арифметических функций (оракулов), значения которых ищутся, вообще говоря, внешним по отношению к теории вычислимости способом. Класс функций, вычислимых относительно набора $\psi_1, \ldots, \psi_l$, называется вычислимым замыканием функций $\psi_1, \ldots, \psi_l$.

Если в формализме относительной вычислимости рассмотреть некоторую программу $P(\psi_1, \ldots, \psi_l)$, то при разных исходных функциях $\psi_1, \ldots, \psi_l$ 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 **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 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 **C**, в частности, замыкания относительно любого вычислимого оператора.

Для каждого вычислимого оператора $\mathrm{R}(\psi)$ можно определить его наименьшую неподвижную точку, которая также оказывается вычислимой функцией (первая теорема о рекурсии). Наконец, класс **C** замкнут относительно диагонализации, т. е. для любой нумерации $\{\varphi_i\}$ вычислимых функций n аргументов функция $\varphi^*(x_1, \ldots, x_n) = \varphi_{x1}(x_1, x_2, \ldots, x_n)$ вычислима.

Вычислимость универсальных функций также является выражением определенных свойств замкнутости класса **C**.

Важные примеры. Объем класса **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; перечис-

лимое, 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 n и $f(n)$, трактуемые как программы, будут задавать одну и ту же вычислимую функцию (вторая теорема о рекурсии). Наряду с этим имеет место принципиальный и, на первый взгляд, парадоксальный факт: программа, задавая исчерпывающую информацию, позволяющую получать 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. свойство, которым обладают не все функции из $\mathbf{C}$), не существует алгоритма, который по программе обнаруживал бы, 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 общего понятия 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 $\mathbf{C}$ (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 конкретного 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 без гарантии того, что нужный нам элемент у принадлежит множеству, и без оценки числа шагов, которые надо сделать, прежде чем добраться до цели.

В геделевых изображениях вычислимых функций существует целая иерархия нормальных форм в виде общерекурсивных предикатов, предваренных 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. 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 ${s_n}^m(\mathrm{p}^{m+n}, x_1,\ldots,x_m)$, 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 $p_x, \ldots, x_n$,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 $\mathbf{N}$ в $\mathbf{N}$. 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, — субрекурсивные классы. Для нумераций нужно уметь сворачивать взаимно однозначно последовательности чисел в одно число; базовой конструкцией является функция пересчета пар, фактически представляющая собой семейство трех функций $z{=}\chi(x,y)$, $x{=}\sigma_1(z)$ и $y = \sigma_2(\mathrm{z})$, удовлетворяющих следующим тождествам

$$\forall z \equiv \chi(\sigma_1(z), \sigma_2(z));$$

$$\forall x, y : x \equiv \sigma_1(\chi(x, y))\&t \equiv \sigma_2(\chi(x, y)).$$

Сложность. В последние годы с общей теорией вычислимости начинает смыкаться 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 понятие протокола или истории вычислений, или просто вычисления: организованная совокупность элементарных шагов, реализующих функциональную зависимость, т. е. приводящая от $x_1, \ldots, x_n$ к $\varphi(x_1, \ldots, x_n)$. 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 вычисления, выполняемые 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 существенно зависят от объема входной информации, сложность обычно указывается как некоторая функция параметра, характеризующего объем.

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 имеют 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 Геделю (Гедель, 1931) установить свой знаменитый результат о неполноте формальных теорий. Очень важную роль для Теории вычислимости и математики вообще сыграла концепция геделевой нумерации, т. е. отображения конструкций формального языка в предметную область, в данном случае — в $\mathbf{N}$. 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, 1936a), дать первые формулировки теорем о нормальной форме (Клини, 1936) и о нумерации (Клини, 1943). Уже в последние годы в виде так называемых базовых трансформаций (*rewritingrules*) 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 следующие наиболее важные универсальные конструкции класса вычислимых функций: отображение функций в предметную область (способность одного и того же терма употребляться и в роли функции, и в роли аргумента), нахождение значения функции от заданного аргумента с помощью универсальной операции (операция (tu)

применения t к u), s-m-n-функцию (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 свойства 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 $|{-}t{\to}u$. 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 Черчем (Черч, 1936), состоит в требовании определить для любого терма, имеет ли он нормальную форму.

Совпадение λ-определимости с общерекурсивностью (Клини, 1936a) позволило Черчу 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; Лэндин (Лэндин, 1965) указал на связь конструкций языка Алгол 60 с λ-исчислением; совсем же недавно Бэкус (Бэкус, 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). Характеризации 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 классифицировать программы по структуре графа переходов (бесцикловые, структурированные программы), в частности, обнаружить, что уже весьма регулярной и простой структуры достаточно, чтобы запрограммировать любую функцию (Петер, 1958; Бем, 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 заданы 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 доказательства свойств программ. Как известно, изобразимость вычислимой функции ф в логическом языке формальной арифметики позволяет средствами логического вывода доказывать принадлежность отдельных точек графику функции φ, но не дает никакого подхода к доказательству общих свойств произвольной вычислимой функции. По программе примитивно рекурсивной функции в ее определении по Геделю 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 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. Это рассуждение, 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; Берсталл, 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, удовлетворяющими разным критериям эффективности, составляет один из центральных разделов теоретического программирования.

Недавно выяснилось, что программные процессоры типа s-m-n-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 десятой проблемы следующую: найти алгоритм, позволяющий для полиномиальных уравнений с целыми коэффициентами (диофантовых уравнений) определить их разрешимость в целых числах. 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 десятой проблеме Гильберта просто еще один пример неразрешимой массовой проблемы, однако гипотеза Дейвиса (Дейвис, 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 историю окончательного установления диофантовости перечислимых множеств, заметим лишь, что построение теории вычислимости в рамках этой модели еще продолжается (Дейвис и др., 1976) . Найден универсальный многочлен, т. е. обладающий параметром, варьирование которого позволяет получать любые перечисленные 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 не более чем четвертой степени), а также получены оценки сложности конкретных множеств. Пожалуй, наиболее загадочной теоремой о нормальной форме является представление любого перечислимого множества из $\mathbf{N}$ 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.

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: (1) 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 дважды 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, 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[1]. 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, нося, 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.

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, предлагаемом теорией доказательств. Выполнимость аксиом абстрактной теории проверяется в конкретном языке программирования данной модели. Уже на

[1]Автору запомнилась сценка, произошедшая на третьем съезде советских математиков в 1966 г. Докладчик, рассказав об одном из уточнений понятия алгоритма, завершил изложение результата сводимости этого уточнения к другим понятиям алгоритма бодрой фразой: „Это, таким образом, лишний раз подтверждает тезис Черча“, в ответ на что из зала раздалось: „В каком смысле лишний?“.

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 **N** и пространства арифметических функций **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 **N**, **A**, **C**, и нумерациями остальных теорий можно получить все утверждения, интересные для математики. Этому противопоставляется такой взгляд, что на практике 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 **N**, **A** и **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 же 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.

1. 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 $cx \in W$ (W – пространство символических выражений) необходимы две функции χ: $D \to W$ и δ: $W \to D$ со свойствами $\chi(x) = cx$, $\delta(cx) = x$, $x_1 = x_2 \Leftrightarrow cx_1 = cx_2$. Для натуральных чисел связь между количеством и его представлением с помощью операции счета предельно симметрична и проста: $n \leftrightarrow 0$ (заметим, что это одно из самых „коварных“ для анализа свойств натурального числа).

3. Любое множество из **N** линейно упорядочено. Любое множество имеет минимальный элемент; любое конечное множество имеет максимальный элемент (это также очень сильное свойство, вырождающее многие алгоритмические операции над числами, особенно поиск).

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 **N**, **A** и **C** возникает еще одна альтернатива: говоря о вычислимости в произвольных областях можно ли позволить себе опираться на теорию вычислимости в **N**? Для пифагорейцев 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 Черча (Черч, 1941) и Карри, (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 $\langle W, \Pi, \Phi\rangle$. Константы достаточно рассматривать как нульместные операции. Если необходимо, операции также наделяются некоторыми 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 $\langle \mathbf{N}, =, 0, \prime\rangle$.

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 (Крайзел, 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. В работах (Вагнер, 1963, 1969) автор рассматривает в качестве исходной предметной области абстрактное множество U. Он постулирует существование геделизации, 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 (Шенфинкель, 1922):

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

и $[u]_1 = [u]$

и определяет для заданной пары $I = \langle U, G\rangle$ функциональное пространство $F(I) = \{[u]_n | u \in U, \mathrm{n} \geqslant \mathrm{l}\}$.

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 (Карри, 1930), также восходящий к Шенфинкелю (Шенфинкель, 1922) и ψ (хорошо известная программистам конструкция **если-то-иначе**), удовлетворяющие следующим аксиомам:

1. $[u](*) = * = [*](u)$;
2. $[\alpha](f, q) \neq *$, $[[\alpha]\ (f, q)]\ (x) = [f](x[q](x))$;

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

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 **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 (сплинтер). Если потребовать, чтобы сплинтер как подмножество в U имел бы в $F(I)$ всюду определенную характеристическую функцию, то тогда $F(I)$, рассматриваемое на сплинтере, совпадает с **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. Стронг (Стронг, 1968) и Фридман (Фридман, 1969), рассматривая пространство функций $F = \{f\} f\colon D^n \to D (n \geqslant 0)$ отдельно от предметной области D, выписывают дополнительные аксиомы, которым должно удовлетворять 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 геделизации, удовлетворяющие аксиомам BRFT, создают изоморфные классы функций.

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

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

2) на F задана операция пересчета пар, при этом функции взятия элементов пары принадлежат F;

3) C замкнуто относительно пересчета пар;

4) на F задана операция **если-то-иначе**;

5) на F задана операция взятия неподвижной точки уравнения ψ — если χ то I иначе $\psi\varphi$, где I — тождественная функция (итерация).

Если Ф — конечный набор элементов из F, то любой элемент из F, который получается из Ф$\cup$ (тождественная функция, взятие элементов пары, истина, ложь) с помощью применения конечного числа операций, заданных на F, называется функцией, рекурсивной в Ф. Для этих функций можно построить теорию рекурсии, включающую аналоги теорем о нормальной форме, первой и второй теоремам о рекурсии и теореме о нумерации.

Вычислительные теории. Московакис (Московакис, 1969*б*) 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 N натурального ряда $\mathbf{N}$, а на C был бы задан пересчет пар. Затем Московакис 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 рефлективности, 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 N,

2) характеристическая функция множеств C и N,

3) функции пересчета пар на C,

4) подстановка и примитивная рекурсия по N,

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

Для того чтобы охарактеризовать класс вычислимых функций, Московакис 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 F объявляется вычислимым, только если длина вычисления F, грубо говоря, не меньше длины вычисления функций, индексы которых являются аргументами вычисления функционала. Кроме того, чтобы предвычислительная теория была вычислительной, требуется — по определению — чтобы длина вычисления по универсальной функции была бы, как скажут программисты, меньше длины частичного вычисления и последующего вычисления по остаточной программе, ср. (Ершов, 1980).

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

Фенстад, (Фенстад, 1974), рассматривая множество вычислений $\Theta = \{e\sigma y\}$, где σ — набор аргументов, вводит вместо длины вычисления транзитивное отношение „быть подвычислением“, т. е. $(e\sigma y) < (e'\sigma' y')$ значит, что вычисление y из σ по программе e является частью вычисления y' из σ' 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 (Московакис, 1969б), где вместо неравенств между длинами вычислений постулируются соответствующие отношения быть подвычислением.

Фенстад, (Фенстад, 1978) строит множество „рекурсивных“ функций в абстрактном функциональном пространстве в виде замыкания $\mathbf{R}_\omega(\Phi)$, 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 следующие „комбинаторы“ — термин, восходящий к Карри (Карри, 1930): $I(f) = f$, $K(f, g) = f$ и $S(f,g,h) = f(h)(g(h))$. Он показывает, что $\mathbf{R}_\omega(\Phi)$ 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 $R: \Phi \cup \Pi \to \mathbf{N}$. Здесь R— тип алгебраической системы, $\Phi = \{\varphi_1, \ldots, \varphi_m\}$ $(m \geqslant 0)$ — функциональные символы, $\Pi = \{\pi_1, \ldots, \pi_n\}$ $(n \geqslant 0)$ — 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, будем для краткости, как правило, опускать. 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 ею класса функций. Поскольку большинство определений абстрактной вычислимости имеют прототипы, среди арифметических функций, группировка разных определений будет аналогичной.

Нумерационная вычислимость. Идея определять абстрактную вычислимость, опираясь непосредственно на арифметические вычислимые функции, принадлежит Мальцеву (Мальцев, 1961). Рассматривая произвольную алгебраическую систему $\langle D, \Phi, \Pi \rangle$, Мальцев постулирует существование для D нумерации α, т. е. отображения $\alpha: \mathbf{N} \to D$, которое позволяет связать произвольную функцию $f: D^r \to D$ с некоторой арифметической $F: \mathbf{N}^r \to \mathbf{N}$ 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 (Лакомб, 1969), рассматривая абстрактную вычислимость в реляционных системах с 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 предиката $P(x_1, \ldots, x_r)$ посредством примитивно рекурсивного множества булевых формул в алфавите $\Pi \cup \{P, =$, предметные

переменные x_1, ..., x_r, конечный набор предметных констант}. Корректность определения абстрактной вычислимости по Лакомбу подтверждается тем, что в $\langle \mathbf{N}, 0, =, ' \rangle$ его вычислимость совпадает с обычной относительной вычислимостью.

Инвариантная определимость является обобщением изобразимости в формальной арифметике. Ее применение к определению абстрактной рекурсивности было предложено Крайзелом, (Крайзел, 1963). Более конкретные определения были даны Фрассе (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 cx для каждого элемента $x\in D$. Пусть $\Phi(\mathrm{P})$ — 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 $\Phi(\mathrm{P})$. Поскольку в реляционной системе $\langle \mathbf{N}, \Pi, =\rangle$ 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 r-местным предикатным символом π связывается счетное множество формул $\Delta(\pi)$ вида $\pi(cx_1, \ldots, cx_r)$или $\pi(cx_1, \ldots, cx_r)$ в зависимости от истинности или ложности $\pi(x_1, \ldots, x_r)$. Пусть $\Delta(\Pi) =\Delta(\pi_1)\cup\ldots\cup\Delta(\pi_n)$. 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

$P(x_1, \ldots, x_r)\Leftrightarrow\Delta(\Pi)\cup\{\Phi(\mathrm{P})\}|-\mathrm{P}(cx_1, \ldots, cx_r)$

и

$P(x_1, \ldots, x_r)\Leftrightarrow\Delta(\Pi)\cup\{\Phi(\mathrm{P})\}|-\mathrm{P}(cx_1, \ldots, cx_r)$.

Простая и поисковая вычислимости. Московакис (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 $(x, \mathrm{y}))$, 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 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 N. Показывается, что так определенные „примитивно рекурсивные“ функции относительно пустого Φ моделируют арифметические примитивно рекурсивные функции.

Московакис приходит к „частично рекурсивным“ функциям, постулируя вычислимость универсальной функции над D^*. При этом, однако, эта функция оказывается определенной лишь в том случае, если первый аргумент этой функции оказывается индексом, построенным в соответствии с заданными индуктивными правилами. Необходимость „синтаксического анализа“ индекса требует допущения операций поиска (аналогичных 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 (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 F — множество всех бинарных отношений в $D^*\times D^*$. Для любых двух бинарных отношений φ и ψ определяются три базовых операции: обычная композиция, сочетание $\{(p,\ s)\colon \exists q\exists r((p,\ q)\in\varphi\&(p,r)\in\psi\&(q,\ r)=s\}$ и итерация (грубо говоря, композиция φ с самим 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 пары q“ и „максимальное“ отношение $U = D^*\times D^*$. Тогда замыкание $\varphi_1,\ldots,\varphi_m$, L и R тремя указанными операциями дает $F\cap$PC, а то же замыкание $\varphi_1,\ldots,\varphi_m,L$, R и U дает $F\cap$SC.

Тьюринговы вычисления в произвольной области. Фридман (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→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→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 Черча (в применении к модели T) и к перечислимости эффективного определения.

Фридман изучает, каким дополнительный свойствам должны удовлетворять T-рекурсивные функции для выполнения основных теорем элементарной общей теории вычислимости. Этих свойств оказывается три:

(1) T-рекурсивность предиката равенства,

(2) T-рекурсивность пересчета пар,

(3) конечная порождаемость D с помощью T-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, 1961). В операторных алгоритмах класс вычислимых функций над алгебраической системой $\langle D, C, \Phi, \Pi\rangle$ — это все функции, программируемые в „системе команд“ (Ф, П). 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 программам, 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 рассмотрены Криницким (Криницкий, 1959, 1970) и получили впоследствии в литературе 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 композиции; операторы присваивания имеют вид y:= t, где t — функциональные термы с операциями из Ф. Важнейшим отношением для схем программ является отношение функциональной эквивалентности: две схемы S_1и S_2в сигнатуре Ф, П 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 (Янов, 1957) и Ратледж (Ратледж, 1964) установили это для сигнатур с унарными операциями и схем, использующих только одну переменную и предикатные термы без функциональных символов. В случае 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).

Помимо этого, 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 говорят программисты. При этом с непосредственной очевидностью обнаруживаются три альтернативных возможности: 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 (Ф, П), используя употребленные в программе символы 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 вхождением 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 (Ф, П). 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 уместно называть детерминантами схем программ.

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

Первый пример такого детерминанта привел Янов, (Янов, 1957.) для указанного выше подкласса схем, получивших название схем Янова. Для его сигнатуры $\Ф = \{A_1, \ldots, A_n\}$и $П = \{p_1, \ldots, p_m\}$ детерминант был образован последовательностями вида

$(\sigma_1^{(0)} \ldots \sigma_m^{(0)})\ A_{i0} \ldots (\sigma_1^{(k)} \ldots \sigma_m^{(k)})\ A_{ik}$

где σ_j— это p_jили 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 (Иткин, 1972). Его протоколы строились следующим образом: брался произвольный путь от входа до выхода из схемы программы. Каждому вхождению предикатного символа в этот путь присваивалось значение, предписанное данным путем (выходная 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, 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 рис. 2 показана схема эквивалентностей и сводимостей разных определений абстрактной вычислимости. Автор не уверен, что схема исчерпывающая. Сделаем к ней некоторые замечания.

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

Униформные рефлексивные структуры (УРС) и базисная рекурсивных функций теория (БРФТ) — это системы разной степени общности. В первой нумерация задана изначально, построение второй обеспечивается более подробными аксиомами. Связь вычислительных теорий с БРФТ лишь только обозначена Фенстадом (Фенстад, 1974). Не исключено, однако, что такая связь изучается в книге Фенстада (Фенстад, 1978а), которая была автору недоступна. От себя добавим, что должен существовать сравнительно прямой способ вложения простой и поисковой вычислимостей (ППВ) в БРФТ в силу явного построения индексов и постулирования вычислимости универсальной функции. Вариант вычислительной теории, предложенный Фенстадом (Фенстад, 1978), сближается с теорией комбинаторных пространств.

Думается, что треугольники сводимостей НВ— ППВ—ИО и ТВ—(ОА, РП)—ЭО на рис. 2 достоверно и интуитивно убедительно формируют объем класса функций, вычислимых в некоторой алгебраической системе $\langle D, C, \Phi, \Pi, R\rangle$. При связи ППВ-ИО и ППВ—НВ (Московакис, 1969а) включает со стороны ППВ в сигнатуру П предикат равенства. Во всех трех моделях — ППВ, ИО и НВ — определение абстрактной вычислимости может быть релятивизировано к любому набору констант C из D. При изучении сводимостей оказывается весьма удобным наличие для нумерационно-вычислимых функций примитивно рекурсивного детерминанта из булевых формул (мы можем использовать этот термин, так как эта характеризация, установленная Лакомбом, вполне соответствует определению детерминанта).

Эффективные определения Фридмана (Фридман, 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 тьюринговых вычислениях и операторных алгоритмах непосредственно транслируются в перечислимый детерминант, образованный клаузами Фридмана. При доказательстве ЭО→ТВ Фридман апеллирует к тезису Черча (в применении к модели Т) и к перечислимости ЭО-детерминанта. В изучении операторных алгоритмов Фридман 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 (Патерсон, 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 клаус детерминанта. Операторные алгоритмы (ОА) и рекурсивные программы (РП) без дополнительных требований к сигнатуре обладают разной силой вычислимости: ОА сводится к РП в той же сигнатуре, однако, как показали Патерсон и Хьюитт (Патерсон, Хьюитт, 1970), существует РП, для которой нет эквивалентного ОА в той же сигнатуре (на самом деле доказывается отсутствие ОА с тем же детерминантом).

Связь эффективной определимости и нумерационной вычислимости аккуратно не прослежена, но трансляция клауз 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 Геделю находит свое прямое обобщение в инвариантной определимости.

Рекурсивная определимость по Эрбрану–Геделю 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 $\varphi_i(\mathrm{c}x_1,\ldots,\mathrm{c}x_{ri})=\mathrm{c}y$, выражающих информацию о том, что $\varphi_i(x_1,\ldots,x_{ri})=y$. 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 $\langle \mathbf{N},0, =, ' \rangle$. Автор не нашел, однако, аналогичного прямого вложения произвольных систем в тот или иной вариант λ-исчисления. Следует, однако, отметить очевидную связь λ-исчисления с теориями абстрактной вычислимости в большом (УРС, БРФТ). Стронг (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 (Фридман, 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, так как 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 базисных теориях, теории комбинаторных пространств, простой и поисковой вычислимостях и тьюринговых вычислениях. Производит большое впечатление доказательная сила аксиом униформных рефлексивных структур. 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.

Абстрагированность от арифметической вычислимости. Анализ этого обстоятельства обнажает самые чувствительные места большинства работ по абстрактной вычислимости. Многие авторы (например, Крайзел, 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 нумерационной вычислимости и у Фридмана в эффективной определимости). 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.

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 обличья 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 Лакомба в НВ. Естественно, не надо забывать и о перечислении графика вычислимой функции элементарными функциями по Кальмару (Кальмар, 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 $\mathrm{X}_n = \{\mathrm{x}_1, \ldots, \mathrm{x}_n\}$ — символы переменных. Определим множество ${T_A}^n$ вычислительных термов ($n \geqslant 0$). Любая константа $\mathrm{c} \in C$ и любая переменная $\mathrm{x}_i \in \mathrm{X}_n$ — это вычислительный терм. Если φ_i — функциональный символ и $\mathrm{R}(\varphi_i) = r$ и $\mathrm{t}_1, \ldots, \mathrm{t}_r$ — вычислительные термы, то $\varphi_i(\mathrm{t}_1, \ldots, \mathrm{t}_r)$ — вычислительный терм. Если π_j — предикатный символ и $R(\pi_j) = s$ и $\mathrm{t}_1, \ldots, \mathrm{t}_s$ — вычислительные термы, то $\pi_j(\mathrm{t}_1, \ldots, \mathrm{t}_s)$ –предикатный терм. Если p — предикатный терм и t вычислительный терм, то (p:t) и (p:t)–вычислительные термы. Определим для вычислительных термов функцию оценки **val**. Если x — значение константы cx или переменной x, то **val**c$x = x$, **val**x$= x$. Далее $\mathbf{val}\varphi_i(\mathrm{t}_1, \ldots, \mathrm{t}_r) = \varphi_i(\mathbf{val}\mathrm{t}_1, \ldots, \mathbf{val}\mathrm{t}_r)$; $\mathbf{val}\pi_j(\mathrm{t}_1, \ldots, \mathrm{t}_s) = \pi_j(\mathbf{val}\mathrm{t}_1,$

$\ldots$,**val**t$_s$). Наконец, **val**(p:t)=**если val**p, **то val**t **иначе** *не определен*,**are**(p:t) ==**если val**p, **то** *неопределен*, **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 $\mathrm{Det}_f \subseteq \mathrm{T}_A{}^n$(детерминант функции f), такое, что

(1) $\forall\ (x_1, \ldots, x_n, y) \in F \exists t(\mathrm{x}_1, \ldots, \mathrm{x}_n) \in \mathrm{Det}_F$:**val** t $(x_1, \ldots, x_n) = y$;

(2) $\forall\ (x_1, \ldots, x_n, y)\ \forall t\ (x_1, \ldots, x_n) \in \mathrm{Det}_F$:**val**t $(x_1, \ldots, x_n) = y \Rightarrow (x_1, \ldots, x_n, y) \in F$.

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 детерминанта. Напомним, что для ОА детерминантом может быть конечный двухленточный автомат. Для рекурсивных программ Розен (Розен, 1975) показал, что детерминант, весьма похожий на наш, описывается контекстно-свободным языком. Этот результат, однако, еще не может считаться окончательным: КС-грамматики имеют неразрешимую проблему эквивалентности, а разрешимость детерминантов рекурсивных программ — открытая проблема с шансами на успех (см. последний результат Сабельфельда (Сабельфельд, 1981)).

Заметим, что это определение — чисто синтаксическое. Оно задает пучок вычислимых функций, которые для разных интерпретаций операционных символов будут иметь один и тот же детерминант, а рисунок графиков будет, помимо конкретных значений, различаться еще и областями 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 отношений и конструкторов, образующих систему операций, которая через то или иное определение вычислимости будет создавать на данном носителе некоторый „абсолютный“ класс вычислимых функций, и этот носитель 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 нас оракулами по существу, и нам не остается ничего другого, как чисто аксиоматически 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 (Крайзел, 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“ составляет один 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 благодаря разделению сущностей. Другой, совершенно не затронутой стороной теории является разработка методов синтеза программ из разных форм априорного значения о задаче в разных предметных областях. Эта сторона проблемы, однако, требует своего анализа.