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

УДК 519.681.2

УНИВЕРСАЛЬНЫЕ КЛАССЫ ПАРАЛЛЕЛЬНЫХ И НЕДЕТЕРМИНИРОВАННЫХ СХЕМ ПРОГРАММ

В.Д. Соловьев

Введение

В многочисленных работах по параллельным программам рассматривались различные средства параллелизма. Обилие и сложность этих средств являются препятствием к построению более или менее завершенной теории параллельных схем программ, аналогичной классиче - ской схематологии - теории последовательных схем программ. В частности, оставался неизученным вопрос, какие из средств параллелизма являются наиболее сильными, в то время как для последовательных схем программ, как известно [I,2,II], - это класс схем программ с массивами и равенством $\varphi(A,=)$, являющийся универсальным.

В настоящей работе вводится класс параллельных схем программ и обосновывается тезис его универсальности в смысле функциональной эквивалентности его в классах детерминированных параллельных схем программ. Отдельно изучается случай недетерминированных схем программ.

Результаты исследования параллельных схем программ применяются и для решения проблемы построения абстрактной теории алгоритмов. Один из путей ее решения состоит в том, чтобы вычислительной теорией над произвольной алгебраической системой $\mathcal{A} = \langle A, L \rangle$ считать замыкание L относительно некоторого класса схем программ [II]. Использование в качестве такового введенного в данной работе класса схем программ позволяет более естественным образом оперировать с частичными функциями и предикатами из L .

Доказан изоморфизм частично упорядоченного по включению множества вычислительных теорий, содержащих все частично рекурсивные функции (ч.р.ф.) со структурой e-степеней.

Все необходимые для понимания статьи определения, отсутствующие в самой статье, можно найти в [I]. Стиль доказательств - не - формальный. Детали программистских конструкций предоставляется восстанавливать читателям.

§I. Схемы с параллельными процедурами. Функциональные возможности

Введем класс параллельных детерминированных схем программ - схемы с параллельными процедурами (обозначим его $\varphi(A,DPP,=)$). Он получается добавлением нового оператора - (детерминированного) параллельного вызова процедур - к средствам класса $\varphi(A,=)$.

Оператор параллельного вызова процедур имеет следующий синтаксис: $F_1(x_1^1,\dots,x_{k_1}^1)\|F_2(x_1^2,\dots,x_{k_2}^2)\|\dots\|F_1(x_1^1,\dots,x_{k_1}^1)$. Здесь $F_1,\dots,F_1$ - имена процедур, x_i^j - параметры. Схема класса $\varphi(A,DPP,=)$ представляет собой главную схему и множество схем процедур, использующих средства класса $\varphi(A,=)$ и оператор параллельного вызова. Памяти всех схем процедур и главной схемы не пересекаются. Оператор вывода находится в главной схеме.

Выполнение интерпретированных детерминированных схем с параллельными процедурами почти аналогично выполнению интерпретированных схем с обычными процедурами [I], но со следующими отличиями:

I. При выполнении оператора параллельного вызова процедур происходит параллельный запуск процедур с именами, указанными в операторе.

2. Управление возвращается на следующий оператор после оператора параллельного вызова, если хотя бы одна из вызванных параллельно работающих процедур завершит свою работу. Остальные из вызванных этим оператором процедур в этом случае прекращают свою работу.

3. Вызванные процедуры не передают данные в вызвавшую их процедуру и не меняют ее состояние памяти.

Введенный класс схем программ характеризуется следующим существенным недостатком: поскольку никакая информация (кроме сообщения об останове) из вызванных процедур не передается в вызвавшую, то последней в случае возврата управления бывает неизвестно, какая из вызывавшихся параллельно работающих процедур завершила свою работу. Чтобы избавиться от этого недостатка, разрешим пере-

дачу в вызывавшую процедуру следующей информации - сообщение об останове вместе с номером закончившей свою работу процедуры. Однако и это приводит к недетерминизму, так как в случае, если останавливаются несколько из параллельно работающих процедур, то сообщение об останове вместе со своим номером может прислать любая из них.

Таким образом, мы приходим к введению класса недетерминированных схем программ с параллельными процедурами (обозначим его $\varphi(A,NPP,=)$).

Его определение такое же, как и в детерминированном случае, за исключением п.3. Теперь его сформулируем следующим образом: вызванные процедуры не передают данные в вызвавшую их процедуру и не меняют ее состояния памяти за исключением одного счетчика процедур, которому присваивается (недетерминированно) значение, равное номеру одной из оставшихся процедур (в заранее определенной нумерации процедур).

Недетерминизм можно было бы устранить введением синхронного выполнения параллельных процедур в едином времени. Однако это представляется нежелательным как в теоретическом, так и в прикладном аспекте.

ПРИМЕР I. Пусть F - имя одной процедуры, имеющей вид z:=F(x); stop.; G - имя другой процедуры, имеющей вид z:= G(x);stop.; c - счетчик процедур, и пусть схема имеет вид: input(x);F(x)‖G(x); if c = 1 then y:= a(x) else y:= b(x); output(y); stop.

При интерпретации $(\forall x)(a(x) = 1 \;\&\; b(x) = 0 \;\&\; F(x) = G(x) = 2)$ по определению c = I или 2 недетерминированно и схема вычисляет многозначную функцию, принимающую на всех аргументах значения I и 0.

ПРИМЕР 2. Для схемы примера I рассмотрим другую интерпретацию: пусть $F(x) = 2$ при $x \in A$ и $F(x)$ не определено при $x \overline{\in} A$; $G(x) = 2$ при $x \overline{\in} A$ и $G(x)$ не определено при $x \in A$. Множество $A \neq \emptyset \;\&\; A \neq N$. Функции a и b те же. При этой интерпретации схема вычисляет характеристическую функцию множества A .

Она, очевидно, отражает идею доказательства теоремы Поста о рекурсивности множества в случае перечислимости его и его дополнения.

Вообще в теории алгоритмов часто встречаются описания конструкций, в которых осуществляется запуск параллельно некоторому числу процессов вычисления в ожидании, который из них остановится.

Предлагаемые классы схем программ дают точные абстрактные описания в программистских терминах средств параллелизма, используемых в таких конструкциях.

Изучим вопрос, насколько же сильны введенные средства. Рассмотрим,в первую очередь, классы схем программ с точки зрения того, какие функциональные отображения они могут реализовать.

Пусть схема вычисляет некоторую функцию (возможно,многозначную), зависящую от входной переменной, все ее предикатные символы интерпретированы вычислимыми предикатами, и все функциональные символы, кроме одного, - вычислимыми функциями. Такая частично интерпретированная схема, естественно, представляет некоторый функционал, сопоставляющий каждой частичной функции (подставляемой вместо единственного оставшегося неинтерпретированным символа схемы) функцию (многозначную), вычисляемую в результате получившейся полностью интерпретированной схемой.

Далее будем предполагать, что схема такова, что при любой такой интерпретации она вычисляет однозначную функцию.

ТЕОРЕМА I. Класс функционалов, представляемых схемами из $\varphi(A,NPP,=)$, совпадает с классом рекурсивных операторов.

ДОКАЗАТЕЛЬСТВО. Пусть схема S представляет функционал Φ и $\Phi(f) = g$. Если имеется некоторый способ перечисления графика функции f, то, ввиду того, что все остальные функции и предикаты схемы эффективно вычислимы по определению, эффективно можно перечислять и график функции g. Для этого детерминированное вычисление функции g необходимо организовать следующим образом. При вызове параллельных процедур начинаем одновременное (по шагам) их вычисление; как только (и если) одна из них завершит свою работу, присваиваем ее номер счетчику процедур в вызывавшей процедуре и возвращаемся к ее выполнению. В остальном вычисление осуществляется как и при интерпретации схем класса $\varphi(A,=)$. Это обеспечивает эффективное вычисление функции g. То, что можно перечислять график g, имея перечисление графика f, означает, что $g = \Phi_z(f)$ для некоторого оператора перечисления Φ_z и $g \leq_e f$. Далее, так как $\Phi(f)$ является по предположению (детерминированной) функцией для любой f, то $\Phi = \Phi_z$ является рекурсивным оператором [3].

Обратно, пусть Φ_z – рекурсивный оператор. По определению [3],

$$g=\Phi_z(f) \leftrightarrow (\forall x)(\langle x,y\rangle \in \tau(g) \leftrightarrow (\exists t)(\langle\langle x,y\rangle,t\rangle \in W_z \;\&\; D_t \subseteq \tau(f))) .$$

Здесь W_z – рекурсивно-перечислимое множество с номером z; D_t – конечное множество с номером t; $\tau(f)$ – график функции f.

Дадим описание схемы S, представляющей функционал Φ_z. Пусть на вход схемы поступило число x. Организуем перечисление элементов множества W_z; если для некоторых y и t окажется, что $\langle\langle x,y\rangle, t\rangle \in W_z$, то вызываем процедуру, проверяющую, будет ли $D_t \subseteq \tau(f)$ (вычисляя значение f в соответствующих точках), и параллельно продолжаем перечислять W_z. Если для некоторых y, t имеет место $\langle\langle x,y\rangle,t\rangle \in W_z$ и $D_t \subseteq \tau(f)$, то выдаем на выход (для этого требуется счетчик процедур в определении схем). Значение функции g на аргументе x вычислено. Так как Φ_z – рекурсивный оператор, то он переводит функции в функции и, следовательно, построенная схема при любой интерпретации функционального символа f будет вычислять однозначную функцию. Теорема доказана.

Для схем класса $\varphi(A,DPP,=)$ более удобно рассматривать представление ими не функционалов, а операторов перечисления, отображающих множества в множества. Для этого будем считать, что единственным не детерминированным символом схемы является предикатный символ $P(x)$ и что схема также вычисляет предикат $R(y)$. В этом случае P и R можно рассматривать как характеристические функции множеств. Используя теорему I, легко получить характеризацию функционалов, представимых схемами из $\varphi(A,DPP,=)$.

СЛЕДСТВИЕ. Класс функционалов, представляемых схемами из $\varphi(A,DPP,=)$, совпадает с классом операторов перечисления.

ДОКАЗАТЕЛЬСТВО. То, что рассматриваемые функционалы являются операторами перечисления, доказывается так же, как и в теореме I. Обратно, если Φ_z – оператор перечисления, $B = \Phi_z(A)$, то $y \in B \leftrightarrow (\exists t)(\langle y,t\rangle \in W_z \;\&\; D_t \subseteq A)$. Схема, представляющая Φ_z, строится так же, как и в теореме I, с той разницей, что нет необходимости искать значение функции $(z$ в доказательстве теоремы I), а нужно лишь выяснить, существует ли число t со свойством $\langle y,t\rangle \in W_z \;\&\; D_t \subseteq A$. Это можно сделать параллельной проверкой всех t, не используя информации, для какого t свойство имеет место, т.е. проверкой средствами схем из класса $\varphi(A,DPP,=)$. Следствие доказано.

Теперь решим вопрос (с точки зрения сравнительной схематологии), как соотносятся введенные средства параллелизма и недетерминированности с другими, уже известными средствами. Легко показать, что, например, недетерминированные схемы программ В.А.Непомнящего и Н.В.Шилова [5] транслируются в $\varphi(A,NPP,=)$.

Не будем, однако, останавливаться на частных средствах недетерминированности и параллелизма, а докажем общий результат, показывающий, что $\varphi(A,NPP,=)$ и $\varphi(A,DPP,=)$ являются универсальными в очень широких классах схем программ и их можно трактовать как разумно определенные классы параллельных детерминированных и недетерминированных схем программ. Аналогичный результат для последовательных детерминированных схем программ получен в [2,II]. Следует подчеркнуть, что в этом исследовании под эквивалентностью схем программ понимается их функциональная эквивалентность. Для параллельных схем программ известны и другие понятия эквивалентности, например, эквивалентность их по истории ячеек [6].

При построении завершенной теории параллельных схем программ необходимо, конечно, решить вопрос об универсальных средствах и для других эквивалентных схем программ. В настоящей работе, однако, рассматривается лишь функциональная эквивалентность. Функциональная эквивалентность наиболее адекватно отражает современную идеологию абстрактных типов данных. Она предполагает, что процедуры, входящие в кластер абстрактных типов данных, используются сугубо функционально, т.е. в зависимости только то того, какие функции и предикаты они вычисляют, но безотносительно к их реализации, истории выполнения и т.д.

В §2 настоящей статьи изучаются абстрактные вычислительные теории, для построения которых концепция функциональной эквивалентности оказывается вполне достаточной.

Обозначим через $Val(R,I)$ результат работы схемы R при интерпретации I всех функциональных, предикатных символов схемы R и входа на области D. Как синоним слова "интерпретация" будет употребляться слово "модель". Разумеется, $Val(R,I)$ может быть и не определен. Кроме того, в случае, если R является недетерминированной схемой, то вполне возможно, что при разных моделях I будут получаться разные результаты, так что $Val(R,I)$ образует множество всех этих результатов. Представляется, что разумно определенный класс схем программ K должен обладать следующими свойства-

ми свойствами (они обобщены на случай параллельных и недетерминированных схем программ в работах [2,II]):

I. Если λ – изоморфизм $I \to I'$, то

$$x \in Val(R,I) \leftrightarrow \lambda(x) \in Val(R,I') .$$

Это свойство означает, что результат работы программы одинаков в изоморфных моделях и дает возможность осуществлять различные переобозначения и перекодировки моделей.

2. Пусть $\gamma(I)$ – наименьшая подмодель модели I, содержащая элемент, являющийся интерпретацией входа в I. Тогда $Val(R,I) = Val(R,\gamma(I))$. Это свойство означает, что при выполнении программы будут появляться только те элементы, которые получаются из уже имеющихся при помощи используемых программой функций.

3. Пусть $y \in Val(R,I)$. Тогда найдется конечное множество $I' \subseteq I$ такое, что $(\forall I_1)(I' \subseteq I_1 \to y \in Val(R,I_1))$. Это свойство означает, что заканчивающееся вычисление может извлекать лишь конечную информацию из модели I.

4. Пусть некоторый конечный набор термов U сигнатуры σ схемы R с непротиворечиво определенными на нем предикатами сигнатуры σ, равенством и предикатами "быть определенным" для функций из σ (эти предикаты истинны в тех точках, в которых функции, которым они соответствуют, определены и не определены в остальных) согласован с R и порождает τ, если $(\forall I \supseteq U)(\tau \in Val(R,I))$. Тогда существует алгоритм, который по любой $R \in K$ задает эффективный пересчет рекурсивно-перечислимого множества пар $\langle U,\tau \rangle$ таких, что U согласован с R и порождает τ. Это свойство является отражением эффективности вычислений в классе K, оно постулирует существование в некотором роде универсальной процедуры вычислений схем из класса K при всех интерпретациях.

ТЕОРЕМА 2. Если класс схем программ K обладает свойствами I–4, то он транслируем в $\varphi(A,NPP,=)$.

ДОКАЗАТЕЛЬСТВО. Пусть R – схема программ из класса K; M – рекурсивно-перечислимое множество всех пар $\langle U,\tau \rangle$ таких, что U согласован с R и порождает τ. Построим схему $\tilde{R} \in \varphi(A,NPP,=)$, эквивалентную R. Схема $\tilde{R}$ будет использовать вспомогательные массивы c_0 и c_1. В массив c_0 заносятся элементы множества M, в массив c_1 – коды всех термов (без повторений) в сигнатуре схемы R. Это может быть осуществлено средствами $\varphi(A,=)$. Технические

детали программирования в $\varphi(A,=)$ здесь опускаются. Их легко восполнить, руководствуясь [4,2].

В остальном работа схемы $\tilde{R}$ выглядит следующим образом. Для каждого элемента массива c_0, т.е. для каждой пары $\langle U,\tau\rangle$, осуществляется следующий процесс: для всех элементов массива (термов) c_1 вычисляются подряд все предикаты сигнатуры схемы R, и предикат равенства и полученные значения сопоставляются со значениями тех же предикатов на тех же термах, закодированных в U. Для предикатов из U "быть определенным" вызывается процедура, работающая параллельно со всем вышеописанным алгоритмом и проверяющая определенность указанных в U функций на указанных термах. Если все сравниваемые значения совпадают, то схема вычисляет значение терма τ, выдает его на выход и параллельно (и недетерминированно !) с этим продолжает весь вышеописанный процесс в поисках пары $\langle U, \tau'\rangle$ с тем же U, но с $\tau' \neq \tau$.

Проверим, что R и $\tilde{R}$ действительно эквивалентны. Рассмотрим произвольную интерпретацию I. Пусть $y \in Val(R,I)$, тогда, по свойству 3, $\exists$ конечное $I' \subseteq I$ такое, что $(\forall I_1)(I' \subseteq I_1 \Rightarrow y \in Val(R,I_1))$. Это, по свойству 2, означает, что для некоторого τ, вычисляющего y, множество I' согласовано с R и порождает τ (по свойству 4).

Но тогда по конструкции схема $\tilde{R}$ при интерпретации I обнаружит его совпадение с I' на I' и выдаст в одном из вариантов значение терма τ в I, равное y, т.е. $y \in Val(\tilde{R},I)$.

Обратно, если $y \in Val(\tilde{R},I)$, то это возможно лишь при совпадении I с I', порождающим τ, причем τ в интерпретации I равно y. Но тогда и $y \in Val(R,I)$. Теорема доказана.

Перейдем к рассмотрению детерминированного случая.

СЛЕДСТВИЕ. Если класс детерминированных схем программ (с возможностью параллельных вычислений) K обладает свойствами I-4, то он транслируем в $\varphi(A,DPP,=)$.

ДОКАЗАТЕЛЬСТВО. Детерминированность R из K означает, что $(\forall I)\,|Val(R,I)| \leq 1$. Построим схему $\tilde{R}$, эквивалентную R, как в доказательстве теоремы 2. Для $\tilde{R}$ также справедливо неравенство $(\forall I)\,|Val(\tilde{R},I)| \leq 1$. Но последнее означает, что значение счетчика процедур не влияет на результат работы интерпретированной схемы. Тогда его, очевидно, можно удалить из схемы, не изменив ре-

зультатов ее работы, т.е. сделать ее детерминированной, принадлежащей к классу $\varphi(A, DPP, =)$. Следствие доказано.

§2. Вычислимость в алгебраических системах

В этом параграфе введенные средства параллелизма будут применены для изучения абстрактной вычислимости в произвольной алгебраической системе $\mathcal{A} = \langle A, L \rangle$, где A – основное множество, L – конечная сигнатура. Проблема построения абстрактной по А.П.Ершову теории вычислимости в $\mathcal{A}$ (можно предполагать, что $A \approx N$) состоит в описании класса функций и предикатов, которые следует считать вычислимыми в предположении, что все функции и предикаты, входящие в L, вычислимы. Одно из возможных решений этой проблемы дано в [II], где в качестве такого класса взято замыкание L относительно совокупности конечных алгоритмических процедур (последовательных) со счетчиками и рекурсией. Предполагается, что L содержит предикат $=$. Известно [7], что класс схем программ с рекурсией и счетчиками взаимнотранслируем с классом $\varphi(A)$, здесь через $\mathrm{FAPCS}(\mathcal{A})$ обозначено замыкание L относительно этого класса схем программ.

В [7,8] показано, что $\mathrm{FAPCS}(\mathcal{A})$ является минимальной вычислимой теорией над $\mathcal{A}$ в смысле аксиоматического подхода [9] и что она совпадает при выполнении некоторых естественных условий на $\mathcal{A}$ с классом $\mathrm{Ind}(\mathcal{A})$ индуктивно определенных функций и предикатов в смысле Платека [I0]. Таким образом, в принципе все три независимо развитых в [II,9,I0] подхода привели к одному и тому же классу функций и предикатов. Стало быть, в интуитивном смысле $\mathrm{FAPCS}(\mathcal{A})$ является наиболее предпочтительным классом функций и предикатов, вычислимых над $\mathcal{A}$.

ЗАМЕЧАНИЕ. В [7,8] программа, использующая частичные функции, работает следующим образом: область A заменяется на $A_u = A \cup \{u\}$, и если какая-либо функция ψ в точке $x \in A$ не определена, то полагают, что $\psi(x) = u$. Кроме того, $\psi(u) = u$ для всех ψ. Тем самым случай частичных функций сводится, по сути, к случаю всюду определенных функций в расширенной области. Поскольку разница между неопределенностью вследствие зацикливания и неопределенностью-отказом нивелируется, то использование частичных функций и предикатов тривиально, т.е. сводится к случаю полной определенности.

Это соответствует подходу, принятому практически во всех остальных работах по схемам программ: если функция $\psi(x)$ (или предикат $p(x)$) не определена, а в программе требуется вычислить

$\phi(x)$ ($p(x)$) , то считают, что процесс выполнения программы в этом месте прерывается, а результат работы не определен. Поэтому чаще всего [I,4] в классической схематологии сразу ограничиваются лишь всюду определенными функциями и предикатами.

Во введенных классах схем с параллельными процедурами появляется другая возможность выполнения программы. Если в некоторой точке требуется вычислить значение частичной функции ϕ на аргументе x, то можно организовать обращение к двум параллельным процедурам, одна из которых является оператором $y := \phi(x)$, а другая представляет собой часть программы, продолжающую вычисления как бы в предположении, что $\phi(x)$ не определена.

Если $\phi(x)$ действительно не определена, то первая из этих процедур не сообщает о своем завершении и тем самым не влияет на процесс вычислений, который тем не менее продолжается второй процедурой. Если же $\phi(x)$ определена, то вычисления также будут продолжены со следующего оператора после оператора обращения к параллельным процедурам.

Аналогично обстоит дело и с предикатами. Рассмотренная возможность позволяет работать с частичными функциями и предикатами более естественным образом.

Пусть L содержит частичные функции и предикаты. Какие средства замыкания следует выбрать для получения вычислительной теории над L? По аналогии со случаем всюду определенных функций и предикатов, когда выбирается универсальный класс последовательных схем программ, для частичных функций и предикатов также необходим универсальный класс параллельных схем программ. Причем универсальность здесь понимается в смысле функциональной эквивалентности.

Отметим, что класс $\varphi(A, DPP, =)$ параллельных детерминированных схем программ явно недостаточен. Действительно, пусть $L = \{ a, b, F, G \}$, где a, b, F, G взяты из примера 2 §I. Тогда программа из примера I будет вычислять характеристическую функцию множества A. В то же время любая детерминированная программа в базисе L не содержит нетривиальных разветвлений и поэтому не может вычислять характеристическую функцию нетривиального множества.

Обозначим через $NFAPCS(\mathcal{CL})$ замыкание L относительно схем программ из класса $\varphi(A, NPP, =)$. При этом используются лишь такие интерпретации схем из класса $\varphi(A, NPP, =)$, которые дают программы, вычисляющие однозначные функции и предикаты.

Рассмотрим упорядоченную по включению совокупность всех вычислительных теорий $NFAPCS(\mathcal{A})$ для всевозможных $\mathcal{A}$ и обозначим ее через NFD .

ПРЕДЛОЖЕНИЕ I. Структура e-степеней вкладывается в NFD как концевой сегмент.

ДОКАЗАТЕЛЬСТВО. Пусть $a \in NFD$, $a = NFAPCS(\mathcal{A})$ и a содержит все ч.р.ф. Тогда a совпадает с совокупностью функций и предикатов, e-сводящихся к некоторой e-степени b , где b - наименьшая верхняя грань e-степеней функций и предикатов из сигнатуры системы $\mathcal{A}$. Это легко вытекает из части "обратно" теоремы I. И наоборот, множество функций и предикатов, сводящихся к некоторой e-степени b, замкнуто относительно $\varphi(A,NPP,=)$.

Действительно, пусть схема из класса $\varphi(A,NPP,=)$ использует функции и предикаты, e-сводящиеся к e-степени b, и вычисляет функцию g . Тогда так же как и в первой части доказательства теоремы I, $g \leq_e b$. Так как, очевидно, отношение $\leq$ на e-степенях и отношение включения в NFD изоморфны, то структура e-степеней изоморфна подструктуре NFD, состоящей из всех вычислительных теорий, содержащих все ч.р.ф. Предложение доказано.

Проследим, что получится, если ограничиться всюду определенными функциями и предикатами, т.е. считать, что сигнатура L системы $\mathcal{A}$ состоит лишь из всюду определенных функций и предикатов и в вычислительную теорию $FAPCS(\mathcal{A})$ входят также лишь всюду определенные функции и предикаты.

Структуру всевозможных $FAPCS(\mathcal{A})$, упорядоченную по включению, обозначим через FD .

ПРЕДЛОЖЕНИЕ 2. Структура T-степеней D вкладывается в FD как концевой сегмент.

ДОКАЗАТЕЛЬСТВО. Схема доказательства та же, что и предложения I. Пусть $a \in FD$, $a = FAPCS(\mathcal{A})$ и a содержит все ч.р.ф. Тогда a совпадает с совокупностью функций и предикатов, сводящихся к некоторой T-степени b , где b - наименьшая верхняя грань T-степеней функций и предикатов из сигнатуры L системы $\mathcal{A}$.Действительно, схему программ,вычисляющую join элементов L ,легко построить, используя имеющиеся,по предположению, ч.р.ф. Если теперь, скажем, $A \leq_T b$ и сведение осуществляется некоторой машиной Тьюринга с

оракулом b, то работу машины Тьюринга можно промоделировать [II] средствами $\varphi(A)$ и тем самым показать, что $A \in a$.

Обратно, множество функций и предикатов, сводящихся к некоторой T-степени b, замкнуто относительно $\varphi(A)$.

Действительно, если схема программ $S \in \varphi(A)$, интерпретированная функциями и предикатами T-степени не выше b, вычисляет всюду определенную (по условию) функцию g, то в силу эффективности работы схем программ класса $\varphi(A)$ функция $g \leq_T b$. Предложение доказано.

Таким образом, переход от последовательных схем программ к параллельным соответствует переходу от сводимости по разрешимости к сводимости по перечислимости в теории алгоритмов.

Литература

I. КОТОВ В.Е. Введение в теорию схем программ. -Новосибирск: Наука, Сиб.отд., 1978. - 257 с.

2. ТРАХТЕНБРОТ Б.А. Об универсальных классах схем программ //Теория программирования. - Новосибирск, 1972. Т.1. - С.239-249.

3. РОДЖЕРС Х. Теория рекурсивных функций и эффективная вычислимость. - М.: Мир, 1972. - 624 с.

4. CONSTABLE R., GRIES D. On classes of program schemata // SIAM Comput. - 1972.- Vol.1.-P.66-118.

5. НЕПОМНЯЩИЙ В.А., ШИЛОВ Н.В. Недетерминированные схемы программ и динамические логики //Трансляция и преобразование программ. - Новосибирск. 1984. - С. 151-169.

6. КАРП Р.М., МИЛЛЕР Р.Е. Параллельные схемы программ //Кибернетический сборник. -М.: Мир, 1976. -Вып.13. - С. 5-61.

7. MOLDESTAD J., STOLTERBERG-HANSEN V., TUCKER J. Finite algorithmic procedures and inductive definability// Math.Scand. - 1980.- Vol.46.-P.62-76.

8. MOLDESTAD J., STOLTENBERG-HANSEN V., TUCKER J. Finite algorithmic procedures and computation theories// Math.Scand.-1980. - Vol.46.-P.77-94.

9. FENSTAD J. General recursion theory: an axiomatic approach.- Berlin: Springer-Verlag,1980.

10. PLATEK R. Foundations of recursion theory//Ph.D.Thesis, Stanford, Stanford University,1966.

11. SHEPHERDSON J.C. Computations over abstract structures : serial and parallel procedures and Friedman's effective definitional schemes// Logic colloqvium'73.- Amsterdam,1975.- P.445-513.

Поступила в ред.-изд.отд.
16 декабря 1986 года