АБСТРАКТНАЯ ВЫЧИСЛИМОСТЬ В АЛГЕБРАИЧЕСКИХ СИСТЕМАХ

А.П.Ершов

Новосибирск, СССР

Введение

Нас интересует абстрактная вычислимость, т.е. общая теория вычислимых функций, по отношению к которой предметная область и элементарные правила вычислений выступают в качестве формальных параметров, наделенных некоторыми аксиоматически заданными свойствами. И хотя автора этот вопрос волнует уже двадцать пять лет, одного интереса, пожалуй, было бы недостаточно, чтобы решиться что-то сказать по поводу этой фундаментальной проблемы, традиционно находящейся в сфере компетенции специалистов по математической логике и основаниям математики. За эти двадцать пять лет, однако, сложилась вычислительная наука (информатика, машинная математика, программирование), которая нуждается в своем взгляде на фундаментальные концепции теории вычислений. Диалог между логиками и вычислителями, характерный для этого симпозиума, должен помочь выработать вычислителям более просвещенный взгляд на основы их науки, отражающий в то же время ее особенности.

Сначала мы кратко изложим побудительные мотивы к поискам концепции абстрактной вычислимости, так как они сложились у автора. В то же время эти соображения можно считать достаточно характерными для математиков-вычислителей. Затем мы сформулируем предлагаемый нами подход к определению абстрактной вычислимости. Поскольку статья писалась некоторое время спустя после симпозиума, оказалось возможным на основе начальной реакции логиков (в целом поощрительной) проанализировать уже имеющиеся подходы к абстрактной и обобщенной вычислимости. Сопоставив эти подходы с нашими, мы излагаем некоторые альтернативы, в рамках которых можно двигаться в дальнейшей проработке указанной проблемы.

Следует признать, что строгой и общепризнанной абстрактной теории вычислимости не существует. Наиболее известные уточнения эффективных процедур и эффективно вычислимых функций (рекурсивные функции, машины Тьюринга, нормальные алгоритмы и еще две-три теории) не только возникли, но и существуют сами по себе со своей собственной литературой. Доказанные взаимные сводимости успокаивают тех, кого интересует только объем запаса вычислимых функций, однако маскируют сущности и смущают специалистов по сложности, поскольку кодирующие функции остаются за пределами теории. На верхнем уровне сложилась так называемая инвариантная теория, однако язык этой теории представляет собой своеобразный полуформализованный жаргон. В доказательствах теорем, служащих своего рода "входом" в инвариантную часть теории, разрыв между простым идейным содержанием и громоздкими правилами программирования выглядит удручающе*). Определение алгоритма, основанное на понятии механического вычисления, выглядит математически незамкнутым, поскольку опирается на принципиально неформальное описание работы машины - универсального алгоритма. Тот факт, что универсальный алгоритм оказывается программируемым в языке той же машины, не меняет положения дел.

Все основные факты общей теории вычислимости извлекаются из теории рекурсивных функций. И хотя значительная часть теории выразима в инвариантной форме, ее слишком тесная связь с системой $\langle \omega, 0, +1, = \rangle$ очевидна. Конечно, никто не станет отрицать особую роль натуральных чисел в математике. В любой аксиоматике натуральные числа, по-видимому, представляют наиболее простой вид конструктивных объектов. Однако именно в силу этой простоты конкретная теория арифметических вычислимых функций сливает сущности, затрудняя усмотреть важные различия (сравни Крайсел 1969, с. 142). Так, вкратце, выглядит для программиста классическая теория вычислимости.

С другой стороны, машинная математика постоянно побуждает работающих в ней к более и более абстрактному взгляду на программирование. Главная тенденция - увидеть и уметь рас-

*) В качестве примера реакции на эту ситуацию см. Глушков, 1979.

суждать о вычислимой функции раньше появления и независимо от наличия задающей ее программы. Вот почему для программистов так важны характеризации вычислимых функций в виде неподвижных точек и конструктивных теорем существования.

Занимаясь поиском алгоритма решения задачи, программист хотел бы как можно дольше оставаться в рамках той предметной области, которая естественна для постановки задачи. Это позволяет легче найти содержательные отношения и свойства, которые окажутся ему полезны при последующем систематическом выводе программы. Часто программист должен осуществлять встречный процесс: отправляясь от априорно заданного языка данных и примитивных операций, определить, в какой степени и как они могут быть использованы при решении задачи. В обоих случаях речь идет о программировании по отношению к заданной алгебраической системе (данные, операции и отношения). Вот почему для программиста относительная вычислимость является скорее первичным понятием, нежели обобщением "абсолютной" вычислимости.

Огромное место в вычислительной науке занимают манипуляции с уже написанными программами. Как правило, применяемые методы приобретают общий характер только в том случае, если они носят схемный характер, т.е. воспринимают константы, переменные и операции программы как формальные символы. Таким образом, программированию по существу необходима теория вычислений в произвольных алгебраических системах.

Естественно, что при манипулировании с программами необходимо сохранять определенные инварианты, гарантирующие правильность работы программы во всех ее обличиях. Этот инвариант также должен носить схемный характер, при этом как можно в более абстрактной форме, чтобы обслуживать разные модели вычислений. Как обычно, особую роль играют такие инварианты, которые обеспечивают разрешимость соответствующей проблемы эквивалентности. Вот почему программированию так нужны схемные характеризации вычислимых функций, которые обладали бы, одновременно, и большой общностью и простой структурой.

Подход

При пояснении основной идеи будем говорить для большей простоты об одноместных функциях. Начнем с двух всем известных

исходных "определений".

Функция $f:D \to D$ вычислима, если существует алгоритм, вычисляющий ее значения.

Алгоритм - это единый эффективный метод получения нужного результата, отправляясь от заданного аргумента, в конечное число элементарных шагов.

Любая точная теория алгоритмов начинается с введения некоторого языка, с помощью которого изображаются данные в виде констант предметной области D , а также - что самое главное - программы. Программы реализуют единство метода тем, что они являются единственным источником информации о способе получения результата при заданных аргументах. Способ, в свою очередь, реализуется некоторым сверхалгоритмом, применяемым единым образом к любой программе.

Эффективность обеспечивается четырьмя "конечностями". Информация о сверхалгоритме конечна, непосредственно обозреваема и одна и та же для любой программы из класса. Информация, содержащаяся в программе, конечна, эффективно обозреваема сверхалгоритмом и одна и та же для любого аргумента вычислимой функции. Информация, содержащаяся в з а д а н н ы х аргументах, конечна, эффективно обозреваема и одна и та же в течение всей реализации вычислительного процесса. Четвертая, уже упомянутая "конечность", - это конечность числа элементарных шагов, выполняемых на пути к результату.

Сформулируем еще раз отправной тезис: функция f может считаться вычислимой, если для каждой точки (x, y) ее графика мы можем "предъявить" "систему" "шагов" (протокол), непосредственно "приводящих" от x к y, систему, "извлеченную" "закономерно" из некоторого одного "источника" информации. Совокупность всех протоколов, т.е. отвечающая всем точкам графика функции f , должна полностью характеризовать функцию f .

"Принципиально разных" шагов должно быть конечное число. Однако каждый шаг, выполнение которого признается элементарным, применяется к бесконечному разнообразию экземпляров данных. Другими словами, одна из совокупностей шагов - это набор базисных операций $\Phi = \{\varphi_1, \ldots, \varphi_m\}$.

Проанализируем, как "источник" информации создает "систему" шагов. В какой-то момент мы можем выполнить только один шаг: так мы получаем цепочки. Разумно также допускать возможность произвольного выбора из конечного числа шагов: так мы получаем деревья. Другими словами, одна из форм предъявления системы шагов для получения y из x (протокол) - это функциональный терм (φ-терм) в сигнатуре Φ с аргументами x (задаваемый извне аргумент функции) и некоторыми константами из D. Для терма $\tau(x)$ взятие его значения $\underline{val}\ \tau(x)$ при значении x аргумента x считается бесспорно выполнимым и это выполнение (так называемое прямое вычисление) не формализуется.

В дополнение к этому мы знаем, что в ряде случаев выбор одного из нескольких шагов делается не произвольно, а на основе некоторого решения. Мы полагаем, что все эти решения должны быть отражены в "системе шагов", которую мы предъявляем в доказательство вычислимости результата. Так у нас появляется сигнатура предикатных символов $\Pi = \{\pi_1, \ldots, \pi_n\}$ с помощью которых строятся обычным образом предикатные термы, или π-термы. Взятие значения π-терма также считается прямым вычислением и не формализуется. Если некоторое прямое вычисление τ обусловлено истинностью или ложностью предикатного терма π, то эту информацию мы изображаем в виде "обусловленных" термов $(\pi:\tau)$ и $(\neg\pi:\tau)$ соответственно. При свободном конструировании обусловленного терма его значение определяется следующим образом: при $\underline{val}\ \pi$ = $\underline{ист}$ $\underline{val}(\pi:\tau) = \underline{val}(\tau)$, а $\underline{val}(\neg\pi:\tau)$ неопределен; при $\underline{val}\ \pi$ = $\underline{ложь}$ $\underline{val}(\pi:\tau)$ неопределен, а $\underline{val}(\neg\pi:\tau) = \underline{val}\ \tau$. Взятие значения обусловленного терма также считается прямым вычислением и не формализуется. Заметим, что противоречивый (т.е. содержащий вместе π и $\neg\pi$) терм всегда неопределен.

Разрешая подставлять обусловленные термы в позиции аргументов φ- и π-символов из сигнатур Φ и Π, мы получаем пространство вычислительных термов T, которые и будем считать протоколами вычисления значений вычислимой функции.

Дадим теперь схему определения вычислимой функции. Рассмотрим алгебраическую систему $A = \langle D, C, \Phi, \Pi, R\rangle$, $R:\Phi \cup \Pi \to \omega$, где ω - натуральный ряд, R - тип алгебраической системы, Φ - функциональные и Π - предикатные символы, D - носитель

(предметная область), C - константы носителя. Пусть T_A - пространство вычислительных термов с аргументами из алфавита $C \cup \{x\}$.

С х е м а о п р е д е л е н и я . Функция $f:D \to D$ называется вычислимой в A , если существует множество $Det_f \subseteq T_A$ (детерминант функции f), такое, что

$$\forall(x, y) \in f \ \exists\tau(x) \in Det_f : \underline{val}\ \tau(x) = y; \qquad (1)$$

$$\forall x \forall\tau(x) \in Det_f : \underline{val}\ \tau(x) = y \Rightarrow (x, y) \in f. \qquad (2)$$

Еще не обсуждая эту схему определения по существу, можно заметить две особенности. Во-первых, определение не исключает многозначных функций. Для обеспечения однозначности необходимо, чтобы для заданного x все протоколы из детерминанта, определенные для y , давали бы одно и то же значение.

Во-вторых, классические определения вычислимости могут быть выражены в этой схеме. Понятие протокола, с теми или иными вариантами, хорошо известно в теории алгоритмов и в программировании. Условие (2) всегда можно обеспечить, сделав протокол достаточно детальным.

Естественно, что детерминант должен быть эффективно породимым множеством. Апостериори он оказывается перечислимым множеством, т.к. может строиться незначительным усовершенствованием универсального алгоритма. Наконец, в конечно порожденной системе с равенством он тривиально оказывается изоморфным графику функции, задаваясь с помощью конструкции

$$Det_f = \{\underline{\text{если}}\ x = c_x\ \underline{\text{то}}\ c_y \mid (x, y) \in f\},$$

где c_x - константа или терм, соответствующий объекту x .

Определять, однако, детерминант, требуя просто его перечислимости, нам представляется неинтересным, т.к. мы ищем определение вычислимости, которое не опиралось бы на эквивалентные понятия. Известно, правда, что перечислимые множества могут перечисляться некоторыми очень простыми субрекурсивными функциями, однако хотелось бы найти предельно узкий класс порождения детерминантов, например, какими-либо автоматами.

Принципиальным допущением на этом пути является разрешение включать в детерминант незначимые термы, т.е. "протоколы", порождаемые порождающим процессом, но не реализуемые никакой моделью алгебраической системы. Ничто не мешает нам включать такие протоколы в детерминант, лишь бы обеспечивалось

требование (2) определения.

Заметим, далее, что так определенная вычислимость не противоречит интуитивному представлению эффективной вычислимости. Действительно, вычислять значение функции по детерминанту можно следующим образом: порождаем вычислимые термы детерминанта один за другим и пытаемся их вычислить для заданного x . Первый терм, определенный для x , даст нужное значение.

Заменяя понятие программы детерминантом, мы абстрагируемся тем самым от многих вещей: прежде всего от конкретного синтаксиса программ, а также от устройства универсального алгоритма.

Следующим очень важным уровнем абстракции является обеспечение возможности описывать и порождать детерминанты независимо от конкретной реализации исходной алгебраической системы. Естественно, что если бы мы требовали от протоколов полной равнообъемности с графиком функции, то это была бы в принципе неразрешимая задача. Разрешение, однако, порождать противоречивые и незначащие компоненты детерминанта позволяет, по крайней мере принципиально, описывать детерминант, учитывая лишь тип алгебраической системы и используя ее сигнатуру как алфавит формальных символов. Естественно, что для доказательства содержательных теорем из теории вычислимости нам потребуются дополнительные сведения о предметной области и об элементарных операциях-оракулах, но можно надеяться на то, что эти свойства удастся выразить аксиоматически.

Анализ родственных работ

Не претендуя на полноту обзора, мы прокомментируем известные автору работы, содержащие материал, который, по нашему мнению, может содействовать построению теории абстрактной вычислимости на основе понятия детерминанта. Мы предпримем попытку с единой точки зрения рассмотреть как работы, выполненные в рамках теоретического программирования, так и работы, находящиеся в сфере математической логики. Если в первой области автор может считать себя экспертом, то в отношении второй области автор претендует в лучшем на роль прилежного читателя.

Нет сомнения, что в основополагающих работах по теории рекурсии и эффективной вычислимости содержится немало предпосылок к абстрактной теории, однако их экспликация потребует специального историко-научного исследования. Мы подведем черту начальному периоду известной монографией Клини, 1952, заметив только, что понятие относительной рекурсивности, хотя оно в классической теории опирается на конкретную систему $\langle \omega, 0, +1, = \rangle$, явилось для многих отправной точкой в поисках абстрактной вычислимости.

Мы сознательно не будем как-либо сортировать ссылки между логикой и программированием, нанизав все работы на ось времени. Статьи, представленные в виде докладов на конференциях, относятся ко времени проведения конференции. Некоторые работы автору оказались недоступны и комментируются, используя промежуточные ссылки. Всюду ниже $\langle D, \Phi, \Pi \rangle$ означает (алгебраическую) систему с носителем D, (функциональными) φ-символами $\Phi = \{\varphi_1, \ldots, \varphi_m\}$ и (предикатными) π-символами $\Pi = \{\pi_1, \ldots, \pi_n\}$. Функциональная часть может отсутствовать (в таких случаях обычно подразумевается наличие равенства). Константы могут трактоваться и как отдельные символы, и как нульместные функции. ω обозначает натуральный ряд с нулем. Система, в которой сигнатуре не сопоставлены конкретные операции и отношения, будет иногда называться абстрактной системой. Конкретные системы, сопоставляемые абстрактной системе, будут называться ее реализациями, или интерпретациями.

Алгебраическая система, над которой рассматриваются программы, их схемы и вычисляемые ими функции, будет называться базовой системой.

Янов, 1957, на основе идей А.А.Ляпунова изучал представление алгоритмов в системах $\langle D, \Phi, \Pi \rangle$, где функции и предикаты только одноместные. Программы изображались в виде граф-схем, где условиями были произвольные булевы функции π-символов, а операторами действия - отдельные φ-символы, работающие только с одной ячейкой памяти. Янов, по-видимому, первый ввел в литературу понятие детерминанта как множества конфигураций, порождаемых из представления программы. Будем записывать произвольный набор значений π-символов в виде $\Delta = \pi_1^{\sigma_1} \ldots \pi_n^{\sigma_n}$,

где π^{σ} - это π (π истинно) или $\neg\pi$ (π ложно). Порождение конфигурации K (сначала пустой) задается так: с произвольным набором значений π-символов поступаем на вход граф-схемы. Пусть теперь с набором Δ мы движемся по граф-схеме однозначно, пока не дойдем до вершины с символом φ или не выйдем на конец (порождение обрывается), или не зациклимся на условиях (порождение обрывается). В первом случае к текущему значению K записываем пару $(\Delta\varphi):K\to K(\Delta\varphi)$ и выбираем произвольным образом новый набор значений π-символов, с которым продолжаем движение по граф-схеме. K может оказаться как конечным, так и бесконечным. Две граф-схемы над общей сигнатурой формально эквивалентны, если их детерминанты совпадают. Янов установил разрешимость этой эквивалентности и построил полное исчисление, позволяющее любую граф-схему преобразовать в любую ей эквивалентную.

Как ни странно, до сих пор никто не исследовал, для каких Φ и Π граф-схемы Янова в системе $\langle\omega, \Phi, \Pi\rangle$ будут задавать все рекурсивные функции.

Ершов, 1958, рассмотрел представление алгоритмов в виде граф-схем в произвольных алгебраических системах $\langle D, \Phi, \Pi\rangle$ без ограничений на тип сигнатуры и число переменных. Операторами действия были последовательности присваиваний $(x:=\tau)$, где x - переменная, а τ - произвольный φ-терм. Логическими условиями являются π-термы или их булевы композиции. Он описал универсальную процедуру выполнения таких программ и содержательное отношение эквивалентности как равенства вычислимых частичных функций. Было введено понятие термального значения (S-представления) переменной как φ-терма, образованного суперпозицией операций, приведших к получению текущего значения переменной. Был найден ряд конкретных так называемых алгоритмически полных систем операций и отношений в конструктивной области D, обеспечивающих вычисление всех вычислимых функций над D.

Эта вычислимая модель, восходящая к блок-схемам фон Неймана и независимо вводившаяся многими авторами, в дальнейшем получила широкое распространение. Представление программ в этой модели мы будем называть граф-программами с памятью.

Криницкий, 1959 (см. также Криницкий, 1970), в своей диссертации рассмотрел программы с памятью (граф-схемы с памятью) в абстрактных системах $\langle D, \Phi, \Pi \rangle$. Он ввел понятие функциональной эквивалентности граф-схем с памятью как вычисление одинаковых функций для любой интерпретации базовой системы. Для случая граф-схем, не содержащих циклов, Криницкий доказал разрешимость функциональной эквивалентности и построил для этого класса схем полную систему преобразований. В литературе граф-схемы с памятью называются также стандартными схемами.

Фрассе, 1959 (по Московакису, 1969б, и Лакомбу, 1969), связал абстрактную рекурсивность (F-рекурсивность) предиката P в системе $\langle D, \Pi \rangle$ с выводимостью в обычном исчислении предикатов с равенством. Для этого в язык L исчисления предикатов добавляются в качестве предикатных символов $\pi_1, \ldots, \pi_n$, символ определяемого предиката P, а также индивидуальные константы c_x для каждого элемента $x \in D$. С каждым предикатным символом π_i^r связывается счетное множество формул $\Delta(\pi_i^r)$ вида $\pi_i(c_{x_1}, \ldots, c_{x_r})$ или $\neg\pi_i^r(c_{x_1}, \ldots, c_{x_r})$ в зависимости от истинности или ложности $\pi(x_1, \ldots, x_r)$. Пусть $\Delta(\Pi) = \Delta(\pi_1) \cup \ldots \cup \Delta(\pi_n)$. Рассмотрим некоторую формулу $\Phi(P)$ в языке L. Тогда говорится, что предикат $P(x_1, \ldots, x_n)$ F-рекурсивен в системе $\langle D, \Pi \rangle$, если он определяется по следующему правилу

$$P(x_1, \ldots, x_n) \leftrightarrow \Delta(\Pi) \cup \{\Phi(P)\} \vdash P(c_{x_1}, \ldots, c_{x_n})$$

и

$$\neg P(x_1, \ldots, x_n) \leftrightarrow \Delta(\Pi) \cup \{\Phi(P)\} \vdash \neg P(c_{x_1}, \ldots, c_{x_n}).$$

В силу полноты исчисления предикатов это, как сказали бы программисты, схемное определение эквивалентно теоретико-модельному определению предиката $P(x_1, \ldots, x_n)$ с помощью формулы $\Phi(P)$. Пусть $\Phi(P)$ - формула исчисления предикатов и пусть $C_\Phi(D, \Pi)$ - класс всех моделей формулы Φ, получаемых всевозможными расширениями системы $\langle D, \Pi \rangle$ с сохранением носителя. Тогда предикат $P(x_1, \ldots, x_n)$, который сохраняет свои значения на каждом расширении из $C_\Phi(D, \Pi)$, и есть F-рекурсивный предикат, задаваемый $\Phi(P)$.

Если взять в качестве носителя ω и в качестве $C_\Phi(\omega, \Pi)$ - класс всех моделей формулы Φ в формальной арифметике, то F-

-рекурсивность в $\langle \omega, \Pi \rangle$ совпадает с обычной относительной рекурсивностью.

Маккарти, 1961, ввел в рассмотрение новую модель для вычислений в произвольных системах $\langle D, \Phi, \Pi \rangle$, названную им рекурсивными программами и - в абстрактной форме - рекурсивными схемами. Его основная конструкция - условный терм, имеющий вид $(\pi \to \varphi, \phi)$, где π - предикатный терм, а φ и ϕ - функциональные или условные термы. Условный терм соответствует определению разбором случаев. Эквивалентные обозначения
if π then φ else ϕ или if π then φ else ϕ fi или $(\pi|\varphi|\phi)$.
Условный или функциональный термы называются операционными. Алфавит операционных термов дополняется символами определяемых функций $F = \{f_1, \ldots, f_k\}$, входных переменных $X = \{x_1, \ldots, x_s\}$ и формальных переменных $U = \{u_1, \ldots, u_t\}$. Рекурсивная программа имеет вид

$$H(X, F, \Phi, \Pi),$$
$$f_1(U_1) = \tau_1(U_1, F, \Phi, \Pi),$$
$$\ldots\ldots\ldots\ldots$$
$$f_k(U_k) = \tau_k(U_k, F, \Phi, \Pi).$$

H - (главная программа) операционный терм, а $f_i(U_i) = \tau_i(U_i, F, \Phi, \Pi)$ - рекурсивное уравнение, в котором τ_i - операционный терм, а U_i - набор формальных переменных.

Маккарти показал, что любая граф-схема с памятью в системе $\langle D, \Phi, \Pi \rangle$ транслируема в рекурсивную схему в той же системе. Им было также показано, что рекурсивные программы в системе $\langle \omega, 0, +1, = \rangle$ вычисляют любую арифметическую рекурсивную функцию.

Эта модель вычислений получила большое распространение в теоретическом программировании благодаря тому, что наименьшие неподвижные точки рекурсивных уравнений оказались хорошим семантическим представлением функций, вычисляемых по рекурсивным программам.

Мальцев, 1961, предложил изучать понятие рекурсивности в произвольных алгебраических системах $\langle D, \Phi, \Pi \rangle$ с помощью отображений α натурального ряда на носитель, связывающий операции $f(u_1, \ldots, u_r)$ в системе с арифметическими функциями $F(x_1, \ldots, x_r)$ с помощью соотношения

$$f(\alpha u_1, \ldots, \alpha u_r) = \alpha F(x_1, \ldots, x_r).$$

Хотя в цитируемой работе собственно теория рекурсии не получила особого развития, высказанная Мальцевым идея не только отразила, но и в значительной степени сформировала тенденцию изучать абстрактные системы в виде их числовых прообразов и самих их нумераций.

Вагнер, 1963 (по Вагнеру, 1969), предложил подход к изучению абстрактной вычислимости, основываясь на, так сказать, аксиоматике высокого уровня, характеризующей весь класс вычислимых функций в целом. Он постулирует априори, что элементы базового множества U являются программами (индексами) функций над этим же множеством и что существует априорно заданная операция взятия функции по ее индексу u и ее применения к аргументу x, обозначаемая $[u](x)$. Точно так же в качестве постулата берется и s-m-n-теорема (Клини, 1952, гл. XП, теорема XXШ), которая, собственно говоря, используется в качестве определения функции нескольких аргументов: по определению $[u](x,y) = [[u](x)](y)$. В результате для построения содержательной теории достаточно постулировать существование в U неопределенного элемента $*$ с аксиомой

$$[u](*) = * = [*](u),$$

некоторой "функции смешивания" (blending function) α с аксиомой, обобщающей правило подстановки:

$$\left.\begin{aligned} &[\alpha](f, g) \neq * \\ &[[\alpha](f, g)](x) = [f](x, [g](x)) \end{aligned}\right\}$$

и функции вычисления разбором случаев ϕ с аксиомой

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

Множества, удовлетворяющие этим аксиомам, названы Вагнером униформно рефлексивными структурами (УРС). Из этих аксиом вытекает существование многих "стандартных" вычислимых функций (константы, тождественная функция, функции выбора аргумента), наличие мощных теорем замыкания и ряд других свойств, обычно адресуемых вычислительным функциям и их классам. Показывается далее, что на натуральном ряде существуют рекурсивные конструкции, которые обеспечивают выполнение на ω аксиом УРС. Возникающий на этой структуре класс функций в точности совпадает с частично-рекурсивными функциями.

В то же время отсутствие аксиоматически задаваемых свойств

конструктивных объектов не позволяет в произвольной УРС выделить без дополнительных предположений класс функций, интуитивно удовлетворяющих представлению об эффективной вычислимости, в частности, перечислимости. Для выделения такого класса Вагнер использует тот факт, что в любой УРС существует одноместная функция и элемент, могущие играть роль функции счета и нуля. В результате в УРС возникает образ натурального ряда. Далее постулируется, что этот образ является вычислимым в том смысле, что в УРС существует его характеристическая функция. Тогда в УРС создается возможность промоделировать операторы композиции (в любой УРС) примитивной рекурсии (в любой УРС) и μ-оператор (только в УРС с вычислимым натуральным рядом), а, стало быть, и весь класс частично рекурсивных функций.

Работа Вагнера сыграла, как кажется, весьма побудительную роль в дальнейших исследованиях по абстрактной вычислимости.

Крайсел, 1963 (по Московакису, 1969б, и Лакомбу, 1969), ввел весьма общее понятие инвариантной определимости предиката с помощью некоторой формулы α в языке вычисления предикатов с равенством. Под инвариантностью имеется в виду выполнимость α на всех интерпретациях функциональных и предикатных символов на рассматриваемой предметной области. Опираясь на совпадение понятия инвариантной определимости и рекурсивности для арифметических предикатов, Крайсел обратил внимание на полезность использования этого понятия для изучения обобщенной и абстрактной рекурсии. Эта точка зрения находит свое подтверждение и в литературе по теоретическому программированию, где также инвариантные конструкции иногда оказываются равнообъемными со схемными конструкциями. Ср., например, с Яновым, 1957; Ратледжом, 1964, и Фрассе, 1959.

Ратледж, 1964, показал, что детерминанты в схемах программ Янова образуют язык, воспринимаемый конечным автоматом. Он ввел понятие функциональной эквивалентности схем Янова (вычисление одинаковых функций для любой интерпретации базовой системы) и показал, что оно равнообъемно отношению формальной эквивалентности (совпадение детерминантов).

Глушков, 1965, ввел в качестве абстрактной модели вычислений понятие дискретного преобразователя. Дискретный преоб-

разователь работает над некоторым информационным множеством S и представляет собой п а р у автоматов: управляющий автомат с входным алфавитом X и выходным алфавитом Y и операционный автомат с входным алфавитом Y и выходным алфавитом X . Грубо говоря, X связывается с сигнатурой предикатов, Y - с сигнатурой функций, а информационное множество - с носителем базовой алгебраической системы. Воспринимая входный символ $x \in X$, управляющий автомат выдает выходной символ $y \in Y$ операционному автомату, который определяет некоторое преобразование $f_y: S \to S$, и выдает соответствующий результату преобразования выходной символ из X . Эта конструкция отчетливо вскрыла носившееся в то время в воздухе различие "логической" и "вычислительной" сторон работы алгоритма.

Патерсон, I968, независимо ввел в рассмотрение модель граф-схем с памятью и показал для них неразрешимость функциональной эквивалентности. Этот отрицательный результат оказал заметное стимулирующее влияние на поиски формальных отношений эквивалентности схем программ с использованием того или иного понятия детерминанта (см. Иткин, I972).

Стронг, I968, придает понятиям Вагнера, I963, более алгебраическую трактовку, а также подвергает его постулаты и аксиомы более детальному анализу. Отделяя пространство функций $F=\{F\}\ f:D^n \to D$ $(n=0,1,\ldots)$ от множества D , пробегаемого аргументами, он выписывает два варианта аксиом, которым должно удовлетворять F , чтобы на D можно было бы задать некоторую униформную рефлексивную структуру (УРС). Первый вариант задает так называемую (употребим старорусский оборот, чтобы сохранить английский порядок слов) базисную рекурсивных функций теорию (БРФТ):

(I) F содержит константные функции для любого элемента D, а также 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 каждый $m,n>0$ в F имеется тотальная функция, удовлетворяющая условиям s-m-n-теоремы (или, как сказали бы

программисты, универсальная функция для частичных, или смешанных вычислений).

Второй вариант аксиом предназначается для пространства F_1 одноместных функций:

(I) F_1 содержит константные функции для любого элемента D, тождественную функцию и обе обратных функции к некоторой фиксированной и внешне заданной функции пересчета пар (pairing function).

(2) F_1 содержит функцию, которая в сочетании с пересчетом пар задает характеристическую функцию равенства константе.

(3) F_1 содержит функцию, которая в сочетании с пересчетом пар задает универсальную функцию для функций из F_1.

(4) F_1 содержит функцию, которая в сочетании с пересчетом пар задает универсальную функцию для частичных вычислений (s-I-I-функция).

Фридман, I969a, улучшает аксиоматику Стронга, I968 (ее второй вариант), показывая, что БРФТ может быть описана для множества функций F над D следующими аксиомами:

(I) D имеет не менее двух элементов.

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

(3) F замкнута относительно подстановки.

(4) D содержит тождественную функцию, функцию пересчета пар и обратные к ней.

(5) F содержит все одноместные функции-константы.

(6) F содержит характеристическую функцию равенства.

(7) F содержит универсальную функцию для одноместных функций.

Улучшение состоит в том, что s-m-n-функция не включается в аксиоматику, а ее отсутствие компенсируется тонкими отличиями в других аксиомах.

Лакомб, I969, рассматривает вычислимость в реляционных системах с равенством $\langle D, \Pi, = \rangle$, где D - нумерованный носитель. Аналогично Мальцеву, I96I, свойства рекурсивности в ω, усматриваемые в образах нумераций, объявляются таковыми для прообразов в D. Предикат $P(x_1,\ldots,x_r)$ абстрактно рекурсивен в Π, если он рекурсивен для любой нумерации D. Для так определенной рекурсивности он находит характеризацию вычислимых предикатов примитивно-рекурсивными множествами булевых формул

в алфавите П, Р, их предметных переменных $x_1,\ldots,x_t$ и конечного набора предметных констант $a_1,\ldots,a_k$. Эти множества булевых формул близки к нашему понятию детерминанта. Корректность определения абстрактной вычислимости по Лакомбу подтверждается тем, что в $\langle\omega,0,+1,=\rangle$ его вычислимость совпадает с относительной вычислимостью. Кроме того, вычислимость по Лакомбу совпадает с вычислимостью по Фрассе, 1959.

Московакис, 1969а, судя по всему, независимо от Вагнера, 1963, и Стронга, 1968, определяет класс рекурсивных функций для произвольного множества D. Расширяя D выделенным элементом 0, он постулирует существование функции пересчета пар $z = (x,y)$, отображающей $D^*\times D^*\to D^*$ (D^* - замыкание $D \cup \{0\}$ относительно (x,y)). После этого в D^* выделяется натуральный ряд 0,1,2,... в виде объектов 0,(0,0),((0,0), 0) и т.д. с функцией счета n + 1 в виде (n, 0). С использованием этой модели натурального ряда 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 контролируется индуктивным построением их индексов - последовательностей натуральных чисел, сохраняющих информацию о строении функции и упакованных с помощью пересчета пар в объекты из D^*. Московакис показывает, что так определенные "примитивно-рекурсивные" функции относительного пустого Φ моделируют арифметические ПРФ.

Расширение области вычислимых функций на "частично-рекурсивные" функции Московакис делает, постулируя вычислимость универсальной (нумерующей) функции над D^*. При этом, однако, обеспечивается, что эта функция определена лишь в том случае, если первый аргумент этой функции оказывается индексом, построенным в соответствии с заданными индуктивными правилами. Необходимость в связи с этим "анализировать" объекты, которые претендуют на то, чтобы быть индексами (программисты сказали бы "проводить синтаксический анализ" входной программы), требует включения операций, аналогичных μ-оператору в обычной теории рекурсивных функций. Московакис рассматривает два варианта такого оператора, трактуя их как

операции поиска во множестве, которое может быть как упорядоченным, так и нет. В первом случае получается так называемая простая (prime) вычислимость, а во втором случае - поисковая вычислимость.

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

Московакис, 1969б, сводит воедино ряд предшествующих работ. Он показывает, что F-рекурсивность по Фрассе, 1959, вытекает из инвариантной определимости по Крайселу, 1963. В свою очередь, и это, пожалуй, главный результат, оказывается, что инвариантная определимость предиката P в системе $\langle D^*, \Pi \rangle$ влечет поисковую вычислимость P в системе $\langle D^*, \Pi, = \rangle$. Последующее доказательство сводимости F-рекурсивности к поисковой вычислимости в сочетании с эквивалентностью с вычислимостью по Лакомбу, 1969, устанавливает весьма четкий этап в исследовании понятия абстрактной вычислимости.

Крайсел, 1969, в своем в высшей степени интересном и богатом идеями обзоре подвел итог работам 60-х годов по обобщению теории рекурсии. Подробно анализируя цели и возможные подходы к развитию теории рекурсии, он фиксирует немало невыясненных тонкостей в основах теории и выдвигает интересные постановки, некоторые из которых созвучны целям нашего исследования. В частности, его анализ позволяет уточнить одно существенное различие между абстрактной и обобщенной теориями рекурсии. Последняя в распространении обычной рекурсии на более общие структуры существенно опирается на классическую теорию, либо используя нумерацию, либо опираясь на понятие перечислимого множества, либо благодаря "буквальному" переносу схем рекурсивных определений на абстрактные функциональные пространства. В то же время абстрактная теория вычислимости, в нашем представлении, должна строиться как формальная теория без каких бы то ни было ссылок на теорию вычислимых арифметических функций. Хотелось бы видеть абстрактную вычислимость в виде логической теории, по отношению к которой обычная теория рекурсии выступала бы как модель или реализация.

Фридман, 1969-б, практически предвосхитил наш подход к абстрактной вычислимости. Он обращает внимание на случаи

скрытого использования обычной рекурсии при изучении вычислимости в произвольных системах. Рассматривая произвольную алгебраическую систему $A=\langle D,C,\Phi,\Pi\rangle$ с константами $C=c_0,\ldots,c_k$, Фридман описывает две модели вычислений в A: модель P, которая практически совпадает с граф-схемами по Ершову, 1958, Криницкому, 1959, и Патерсону, 1968, и модель T, которая обобщает понятие машины Тьюринга (на одну клетку ленты помещается один элемент из D или вспомогательный символ). Наконец, он вводит модель эффективных определений, модель Д, которая практически совпадает с нашим понятием детерминанта. Роль протокола у Фридмана играет "клауза" вида $(C \to t)$, где C либо пусто, либо непротиворечивая конъюнкция π-термов или их отрицаний, а t - функциональный терм. Клауза, естественно, означает вычисление ϕ-терма t при выполнении условия C. Пустое C всегда истинно. Роль детерминанта у Фридмана играет множество клауз, рекурсивно перечислимое в смысле обычной теории рекурсии и такое, что конъюнкция условий двух разных клауз всегда противоречива. Фридман показывает, что каждое вычисление в моделях P и T создает протокол, совокупность которых образует детерминант в модели Д. Обратное доказательство Д → T апеллирует к тезису Чёрча (в применении к модели T) и к перечислимости детерминанта. Доказательство P→T более сложно, поскольку P-программы могут использовать только ограниченное (независимо от вычислений) число ячеек памяти. Фридман, во-первых, использует P-машины со счетчиками, которые вычисляют по геделевым номерам клауз детерминанта и, во-вторых, постулирует наличие предиката проверки равенства.

Фридман называет T-рекурсивными функциями все функции, вычисляемые T-машинами в алгебраических системах $\langle D,\Phi,\Pi\rangle$ с использованием любого (конечного) количества констант из D. Он перечисляет 13 основных теорем элементарной теории рекурсии и затем, разбивая их на 6 групп (с пересечениями), показывает, какие комбинации дополнительных свойств, накладываемых на базовую систему, обеспечивают выполнимость теорем из той или иной группы. Этих дополнительных свойств три:

(1) T-рекурсивность предиката равенства.

(2) T-рекурсивность пересчета пар.

(3) Конечная порождаемость D с помощью некоторого числа

констант и T-рекурсивных тотальных функций. (В дальнейшем оказывается, что (I)&(3)→(2)).

Этот в высшей степени интересный анализ, к сожалению, концептуально небезупречен из-за использования по существу обычной теории рекурсии (перечислимость детерминантов).

Фридман отмечает, что программисты указали ему на связь его P-модели с граф-схемами по Патерсону, I968, и выражает надежду, что его анализ P-модели поможет математическому выражению основных различий разных моделей вычислений, известных программистам.

Московакис, I969в, предложил новый подход к аксиоматизации вычислимости на абстрактных множествах, 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, а на множестве 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 $f(x_1, \ldots, x_n)$ Θ - предвычислима, если существует такое $e \in C$, что для любой точки $x_1, \ldots, x_n, y$ графика f набор $(e, x_1, \ldots, x_n, y)$ принадлежит Θ. Благодаря свойству рефлексивности тотальным функционалам и операторам можно сопоставить функции и, стало быть, перенести на них понятие Θ-вычислимости.

Московакис называет множество вычислений Θ предвычислительной теорией, если следующие функции, функционалы и операторы оказываются Θ-вычислимыми:

I. Константы, тождественная функция и функция "счета";

2. Характеристическая функция множеств C и N;

3. Функции пересчета пар;

4. Подстановка и примитивная рекурсия;

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 вычислимыми функциями и вычислимыми функционалами. Предвычислимая функция является вычислимой автоматически, а предвычислимый функционал F объявляется вычислимым, только если длина вычисления F , грубо говоря, не меньше длины вычислений функций, индексы которых являются аргументами вычисления функционала. Кроме того, чтобы предвычислительная теория была вычислительной теорией, требуется, чтобы длина вычисления (по универсальной функции) была бы, как скажут программисты, меньше длины частичного вычисления и последующего вычисления по остаточной программе (сравни Ершов, 1980).

Московакис показывает, что вычисления на машине Тьюринга образуют вычислительную теорию. Он устанавливает справедливость 1-й теоремы о рекурсии для вычислительных теорий и демонстрирует, что простую и поисковую вычислимости по Московакису, 1969а, можно представить в виде вычислительных теорий.

Эта работа Московакиса представляется весьма интересной, и думается, что ее потенциал еще не полностью разряжен.

Патерсон и Хьюитт, 1970, привели пример рекурсивной схемы по Маккарти, 1961, которая не транслируется в какую бы то ни было граф-схему с памятью в той же абстрактной системе операций и отношений.

Иткин, 1972, получил важный результат, показав, что граф-схемы с памятью имеют разрешимый детерминант. Формальные протоколы строятся следующим образом. По граф-схеме прокладывается произвольный путь от входа до выхода. Выписывается последовательность проходимых π-символов с отрицанием или без него в зависимости от выбора then - или else-дуги, исходящей из условия. Для каждого аргумента каждого π-символа выписывается термальное значение (смотри Ершов, 1958) этого аргумента, определяемое однозначно выбранным путем в схеме. Такой протокол называется логико-термальной (лт-) историей. Лт-детерминант - это множество всех конечных лт-историй. Граф-схемы с памятью называются лт-эквивалентными, если они имеют одинаковые лт-детерминанты. Лт-детерминанты полностью характеризуют вычислимые функции, т.к. из лт-эквивалентности следует функциональная эквивалентность.

Успенский, 1974, делает интересное наблюдение о роли протоколов в уточнении понятия относительной вычислимости (или вычислений с оракулом). Он замечает, что в теории рекурсии можно выделить четыре свойства класса K вычислимых функций с оракулом, из которых вытекают все утверждения теории рекурсии, которые релятивизируются к какому бы то ни было оракулу:

(1) Класс K содержит все (обычно) вычислимые функции.

(2) Класс K рекурсивно замкнут (т.е. относительно подстановки, рекурсии и минимизации).

(3) (Аксиома протокола). Для всякой f из K существует множество (кодов), характеристическая функция которого принадлежит K и функции a и b из K такие, что для любой точки (x, y) графика f существует такой элемент q из множества кодов, что $a(q)=x$ и $b(q)=y$.

(4) Класс K содержит универсальную функцию всех одноместных функций из K.

Буда и Иткин, 1974, показали, что лт-детерминант граф-схем с памятью по Иткину, 1972, образует язык, воспринимаемый двухленточным автоматом.

Непомнящий, 1974, изучает арифметические функции, задаваемые граф-программами с памятью над системами $\langle \omega, \Phi, \Pi \rangle$ и интересуется, каким свойством должны удовлетворять Φ и Π, чтобы множество задаваемых функций совпадало бы со всем классом частично-рекурсивных функций. Это свойство называется алгоритмической полнотой. Им получена теорема редукции, сводящая полноту системы $\langle \omega, \{\varphi(y)\}, \{p(y_1, \ldots, y_m)\} \rangle$ к полноте системы $\langle \omega, \{y+1\}, \{\pi(y_1, \ldots, y_m)\} \rangle$. Для систем вида $\langle \omega, \{y+1\}, \{\pi(y)\} \rangle$ указывается критерий полноты в терминах множеств, допустимых обобщенными конечными автоматами. Для системы вида $\langle \omega, \{\varphi(y)\}, \{\pi(y)\} \rangle$ дается простое достаточное условие полноты. Для функций одним из свойств является исчерпание φ-термами всего натурального ряда, а для предикатов таким свойством является наличие автоматной схемы испытаний, позволяющей узнавать о равенстве числа константе.

Грильо, 1974, также подвергает логическому анализу градации абстрактной рекурсии в зависимости от исходных допущений на базовую алгебраическую систему. Он также начинает свой

анализ важными для нас предупреждениями, говоря следующее:

"... Рекурсивность F в $G_1,\ldots,G_n$ значит, что F вычислимо или может быть комбинаторно построено в системе $\langle\omega;0,+1,=,G_1,\ldots,G_n\rangle$ ... Естественно заменить систему $\langle\omega;0,+1,=,G_1,\ldots,G_n\rangle$ на произвольную систему $\langle A;G_1,\ldots,G_n\rangle$, где A - множество, а $G_1,\ldots,G_n$ - функции или предикаты на A. Это в точности то, о чем наше исследование: найти, что значит рекурсивность в произвольной системе... К сожалению, дело не так просто, поскольку система натуральных чисел в высшей степени уникальна по отношению к другим системам. Следовательно, абстрактное изучение рекурсивности может оказаться обусловленным предположениями, неотъемлемыми от обычной рекурсивности в натуральных числах."

Грильо формулирует три основных допущения, характеризующих системы, над которыми строится рекурсивное замыкание:

C-схема: функции-константы должны быть рекурсивны;

E-схема: отношение равенства должно быть рекурсивно;

S-схема: неупорядоченный поиск (во множестве) должен быть рекурсивен (эквивалентная формулировка: полу-вычислимые предикаты замкнуты по отношению к навешиванию квантора существования).

Грильо анализирует разные определения рекурсивного замыкания. В качестве основного (он называет его синтаксическим) определения он берет схемное определение выводимости в языке исчисления предикатов, по Фрассе, 1959, и напоминает, что, в силу полноты исчислений предикатов, схемное определение эквивалентно семантическому (т.е. с использованием моделей, интерпретирующих базовую сигнатуру) с помощью инвариантной определимости по Фрассе, 1959; Крайселу, 1963, и Московакису, 1969а и б. В качестве критерия полноты рекурсивного замыкания Грильо берет инвариантность рекурсивности по отношению к произвольным нумерациям носителя по Лакомбу, 1969.

Грильо отмечает, что, по его мнению, теоретико-модельное (семантическое) определение рекурсивности дает ощущение полноты, отсутствующее в чисто комбинаторных определениях, и оказывается, тем самым важным содержательным подтверждением тезиса Чёрча.

Фенстад, (1974) строит вариант теории Московакиса (1969в), вводя на множестве вычислений Θ вместо длины вычислений 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 аксиомам Московакиса (1969в), где вместо постулирования неравенств на длины вычисления постулируются аналогичные отношения "быть подвычислением".

Розен (1974) рассмотрел для рекурсивных схем детерминант, практически совпадающий с нашим понятием детерминанта, введенного в предыдущем разделе. Он показал, что схемы с одинаковыми детерминантами для любой интерпретации исходной системы вычисляют одинаковые функции. Он показал также, что детерминант рекурсивных программ является контекстно-свободным языком. Этот результат нельзя однако считать окончательным, так как проблема распознавания эквивалентности кс-грамматик в общем случае неразрешима, а для детерминанта рекурсивных схем - это открытая проблема с надеждами на положительное решение.

Скордев (1976), развивая идеи Платека, 1966, и Московакиса, 1969а, изучает рекурсию в абстрактных функциональных пространствах, содержащих тождественную функцию, замкнутых относительно композиции и с отношением частичного порядка: $f \leq g \Leftrightarrow$ график $f \subseteq$ график g. Функциональное пространство $\mathcal{F}$ называется комбинаторным итеративным пространством, если оно:

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

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

3) C замкнуто относительно пересчета пар;

4) на $\mathcal{F}$ задана операция определения функции разбором случаев;

5) на $\mathcal{F}$ задана операция взятия неподвижной точки уравнения $\phi =$ если x то I иначе $\phi\phi$, где I - тождественная функция.

Если Φ - конечный набор элементов из $\mathcal{F}$, то любой элемент из $\mathcal{F}$, который получается из $\Phi \cup$ {тождественная функция, взятие элементов пары, истина, ложь} с помощью применения конечного числа операций, заданных на $\mathcal{F}$, называется функцией, рекурсивной в Φ .

Скордев показывает, что так определенные рекурсивные функции содержат (в естественном изоморфизме) все обычные рекурсивные функции, удовлетворяют аналогу теорем о нормальной форме, I-й и 2-й теоремам рекурсии и теореме нумерации. Рекурсивность некоторой конкретизации комбинаторных итеративных пространств совпадает с определением абсолютной простой рекурсивности по Московакису, I969а.

Сабельфельд, I976, построил для граф-программ с памятью полную систему преобразований, сохраняющую лт-эквивалентность, введенную Иткиным, I972.

Фенстад, I978, ссылаясь на диссертацию Платека, I966, излагает подход к описанию рекурсивного замыкания $R_{\omega}(\Phi)$, где Φ - множество функций, как наименьшего множества, замкнутого относительно операторов композиции, взятия неподвижной точки, содержащего правило определения функции разбором случаев и содержащего "комбинаторы" $I(f)=f, K(f,g) = f$ и $S(f,g,h) = f(h)\ (g(h))$. Он, в частности, показывает, что $R_{\omega}(\Phi)$ представима как вычислительная теория по Московакису, I969в. Подобного рода рекурсивные замыкания могут быть описаны как неподвижные точки некоторого индуктивного оператора - одного из главных понятий теории индуктивных определений - недавно сложившегося раздела математической логики (см., например, Московакис, I973).

Шень, I980, развив наблюдение Успенского, I974, показал, что класс арифметических функций может быть частично-рекурсивным относительно некоторого множества тогда и только тогда, когда он удовлетворяет аксиомам Успенского.

Варианты дальнейшего продвижения

Надо сказать, что уже одно только сопоставление накопленного материала из теоретического программирования и абстрактной теории вычислимости произвело на автора очень большое впечатление. Можно только удивляться, насколько, как кажется,

логика и программирование подготовлены к тому, чтобы вступить в более тесное взаимодействие. Не претендуя на чрезмерные обобщения, скажем лишь в контексте нашего исследования, что программирование может дать абстрактной рекурсии цель, взяв у нее метод. Насколько произвольными в работах по абстрактной рекурсии выглядят иногда предпосылки исследования в части целей и исходных допущений, настолько случайными и недостаточно освоенными выглядят математические методы, используемые в работах по теоретическому программированию. Очень хотелось бы надеяться, что намечающееся сближение математики и вычислительной науки, столь явственно демонстрируемое на нашем симпозиуме, окажется более глубоким и постоянным, нежели несостоявшееся обручение логики и программирования в нашей классификации наук, которое было отмечено в середине 60-х годов формированием общей специальности "математическая логика и программирование", просуществовавшей, к сожалению, лишь до реформы ВАКа в 1971 г.*)

Естественно, прежде всего спросить, не содержится ли среди работ по абстрактной рекурсии уже готовых решений интересующей нас проблемы. Думается, что если речь идет об объеме понятия абстрактно-вычислимых функций, то он уже нащупан: в этом нас убеждают множества теорем эквивалентности (Московакис, 1969б; Лакомб, 1969; Грильо, 1974). Однако, каковы м и н и м а л ь н ы е требования к управлению выполнением алгоритма (чисто комбинаторная часть рекурсии) - этот вопрос представляется неясным до сих пор. Естественно, что мы имеем теоремы о нормальной форме, где оператор поиска включается только, так сказать, на последней стадии, а вся "основная часть" - это примитивно-рекурсивные и даже субрекурсивные вычисления. Однако, автору кажется, что здесь возможно дальнейшее продвижение. Требование, чтобы детерминант был перечислимым множеством (Лакомб, 1969; Фридман, 1969б) или чтобы кодирующая схема для индексов была бы вычислимой (Московакис, 1969в), представляется нам слишком слабым. В этом нас убеждают результаты о структуре схемных детерминантов в разных моделях вычислений: контекстно-свободный язык для рекурсивных

*) Речь идет о формальном перечне специальностей, к которым должны относиться кандидатские и докторские диссертации.

программ (неокончательный результат) - Розен, 1974; двухленточный автомат для граф-схем с памятью (неокончательный результат) - Буда и Иткин, 1974; конечный автомат (!) для схем Янова - Ратледж, 1964.

В теоретическом программировании имеется совершенно необыкновенная теорема о нормальной форме (Харел, 1980), которая показывает, что любая граф-программа с памятью может иметь вид только одного цикла типа while , телом которого является программа, задающая только прямые вычисления. Думается, что этот результат еще не получил полного осмысления.

Другим выражением имеющегося запаса в используемой структуре управления вычислениями является эквивалентность синтаксического и семантического определений вычислимости (Фрассе, 1959; Грильо, 1974). В основе этой эквивалентности лежат два факта: один общий - полнота исчисления предикатов и один частный - беря в качестве посылок утверждения об истинности базовых предикатов (диаграммы) по Лакомбу, 1969, и Московакису, 1969б (что требует конкретного знания базовой системы), мы в процессе вывода имитируем процесс вычислений и в результате того или иного способа разборки формулы-программы, выводим те или иные утверждения об истинности или ложности искомого предиката.

Попробуем выводить теперь не формальные утверждения об истинности, но протоколы, при этом не только инвариантные, т.е. выполнимые в любой модели, но и что-то сверх того. В этом случае вывод не только не будет требовать конкретного знания базовой системы, но и возможно, потребует гораздо более слабых аксиом и правил вывода, нежели стандартное исчисление предикатов. Найти самую слабую систему вывода протоколов, но порождающую все необходимые термы-программы прямых вычислений, - еще один подход к изучению структуры управления.

В полезности делать различие между синтаксическим и семантическим определением вычислимости убеждает также следующий результат Иткина и Звиногродского 1972. Детерминант граф-схемы с памятью называется точным, если существует интерпретация базовой системы, в которой все протоколы этого детерминанта будут значимы. Иткин и Звиногродский показали, что любое отношение эквивалентности граф-схем с памятью, основанное

на точном детерминанте, будет неразрешимым. В то же время для граф-схем с памятью имеется схемный (синтаксический) детерминант с разрешимой эквивалентностью (Иткин, 1972).

В исследованиях по абстрактной рекурсии можно усмотреть еще одно противоречие. Теория рекурсии, вся пронизанная разными "конечностями", не может быть построена без конкретной комбинаторной системы, реализующей эти "конечности". Вот почему обычная теория рекурсии так непобедимо пролезает практически в любую абстрактную теорию через нумерации Мальцева (1961) и Лакомба (1969) через сплинтеры Стронга (1968), через B*-множества Московакиса (1969а) или в чистом виде через перечислимость детерминантов Фридмана (1969-б). Частично авторы избавляются от конкретной комбинаторики, постулируя вычислимость универсальных функций высокого уровня: пересчет пар, s-m-n-функции, функции нумерации, схемы рекурсии. Все эти приемы имеют смысл и разъясняют многие сущности, но не все.

Один из подходов к абстрактной вычислимости, который не игнорирует комбинаторику, мог бы состоять в выработке общей аксиоматики конструктивных объектов. Какова общая природа конструктивных объектов - это вопрос, о котором думают многие, но мало кто пишет. Очень грубо можно выделить два подхода: один, "индуктивный", обобщающий аксиоматику Пеано, второй, "теоретико-множественный", трактующий конструктивный объект как конечное множество с заданным на нем отношением. Для первого даже есть аксиоматика: см. обобщенную арифметику у Клини, 1952, § 50 и добавление I переводчика к русскому изданию (Клини, 1957). Второй связан с понятием алгоритма по Колмогорову (1953) и Колмогорову и Успенскому (1958).

Какова бы ни была еще не найденная общая аксиоматика конструктивных объектов, она в любом случае будет задавать для конструктивного множества некоторую в н у т р е н н е е м у п р и с у щ у ю систему операций и отношений, над которой любым образом можно определить вычислимые функции, например, в формализме Маккарти (1961) т.е. определяемые как неподвижные точки рекурсивных уравнений какого-нибудь стандартного вида.

Построенная таким образом теория, однако, заведомо не будет решать все проблемы абстрактной теории вычислимости. В математике и вычислительной науке созревает такое понятие как конструктивное, вычислимое, рекурсивное замыкание произвольных, т.е. не только конструктивных функций и множеств. Это понятие обобщает практику вычислений в общей математике, а также понятие вычисления с оракулом и относительной вычислимости в логике. Один из возможных подходов - выделение в произвольном носителе D конструктивного подмножества (чего-то вроде ε-сети) и построения над ним теории рекурсии в духе предыдущего параграфа. Намек на такой подход можно найти, например, у Энгелера [1973].

Другой подход к введению конкретной комбинаторики в абстрактную рекурсию состоит в следующем. Пусть $\langle D, \Phi, \Pi \rangle$ - базовая алгебраическая система. Тогда в теорию каким-то образом вводится исчисление (порождающая, определительная или вычислительная системы), в котором Φ и Π используются как элементарные символы (алфавит), составляющие конструктивные объекта исчисления. Это исчисление используется как комбинаторное средство для построения теории.

Опять-таки, можно наметить два варианта к построению такой теории. Один из них - это построение стандартной интерпретации в духе Эрбрана, в которой D предстает в виде некоторого множества $T(\Phi,\Pi)$ термов в алфавите $\Phi \cup \Pi$, а операция Φ и Π задают правила конструирования и анализа термов из $T(\Phi,\Pi)$. Конкретные свойства стандартной интерпретации будут выражать абстрактные свойства вычислимости по отношению к любым интерпретациям базовой системы. Определенное подтверждение этому подходу можно найти в некоторых работах по теоретическому программированию. Неясным, по крайней мере для автора, вопросом является то, как естественно задать стандартную интерпретацию предикатов.

Второй подход можно было бы назвать грамматическим. Его можно продемонстрировать на примере того, как, скажем, пытаться осуществить отражение конечной и поддающейся анализу информации о вычислимой функции на объекты, выраженные в алфавите базовой сигнатуры. Поскольку мы характеризуем вычислимую функцию детерминантом, т.е. некоторым множеством слов в

алфавите Ф ∪ П, то можно считать, что функция задается грамматикой своего детерминанта. Пусть все грамматики подобного рода принадлежат некоторому классу $\mathcal{G}$. Если мы преуспеем в том, чтобы некоторым естественным образом закодировать грамматику любого детерминанта тоже в алфавите Ф ∪ П, то тогда можно говорить о некоторой "универсальной грамматике" U, которая допускает множество слов (над алфавитом Ф ∪ П) вида $g * s$, такого и только такого, что g является кодом грамматики некоторого детерминанта, а s - строка, воспринимаемая этой грамматикой. Если нам удастся показать, что U может относиться к классу $\mathcal{G}$, то необходимое отражение программ в предметную область будет обеспечено. Проблемы универсальных грамматик хорошо известны в теории формальных языков, однако есть определенное разнообразие в постановке этих проблем, которое для одних и тех же классов приводит либо к разрешимым, либо к неразрешимым, либо к открытым проблемам, и еще предстоит исследовать, какие из этих постановок наиболее адекватны нашим целям (Касаи, 1975; Розенберг, 1977).

Хочется особенно внимательно исследовать логический, или исчислительный подход к построению абстрактной вычислимости, основанный на понятии индуктивной и инвариантной определимости (Фрассе, 1969; Крайсел, 1963; Грильо, 1974). Сейчас очень интенсивно изучается вопрос систематического извлечения программы из конструктивного доказательства теоремы о существовании решения задачи (см., например, Крайсел, 1975; Непейвода, 1982). Уже одна постановка этой проблемы делает формулы исчисления предикатов и программы, так сказать, однопорядковыми объектами. Далее, обнаруживается, что извлечение э ф ф е к т и в н о й программы из теоремы существования требует добавления к исчислению некоторых нетривиальных посылок, составляющих специальную теорию задачи. Наконец, сейчас оказывается, что многие виды обработки программ, встречающиеся на практике, адекватно описываются некоторыми базовыми трансформациями программного текста, которые напоминают логический вывод в исчислении формул, по Лакомбу (1969) и Грильо (1974).

Одной из важных манипуляций является адаптация программы к специфической информации, заданной в виде конкретных значений ее аргументов. Принципиальная возможность такой адаптации,

устанавливаемая s-m-n-теоремой, получает очень серьезное развитие в теоретическом и системном программировании (Ершов, 1977, 1980). Естественно, однако, что специфическая информация об аргументе может не обязательно иметь вид x=a, но быть заданной некоторым предикатом D(x)и, как таковая, быть использованной не только в программе, но и в некотором рассуждении о программе или вычисляемой ею функции. Совсем недавно Гоад, 1980, выполнил исследование, как адаптировать конструктивные доказательства к дополнительной информации о задаче или о ее данных. Логика и программирование поистине стремятся в объятия друг к другу!

Автор признателен А.А.Летичевскому, Ю.И.Манину, В.А.Непомнящему и В.А.Успенскому за стимулирующие дискуссии, отзвуки которых они найдут в этой работе; С.Дворникову, обратившего внимание на обобщенную теорию рекурсии; Дж.Крайселу, любезно приславшему автору очень полезные работы, в особенности Крайсел, 1969 и Гоад, 1980; Д.Скордеву за конструктивные замечания.

Литература

Буда и Иткин
[1974] Буда А.О., Иткин В.Э. Сводимость эквивалентности схем программ к термальной эквивалентности. - В кн.: Тр. 3 Всесоюз. симпоз. "Системное и теоретическое программирование". Кишинев, КГУ, 1974, т.1, с. 293-324.

Глушков
[1965] Глушков В.М. Теория автоматов и вопросы проектирования структур вычислительных машин. - Кибернетика, 1965, № 1, с. 3-11.

1979 Глушков В.М. Теорема о неполноте формальных теорий с позиций программиста. - Кибернетика, 1979, № 2, с. 1-5.

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

1977 Ершов А.П. О сущности программирования. - Программирование, 1977, № 5, с. 21-39.

1980 Ершов А.П. Смешанные вычисления: потенциальные применения и проблемы исследования. - В кн.: Всесоюз. конф. "Методы матем. логики в проблемах иск. интелл. и сист. программирование". Паланга, 1980 г., ч. 2. Вильнюс, Ин-т матем. и кибер. АН Лат.ССР, 1980, с. 26-55.

Иткин

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

Клини

1952 1957 Клини С.К. Введение в метаматематику. - М.: ИЛ, 1957, - 526 с.

Колмогоров

1953 Колмогоров А.Н. О понятии алгоритма. - УМН, 1953, т. 8, вып. 4, с. 175-176.

Колмогоров и Успенский

1958 Колмогоров А.Н., Успенский В.А. К понятию алгоритма. - УМН, 1958, т. 13, вып. 4, с. 3-28.

Криницкий

1959 Криницкий Н.А. Равносильные преобразования логических схем. Автореф. дис. на соиск. учен. степени канд. физ.-мат. наук. - М., 1959. - МГУ.

1970 Криницкий Н.А. Равносильные преобразования алгоритмов и программирование. - М., Советское Радио, 1970.

Мальцев

1961 Мальцев А.И. Конструктивные алгебры. - УМН, 1961, т. 16, вып. 3, с. 3-60.

Непейвода

1982 Непейвода Н.Н. Логический подход к программированию (в этой книге с. 32-63).

Непомнящий В.А.

1974 Непомнящий В.А. Критерии алгоритмической полноты систем операций. - В кн.: Теория программирования. Новосибирск, ВЦ СО АН СССР, 1972, т. 1, с. 267-279.

Патерсон и Хьюитт

1970 Патерсон М.С., Хьюитт К.Е. Сравнительная схематология. - В кн.: Кибернетический сборник. М.: Мир, 1976, вып. 13 (новая серия), с. 62-72.

Сабельфельд

1976 Сабельфельд В.К. Эквивалентные преобразования стандартных схем. - В кн.: Проблемы программирования. Новосибирск: ВЦ СО АН СССР, 1976, с. 94-121.

Успенский

1974 Успенский В.А. Теорема Гёделя о неполноте в элементарном изложении. - УМН, 1974, т. 29, вып. 1, с. 3-47.

Шень

1980 Шень А. Аксиоматический подход к теории алгоритмов и относительная вычислимость. - Вест. Моск. ун-та. Сер. 1.Математика. Механика, 1980, № 2, с. 27-29.

Янов

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

Вагнер

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

1969 Wagner E.G. Uniformly reflexive structures: an axiomatic approach to computability. Information Sciences, 1969, v.1, no.4, p. 343-362.

Гоад

1980 Goad C.A. Proofs as descriptions of computation. - Preprint, Stanford, Stanford University, 1980, 14 p.

Грильо

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

Ершов

1977 Ershov A.P. On the essense of compilation. -In: E.J.Neuhold (Ed). Formal Description of Programming Concepts. Amsterdam a.o., North-Holland, 1977, p.391-420.

1980 Ershov A.P. Mixed computation: potential applications and problems for study. - Theor. Comp. Sci., 1982, v.18, no.1, p. 41-67.

Иткин и Звиногродский

1972 Itkin V.E., Zwinogrodski Z. On program schemata equivalence. - J. Comp. Syst. Sci., 1972, v.6, no.1, p. 88-101.

Касаи

1975 Kasai T. A universal context-free grammar. - Information and Control, 1975, v.28, p. 30-34.

Клини

1952 Kleene S.C. Introduction to metamathematics. - New York, Van Nostrand, 1952.

Крайсел

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

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

1975 Kreisel G. Some uses of proof theory for finding computer programs. -In: Colloque International de Logique, Clermont-Ferrand (July 1975). Colloques Internationaux du CNRS, 1975, no.249, p. 123-124.

Лакомб

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

Маккарти

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

Московакис
1969а Moschovakis Y.N. Abstract first order computability. I. - Trans. Amer. Math. Soc., 1969, v.138, p.427-464.
1969б Moschovakis Y.N. Abstract computability and invariant definability. - J. Symb. Logic, 1969, v.34, no.4, p. 605-633.
1969в Moschovakis Y.N. Axioms for computation theories. In: Gandy R.O., Yates C.M.E. (Eds). Logic Colloquium'69, Amsterdam, North-Holland, 1971, p.199-256.
1973 Moschovakis Y.N. Elementary induction on abstract structures. - Amsterdam, North-Holland, 1973.
Непомнящий
1974 Nepomniaschy V.A. Criteria for the algorithmic completeness of the system of operations. -In: Ershov A.P. et al. (Eds). International Symposium in Theoretical Programming. Lect. Notes Comp. Sci., v.5, Berlin a.o., Springer Verlag, 1974, p. 172-186.
Патерсон
1968 Paterson M.S. Program schemata. -In: Machine Intelligence, v.3, Edinburgh, Edinburgh Univ. Press,1968, p. 19-31.
Платек
1966 Platek R.A. Foundation of recursion theory. Dissertation. Stanford University, January 1966, 215 p.
Патерсон и Хьюитт
1970 Paterson M.S., Hewitt C.E. Comparative schematology. -In: Records of Project MAC Conf. on Concur. Syst. and Parall. Comp., ACM, 1970, p. 119-128.
Ратледж
1964 Rutledge J.D. On Ianov's program schemata. - J.ACM, 1964, v.11, no.1, p. 1-9.
Розен
1964 Rosen B.K. Program equivalence and context-free grammars. - J. Comp. Syst. Sci., 1975, v.11, p. 358-374.
Розенберг
1977 Rosenberg G. A note on universal grammars. - Information and Control, 1977, v.34, p. 172-175.

Сабельфельд
1976 Sabelfeld V.K. Äquivalente Transformationen für Flussdiagramme. - Acta Informatica, 1978, v.10, Fasc.2, S. 127-156.

Скордев
1976 Skordev D. Recursion theory on iterative combinatory spaces. - Bull. Acad. Polon. Sci., Ser. Math. Astr. Phys., 1976, v.24, no.1, p. 23-31.

Стронг
1968 Strong H.R. Algebraically generalized recursive function theory. - IBM J. Res. Devel., 1968, v.12, no.6, p. 465-475.

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

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

Фрассе
1959 Fraissé R. Une notion de récursivitè relative. -In: Infinitistic Methods. Proceedings of the Symposium on Foundations of Mathematics, Warsaw 1959. Oxford a.o., Pergamon Press, 1961, p. 323-328.

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

1969 Friedman H. Algorithmic procedures, generalised Turing algorithms and elementary recursion theories. -In: Gandy R.O., Yates C.M.E. (Eds). Logic Colloquium'69. Amsterdam, North-Holland, 1971, p.361-390.

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

Энгелер

1970 Engeler E. On the structure of algorithmic problems. -In: Böhling K.-H., Indermark K. (Eds). 1. Fachtagung über Automatentheorie unf Formale Sprachen. Lect. Not. Comp. Sci., v.2, Berlin u.a., Springer Verlag, 1973, p. 2-15.