ПРИКЛАДНЫЕ АСПЕКТЫ МАТЕМАТИЧЕСКОЙ ЛОГИКИ
(Вычислительные системы)
1987 год Выпуск 122

510.67+519.68:510

ЛОГИЧЕСКИЕ АСПЕКТЫ ТЕОРИИ АБСТРАКТНЫХ ТИПОВ ДАННЫХ

Н.Х.Касымов, А.С.Морозов

Введение

В настоящем обзоре авторы попытались отразить математическую, точнее логико-математическую, сторону исследований по одному из важнейших направлений современного теоретического программирования - теории абстрактных типов данных (АТД). Сознательно сузив предмет обзора, авторы не стремились освещать технические, программистские, методологические и другие аспекты этой теории. В стороне остались также вопросы, связанные с параметризацией абстрактных типов данных, интерпретацией одного типа другим, пересечением типов данных, представлением абстрактных типов данных в виде решеток. Конкретно, задача настоящего обзора чисто методическая: показать математику, применяемую в теории абстрактных типов данных, и показать математику, которая могла бы оказать помощь в развитии этой теории.

Введение обзора написано авторами совместно; §1 принадлежит Н.Х.Касымову, остальные параграфы написаны А.С.Морозовым.

Всякий алгоритм работает над некоторым программистским типом (областью) данных (ПТД) - это могут быть числа, буквы, строки, матрицы, конечные отображения и т.д. Создавая тот или иной алгоритм, нам удобно вести рассуждения о нем в терминах некоторого ПТД. В этих же терминах мы записываем этот алгоритм для передачи программисту. Так, например, в матричном виде мы записываем итерационный процесс для решения системы линейных уравнений. Для его программирования, например, на ФОРТРАНЕ нам необходимо описывать в программе двумерные массивы и затем программировать вычисление мат-

риц, расписывая этот процесс покоординатно. С гораздо большими трудностями встречается программист, работающий на этом языке со строками.

Одна из идей облегчения его труда состоит в том, чтобы в языке программирования предусмотреть возможность записывать программу в тех же терминах, т.е. того же ПТД, в которых был записан исходный алгоритм. Для этого необходимо первоначально описать ПТД, в котором функционирует алгоритм так, чтобы этого описания было достаточно для выполнения последующей программы.

Что же такое программистский тип данных? Прежде всего это набор (множество) значений. Эти значения мы можем записывать в некотором виде. Для нас здесь важно то, что каждый элемент нашей области данных может быть записан в некотором виде и однозначно определен этой записью. Как правило, не запрещается, чтобы две различные записи обозначали один и тот же элемент ПТД. Однако тип данных - это не только множество значений [2,10,31-35,52-57]. Существенной чертой его является наличие в нем специфических операций и предикатов (которые можно заменить на операции, добавив к типу данных логические значения "истина" и "ложь"). Разумеется, эти операции должны быть алгоритмически вычислимы.

В [37-39, 44,45,58] предлагается считать, что тип данных полностью характеризуется системой операций, однако такой подход игнорирует информацию о том, как эти операции должны действовать.

Таким образом, для понимания ПТД существенны как его объекты, так и операции над ними. Стало быть, программистский тип данных можно рассматривать как структуру данных (СД). Определение структуры данных мы дадим, следуя [24,44,58].

Сигнатура Σ есть множество имен для сортов и имен для операций, причем каждой операции поставлен в соответствие тип операции вида: $(s_1, \ldots, s_n \to s)$, где $s_1, \ldots, s_n$, s - символы сортов. Обозначим $\Sigma = (S, F)$, где S - сорта, F - операции. Структура данных $\mathcal{A}$ сигнатуры Σ есть множество $A = \bigcup_{s \in S} A_s$ и интерпретация γ такая, что если $f \in F$ и имеет тип $(s_1, \ldots, s_n \to s)$, то $\gamma(f): A_{s_1} \times \ldots \times A_{s_n} \to A_s$.

Как правило, считается [44], что множества различных сортов не пересекаются. Некоторые авторы включают в определение СД требование конечной порожденности ее значениями сигнатурных констант, т.е. нульместных операций [19], другие - конечность сиг -

натуры. Это связано с тем, что таковы, как правило, СД , возникающие на практике [II] (каждый элемент можно как-то "назвать" в виде терма). Отметим, что случай пустых сортов не исключается [49]. Разумеется, если f - нульместная операция сорта s, то этот сорт пустым быть не может. Взгляд на СД как на многосортные алгебры позволяет привлекать для их изучения мощный аппарат теории универсальных алгебр [I3,23,24,36,58].

Теперь, когда выяснены основные черты ПТД, возникает вопрос, как объяснить машине, какой тип данных мы имеем в виду при написании программы.

Следует отметить, что пытаться искать общий метод описания СД с точностью до изоморфизма - дело бесполезное (об этом будет написано ниже). Поэтому нужно описывать только то, что понадобится нам для вычислений в СД.

Мы будем подразумевать под абстрактным типом данных (АТД) некоторое конечно задаваемое описание структуры данных. (Дикуссию об этом см,например,в [3,7,20-22,29,37-39,44,57].) Не следует путать понятия АТД и СД. СД - это конкретная алгебраическая система, а АТД - это некоторое абстрактное описание, которому, вообще говоря, могут удовлетворять и неизоморфные системы (СД). Наша СД дана машине именно в виде такого описания и ничего другого, т.е. типом данных для машины является как раз это описание. В этом и состоит абстрактность типа данных.

Приведем теперь основные требования, которым должен удовлетворять язык спецификаций СД.

I. Поскольку адекватность описания описываемому, возможность эффективной реализации этого описания и построения некоторых "канонических" структур должны гарантироваться структурой самого описания, то этот язык должен иметь корректно определенный синтаксис и средства генерирования новых выражений из заданных, т.е. он должен быть логическим.

2. Ясно, что метод должен обладать достаточно широкой областью применимости.

3. Требование гибкости заключается в том, что незначительное изменение специфицируемого понятия не должно приводить к резкому усложнению спецификации.

4. Требование легкости восприятия и относительной простоты составления спецификаций желательны, но не необходимы,так как,повидимому, сложные спецификации всегда будут составляться специалистами.

5. Язык должен располагать возможностью адекватного описания (это требование будет обсуждено ниже).

Все эти требования, а также некоторые их модификации широко обсуждаются, например, в [I,I0,I5,25,27-29,48,5I,55].

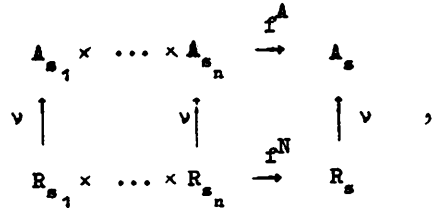
Перейдем теперь к вопросу об адекватности спецификации специфицируемому. Поскольку специфицируемыми объектами являются СД, то в идеале желательно, чтобы спецификация имела одну, и, с точностью до изоморфизма, только одну удовлетворяющую ей структуру. Однако для большинства хорошо изученных логических языков такое однозначное определение СД возможно далеко не всегда. Например,если хотя бы один сорт структуры бесконечен, то всякое множество предложений многосортной логики первого порядка, истинное в этой структуре, истинно и в некоторой другой, ей не изоморфной [5,I3]. Имеются и другие результаты, показывающие несостоятельность такого требования. Эту трудность можно обойти, считая представлениями АТД не все, а лишь некоторые "канонические" структуры, удовлетворяющие спецификации [54].

Еще более простая попытка решения этого вопроса состоит в том, чтобы отказаться от той, для наших целей, вообще говоря,излишней роскоши, какой является характеризация с точностью до изоморфизма, и потребовать, например, характеризацию с точностью до элементарной эквивалентности в общем языке первого порядка [2I, 53]. Однако здесь нас подстерегают трудности другого рода.Для многих СД множество всех истинных в них предложений не рекурсивно-перечислимо, а такую информацию трудно мыслить как спецификацию [44,49]. Если мы внимательно посмотрим на то, что же нам нужно отразить в спецификации СД, то заметим, что можно отказаться и от элементарной эквивалентности. Действительно, важным является то, чтобы для любого представления АТД какой-либо структурой результат вычисления в этой структуре (представляемый замкнутым термом) совпадал с результатом вычисления на соответствующих элементах в той структуре, для которой записан тип.

У вопроса об адекватности есть и другая, гораздо менее изученная сторона, касающаяся того, что интересующие нас СД должны быть в том или ином смысле эффективно представимыми [4,I9,26,40, 47]. Уточним это, следуя [3,I4,I9].

Пусть СД $\mathcal{A} = (A,\Sigma)$, где $A = \bigcup_{s\in S} A_s$; $\Sigma = (S,F)$. Обозначим $N = \{0,1,2,\ldots\}$ и $\nu: N \to A$, причем ν – наложение и $\nu^{-1}A_s = R_s$ – рекурсивное множество для каждого $s \in S$. Если для всякой операции

f^A СД $\mathcal{A}$ типа $(s_1,\dots,s_n \to s)$ найдется такая рекурсивная функция f^N, что будет коммутативной следующая диаграмма



то ν называется нумерацией $\mathcal{A}$.

Всякая СД со счетным носителем и рекурсивно-перечислимой сигнатурой имеет нумерацию [5].

Если ν – нумерация $\mathcal{A}$ и $\eta_\nu = \{(x,y) \mid \nu x = \nu y\}$ рекурсивно (рекурсивно-перечислимо), то ν называется конструктивизацией (позитивной нумерацией) $\mathcal{A}$, а пара $(\mathcal{A},\nu)$ – конструктивной (позитивной) СД. СД $\mathcal{A}$ называется конструктивизируемой (позитивно представимой), если существует конструктивизация (позитивное представление) $\mathcal{A}$. Мы будем отождествлять эффективную представимость с позитивной.

Интересно, что проблема характеризации с точностью до изоморфизма в логике первого порядка, может решаться положительно и для бесконечных СД, если ограничиться лишь эффективно представимыми. В [4] показано, что система аксиом арифметики Пеано имеет одну и, с точностью до эффективной представимости, только одну позитивную модель натуральных чисел с естественными интерпретациями символов для операций следования, сложения, умножения и нуля. Такой подход, при котором акцентируется внимание на изучение конструктивизируемых и, более общо, позитивно представимых СД, удовлетворяющих спецификациям, представляется нам особенно важным, и вот почему. Во-первых, "стандартные" модели таких общеупотребительных понятий, как стек, массив, очередь, размеченное объединение и т.п., являются конструктивизируемыми по своей природе. Во-вторых, спецификацию нужно уметь, хотя бы в принципе, эффективно реализовать, а для этого требуется наличие эффективно представимой модели, причем желательно в некотором смысле модели "канонической", строящейся из синтаксического материала, заключенного в самой спецификации, поскольку другого материала для такого построения нет. Все предложенные для спецификаций абстрактных типов данных языки (хотели того авторы или нет) были построены на основе существующих, хорошо

изученных логических языков,и проблема спецификации СД, таким образом, является логической проблемой.

Основная тема обзора - это способы и возможности описания алгебр, представляемые различными логическими языками, и способы построения структур данных по их описаниям.В обзоре мы будем двигаться от простых логических языков к сложным.

Есть две стороны правильности спецификации - это соответствие алгоритма этой спецификации и соответствие модели со спецификацией нашей реальной ситуации.

§I. Эквациональный и квазиэквациональный подходы

Инициаторами эквационального подхода являются авторы [I8, 29-35,37-39,57,59].Спецификацией в этом случае считается некоторое множество тождеств некоторой сигнатуры Σ,т.е. равенств вида $t=t'$, $t, t' \in T_\Sigma(X)$, где $T_\Sigma(X)$ - множество всех термов сигнатуры Σ с переменными из множества X. Основополагающей здесь является теорема Биркгофа, характеризующая многообразия [5,I3,23,24], т.е. классы всех алгебр, удовлетворяющих некоторой совокупности тождеств. Если $\mathcal{M}$ - многообразие алгебр, $A \in \mathcal{M}$ и X - такое порождающее множество для A, что всякое отображение X в любую алгебру из $\mathcal{M}$ продолжается до гомоморфизма A в эту алгебру,то A называется свободной в $\mathcal{M}$ алгеброй ранга $|X|$. Из теоремы Биркгофа следует,что в неединичном многообразии существуют свободные алгебры любого ранга. Структуры данных, интересующие нас, как правило, являются конечно-порожденными значениями сигнатурных констант [I0,44], поэтому представлениемАТД в этом случае естественно считать свободную структуру ранга o, т.е. порожденную значениями констант [I0, 20,35,4I,42,45].

При категорном подходе представлением считается инициальный объект в категории, объектами которой являются модели спецификации, а морфизмами - гомоморфизмы между ними [I0],но этот инициальный объект и есть свободная структура ранга o в соответствующем многообразии, так как для нее существует один и только один гомоморфизм в любую структуру из многообразия.

Заметим, что многие авторы, по существу, передоказывают теорему Биркгофа [23], которая хорошо известна на протяжении более пятидесяти лет.

Эквациональный подход в полной мере удовлетворяет требованию формальности спецификации и возможности эффективного построения стандартного представления по спецификации (если спецификация эффективно задана). Действительно, если SP - спецификация и $\mathcal{A}$ - ее стандартное представление, то

$$SP \vdash t_1 = t_2 \Leftrightarrow \mathcal{A} \models (t_1 = t_2) ,$$

где $t_1, t_2 \in T_\Sigma(\emptyset)$, здесь знак "$\vdash$" означает выводимость в обычной логике равенства, а "$\models$"-истинность предложения на модели.

Необходимо отметить, что в многосортном случае обычную логику равенства необходимо слегка модифицировать [34]. Это связано с тем, что мы допускаем те случаи, когда некоторые сорта пустые, и может случиться так, что равенство, выводимое из спецификации, оказывается ложным в некоторой структуре, удовлетворяющей SP: инициальной с некоторыми пустыми сортами [49].

Поскольку спецификация должна быть эффективно задаваемым объектом, то необходимым условием для нее должна быть ее рекурсивная перечислимость, и желательна, с практической точки зрения, ее конечность. В связи с этим возникает вопрос: насколько широк класс СД, обладающих конечными эквациональными спецификациями? В [50] предложен простой пример просматриваемого стека, для которого в [43,46,56] показана его неспецифицируемость конечным числом тождеств, т.е. $T_\Sigma(\emptyset)/\equiv_E \neq \mathcal{A}$, где Σ - сигнатура понятия "просматриваемый стек", E - произвольное конечное множество тождеств сигнатуры Σ, $\equiv_E$ - наименьшая конгруэнтность на $T_\Sigma(\emptyset)$, порожденная E, и $\mathcal{A}$ - "стандартный" просматриваемый стек. В математике примеры подобных структур, т.е. алгебр, не являющихся свободными, ранга o, подходящими конечнобазируемыми многообразиями, хорошо и давно известны [5,13]. Ситуация резко меняется, если допустить конечные спецификации в обогащениях. Если $\mathcal{A}$ - структура данных сигнатуры $\Sigma = (S,F)$ и $\mathcal{A}'$ - сигнатуры $\Sigma' = (S',F')$, причем $S \subset S'$ & & $F \subset F'$, то $\mathcal{A}'$ называется Σ'-обогащением $\mathcal{A}$, если Σ - обеднение $\mathcal{A}'$ изоморфно $\mathcal{A}$. В обозначениях $\mathcal{A}' \restriction \Sigma \cong \mathcal{A}$. В [19] показано, что всякая конструктивизируемая структура данных $\mathcal{A}$ сигнатуры Σ имеет Σ'-обогащение $\mathcal{A}'$, являющееся свободной структурой ранга o конечнобазируемого многообразия, причем $S' = S$, т.е. соответствующие обогащения можно задавать одними лишь операциями. Некоторые авторы такую специфицируемость в обогащениях не признают. Например, в [1] утверждается, что "использование посторонних

операций ... противоречит идее абстрактности спецификации, кото - рая предполагает, что спецификация должна содержать только суще - ственные для понятия операции". Нам кажется, что идея абстрактно- сти заключается не в том, что описание понятия должно даваться только в терминах (быть может, достаточно произвольных), обозна - чающих это понятие, а в том, что класс объектов определяется сво- им описанием (именно поэтому под АТД мы понимаем спецификацию объектов, а не сами объекты). Более того, неспецифицируемость по- нятия без скрытых операций показывает существенность этих скрытых операций для понятия.

Для позитивно представимых СД также существуют обогащения, за- даваемые конечным числом тождеств [I7], однако структуру исходной структуры в этом случае нужно обогащать не только операциями, но и сортами, т.е. по существу речь идет об обогащении расширения, кога $S' \setminus S = \emptyset$. В [9] построен пример конечно-порожденной позитив- но представимой алгебры конечной сигнатуры, у которой никакое ко- нечное обогащение (операциями и отношениями) не является свобод - ной системой ранга o, подходящего конечнобазируемого многообра - зия. Таким образом, проблема конечной специфицируемости тождест - вами в обогащениях без сортов для конечно-порожденных позитивно представимых структур данных решается отрицательно.

Введение языка квазитождеств (т.е. формул вида $t_1^0(\bar{x}) = t_1^1(\bar{x}) \& \ldots \& t_k^0(\bar{x}) = t_k^1(\bar{x}) \to s_0(\bar{x}) = s_1(\bar{x})$) вместо языка тождеств расширяет класс специфицируемых объектов в том смысле, что суще - ствуют структуры данных, которые задаются конечным числом квази - тождеств (в квазимногообразиях также существуют свободные системы любого допустимого ранга, хотя далеко не все их гомоморфные образы лежат в этом квазимногообразии [5, I3]) и не задаются тождествами (в необогащенной сигнатуре). Примеры таких структур опять-таки хо- рошо и давно известны в математике так же, как и примеры простых структур, не имеющих конечных квазиэквациональных спецификаций [I3]. Известно, что существует позитивно представимая алгебра ко- нечной сигнатуры, которая не обладает обогащением, являющимся сво- бодной системой ранга o конечнобазируемого квазимногообразия [8], но для конечно-порожденных алгебр вопрос остается открытым [I2, 44].

Приведем простой пример, демонстрирующий метод эквациональ - ных спецификаций. Пусть $\mathcal{O}\ell = (N;0,s,f)$, где $N = \{0,1,2,\ldots\}$,

$s(n) = n+1$ и $f(n) = n^2$. Несмотря на простоту, эта структура не специфицируется конечным числом тождеств в своей сигнатуре $\Sigma = (0,s,f)$ (см. [I7]). Поскольку сорт один, мы его явно не указываем.

Очевидно, $f(0) = 0$ и $f(x+1) = f(x) + 2x + 1$. Теперь если $\Sigma' = (0,s,f,d,h)$, причем $d(x) = 2x$, $h(x,y) = x + y$, то ясно, что

$$\begin{cases} d(0) = 0, ds(x) = ssd(x), \\ h(x,0) = x, h(x,s(y)) = sh(x,y), \\ f(x) = 0, fs(x) = sh(f(x),d(x)). \end{cases}$$

Обозначим эти шесть тождеств через E. Легко показать, что $T_{\Sigma'}(\emptyset)/\equiv_E \restriction \Sigma \cong \mathcal{A}$.

Более точный в смысле процедуры подход к использованию квазиэквациональных спецификаций заключается в следующем [I9]. Будем говорить, что структура данных $\mathcal{A}$ сигнатуры Σ хорошо специфицируется тождествами (квазитождествами) в обогащении Σ', если существует алгебраический предпорядок $\geq$ на множестве $T_{\Sigma'}(\emptyset)$, который обладает свойствами нётеровости и Черча-Россера и индуцируется конечным числом тождеств (квазитождеств) сигнатуры Σ' вида $t_1(\bar{x}) \geq t_2(\bar{x})$ $(t_1^1(\bar{x}) \geq t_2^1(\bar{x}) \,\&\, \ldots \,\&\, t_1^n(\bar{x}) \geq t_2^n(\bar{x}) \to t_1(\bar{x}) \geq t_2(\bar{x}))$ и такой, что $T_{\Sigma'}(\emptyset)/\equiv \restriction \Sigma \cong \mathcal{A}$, где $\equiv$ - конгруэнтное замыкание $\geq$.

Если SP - хорошая спецификация некоторого обогащения структуры $\mathcal{A}$, то всякий терм сигнатуры $\mathcal{A}$ редуцируется посредством SP в единственную нормальную форму (возможно, обогащенной сигнатуры). Известно следующее:

Структура $\mathcal{A}$ конечной сигнатуры конструктивизируема $\Leftrightarrow$ хорошо специфицируется тождествами (в обогащении одними лишь операциями) [I9].

Структура $\mathcal{A}$ конечной сигнатуры позитивно представима $\Leftrightarrow$ хорошо специфицируется квазитождествами (в обогащении сортами и операциями) [6].

Заметим, что никакая позитивно представимая неконструктивизируемая структура не обладает хорошей спецификацией в виде тождеств даже в обогащениях, допускающих не только новые операции, но и новые сорта, так как хорошая специфицируемость тождествами влечет конструктивизируемость.

Возможность добавления к квазиэквациональным спецификациям различных дополнительных конструкций (параметризованные спецификации, свободные спецификации, конструкция вынуждения данных и т.д.), рассматриваемая в [20,30,41,42,57,59], естественно, расширяет класс специфицируемых объектов, однако в этом случае многие спецификации не допускают эффективных реализаций [20,49].

§2. Описания в языке исчисления предикатов

Язык исчисления предикатов (все определения см. в [13]) представляет существенно большие возможности для описания моделей, чем язык тождеств и квазитождеств. Однако следует сразу заметить, что и этого языка в большинстве случаев недостаточно для описания моделей с точностью до изоморфизма. Это утверждение конкретизируется следующими классическими результатами.

Сначала дадим два определения.

ОПРЕДЕЛЕНИЕ 1. Теорией в языке первого порядка называется всякое непротиворечивое множество формул этого языка.

ОПРЕДЕЛЕНИЕ 2. Теория T называется тотально категоричной, если все ее модели изоморфны между собой, и ω-категоричной, если все ее счетные модели изоморфны между собой.

РЕЗУЛЬТАТ 1 [60]. Если теория T тотально категорична, то ее единственная с точностью до изоморфизма модель конечна.

РЕЗУЛЬТАТ 2 [60] (теорема Левенгейма-Скулема). Пусть T - счетная теория, которая имеет бесконечную модель. Тогда T имеет модель в любой бесконечной мощности.

Но ведь нас интересуют лишь счетные модели теории. Однако и здесь ситуация в целом не столь благополучна, как показывают следующие результаты.

ОПРЕДЕЛЕНИЕ 3. Формулы $\varphi(x)$ и $\phi(\bar{x})$ эквивалентны относительно теории T, если для любой модели $\mathcal{M}$ теории T $\mathcal{M} \models \forall \bar{x}(\varphi(\bar{x}) \leftrightarrow \phi(\bar{x}))$.

РЕЗУЛЬТАТ 3 [60] (теорема Рыль-Нардзевского.) Теория T ω-категорична тогда и только тогда, когда для любого $n \geq 0$, любого набора переменных $x_0, \ldots, x_n$ существует лишь конечное число попарно неэквивалентных относительно T формул, в которых все свободные переменные содержатся среди $x_0, \ldots, x_n$.

Предположим теперь, что нам необходимо описать некоторую бесконечную конечно-порожденную алгебру $\mathcal{A}$. Пусть $a_0, \ldots, a_k$ - порождающие этой алгебры. Существует бесконечная последовательность

термов $t_0(x_0,\dots,x_k)$, $t_1(x_0,\dots,x_k)$,... такая, что при $i \neq j$ $\mathcal{A} \models t_i(a_0,\dots,a_k) \neq t_j(a_0,\dots,a_k)$. Тогда формулы $x_{k+1} = t_i(x_0,\dots,x_k)$ попарно неэквивалентны в теории модели $\mathcal{A}$ при различных i. Таким образом, описать полностью нашу алгебру $\mathcal{A}$, даже взяв все предложения языка исчисления предикатов, истинные на $\mathcal{A}$, не удастся.

Мы опять вынуждены среди всех моделей теории T выделять те особые модели, про которые мы "рассказываем" с помощью T.

Первое, что приходит в голову, - это выделить "самую маленькую" так называемую простую, либо "самую большую" так называемую насыщенную модель (определения см. в [60]). Об этом пишет Ф.Нурани [54], учитывая, что не всегда теория имеет простую или насыщенную модель и что не существует общих методов построения таких моделей. Тем не менее для некоторых абстрактных типов данных этот подход применим. Так,в [7], например, приведены рекурсивно-аксиоматизируемые и даже разрешимые теории (т.е. множество выводимых в этой теории предложений рекурсивно) для СД "списки" и "стеки", для части которых стандартные модели являются простыми моделями.

Существует способ построения некоторой модели теории по самой этой теории - так называемый метод Хенкина [60]. Для того чтобы это построение осуществлять эффективно с помощью алгоритма, нужно потребовать от теории T ее разрешимость. Е.И.Латкин [62] получил обнадеживающие результаты о сложности построения таких моделей.

Тем не менее существует немало хороших, с точки зрения программирования, моделей, которые имеют неразрешимую теорию. Например, модель арифметики $\langle N; +, \cdot, s, <, 0 \rangle$ имеет алгоритмически очень сложную неразрешимую теорию [61].

Ввиду этого хорошо было бы иметь некоторые способы эффективного построения моделей для неразрешимых теорий. Естественно, при этом на теорию придется наложить некоторые ограничения, так как не всякая теория имеет эффективно конструктивизируемую модель.(Более точный смысл последнего утверждения будет сообщен ниже.) Один из способов эффективного построения дает теорема о теориях с конечными препятствиями [5], которую мы здесь приведем.

Пусть Φ - некоторая совокупность предложений конечной сигнатуры $\sigma = \langle P_0^{b_0},\dots,P_n^{b_n},c_0,\dots,c_k \rangle$, которая не содержит функциональных символов, находящихся в приведенной форме, а Φ^* - сово -

купность эрбрановых форм предложений из Φ. Рассмотрим сигнатуры $\sigma' = \sigma \cup \langle f_0^{n_0}, f_1^{n_1}, \dots \rangle$, где $f_i^{n_i}$ – введенные при образовании эрбрановых форм скулемовские функции, и $\sigma^* = \sigma' \cup \langle a_0, a_1, \dots \rangle$, где a_i – константы, не содержащиеся в σ'. Занумеруем все термы $t_0, t_1, \dots$ вида $f_i(a_{k_1}, \dots, a_{k_{n_i}})$ так, что если $t_k = f_i(a_{k_1}, \dots, a_{k_{n_i}})$, то $i, k_1, \dots, k_{n_i} \leq k$. Рассматривая термы t_i как константы, определим сигнатуры $\sigma_{m,n} \leftrightarrows \sigma \cup \langle a_0, \dots, a_{m-1} \rangle \cup \langle t_0, \dots, t_{n-1} \rangle$ для $m, n < \omega$. Также положим $\sigma_m \leftrightarrows \sigma_{m,m}$. Если $\mathcal{M}$ – конечная модель сигнатуры $\sigma_{m,n}$, где $n \leq m$, то через $\exists\Delta(\mathcal{M})$ будем обозначать формулу сигнатуры σ', определенную так: пусть $\mathcal{M}$ содержит ровно k элементов и $\nu: \{0, \dots, k\} \to |\mathcal{M}|$ – отображение "на". Элементарную формулу или ее отрицание для $\mathcal{A}$ сигнатуры σ' назовем $\mathcal{M}$-формулой, если выполнены условия:

1) все свободные переменные $\mathcal{A}$ принадлежат множеству $\{x_0, \dots, x_{k-1}\}$;

2) если терм t имеет вхождение в $\mathcal{A}$, то он имеет вид x_i либо $f_j(x_{i_1}, \dots, x_{i_{n_j}})$, причем существует терм t_l, где $l < n$, имеющий вид $f_j(a_{s_1}, \dots, a_{s_{n_j}})$ такой, что значения констант a_{s_t} в модели $\mathcal{M}$ совпадают с $\nu(i_t)$ при $1 \leq t \leq n_j$;

3) $\mathcal{A}$ истинна в $\mathcal{M}$, когда значениями x_i являются элементы $\nu(i)$, а значениями термов $f_j(x_{i_1}, \dots, x_{i_{n_j}})$ – значения "констант" $t_l = f_j(a_{s_1}, \dots, a_{s_{n_j}})$, где t_l соответствует терму $f_j(x_{i_1}, \dots, x_{i_{n_j}})$, как в п.2.

Пусть $\Delta(\mathcal{M})$ – конъюнкция всех $\mathcal{M}$-формул (их конечное число). Формулу $\exists x_0 \dots x_{k-1}\, \Delta(\mathcal{M})$ обозначим через $\exists\Delta(\mathcal{M})$.

ОПРЕДЕЛЕНИЕ 4. Модель $\mathcal{M}$ сигнатуры $\sigma_{n,n}$, $n \leq m$, называется препятствием, если $\Phi^* \cup \{\exists\Delta(\mathcal{M})\}$ противоречиво.

ОПРЕДЕЛЕНИЕ 5. Будем говорить, что Φ^* – с конечными препятствиями, если для любого $m \in \omega$ существует конечное число моделей $\mathcal{M}_0, \dots, \mathcal{M}_k$ сигнатуры σ_m таких, что любая конечная модель $\mathcal{M}'$ сигнатуры σ_m является препятствием тогда и только тогда, когда для некоторого $i \leq k$ модель $\mathcal{M}_i$ изоморфно вкладывается в $\mathcal{M}'$.

ТЕОРЕМА [5]. Если Φ рекурсивно-перечислима, а Φ^* – с конечными препятст-

виями, то для Φ существует конструктивная модель.

Еще одно ограничение на теорию, достаточное для существования эффективной модели (не всегда, правда, уже у этой теории), дает так называемая теорема о ядре, которая сейчас будет сформулирована.

Пусть Φ – некоторое множество формул сигнатуры σ, содержащих единственную свободную переменную, а $\mathcal{M}$ – модель сигнатуры σ. Элемент a из $\mathcal{M}$ назовем Φ-алгебраическим, если для некоторой формулы $\varphi(x) \in \Phi$ $\mathcal{M} \models \varphi(a)$ и множество $\{x | \mathcal{M} \models \varphi(x)\}$ конечно. Обозначим через $A_\Phi(\mathcal{M})$ множество всех Φ-алгебраических элементов модели $\mathcal{M}$. Пусть подмножество $\sigma' \subseteq \sigma$ таково, что $(x=c) \in \Phi$ для любого константного символа $c \in \sigma'$. Для модели $\mathcal{M}$ через $[\mathcal{M}]^{\sigma'}_\Phi$ обозначим σ'-подмодель модели $\mathcal{M}$ с основным множеством $A_\Phi(\mathcal{M})$ (разумеется, если оно не пусто).

Будем говорить, что теория T сигнатуры σ имеет (σ', Φ)-ядро, если существует такая модель $\mathcal{M}_0 \models T$, что для любой модели $\mathcal{M} \models T$ существует σ'-изоморфизм λ: $[\mathcal{M}_0]^{\sigma'}_\Phi \to [\mathcal{M}]^{\sigma'}_\Phi$ такой, что для любого $a \in A_\Phi(\mathcal{M}_0)$ и любой $\varphi \in \Phi$ имеет место $\mathcal{M}_0 \models \varphi(a) \leftrightarrow \mathcal{M} \models \varphi(\lambda a)$. Оказывается, если теория T имеет (σ', Φ)-ядро, то модель $[\mathcal{M}_0]^{\sigma'}_\Phi$, про которую говорилось выше, определена однозначно с точностью до изоморфизма, т.е., грубо говоря, не зависит от выбора $\mathcal{M}_0$. Эту модель мы назовем (σ', Φ)-ядром теории T и будем обозначать через $c(T)^{\sigma'}_\Phi$.

ТЕОРЕМА О ЯДРЕ [5]. *Если T – рекурсивно-перечислимая аксиоматизируемая теория сигнатуры σ и σ' – рекурсивно-перечислимое подмножество σ, Φ – рекурсивно-перечислимое множество формул и T имеет (σ', Φ)-ядро, то модель $c(T)^{\sigma'}_\Phi$ конструктивизируема.*

Еще одним достаточным условием существования конструктивной модели является

ТЕОРЕМА (Баур [63]). *Если T – рекурсивно-перечислимая $\forall\exists$-теория, имеющая модель с перечислимой $\exists$-теорией, то T имеет конструктивную модель.*

В начале нашего обзора, когда обсуждались требования, предъявляемые к спецификации СД, в которой будут производиться вычисления, отмечалось, что реализаций описания в виде алгебр может быть много (и так оно часто и есть, когда язык спецификаций строится на основе языка логики предикатов), но не это главное. Важно, чтобы результат вычисления не зависел от реализации спецификации.

Одна из конкретизаций этого условия состоит в требовании, чтобы у моделей теории была одинаково устроенная подмодель. Если брать исходные данные из этой подмодели, то в процессе вычисления мы никогда не выйдем за пределы этой подмодели, и, таким образом, производя вычисления фактически при всех реализациях в одной и той же модели, мы всегда придем к одному и тому же результату.

На этой идее основан подход, предлагаемый Орнеги, Маури, Бертони и Миглиоли [64,65], основанный на следующем логическом результате Крейсела:

ТЕОРЕМА [66]. Пусть $\mathcal{M}$ - такая модель теории T, которая изоморфно вкладывается в любую модель теории T и притом единственным образом. Тогда для любого $d \in |\mathcal{M}|$ существует бескванторная формула $\Delta(x,\bar{y})$ такая, что для любой модели $\mathcal{A}$ и изоморфного вложения $\varphi: \mathcal{M} \to \mathcal{A}$ справедливо $\mathcal{A} \vDash \exists \bar{y} \Delta(\varphi(d),\bar{y})$ и $\mathcal{A} \vDash \exists^1 x \exists \bar{y} \Delta(x,\bar{y})$ (здесь $\exists^1 x$ обозначает "существует единственный x такой, что").

ОПРЕДЕЛЕНИЕ 6. Модель $\mathcal{M}$ теории T назовем изоинициальной, если для любой модели $\mathcal{A}$ теории T существует единственное изоморфное вложение $\mathcal{M}$ и $\mathcal{A}$.

Оказывается, такие модели имеют весьма простое, с точки зрения теории моделей, строение.

Пусть T - теория, для которой существует бескванторная формула $\Delta(x,\bar{y})$ такая, что $T \vdash \exists^1 x \exists \bar{y} \Delta(x,\bar{y})$. Определим модель A_T этой теории следующим образом. Положим

$$A \leftrightarrows \{\Delta(x,\bar{y}) \mid T \vdash \exists^1 x \exists \bar{y} \Delta(x,\bar{y})\}.$$

Элементы A называются абстрактными данными.

Определим эквивалентность абстрактных данных:

$$\Delta_0(x,\bar{y}) \equiv \Delta_1(t,\bar{z}) \Leftrightarrow T \vdash$$

$$\vdash \exists x \exists t \exists \bar{y} \exists \bar{z}(\Delta_0(x,\bar{y}) \& \Delta_1(t,\bar{z}) \& x = t).$$

Положим $|A_T| \leftrightarrows A/_{\equiv}$. Пусть f - функциональный символ теории T. Определим

$$f^{A_T}(\Delta_0(x_0,\bar{y}_0)/\equiv,\ldots,\Delta_n(x_n,\bar{y}_n)/\equiv) = \Delta_{n+1}(x_{n+1},\bar{z})/\equiv \Leftrightarrow$$

$$\Leftrightarrow T \vdash \exists x_0 \ldots x_n \bar{y}_0 \ldots \bar{y}_n \bar{z}(\Delta_0(x_0,\bar{y}_0) \& \ldots \& \Delta_n(x_n,\bar{y}_n) \&$$

$$\& \Delta_{n+1}(x_{n+1},\bar{z}) \& \ldots \& f(x_1,\ldots,x_n) = x_{n+1})$$

и для предикатного символа P определим:

$$P^{A_T}(\Delta_0(x_0,\bar{y}_0)/\equiv,\ldots,\Delta_n(x_n,\bar{y}_n)/\equiv) \Leftrightarrow$$

$$\Leftrightarrow T \vdash \exists x_0 \ldots x_n \bar{y}_0 \ldots \bar{y}_n \Big(\&_{i=0}^{n} \Delta_i(x_i,\bar{y}_i) \& P(x_0,\ldots,x_n)\Big).$$

ОПРЕДЕЛЕНИЕ 7 [64]. Теория T допускает абстрактную структуру данных, если

1) существует бескванторная формула $\Delta(x,\bar{y})$ такая, что $T \vdash \exists^1 x \exists \bar{y} \Delta(x,\bar{y})$;

2) $A_T \models T$;

3) все отношения на A_T разрешимы в T, т.е. для любого n, любого предикатного символа P арности n и любых абстрактных данных $\Delta_0(x_0,\bar{y}_0),\ldots,\Delta_{n-1}(x_{n-1},\bar{y}_{n-1})$ в T доказуема одна из следующих формул:

$$\exists x_0 \ldots x_{n-1} \bar{y}_0 \ldots \bar{y}_{n-1} \Big(\&_{i=0}^{n-1} \Delta_i(x_i,\bar{y}_i) \& P(x_0,\ldots,x_{n-1})\Big),$$

$$\exists x_0 \ldots x_{n-1} \bar{y}_0 \ldots \bar{y}_{n-1} \Big(\&_{i=0}^{n-1} \Delta_i(x_i,\bar{y}_i) \& \neg P(x_0,\ldots,x_{n-1})\Big).$$

В [64] доказывается

ТЕОРЕМА. Следующие два условия эквивалентны:

1) T допускает абстрактную структуру данных;

2) Т и м е е т и з о и н и ц и а л ь н у ю м о д е л ь.

В любом из этих случаев A_T является изоинициальной моделью для Т.

В [64] показано, что в случае, когда Т рекурсивно-перечислимо аксиоматизируема и допускает абстрактную структуру данных, A_T является конструктивизируемой, причем ее конструктивизация строится естественным образом по аксиоматизации Т. Там же рассмотрены различные ослабления указанных выше условий.

В этих же терминах изучался вопрос о расширении абстрактных СД путем добавления новых операций [65].

Можно пытаться отказаться в определении изоинициальности от требования единственности изоморфного вложения. Мы получим класс так называемых алгебраически простых моделей. Для них теория оказывается сложнее (см. [67]), и авторам пока что неизвестны ее применения в теории абстрактных типов данных.

В связи с попытками описания СД в языке исчисления предикатов, особую важность приобретает проблема поиска логического вывода и накопления опыта построения быстрых разрешающих алгоритмов для разрешимых элементарных теорий.

§3. Способы описания алгебраических систем в других языках

Богатые возможности для описания алгебраических систем дают языки более высокого уровня, чем языки исчисления предикатов.

Здесь будут рассмотрены два языка алгоритмической логики, и на примерах продемонстрированы их возможности.

Один из вариантов алгоритмической логики допускает выражения вида $A\{P\}B$, где A и B - некоторые утверждения, а P - некоторая программа. Считается, что это выражение истинно на алгебраической системе, если программа P, запущенная на любых входных данных, удовлетворяющих A, заканчивает свою работу через конечное число шагов и после окончания работы программы P утверждение B истинно.

Сальвицкий [68] показал, как применять алгоритмическую логику для описания структур данных. Вот пример, как можно описать абстрактный тип данных "стеки" с помощью подобного языка. Сигнатура языка состоит из двух сортов: E (сорт элементов) и S (сорт стеков), а также выделенного из S элемента empty и операций:

top: S → A

pop: S → S

push: $S \times A \to S$

A1: true{ begin while $\neg$(empty $=_s$ s) do s:= pop(s) end, true,

A2: $\neg$(empty $=_s$ s) $\to$ s = push(top(s),pop(s)),

A3: e $=_E$ top(push(e,s)),

A4: s $=_s$ pop(push(e,s)),

A5: $\neg$((push(e,s)) $\equiv_s$ empty) ,

A6: s = s' $\leftrightarrow$ true{begin s_1= s; s_2= s'; bool = true;

while bool & $\neg$(empty $\equiv_s$ s_1) & $\neg$ (empty $\equiv_s$ s_2) do

begin bool:= bool & top(s_1) = top(s_2);

s_1 = pop(s_1); s_2 = pop(s_2);

end; end}(bool & (empty $\equiv_s$ s_1) & (empty $\equiv_s$ s_2)) .

Все СД, удовлетворяющие AI-A6, являются стеками в обычном смысле. (Мы опускаем здесь детали в определении значений top(empty) и pop(empty) .) Важным свойством языка этой логики является то, что при описании СД в ней нет необходимости привлекать те или иные категорные условия (изоинициальность той самой модели, которую мы описываем, либо ее изоинициальность, простоту и т.д.). Так, например, аксиома AI обеспечивает конечность любого стека, что уже дает однозначность понимания стека как конечной последовательности элементов E.

К сожалению, пока что не известно, существуют ли регулярные методы построения СД по подобным описаниям.

Не исключено, что поиск такого метода лежит на пути выделения подходящего довольно узкого и удобного класса описаний данного языка.

Другой любопытный язык, являющийся расширением обычной логики первого порядка, - это логика эффективных определений LED. Она изучается в работе Тюрина [69].

Язык логики LED строится следующим образом: возьмем какой-либо язык L конечной сигнатуры σ и обозначим у него через F(L,n) множество всех бескванторных формул I-го порядка, все переменые которых содержатся среди $\{x_0, \ldots, x_{n-1}\}$. Через T(L,n) обозначим

множество всех термов языка L от переменных $\{x_0,\dots,x_{n-1}\}$. Сначала введем понятие функционального эффективного определения (фэо). Функциональным эффективным выделением арности n называется произвольное вычислимое отображение: $S:\omega \to P(L,n)\times T(L,n)$. Введем обозначения $S_n = \langle \alpha_n(\bar{x}), t_n(\bar{x})\rangle$. Если $\mathcal{M}$ – модель сигнатуры σ, то всякое фэо S-арности k определяет некоторое частичное отображение $\hat{S}:|\mathcal{M}|^k \to |\mathcal{M}|$ следующим образом: чтобы вычислить $\hat{S}(\bar{a})$ для $\bar{a} \in |\mathcal{M}|^k$, нужно найти минимальное l такое, что $\mathcal{M} \models \alpha_l(\bar{a})$, и взять в качестве ответа $t_l(\bar{a})$.

Формулы LED определяются следующим образом:

1) если S_1 – и S_2 – фэо арности n, то $S_1 \doteq S_2$ есть формула LED (на элементах $\bar{a}$ модели $\mathcal{M}$ она интерпретируется как утверждение: $\hat{S}_1(\bar{a})$, $\hat{S}_2(\bar{a})$ определены и равны);

2) формулы, получаемые из LED-формул с помощью связок &, $\vee$, $\neg$ и навешивания кванторов $\forall x_i$ и $\exists x_i$, являются LED-формулами;

3) других LED-формул нет.

Несмотря на то, что формулы языка LED бесконечны, они являются эффективными объектами и могут быть описаны полностью заданием конечного количества информации.

Бергстрой и Тюрином [70] доказана следующая

ТЕОРЕМА. Для алгебры $\mathcal{A}$ сигнатуры σ содержащей константный символ, эквивалентны следующие условия:

1) $\mathcal{A}$ описывается с точностью до изоморфизма некоторым семейством формул вида $\forall \bar{x}(s \doteq s)$;

2) $\mathcal{A}$ описывается с точностью до изоморфизма некоторым семейством формул вида $\forall \bar{x}\varphi(\bar{x})$, где φ не содержит кванторов;

3) $\mathcal{A}$ не имеет собственных подалгебр.

В этой же работе они ставят вопрос (который на самом деле они впервые поставили в 1977 году) о том, верна ли эта теорема без ограничений на сигнатуру. Отрицательный ответ на этот вопрос получен в [71]. Тут следует отметить, что СД всегда содержат хоть одну константу в сигнатуре (ведь надо же как-то называть ее элементы по имени! Вот тут и нужны термы от констант).

Еще большие возможности для описания алгебраических систем представляют языки, допускающие бесконечно длинные выражения (см., например, [72]). Не исключено, что на практике возможно применение некоторых финитно задаваемых формул этого языка для описания СД.

§4. Абстрактные типы данных в семантическом программировании

В основе концепции семантического программирования лежит идея, по которой исполнению соответствует проверка истинности. В этой концепции отражен богатый опыт изучения понятия алгоритма, накопленный в математической логике: от различных понятий вычислимости с применением идей дескриптивной теории множеств до теории допустимых множеств. Главное ее отличие от последующих концепций состоит в наличии некоторого универсального типа данных - наследственно конечных списков, в которых может быть эффективно проинтерпретирована любая конструктивная или перечислимая СД. Но следует отметить, что теория абстрактных типов данных в семантическом программировании, как и само это программирование, пока существует на уровне идей и теоретических исследований.

С основными идеями и понятиями семантического программирования можно познакомиться в работах [73,74,75].

Программа в языке семантического программирования есть Σ-формула, которая сама является собственной спецификацией, причем эта спецификация сама может быть исполнена. Здесь мы лишь наметим основные идеи развития теории абстрактных типов данных в семантическом программировании.

Для задания конструктивной или перечислимой СД необходимо задать ее сорта и основные операции. Сорта могут быть заданы с помощью Σ-формул, основные операции также могут быть заданы Σ-формулами. Таким образом, абстрактный тип данных в семантическом программировании может быть задан обычными языковыми средствами. При этом элементы СД будут интерпретироваться как некоторые наследственно конечные списки (или их классы), операции СД будут вычислимыми операциями над этими списками. Тогда для вычислений отпадет нужда в задании сортов СД, и сам абстрактный тип данных будет выступать как набор процедур или как модуль, и пользователю (или машине) остается лишь проинтерпретировать полученный результат - наследственно конечный список - в удобных ему терминах. Ав-

торы также надеются, что с разработкой концепции абстрактных ти - пов данных в семантическом программировании упростится решение проблем, связанных с перечислением СД и сортов в СД, а также с интерпретацией одного типа данных в другом.

В связи с этим нельзя не отметить работу Ю.Л.Ершова [76], содержащую предпосылки семантического программирования, а также представление в Σ-языке динамической логики. Там же вводится наиболее общее и естественное понятие интерпретации одной СД в дру - гой.

Литература

I. АГАФОНОВ В.Н. Типы и абстракция данных в языках программирования //Данные в языках программирования. -М., Мир, 1982. -С. 256-327.

2. БРОСГОЛ Б.М. Дискуссионные вопросы из области типов данных и контроля типов //Там же. -С. 170-195.

3. ГОНЧАРОВ С.С. Матрицы как абстрактный тип данных //Мате - матическое обеспечение ВС из микро-ЭВМ.-Новосибирск.-1983.-Вып.96: Вычислительные системы. -С. 75-86.

4. Его же. Модели данных и языки их описания //Логико-математические основы проблемы МОЗ. - Новосибирск. - 1985. - Вып. 107: Вычислительные системы. - С. 52-70.

5. ЕРШОВ Ю.Л. Проблемы разрешимости и конструктивные модели -М.: Наука, 1980.

6. КАСЫМОВ Н.Х. Алгебраическое описание рекурсивно-перечислимых типов данных //Структурный анализ символьных последовательностей. -Новосибирск. - 1984. - Вып. 101: Вычислительные системы. - С. 130-140.

7. Его же. Вопросы разрешимости для некоторых абстрактных типов данных //Логические вопросы теории типов данных.-Новосибирск. - 1986. Вып. 114: Вычислительные системы. -С. 107-121.

8. КАСЫМОВ Н.Х., ХУСАИНОВ Б.М. Конечно-порожденные перечислимые и абсолютно локально-конечные алгебры //Прикладная логика.-Новосибирск. - 1986. -Вып. 116: Вычислительные системы.-С. 3-15.

9. КАСЫМОВ Н.Х. Об алгебрах с финитно-аппроксимируемыми позитивно представимыми обогащениями //Алгебра и логика,1987.-Т.26. - №6.

10. ЛЕМАН Д., СМИТ М. Типы данных //Данные в языках програм - мирования. М., Мир, 1982. - С.196-213.

11. ЛИСКОВ Б., ЗИЛЛЕС С. Методы спецификации,используемые для абстракции данных //Там же. -С.91-122.

12. Логическая тетрадь. - Новосибирск, 1986.

13. МАЛЬЦЕВ А.И. Алгебраические системы. - М.: Наука, 1970.

14. Его же. Конструктивные алгебры.I //Избранные труды. М., Наука, 1976. - Т.П. - С.134-185.

I5. ПАРНАС Д. Метод спецификации модулей програ много обес - печения //Данные в языках программирования. М., Мир, 982.-С.9-24.

I6. BERGSTRA J.A., MEYER J.-J.Ch. On specifing se s of integers//Afdelfing informatica,IW 237/83.

17. BERGSTRA J.A., TUCKER J.V. Algebraic specifications of computable and semicomputable data structures// Afdelfing infor - matica IW 115/79.

18. BERGSTRA J.A., TUCKER J.V. On the adequacy of finite equational metods for data type specification// SIGPLAN Notices,1979.- Vol.14,N 11.

19. BERGSTRA J.A., TUCKER J.V. A characterization of compu - table data types by means of a finite equational specification method// Proc.7th ICALP LNCS.-1980.-Vol.85.-P.76-90.

20. BERGSTRA J.A., BROY M., TUCKER J.V., WIRSING M. On the power of algebraic specifications// Proc.10th Int.Symp.Math.Foundation of comp.Sci.//LNCS.- 1981.- Vol.18.- P.193-204.

21. BERTONI A., MAURI G., MIGLIOLI P.A. A characterization of abstract data types as model-theoretic in variants// LNCS. - 1979.- Vol.71. - P.26-37.

22. BERTONI A., MAURI G., MIGLIOLI P.A. Towards a theory of abstract data types: a discussion of problems and tools// LNCS. - 1980.-Vol.83.-P.44-58.

23. BIRKHOFF G. On the structure of abstract algebras//Proc. Cambr.Phil.Soc.- 1935.- Vol.31.- P.433-454.

24. BIRKHOFF G., LIPSON D. Heterogeneous algebras// J.Combinatorial Theory.- 1970.-Vol.8,N 1.- P.115-133.

25. BRAND D. A note on data abstractions// SIGPLAN Not.-1978. -Vol.13,N 1.-P.21-24.

26. COMYN G., WERNER G. Computable data types// LNCS.- 1979. -Vol.74.- P.228-236.

27. CONSTABLE R.L. Programs and types// IEEE Symp.on Foundations of Comput.Sci.-1980.-P.118-128.

28. DONAHUE J.E. On the semantics of "data type"// SIAM J. Comp.-1979.-Vol.8,N 4.- P.546-560.

29. EHRIG H.-D. Extensions and implementations of abstract data type specifications// LNCS.- 1978.- Vol.64.- P.115-163.

30. EHRIG H.-D., FEY W., HANSEN. ACTONE: An algebraic spe - cification language with two levals of semantics// Institut für Software und Theoretische Informatik, Tech.Univ.Berlin, W.Germany. -1983.- Rep.83-03.

31. EHRIG H.-D., WAGNER E.G., THATCHER J.W. Algebraic constraints for specifications and canonical form results// Inst. für Software und Theoretische Informatik. Tech.Univ.Berlin, W.Germany. -1982.- Rep.82-09.

32. EHRIG H.-D., WAGNER E.G., THATCHER J.W. Algebraic specifications with generating constraints// Proc.10th Int.Collog.Automata, Lang., Programming// LNCS.- 1983.-Vol.54.-P.188-202.

33. GOGUEN J.A. Abstract errors for abstract data types// Formal description of programming concepts.North.-Holl., 1978. - P.491-523.

34. GOGUEN J.A., MESEGUER J. Completeness of many-sorted equational logic// SIGPLAN Not. - 1981.- Vol.16,N 7.-P.24-32.

35. GOGUEN J.A., THATCHER J.W., WAGNER E.G. An initial algebra approach to the specification, correctness and implementation of abstract data types// IBM Res.Rep.RC 6487, 1976; see also Current trends in Programming Methodology// Data Structuring. - 1978. - Vol.4. - P.80-149.

36. GRATZER G. Universal Algebra// 2nd ed.-New York: Sprin - ger,1979.

37. GUTTAG J.V. Abstact data types and the development of data structures// Comm.ACM.- 1977.- Vol.20,N 6.- P.396-404.

38. GUTTAG J.V., HORNING J.J. The algebraic specification of abstract data types// Acta Informatica.- 1978.- Vol.10, N 1.- P.27-52.

39. GUTTAG J.V., HOROWITZ E., MUSSER D.R. Abstract data types and software validation// Comm.ACM.- 1978.-Vol.21,N 12.- P.1048-1064.

40. HOARE C.A.R. Recursive data structures// Intern J.Comp. Inf.Sci.- 1975.- Vol.4, N 2. - P.105-132.

41. HUPBACH U.L. Abstract implementation of abstract data types// LNCS.- 1980.- Vol.88.- P.291-304.

42. HUPBACH U.L., KAPHENGST H., REICHEL H. Initial algebraic specification of data types, parametrized data types and algorithms// VEB Robotron, Zentrum für Forschung und Technik, Dresden, East Germany,1980.

43. JONES D.W. A note of some limits of the algebraic specification method// SIGPLAN Not.-1978.-Vol.13,N 4.-P.64-67.

44. KAMIN S. Some definitions for algebraic data type specifications// SIGPLAN Not.- 1979.- Vol.14, N 3.- P.28-37.

45. KANDA A. Data types as initial algebras: a unification of Scottery and ADJery//IEEE Symp.on Found. of Comp.Sci,1978. - P.221-230.

46. KAPUR D. Specifications of Majster's traversable stack and Veloso's traversable stack// SIGPLAN Not.-1979.- Vol.14, N 5. - P.46-53.

47. LEVIS C.H., ROSEN B.K. Recursively defined data types// ACM Symp.on Principles of Progr.Lang.,1973.- P.125-138.

48. LISKOV B., ZILLES S. Programming with abstract data types// SIGPLAN Not.-1974.- Vol.9,N 4.- P.50-59.

49. MACQUEEN D.B., SANNELLA D.T. Completeness of Proof Sys - tems for equational specifications// IEEE Trans on Software Engineering.- 1985.- Vol.SE-11,N 5.-P.454-461.

50. MAJSTER M. Limits of the "algebraic" specification of abstract data types// SIGPLAN Not.- 1977.-Vol.12,N 10.-P.37-42.

51. MAJSTER M. Data types, abstract data types and their specification problem// Theor.Comp.Sci.-1979,-Vol.8,N 1.-P.89-128.

52. MORRIS J. Types are not sets// ACM Symp.on Principles of Progr.Lang.,1973.-P.120-124.

53. NAKAJIMA R., HONDA M., NAKAHARA H. Hierarchical Program specifications and verification - a many-sorted logical approach // Acta Inf.-1980.-Vol.14.-P.135-155.

54. NOURANI F. A model-theoretic approach to specification, extension and implementation// LNCS.- 1980.- Vol.83.-P.282-297.

55. REYNOLDS J.C. Towards a theory of type structure// LNCS. - 1974.- Vol.19. - P.408-423.

56. SUBRAHMANYAM P. On a finite axiomatization of the data type//SIGPLAN Not.-1978.- Vol.13, N 4.-P.80-84.

57. THATCHER J.W., WAGNER E.G., WRIGHT J.B. Data type specification: parametrization and the power of specification techniques// ACM Trans.Prog.Lang.Syst.-1982.- Vol.4,N 4.- P.711-732.

58. ZILLES S.N. Introduction to data algebras// LNCS.-1980.- Vol.86.-P.248-272.

59. ZILLES S.N., LUCAS P., THATCHER J.W. A look at algebraic specifications// IBM Res.Rep.RJ 3568,1982.

60. КЕЙСЛЕР Г., ЧЭН Ч.Ч. Теория моделей. -М.: Мир, 1977.

61. РОДЖЕРС Х. Теория рекурсивных функций и эффективная вычислимость. - М.: Мир, 1976.

62. ЛАТКИН Е.И. Модели абстрактных данных с ограниченной алгоритмической сложностью //Всесоюзн. конф. по прикладной логике. - Новосибирск, 1985.

63. BAUR W. Über rekursive Strukturen// Inven.Math.- 1974. - Vol.23,N 2.- P.89-95.

64. BERTONI A., MAURI P. MIGLIOLY P. On the power of model theory in specifying abstract data types and in capturing their recursiveness//Fund.Inf.- 1983.-Vol.6,N 2.- P.127-170.

65. BERTONI A., MAURI G., MIGLIOLY P., ORNAGHI M. Abstract data types and their extensions within constructive logic// LNCS. -1984.-Vol.173.-P.173-195.

66. KREISEL G. Model-Theoretic invariants: applications to recursive and hyperarithmetic operations// Proc.Symp.on the th.of models.-Amsterdam,North.-Holl,1965.

67. BALDWIN J.T., KUEKER D.W. Algebraically prime models // Ann.Math.Log.- 1981.- Vol.20,N 3.

68. SALWICKI. On the algorithmic theory of stacks// Fund. Inf.- 1980.- Vol.3,N 3.- P. 311-332.

69. TIURYN J. Logic of effective definitions// Fund.Inf. - 1981.- Vol.4,N 3.- P.629-660.

70. BERGSTRA J., TIYRYN J. Implicit definability of algebraic structures by means of program properties// Fund.Inf.- 1981.- Vol.4,N 3.- P.661-674.

71. МОРОЗОВ А.С. Об одном вопросе Бергстры и Тыорина //Алгебра и логика. - 1986. - Т.25, №5. - С.566-583.

72. KEISLER H.J. Model theory of infinitary Languages.- Amsterdam, North.-Holl,1971.

73. ГОНЧАРОВ С.С., СВИРИДЕНКО Д.И. Σ-программирование //Логико-математические основы проблемы МОЗ. -Новосибирск, 1985.-Вып. 107: Вычислительные системы. - С. 3-29.

74. GONCHAROV S.S., ERSHOV Yu.L., SVIRIDENKO D.I. Semantic programming// Information Processing.- Dublin,Ireland, North.-Holl, 1986.- P.1093-1100.

75. ВОРОНКОВ А.А. Естественное исчисление для Σ-программ //Логические методы в программировании. - Новосибирск. - 1987. -Вып. 120: Вычислительные системы. -С. 14-23.

76. ЕРШОВ Ю.Л. Динамические логики над допустимыми множествами //Докл. АН СССР. - 1983. -Т. 273, №5. - С. 1045-1048.

Поступила в ред.-изд.отд.
13 июля 1987 года