ТЕОРИЯ ВЫЧИСЛИМОСТИ И ЯЗЫКИ СПЕЦИФИКАЦИЙ
(Вычислительные системы)
1991 год Выпуск 139

УДК 519.685+510.52

СИНТЕЗ РЕШЕНИЯ ПРОСТЫХ ЛОГИЧЕСКИХ ЗАДАЧ В СЕМАНТИЧЕСКОМ ПРОГРАММИРОВАНИИ

И.В. Пивкина

1. Постановка задачи

Пусть $\mathcal{A}$ - модель сигнатуры σ.

Рассматриваются задачи вида:

$$!\bar{x}\ ?\bar{y}\ P(t_1(\bar{x},\bar{y}),\ldots,t_m(\bar{x},\bar{y})),$$

где $\bar{x} = x_1,\ldots,x_k$, $\bar{y} = y_1,\ldots,y_s$ - переменные, P - предикатный символ, $t_1,\ldots,t_m$ - термы сигнатуры σ. Требуется найти при заданных $\bar{b} = b_1,\ldots,b_s$ все наборы значений $\bar{a} = a_1,\ldots,a_k$ такие, что $\mathcal{A} \models P(t_1(\bar{a},\bar{b}),\ldots,t_m(\bar{a},\bar{b}))$ (при этом наборы $a_1,\ldots,a_k$ называются решениями).

Считаем, что область значений переменных рекурсивно-перечислима и существует эффективная процедура G, генерирующая все возможные значения переменной.

Возможны различные процедурные прочтения n-местного предиката или функции (см. [2]).

Например, функция $f(\_,\_)$ может рассматриваться как процедура

1) находящая значение $x = f(y_1,y_2)$ по двум аргументам y_1,y_2;

2) находящая второй аргумент y_2, такой что $x = f(y_1,y_2)$, по первому аргументу y_1 и значению x;

3) определяющая, верно ли, что $f(y_1,y_2) = x$ для данных аргументов y_1,y_2 и значения x, и так далее.

Набор заданных переменных назовем входом процедуры, набор искомых переменных - выходом. В случае 3 процедура будет иметь три входных параметра и один выходной, выдающий ответ истина/ложь.

Назовем конкретное процедурное прочтение реализацией предиката или функции, если у нас имеется программа (или устройство), осуществляющая требуемое действие.

В настоящей работе описывается алгоритм, позволяющий находить все решения задачи указанного типа при условии, что предикат и функции, входящие в формулу, имеют хотя бы по одной реализации. В п.2 изложено описание алгоритма. В п.3 показано, что сложность вычисления зависит от способа кодировки модели.

2. Описание алгоритма

Предлагаемый алгоритм решения задачи будет состоять из двух этапов. Первый этап - построение схемы вычислений, которая устанавливает связи между входами и выходами процедур, второй этап - счет по схеме.

Схема вычислений описывается ориентированным графом $\vec{S}_\varphi = (I_\varphi, \vec{R}_\varphi)$, где I_φ - множество вершин, а $\vec{R}_\varphi$ - множество дуг (ориентированных ребер). Стыковка ребра (дуги) j и вершины i производится через буфер - некоторое целое число. Ребро, снабженное буфером k, будем называть k-м ребром вершины i. (Ребро $\{i(k_1), j(k_2)\}$ есть k_1-е ребро вершины i и k_2-е ребро вершины j; $(i(k_1), j(k_2))$ - дуга, выходящая из вершины i с буфером k_1 и входящая в вершину j с буфером k_2.)

2.1. Построение схемы вычислений $(\vec{S}_\varphi)$. Пусть дана задача: $!\bar{x}\ ?\bar{y}\ \varphi(\bar{x},\bar{y})$, где φ - формула указанного вида. Построение схемы вычислений можно разбить на два этапа: построение неориентированной схемы, которая однозначно определяется формулой φ, и ориентацию схемы - выбор конкретных реализаций, входящих в формулу предиката и функций.

2.1.1. Построение неориентированной схемы S_φ. Положим $T_\varphi = \{\varphi\} \cup \{$все различные подтермы $\varphi\}$. (Если $\varphi = P(t_1,\dots,t_s)$, то $t_1,\dots,t_s$ - подтермы φ; если $t = f(g_1,\dots,g_k)$ - подтерм φ, то $g_1,\dots,g_k$ - также подтермы φ.) Число вершин в графе S^0_φ возьмем равным числу $|T_\varphi|$. Для удобства вершины обозначим натуральными числами. В результате получим множество вершин I^0_φ (в дальнейшем будем называть их основными; вершины, не являющиеся основными, будем называть соединительными). Произвольным образом определим на I^0_φ функцию $\mathrm{term}: I^0_\varphi \to T_\varphi$, задающую взаимно-однозначное соответствие между вершинами и различными элементами T_φ. Определим на T_φ отношения "$<$" и "$<_i$", $i \in \mathbb{N}$. Будем считать, что $g <_i h$ $(g,h \in T_\varphi)$, если в формуле φ встречается $h(t_1,\dots,t_k)$, и $g = t_i$, $1 \le i \le k$; аналогично, $g < h$, если $(\exists i)(g <_i h)$. В дальнейшем будем использовать также функцию $\mathrm{name}: \{$множество вершин$\} \to \{$множество функций, предикатов и переменных, входящих в $\varphi\} \cup \{G, =^k, \mathrm{pl}^k, \&^k\}$. Для $i \in I^0_\varphi$ определим

$$\mathrm{name}(i) = \begin{cases} z, & \text{если } \mathrm{term}(i) = z \text{ - переменная,} \\ h, & \text{если } \mathrm{term}(i) = h(g_1,\dots,g_k), \end{cases}$$

где $g_1,\dots,g_k$ - термы. Положим

$$R^0_\varphi = \bigcup_{\substack{i,j \in I^0_\varphi \\ \mathrm{term}(i) < \mathrm{term}(j)}} \ \bigcup_{k \ \mathrm{term}(i) <_k \mathrm{term}(j)} \{j(k), i(0)\}.$$

Получим некоторый граф $S^0_\varphi = (I^0_\varphi, R^0_\varphi)$. Видно, что у любой вершины $i \in I^0_\varphi$ не более одного k-го ребра, $k \geq 1$, но нулевых ребер может быть несколько.

Пусть у вершины i_0 имеется m штук нулевых ребер: $r_1 = \{i_0(0);\ j_1(k_1)\},\ \ldots\ , r_m = \{i_0(0);\ j_m(k_m)\}$. Добавим к нашему графу соединительную вершину $j \notin I^0_\varphi$ с неопределенными пока значением и буферами так, что

$$I''_\varphi = I^0_\varphi \cup \{j\},$$

$$R''_\varphi = \Big[R^0_\varphi \cup \{\{i_0(0);j\},\{j;j_1(k_1)\},\ldots \ldots,\{j;j_m(k_m)\}\}\Big] \setminus (r_1,\ldots,r_m).$$

Получим граф $S''_\varphi = (I''_\varphi, R''_\varphi)$.

То же самое проделываем при необходимости для любой другой основной (из I^0_φ) вершины.

В результате получим граф $S^*_\varphi = (I^*_\varphi, R^*_\varphi)$, у которого каждая основная вершина имеет не более одного k-го ребра для всех $k = 0,1,\ldots$

В середину каждого ребра, соединяющего две основные вершины, тоже вставляем по соединительной вершине:

$$R_\varphi = \Big[R^*_\varphi \setminus R^0_\varphi\Big] \cup \bigcup_{\substack{r \in (R^0_\varphi \cap R^*_\varphi) \\ r=\{j(k),i(0)\}}} \{\{j(k);n_{ij}\},\{n_{ij};i(0)\}\},$$

где все n_{ij} различны при различных i,j и $n_{ij} \notin I^*_\varphi$,

$$I_\varphi = I^*_\varphi \cup \{n_{ij} \mid r = \{j(k);i(0)\} \in R^0_\varphi \cap R^*_\varphi\}.$$

Непосредственно из построения следует, что граф $S_\varphi = (I_\varphi, R_\varphi)$ обладает следующими свойствами:

а) любое ребро в S_φ соединяет одну основную и одну соединительную вершины;

б) от любой основной вершины i отходят n ребер, где

$$n = \begin{cases} 1, & \text{если } name(i) = x \text{ - переменная,} \\ k+1, & \text{если } name(i) = f \text{ - } k\text{-местная функция,} \\ k, & \text{если } name(i) = p \text{ - } k\text{-местный предикат;} \end{cases}$$

в) от любой соединительной вершины j отходят не менее двух ребер, причем одно из них имеет вид $\{j; m(0)\}$, $m \in I^0_\varphi$.

2.1.2. Построение ориентированного графа $\vec{S}_\varphi$. Для каждой основной вершины $i \in I^0_\varphi$ фиксируем определенным образом какую-нибудь реализацию $name(i)$ (хотя бы одна реализация существует по условию).

Пусть $\{i;j(k)\}$ - ребро графа S_φ (i - номер соединительной вершины, $j \in I^0_\varphi$. Согласно выбранной нами реализации $name(j)$ для вершины j смотрим, является k-й параметр входным или выходным. Если он является входом (выходом), то в ориентированном графе будет дуга $(i,j(k))$ (соответственно $(j(k),i)$). Таким образом ориентируем все ребра S_φ. Получим ориентированный граф $\vec{S}'_\varphi$ с множеством вершин I_φ, множеством дуг $\vec{R}'_\varphi$.

Назовем вершину ориентированного графа достижимой, если все дуги, входящие в эту вершину, выходят из достижимых вершин. В частности, генераторы (вершины, не имеющие входов) являются достижимыми.

Для того чтобы по ориентированному графу можно было проводить вычисления, необходимо, чтобы все его вершины были достижимыми.

Предположим, что $\exists i_1 \in I_\varphi$, i_1 - недостижимая. Это означает, что существует недостижимая вершина $i_2 \in I_\varphi$:

$i_2 \neq i_1$ и $(i_2, i_1) \in \vec{R}'_\varphi$. Следовательно, $\exists i_3 \in I_\varphi$: $i_3 \neq i_2$, $(i_3, i_2) \in \vec{R}'_\varphi$ и i_3 - недостижимая. И так далее. Поскольку вершин конечное число, то $\exists i_p \in I_\varphi$ такое, что для некоторого $k < p$: $\{(i_k, i_p), (i_p, i_{p-1}), \ldots, (i_{k+1}, i_k)\} \subseteq \vec{R}'_\varphi$. (Вершины $i_k, i_{k+1}, \ldots, i_p$ образуют цикл.) Таким образом, доказана

ЛЕММА. *В графе есть недостижимые вершины* $\Leftrightarrow$ *в нем есть цикл из недостижимых вершин.*

Возьмем какую-нибудь соединительную вершину i из этого цикла. Пусть в нее входят дуги $d_1 = (n_1(k_1), i), \ldots, d_s = (n_s(k_s), i)$, а выходят $d_{s+1}, \ldots, d_{s+l}$. Добавим к нашему графу еще одну соединительную вершину $j \notin I_\varphi$ так, чтобы получился новый ориентированный граф:

$$\vec{S}''_\varphi = \Big[I_\varphi \cup \{j\}, (\vec{R}_\varphi \setminus \{d_1, \ldots, d_s\}) \cup \{(i,j), (n_1(k_1), j), \ldots, (n_s(k_s), j)\}\Big].$$

В результате вершина i становится генератором, т.е. достижимой вершиной. Добавленная вершина j не имеет выходов, значит, ни в какой цикл входить не может. Следовательно, мы построили граф, в котором число циклов из недостижимых вершин уменьшилось. Проделывая, если нужно, эту процедуру несколько раз, получим граф $\vec{S}''_\varphi = (I''_\varphi, \vec{R}''_\varphi)$, не имеющий недостижимых вершин.

Рассмотрим множество соединительных вершин в $\vec{S}''_\varphi$: $I''_\varphi \setminus I^0_\varphi$. Пусть $i \in I''_\varphi \setminus I^0_\varphi$. Если в i входят дуги d_1, $d_2 = (n_2(k_2), i), \ldots, d_s = (n_s(k_s), i)$, $s \geq 2$, а выходят дуги $d_{s+1}, \ldots, d_{s+l}$, $l \geq 1$, то преобразуем наш граф в граф $\vec{S}^*_\varphi = (I^*_\varphi, \vec{R}^*_\varphi)$, где

$$I^*_\varphi = I''_\varphi \cup \{j\}\ ,\quad j \notin I''_\varphi\ ,$$

$$\vec{R}^*_\varphi = \left[\vec{R}^*_\varphi \setminus \{d_2,\ldots,d_s\}\right] \cup \{(i,j),(n_2(k_2),j),\ldots$$
$$\ldots,(n_s(k_s),j)\}.$$

Если у вершины i нет входов, то положим $\mathbf{name}(i) = G$.

Если у i имеется k входов, но нет выходов, то $\mathbf{name}(i)$ – "k-местный предикат равенства" $(=^k)$.

Если у i один вход и k выходов, $k \geq 1$, то $\mathbf{name}(i) = pl^k$, где pl^k - процедура, которая значение входа передает без изменений на все выходы.

Таким способом мы построим граф $\vec{S}^*_\varphi = (I^*_\varphi, \vec{R}^*_\varphi)$ с функцией $\mathbf{name}$, определенной на всех вершинах.

Если в построенном графе будет $n > 1$ не имеющих выходов вершин (результатом работы которых является $\mathbf{true}$ или $\mathbf{false}$) с номерами $i_1,\ldots,i_n$, то полагаем $I^{*\,\text{ит}}_\varphi = I^*_\varphi \cup \{j\}$, где $j \notin I^*_\varphi$, $\mathbf{name}(j) = \&^n$ ($\&^n$ - конъюнкция n членов, принимающая значение $\mathbf{true}$ или $\mathbf{false}$),

$$\vec{R}^{*\,\text{ит}}_\varphi = \vec{R}^*_\varphi \cup \{(i_1(n),j),\ldots,(i_n(j),j),$$
$$(j(b),n_1(1)),\ldots,(j(b),n_k(1))\},$$

где $n_1,\ldots,n_k$ таковы, что $\mathbf{name}(n_i) = x_i$ - выход исходной задачи (полученные в результате счета по схеме значения $x_1,\ldots,x_k$ будут решением, только если все встречающиеся в I^*_φ вершины без выходов дадут $\mathbf{true}$). Буфер b означает, что по дуге могут передаваться только логические константы $\mathbf{true}$. или $\mathbf{false}$. При этом набор значений $x_1,\ldots,x_k$ считается результатом работы схемы только в том случае, когда по дуге с буфером b будет передано $\mathbf{true}$.

Соответственно при $n=1$:

$$I_{\varphi}^{ит} = I_{\varphi}^{*},\quad \vec{R}_{\varphi}^{ит} = \vec{R}_{\varphi}^{*} \cup \{(i(b),n(1)),\ldots,(i(b),n(1))\},$$

где i - номер вершины, не имеющей выходов.

Полагаем $\vec{S}_{\varphi} = (I_{\varphi}^{ит}, \vec{R}_{\varphi}^{ит})$.

Таким образом, хотя бы один ориентированный граф $\vec{S}_{\varphi}$ для задачи вида $!x_1,\ldots,x_k?y_1,\ldots,y_l\varphi$ всегда можно построить.

ЗАМЕЧАНИЕ. Одному неориентированному графу S_{φ} могут соответствовать несколько различных ориентированных графов $\vec{S}_{\varphi}$ (в случае, когда у основных вершин существуют несколько различных реализаций).

2.2. Счет по схеме. Опишем процесс, позволяющий по схеме вычислений синтезировать все возможные решения. Нас будет интересовать только принципиальная возможность найти все решения, а не минимизация используемых при этом ресурсов времени и памяти. Поскольку возможны процедуры, в которых одному входу соответствует счетное число выходов, то для единообразия будем считать, что если какая-то реализация процедуры существует, то она обладает следующими свойствами:

- при любом входе каждый возможный выход, удовлетворяющий данному входу, рано или поздно будет сгенерирован (время работы считается бесконечным);

- либо при работе процедуры есть возможность запомнить состояние (т.е. значения всех используемых величин, а также на какой стадии счета находимся), либо для каждой процедуры есть счетное число устройств, производящих вычисления каждое для своего входа, чтобы можно было сделать перерыв в счете, а затем продолжить счет в точности с того момента, на котором прервались (в любой момент времени счет ведется только на одном устройстве);

- за конечное время работы при заданном входе может быть сгенерировано лишь конечное число наборов выходов.

Фиксируем некоторый конечный промежуток времени - "единицу времени".

Пусть $1,\ldots,k$ - номера вершин, входящих в схему. Вычисления будем производить по шагам.

i-й шаг. Начинает "работу" вершина 1. Через конечное число единиц времени она прервется. Затем начинает "работу" вершина 2,..., затем вершина k. Конец i-го шага.

Под словами "работа вершины i" подразумеваются следующие действия:

а) если на предыдущих шагах уже начались вычисления по фиксированной в $\vec{S}_\varphi$ реализации $\mathrm{name}(i)$ для конечного числа входов, то для каждого из этих входов продолжается счет в течение единицы времени (так как "старых" входов конечное число, то за конечное время это будет сделано);

б) из вновь прибывших значений входов и имеющихся старых составляется конечное число новых входов (отличных от старых с учетом согласованности переменных, например, если один аргумент получен при одном значении выхода данного генератора, а другой при другом, то пару они образовать не могут);

в) для каждого из полученных новых входов в течение единицы времени производятся вычисления (по реализации $\mathrm{name}(i)$).

Если при счете по п. "а" или п."в" генерируются выходы, то они передаются на входы нужным вершинам, которые накапливают их, дожидаясь следующего шага своих вычислений.

Поскольку все вершины были достижимыми, то каждая на некотором шаге начнет "работу". Так как вершин конечное число, и каждая на i-м шаге "работает" конечное число единиц времени, то $(i+1)$-й шаг всегда наступит.

2.3. Корректность и полнота алгоритма. Положим по определению $\forall i \in I^0_\varphi$: $v_r(i) = j \leftrightarrow \{i(r);j\} \in R_\varphi$. Заме -

тим, что набор $\bar{a}$ может быть получен в качестве решения исходной задачи в том и только том случае, когда выполняются следующие условия.

Условие 1. Для любой соединительной вершины $j \in I_\varphi^{ит} \setminus I_\varphi^0$ значения всех ее входов и выходов совпадают.

Выполнение этого условия дает возможность определить на множестве соединительных вершин функцию μ как значение входов и выходов данной вершины.

Условие 2. Для любой основной вершины $i \in I_\varphi^0$:

а) если $\mathrm{name}(i) = P$ - n-местный предикат, то $P(\mu(v_1(i)), \ldots, \mu(v_n(i)))$ - истина;

б) если $\mathrm{name}(i) = f$ - n-местная функция, то $f(\mu(v_1(i)), \ldots, \mu(v_n(i))) = \mu(v_0(i))$;

в) если $\mathrm{name}(i) = z$ - переменная, то $z = \mu(v_0(i))$.

ТЕОРЕМА 1.

1) Любой полученный описанным способом набор $x_1, \ldots, x_k$ *будет решением исходной задачи* $!x_1, \ldots, x_k ?y_1, \ldots, y_1 P(t_1, \ldots, t_s)$;

2) каждое решение на некотором шаге будет получено.

ДОКАЗАТЕЛЬСТВО. Пусть S_φ - граф, соответствующий формуле $\varphi = P(t_1, \ldots, t_s)$. Граф $\vec{S}_\varphi$ - один из ориентированных графов, построенных по S_φ.

1) Пусть получили набор $\bar{a} = a_1, \ldots, a_k$. Рассмотрим множество термов $T_\varphi \setminus \{\varphi\}$. Докажем индукцией по глубине терма, что если $t \in T_\varphi \setminus \{\varphi\}$, то $t|^{a_1, \ldots, a_k}_{x_1, \ldots, x_k} = \mu(v_0(i))$, где $\mathrm{term}(i) = t$.

Действительно, если $t = z$ - переменная, то утверждение очевидно. Пусть $t = f(g_1, \ldots, g_n)$, $\mathrm{term}(i) = t$,

$term(i_1) = g_1, \ldots, term(i_n) = g_n$ . . По индукционному предположению имеем: $\mu(v_0(i_j)) = g_j|^{a_1,\ldots,a_k}_{x_1,\ldots,x_k}$, $j = 1, \ldots, n$. Таким образом, $t|^{a_1,\ldots,a_k}_{x_1,\ldots,x_k} =$

$= f\left(g_1|^{\bar{a}}_{\bar{x}}, \ldots, g_n|^{\bar{a}}_{\bar{x}}\right) = f(\mu(v_0(i_1)), \ldots, \mu(v_0(i_n))) =$

$= f(\mu(v_1(i)), \ldots, \mu(v_n(i))) = \mu(v_0(j))$.

Для выполнения преобразования используются условие 2б и следующая

ЛЕММА. *Для* $j = 1, \ldots, n$ *выполняется равенство* $v_0(i_j) = v_j(i)$.

ДОКАЗАТЕЛЬСТВО. Имеем $t = f(g_1, \ldots, g_n)$, $t = term(i)$, $g_j = term(i_j)$, $j = 1, \ldots, n$. Следовательно, $g_j <_j t$, т.е. $term(i_j) <_j term(i)$. Значит, $\{i_j(0); i(j)\} \in R^0_\varphi$ (это следует из построения $\vec{S}_\varphi$). Существует единственное $k \in I_\varphi \setminus I^0_\varphi$ такое, что $\{i_j(0); k\}, \{k; i(j)\} \in R_\varphi$. Следовательно, $v_0(i_j) = k = v_j(i)$, $j = 1, \ldots, n$. Лемма доказана.

Используя условие 2а, аналогично получаем, что $\varphi|^{\bar{a}}_{\bar{x}}$ - истина.

2) Предположим, что набор $\bar{a} = a_1, \ldots, a_k$ - решение исходной задачи, т.е. $\varphi|^{\bar{a}}_{\bar{x}}$ - истина. Для любого $i \in I^0_\varphi$ положим $\mu(v_0(i)) = term(i)|^{\bar{a}}_{\bar{x}}$. Если $(k, k') \in \vec{R}^{ит}_\varphi$, $k, k' \in I^{ит}_\varphi \setminus I^0_\varphi$, то $\mu(k') = \mu(k)$. Условия 1 и 2 тогда будут выполнены. Тем самым мы нашли значения входов и соответствующих им выходов, которые должны быть у процедур (вершин), участвующих в схеме, чтобы набор $\bar{x} = \bar{a}$ был выдан

как результат работы схемы. Поскольку по свойству реализаций каждый выход, удовлетворяющий заданному входу, на некотором шаге будет получен и число вершин в схеме конечно, то $\exists t^*$ такое, что набор $\bar{x} = \bar{a}$ будет выдан как результат работы схемы на шаге t^*. Теорема доказана.

2.4. Анализ алгоритма. В общем случае решение задачи может потребовать бесконечной памяти и вычислений.

Из результата Ю.В.Матиясевича [3] следует возможность указать конкретный многочлен $W(a, z_1, \ldots, z_n)$ с целыми коэффициентами такой, что не существует алгоритма, позволяющего по значению параметра a узнавать, разрешимо ли уравнение $W(a, z_1, \ldots, z_n) = 0$ относительно $z_1, \ldots, z_n$. Значит, не существует реализации, позволяющей по значению параметра a находить значения $z_1, \ldots, z_n$ такие, что $W(a, z_1, \ldots, z_n) = 0$. Поэтому единственным способом решения задачи $!z_1, \ldots, z_n ?a.W(a, z_1, \ldots, z_n) = 0$ остается генерация всех возможных наборов $z_1, \ldots, z_n$ и подстановка каждого из них в уравнение. При этом, если уравнение не разрешимо, то ни на каком шаге вычислений t мы не сможем узнать, будет ли получено решение на некотором шаге $t^* > t$ или его вообще не существует.

Описанный способ решения задачи дает возможность строить различные алгоритмы синтеза решений. Как уже отмечалось, это различие возникает при выборе конкретных реализаций предиката и функций при ориентации схемы вычислений. Подбирая соответствующим образом конкретные реализации, можно получать наилучшие в каком-либо смысле алгоритмы решения данной задачи.

Следует отметить, что для практического использования алгоритма на процедуры необходимо накладывать определенные ограничения, например, рассматривать их конечные аппроксимации.

3. Зависимость сложности вычислении от конструктивизации модели

Одним из способов уменьшения используемых вычислительных ресурсов является также выбор подходящей конструктивизации модели. В этом пункте рассматриваются вопросы, связанные с существованием конструктивизаций, в которых рассматриваемые функции являются полиномиально вычислимыми.

Для формального определения полиномиально вычислимых функций необходимо зафиксировать определенную модель процесса вычислений. Пусть это будет детерминированная одноленточная машина Тьюринга с множеством входных символов Σ. Программа M для такой машины Тьюринга называется полиномиальной, если существует полином p такой, что $\forall x \in \Sigma^*$: вычисление по программе M на входе x требует не более $p(|x|)$ шагов (здесь $|x|$ - количество символов в записи числа x).

ОПРЕДЕЛЕНИЕ 1. Функция $f: \Sigma^* \to \Sigma^*$ называется полиномиально вычислимой, если существует полиномиальная программа, вычисляющая функцию f.

Обозначим класс полиномиально вычислимых функций через P.

Как показано в [1], формальное определение класса P можно сформулировать в терминах программ для любой "реалистичной" модели ЭВМ, причем в результате получится один и тот же класс языков, поэтому нет необходимости вдаваться в детали машины Тьюринга, и можно обсуждать полиномиально вычислимые функции в машинно-независимом стиле (подразумевается, что для любой полиномиально вычислимой функции можно построить соответствующую полиномиальную программу).

Рассмотрим функцию $f: \mathbb{N} \xrightarrow[\text{на}]{1-1} \mathbb{N}$, определенную следующим образом:

$$f(x)=\begin{cases}2k, & \text{если } x=2^k;\\ 2(x-[\log_2 x])-1, & \text{если } x\ge 3,\ x\neq 2^k;\\ 1, & \text{если } x=0.\end{cases}$$

Чтобы найти $[\log_2 x]$, достаточно не более $c\cdot|x|$ раз (c - константа, зависящая от схемы кодирования) разделить число, не превосходящее x, пополам, т.е. выполнить $O(|x|^2)$ действий. Таким образом, f является полиномиально вычислимой.

Обратной к f будет функция

$$f^{-1}(y)=\begin{cases}2^k, & \text{если } y=2k;\\ k+2[\log_2 k]-[\log_2(k+[\log_2 k])], & \text{если } y=2k-1,\ k\ge 2;\\ 0, & \text{если } y=1.\end{cases}$$

Очевидно, она не полиномиально вычислима. (Иначе существовал бы полином p, такой что $\forall k \quad |2^k| \le p(|2k|)$, а этого не может быть.)

ОПРЕДЕЛЕНИЕ 2. Функция $t: A \to A$ (A - основное множество модели $\mathcal{A}$) называется полиномиально вычислимой в конструктивизации μ модели $\mathcal{A}$, если существует функция $p\in P$ такая, что $t(\mu(x)) = \mu(p(x))\ \forall x\in\mathbb{N}$.

Рассмотрим алгебраическую систему $\mathcal{A}=\langle\mathbb{N},f,f^{-1}\rangle$ и две ее конструктивизации α и α', определенные следующим образом:

$$\alpha x = x,$$

$$\alpha' x=\begin{cases}n, & \text{если } x=3n;\\ f(\alpha' n), & \text{если } x=3n+1;\\ f^{-1}(\alpha' n), & \text{если } x=3n+2.\end{cases}$$

В конструктивизации α, как показано выше, f - полиномиально вычислима, а f^{-1} - нет. В α' и f, и f^{-1} полиномиально вычислимы. Действительно, $f(\alpha' x) = \alpha'(3x+1)$, $f^{-1}(\alpha' x) = \alpha'(3x+2)$.

ОПРЕДЕЛЕНИЕ 3. Пусть $\mathcal{A}$ - модель сигнатуры σ. Отображение $\alpha: \mathbb{N} \xrightarrow{\text{на}} A$ называется полиномиальной конструктивизацией модели $\mathcal{A}$, если

а) $(\forall f \in \sigma)\ (\exists F \in P)\ (\forall x_1, \ldots, x_i \in \mathbb{N})\ f(\alpha x_1, \ldots, \alpha x_i) = \alpha F(x_1, \ldots, x_i)$;

б) $(\forall p \in \sigma)\ (\exists G \in P)\ (\forall x_1, \ldots, x_i \in \mathbb{N})\ p(\alpha x_1, \ldots, \alpha x_i) \Leftrightarrow G(x_1, \ldots, x_i) = 0$;

в) $(\exists R \in P)\ (\forall x, y \in \mathbb{N})\ \alpha x = \alpha y \Leftrightarrow R(x, y) = 0$.

УТВЕРЖДЕНИЕ 1. *Если* $f_1, \ldots, f_n$ *- одноместные рекурсивные функции, то алгебраическая система* $\mathcal{A} = \langle \mathbb{N}, f_1, \ldots, f_n \rangle$ *имеет конструктивизацию, в которой функции* $f_1, \ldots, f_n$ *полиномиально вычислимы.*

ДОКАЗАТЕЛЬСТВО. Положим

$$\alpha x = \begin{cases} k, & \text{если } x = (n+1)k; \\ f_i(\alpha k), & \text{если } x = (n+1)k+i. \end{cases}$$

Имеем $\forall i: f_i(\alpha x) = \alpha((n+1)x+i)$. Функция $\alpha: \mathbb{N} \to \mathbb{N}$ рекурсивна. Следовательно, α - требуемая конструктивизация.

ЗАМЕЧАНИЕ 1. Равенство в α не будет полиномиально вычислимым, поэтому α не полиномиальная конструктивизация.

ЗАМЕЧАНИЕ 2. Цензером и Реммелом [4] доказано существование полиномиальной конструктивизации для булевых алгебр и векторных пространств.

ОПРЕДЕЛЕНИЕ 4. Конструктивизации α и α' алгебраической системы $\mathcal{A}$ называются полиномиально эквивалентными, если су -

ществуют полиномиально вычислимые функции f и f' такие, что $\alpha \circ f = \alpha'$ и $\alpha' \circ f' = \alpha$.

УТВЕРЖДЕНИЕ 2. *Если конструктивизации* α *и* α' *полиномиально эквивалентны и функция* t *полиномиально вычислима в* α, *то* t *полиномиально вычислима и в* α'.

ДОКАЗАТЕЛЬСТВО. По условию существуют функции f, f', $p \in P$ такие, что $\alpha \circ f = \alpha'$, $\alpha' \circ f' = \alpha$ и $t(\alpha(x)) = \alpha(p(x))$. Следовательно, $t(\alpha'(x)) = t(\alpha \circ f(x)) = \alpha(p(f(x)) = \alpha'(f'(p(f(x))))$. Класс P замкнут относительно операции суперпозиции, значит, $f' \circ p \circ f \in P$.

В стандартной модели арифметики $\mathbb{N} = \langle \mathbb{N}, +, \cdot, s, \leq, 0 \rangle$ функция $g(x) = 2^x$ не полиномиально вычислима. Пусть α' - конструктивизация $\mathbb{N}$, в которой g полиномиальна. Тогда конструктивизации $\alpha = id_{\mathbb{N}}$ и α' не полиномиально эквивалентны.

ТЕОРЕМА 2. *Существует бесконечно много не полиномиально эквивалентных конструктивизаций арифметики.*

ДОКАЗАТЕЛЬСТВО. Рассмотрим следующие конструктивизации $\mathbb{N}$:

$$\nu_i x = \begin{cases} k, & \text{если } x = 2k; \\ p_i^{\nu_i k}, & \text{если } x = 2k+1, \quad i = 0,1,\ldots \end{cases}$$

Здесь p_i - i-е простое число. По построению для всех i: $p_i^{\nu_i k} = \nu_i(2k+1)$, т.е. $\lambda x(p_i^x)$ - полиномиально вычислимая в ν_i функция.

Покажем, что $\forall i,j$: если $i \neq j$, то $\lambda x(p_i^x)$ не полиномиально вычислима в ν_j. Предположим противное,

пусть $\exists f \in P: p_i^{\nu_j x} = \nu_j f(x)$. При $x = 2k, k \geq 1$ имеем

$$p_i^{\nu_j(2k)} = p_i^k = \nu_j f(2k).$$

Если $\exists k^*: f(2k^*) = 2n+1$ для некоторого n, то

$$p_i^k = \nu_j(2n+1) = p_j^{\nu_j n}.$$

Но такое равенство не может выполняться при $i \neq j, k \geq 1$. Значит, $\forall k \geq 1: f(2k)$ четное. Следовательно, $\nu_j f(2k) = f(2k)/2 \quad \forall k \geq 1$. Тогда $\lambda k(p_i^k) = f(2k)/2 \in P$. Противоречие. Используя утверждение 2, получаем, что $\forall i \neq j$ ν_i не полиномиально эквивалентна ν_j. Теорема доказана.

Автор благодарит С.С.Гончарова и Д.И.Свириденко за постановку задачи и внимание к работе.

Литература

1. ГЭРИ М., ДЖОНСОН Д. Вычислительные машины и трудноре - шаемые задачи. - М.: Мир, 1982.

2. КОТОВ С.В. К операционной интерпретации $\Sigma^{\pm}$-программ. Язык r-процедур //Системология и методологические проблемы информационно-логических систем. - Новосибирск, 1990.-Вып. 135: Вычислительные системы. -С. 131-159.

3. МАТИЯСЕВИЧ Ю.В. Диофантовы множества //Успехи мат.наук. - 1972. - Том 27, вып. 5. - С.185-222.

4. CENZER D., REMMEL J. Polynomial-time complexity of models //Association for Symbolic Logic, Annual neeting, Univ. of California. Los Angeles, January, 14-17, 1989.

Поступила в ред.-изд.отд.

27 июня 1991 года