П Р И К Л А Д Н А Я   Л О Г И К А
(Вычислительные системы)

1986 год Выпуск 116

УДК 519.681.3

КАК УЧИТЫВАТЬ СВОЙСТВА ОПЕРАЦИЙ ПРИ ГЛОБАЛЬНОМ АНАЛИЗЕ ПРОГРАММ?

В.К. Сабельфельд

Глобальный анализ программ используется для преобразования программ при их разработке, оптимизации и верификации. Единый подход к построению алгоритмов анализа различных программных свойств был предложен в [1]. В работах [3,5] был построен основанный на этом подходе алгоритм поиска программных инвариантов, представляющих собой множества соотношений равенства термов. В настоящей работе предлагается такое обобщение этого алгоритма, которое позволяет учитывать свойства базисных функций, заданные в форме правил переписывания термов, а также исключать из рассмотрения некоторые заведомо нереализуемые пути программы.

1. Задача глобального поиска и идея алгоритма

Пусть X, F, P - счетные множества переменных, функциональных и предикатных символов соответственно, а $d(f)$ означает количество аргументных позиций символа $f \in F \cup X \cup P$, причем $d(x) = 0$ для $x \in X$; $FALSE, TRUE \in P$, $d(FALSE) = d(TRUE) = 0$. Обозначим через $Fterm(X)$ множество функциональных термов над (X, F), а через $Pterm(X)$ множество предикатных термов над (X,F,P), т.е. термов вида $p(t_1, \dots, t_n)$, где $p \in P$, $n = d(p) \geq 0$, $\forall i\ (1 \geq i \geq n)\ t_i \in Fterm(X)$; $var(t)$ будет означать множество переменных, встречающихся в терме t, а $t[t_1/x_1, \dots, t_n/x_n]$ - терм, получающийся из терма t одновременной заменой всех вхождений переменной x_i на терм t_i $(i = 1, \dots, n)$, $Term(X) = Fterm(X) \cup Pterm(X)$.

Пусть задана полурешетка (свойств) с нулем $\mathcal{M} = \langle M, \wedge, 0 \rangle$, т.е. $\forall e_1, e_2, e_3 \in M$ $e_1 \wedge e_2 = e_2 \wedge e_1$, $e_1 \wedge (e_2 \wedge e_3) = (e_1 \wedge e_2) \wedge e_3$,

$e_1 \wedge e_1 = e_1$, $e_1 \wedge \mathbb{0} = \mathbb{0}$, а также семейство $\Phi_{s,i}: M \to M$ преобразователей свойств, описывающих изменение свойств в результате выполнения оператора присваивания или теста s с выходом по i-й дуге этой программной единицы. Порядок на элементах полурешетки согласован с операцией $\wedge$: $e_1 \le e_2 \Leftrightarrow e_1 \wedge e_2 = e_1$.

Пусть задана стандартная схема, в ней путь w и $e \in M$. Положим

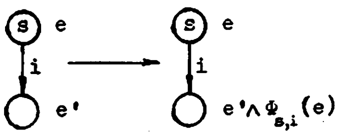
$$G(e,w) = \begin{cases} e \text{ ,если } w \text{ - пустой путь;} \\ \Phi_{s,i}(G(e,w')), \text{ если } w = w'va \text{ , где } v \text{ - вершина с оператором присваивания или тестом} \\ s, \quad a \text{ - } i\text{-я дуга, выходящая из вершины } v. \end{cases}$$

Для вершины v в схеме S положим $H(v) = \bigwedge_{w \in W} G(e_0, w)$, где W - множество всех путей в S, начинающихся с ее выделенной входной вершины и кончающихся дугой, ведущей к вершине v, а e_0 - свойство на входе схемы S.

Задача глобального анализа схемы S состоит в том, чтобы определить $H(v)$ для всех вершин схемы S. Известно [I,2] несколько различных подходов к решению этой задачи. Например (см. [I]), широко распространенный подход состоит в использовании следующего итеративного алгоритма $\mathfrak{A}$.

I. Фиксируем начальную разметку μ: $V \to M$, полагая $\mu(v) = e_0$ для входной вершины схемы и $\mu(v) = \mathbb{1}$ для всех других вершин (здесь $\mathbb{1}$ - искусственно добавляемое к M "парадоксальное" свойство, такое, что $\forall e \in M$ $e \wedge \mathbb{1} = \mathbb{1} \wedge e = e$).

2. Применяем правило изменения разметки



до стабилизации.

В предположении ограниченности полурешетки $\mathcal{M}$ (т.е. все ее строго убывающие цепи имеют конечную длину) и дистрибутивности всех преобразователей свойств (т.е. $\forall e, e' \in M$ $\Phi_{s,i}(e \wedge e') = \Phi_{s,i}(e) \wedge \Phi_{s,i}(e')$) этот алгоритм дает точное решение задачи глобального анализа (см. [I]). В предположении ограниченности полурешетки $\mathcal{M}$ и монотонности всех преобразователей свойств (т.е.

$\forall e, e' \in M \quad e \le e' \Rightarrow \Phi_{s,i}(e) \le \Phi_{s,i}(e'))$ этот алгоритм дает приближенное надежное решение μ такое, что $\mu(v) \le H(v)$ для всех вершин v (см. [2]).

2. Пометки для представления программных инвариантов

Понятие "пометки", вводимое ниже, используется для представления (бесконечных) множеств соотношений равенства термов. Это понятие учитывает симметричность, рефлексивность и транзитивность отношения равенства термов, замкнутость относительно подстановки термов; дает возможность простой формулировки операций над множествами равенств. Оно является развитием конструкций, описанных в [3,5,6].

Пометка $e = \langle V_e, \Psi_e, \Gamma_e \rangle$ – это функциональная сеть, содержащая конечное множество вершин $V_e = \{ v \}$, каждая из которых, в свою очередь, содержит конечное множество $v = \{ c \}$ элементов вершины. Всякому элементу c приписан некоторый символ $\Psi(c) \in X \cup F \cup P$, от него ведут $d(\Psi(c))$ пронумерованных дуг к другим вершинам пометки; $\Gamma(c,i)$ означает вершину, к которой ведет i-я дуга элемента c. На рисунках с примерами пометок вершины изображены овалами, а их элементы – прямоугольниками внутри овалов.

Для вершин v и элементов c пометки e определим множества термов $know_e(v)$ и $know_e(c)$:

$$know_e(c) = \begin{cases} \{g\}, \text{ если } \Psi_e(c) = g \ \&\ d(g) = 0; \\ \{f(t_1, \dots, t_n) : \forall i \ (1 \le i \le n) \quad t_i \in know_e(\Gamma_e(c,i))\}, \\ \qquad \text{если } \Psi_e(c) = f \ \&\ d(f) = n > 0; \end{cases}$$

$$know_e(v) = \bigcup_{c \in v} know_e(c);$$

$$as(e) = \bigcup_{v \in V_e} \{t = t' : t, t' \in know_e(v)\}.$$

Если интерпретировать пометку как состояние вычислений, то информацию, содержащуюся в ней, можно понимать следующим образом: в состоянии e все термы $t \in know_e(v)$ для $v \in V_e$ вычислены, их значения доступны и совпадают.

Если задано некоторое множество m соотношений равенства термов, то его замыканием $\bar{m}$ будем называть минимальное среди множеств m', обладающих следующими свойствами:

а) $m' \supset m$,

б) $\forall t, t' (t = t') \in m' \Rightarrow (t' = t) \in m'$,

в) $\forall t, t', t_1, t_2 \ (t = t') \in m' \ \& \ (t_1[t/x] = t_2) \in m' \Rightarrow$

$$\Rightarrow (t_1[t'/x] = t_2) \in m'.$$

В дальнейшем важную роль будет играть $eq(e) = \overline{as(e)}$ – множество равенств, кодируемых пометкой e. Для линейного представления пометок мы введем понятие базиса. Базисом пометки e называется всякое такое конечное множество соотношений равенства $basis(e)$, что

а) $\overline{basis(e)} = eq(e)$,

б) $\forall m \ (m \subset basis(e) \ \& \ basis(e) \neq m \Rightarrow \overline{m} \neq eq(e))$.

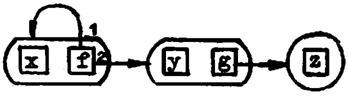


Рис.I

Очевидно, всякая пометка имеет хотя бы один базис и строить его по пометке можно за линейное время. Например, $\{ x = f(x,y), y = g(z), z = z \}$ – базис пометки, изображенной на рис.I.

Две пометки $e = \langle V, \Psi, \Gamma \rangle$ и $e' = \langle V', \Psi', \Gamma' \rangle$ называются изоморфными (или совпадающими, обозначение: $e = e'$), если существует взаимно-однозначное отображение $ID: V \to V'$, а также взаимно-однозначные отображения $id_v : v \to ID(v)$ для всех $v \in V$ такие, что

$$\forall v \in V \ \forall c \in v (\Psi'(id_v(c)) = \Psi(c) \ \& \ \forall i \Gamma'(id_v(c), i) = ID(\Gamma(c,i))).$$

Через 0 будем обозначать пометку с $V_0 = \emptyset$.

Вершину v (или ее элемент c) в пометке e назовем бесполезной, если $know_e(v) = \emptyset$ (или $know_e(c) = \emptyset$). Для поиска и удаления бесполезных элементов и вершин можно использовать следующий алгоритм.

I. Все элементы c такие, что $d(\Psi(c)) = 0$, объявим полезными.

2. Если вершина содержит хотя бы один полезный элемент, то объявим ее полезной.

3. Если все наследники элемента c полезны, то считаем полезным и элемент c.

4. Когда процесс применения правил I–3 стабилизируется, т.е. не будет давать новых полезных вершин и элементов, удаляем все те вершины и элементы, которые не стали полезными.

ЛЕММА I. Если пометка e' получается из пометки e удалением бесполезных вершин и элементов, то $as(e) = as(e')$.

Определим теперь бинарную операцию "$\times$" произведения пометок: для $e = \langle V, \Psi, \Gamma \rangle$ и $e' = \langle V', \Psi', \Gamma' \rangle$ положим $e \times e' = \langle V'', \Psi'', \Gamma'' \rangle$, где $V'' = V \times V'$ (декартово произведение множеств), $(v.v') = \{(c, c'): c \in v \ \& \ c' \in v' \ \& \ \Psi(c) = \Psi'(c')\}, \Psi''((c\ c')) = \Psi(c), \forall i \ \Gamma''((c,c'),i) = (\Gamma(c,i), \Gamma'(c',i))$.

Пример применения операции произведения пометок показан на рис.2.

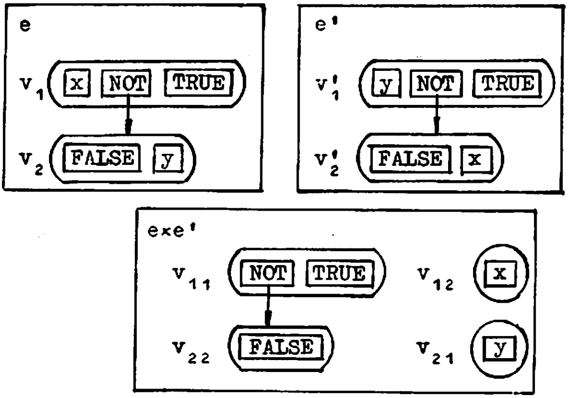


Рис. 2

ЛЕММА 2. Для всех вершин $v \in V_e$ и для всех вершин $v' \in V_{e'}$,

$$know_{e \times e'}((v, v')) = know_e(v) \cap know_{e'}(v').$$

Две вершины v_1, v_2 в некоторой пометке назовем связанными, если $know(v_1) \cap know(v_2) \neq \emptyset$. Транзитивное замыкание отношения связанности вершин пометки называется подобием. Близнецами назовем два любых различных элемента c, c' одной и той же вершины, для которых $\Psi(c) = \Psi(c') \ \& \ (d(\Psi(c)) = 0 \vee \forall i \ (1 \leq i \leq d(\Psi(c))) \ \Gamma(c,i) = \Gamma(c',i))$.

Операция объединения двух различных вершин v_1, v_2 в некоторой пометке состоит в следующем:

1) все элементы вершины v_2 (вместе с выходящими из них дугами) переносятся в вершину v_1 ;

2) все дуги, которые вели к v_2, направляются к v_1;

3) вершина v_2 удаляется.

Операция склеивания близнецов состоит в удалении одного из близнецов.

Операция сжатия пометки состоит в повторном применении операций объединения связанных вершин и склеивания близнецов до тех пор, пока в пометке не останется различных связанных вершин и близнецов.

ЛЕММА 3. Если пометка $e' = \mathrm{comp}(e)$ получается сжатием пометки e, то $eq(e) = eq(e')$ и $\mathrm{know}_{e'}(v_1) \cap \mathrm{know}_{e'}(v_2) = \emptyset$ для всех пар v_1, v_2 различных вершин в e'.

Заметим, что после объединения двух связанных вершин в новой пометке связанными могут стать вершины, которые не были связанными в исходной пометке. Пример применения операции сжатия показан на рис.3, звездочками помечены объединяемые вершины пометки. Ясно,

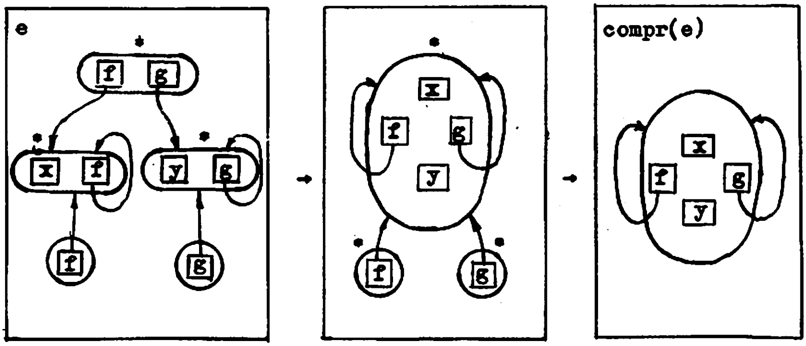


Рис. 3

однако, что этот процесс конечен, поскольку количество вершин монотонно убывает.

Пометка e называется противоречивой, если $\exists f,g \in F \cup P,\ f \neq g \,\&$ $\&\ d(f) = d(g) = 0\ \&\ (f = g) \in eq(e)$, и непротиворечивой в противном случае. Множество всех пометок пополним новым, искусственно добавляемым элементом с обозначением $\mathbf{1}$, для которого положим по определению $as(\mathbf{1}) = \{ (t = t') : t, t' \in Term(X) \}$. Приведенной будем называть пометку $\mathbf{1}$, а также всякую непротиворечивую пометку, не имеющую бесполезных вершин и элементов, а также различных подобных вершин и близнецов. Для приведения пометок будет использоваться операция fold, которая выполняется следующим образом. Прежде всего удаляются все бесполезные вершины и элементы исходной пометки e. К результату применяется операция сжатия пометки. Если при этом получается непротиворечивая пометка e', то полагаем

$\mathrm{fold}(e) = e'$, в противном случае $\mathrm{fold}(e) = \mathbb{1}$. Мы полагаем также $\mathrm{fold}(\mathbb{1}) = \mathbb{1}$ по определению.

ЛЕММА 4. Пометка e противоречива тогда и только тогда, когда для $e' = \mathrm{compr}(e)$ существуют вершина $v \in V_{e'}$ и ее элементы c, c' такие, что $\Psi(c) \neq \Psi(c')$, $\Psi(c)$, $\Psi(c') \in F \cup P$, $d(\Psi(c)) = d(\Psi(c')) = 0$.

ЛЕММА 5. Для непротиворечивых пометок e

$$e \neq \mathbb{1} \rightarrow eq(\mathrm{fold}(e)) = eq(e) .$$

ЛЕММА 6. Для приведенных пометок e, e'

$$e = e' \leftrightarrow eq(e) = eq(e') .$$

Обозначим через M множество всех приведенных пометок. На M введем бинарную операцию пересечения $\wedge$, полагая $e \wedge \mathbb{1} = \mathbb{1} \wedge e = e$ для всех $e \in M$ и $e \wedge e' = \mathrm{fold}(e \times e')$ для всех $e, e' \in M$ при $e \neq \mathbb{1}$ и $e' \neq \mathbb{1}$.

ТЕОРЕМА I. Для всех приведенных пометок $e, e', e'' \in M$

$$e \wedge e' = e' \wedge e, \quad e \wedge (e' \wedge e'') = (e \wedge e') \wedge e'' ,$$

$$e \wedge e = e, \quad e \wedge \mathbf{0} = \mathbf{0}, \quad eq(e \wedge e') = eq(e) \cap eq(e') .$$

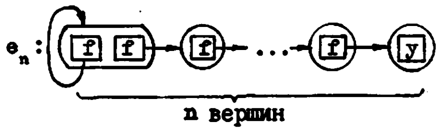


Рис. 4

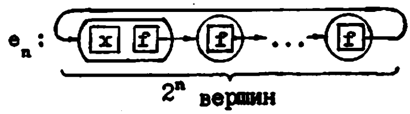


Рис. 5

Таким образом, $\mathcal{M} = \langle M, \wedge, \mathbf{0} \rangle$ - полурешетка с нулем. Однако, как видно из примеров на рис. 4 и 5, эта полурешетка не является ограниченной (на рис. 4 и 5 изображены примеры бесконечных убывающих цепей пометок из M , $\forall n (n \geq 2)\ e_n \wedge e_{n+1} = e_{n+1}$).

Будем говорить, что вершина v в пометке e "знает" терм t, если $t \in \mathrm{know}_e(v)$. Нетрудно видеть, что нахождение вершины v, "знающей" терм t в (не обязательно приведенной) пометке e, или распознавание факта, что такой вершины в e нет, можно выполнять за время $O(N \log N)$, где N - сумма размеров пометки e и терма t . В приведенной пометке e имеется не более одной вершины, "знающей" t .

Определим операцию add, которая по пометке e и терму t дает пометку add(e,t), в которой (при $e \neq \mathbb{1}$) есть вершина, "знающая" t. Если $e=\mathbb{1}$ или в e уже есть вершина, "знающая" t, то add(e,t)= =e. В противном случае пусть $t = f(t_1,\dots,t_n)$, где $n \geq 0, f \in X \cup F \cup P$. Построим тогда пометки $e_0=e$, $e_i=add(e_{i-1},t_i)$ для $i=1,\dots,n$ и получим add(e,t) из e_n добавлением новой вершины v с единственным элементом c, для которого $\Psi(c)=f$ и $\forall i(1 \leq i \leq n)\ \Gamma(c,i)=v_i$, где v_i - вершина из e_n, "знающая" терм t_i.

Для $t_1,t_1',\dots,t_n,t_n' \in Term(X)$ пометка $\langle t_1 \uparrow t_1',\dots,t_n \uparrow t_n' \rangle e$ строится по пометке e следующим образом: пусть $e_0 = e$, $e_i = add(add(e_{i-1},t_i), t_i')$ для $i = 1,\dots,n$ и пусть v_i, v_i' - вершины пометки e_n, "знающие" термы t_i и t_i' соответственно. Пусть, далее, пометка e' получается из пометки e_n попарным объединением вершин v_i и v_i' для всех i таких, что $v_i \neq v_i'$. Положим тогда $\langle t_1 \uparrow t_1',\dots, t_n \uparrow t_n' \rangle e = fold(e')$.

Для списка переменных $y_1,\dots,y_n$ пометка $\langle y_1,\dots,y_n - \rangle e$ строится по пометке e следующим образом: пусть пометка e' получается из e удалением всех таких элементов c, что $\Psi(c) \in \{y_1,\dots,y_n\}$. Положим тогда $\langle y_1,\dots, y_n - \rangle e = fold(e')$.

ЛЕММА 7.

$$\langle y_1,\dots, y_n - \rangle e \wedge e' = \langle y_1,\dots, y_n - \rangle e \wedge \langle y_1,\dots,y_n - \rangle e'$$

$$\langle t_1 \uparrow t_1',\dots,t_n \uparrow t_n' \rangle\, e \wedge e' \leq \langle t_1 \uparrow t_1',\dots,t_n \uparrow t_n' \rangle e \wedge \langle t_1 \uparrow t_1',\dots,t_n \uparrow t_n' \rangle\, e'.$$

ДОКАЗАТЕЛЬСТВО вытекает из того, что для любых множеств равенств m, m',m" и для любых $t_1,t_2 \in Term(X)$

$$(m \cap m') \setminus m'' = (m \setminus m'') \cap (m' \setminus m''),$$

$$\overline{(\overline{m} \cap \overline{m'}) \cup \{t_1 = t_2\}} \subset \overline{(\overline{m} \cup \{t_1 = t_2\})} \cap \overline{(\overline{m'} \cup \{t_1 = t_2\})}.$$

Приведем контрпример, опровергающий дистрибутивность операции $\langle t_1 \uparrow t_1',\dots, t_n \uparrow t_n' \rangle$. Пусть e_1,e_2 - пометки с базисами $basis(e_1) = \{x = y\}$ и $basis(e_2) = \{x = x,\ y = y,\ p(y) = TRUE\}$ соответственно. Тогда $basis(\langle p(x) \uparrow TRUE \rangle e_1 \wedge e_2) = \{x = x,\ y = y,\ p(x) = TRUE\}$, но $basis(\langle p(x) \uparrow TRUE \rangle e_1 \wedge \langle p(x) \uparrow TRUE \rangle e_2) = \{x = x,\ y = y,\ p(x) = TRUE,\ p(y) = TRUE\}$, поэтому $\langle p(x) \uparrow TRUE \rangle e_1 \wedge e_2 \neq \langle p(x) \uparrow TRUE \rangle e_1 \wedge \langle p(x) \uparrow TRUE \rangle e_2$.

3. Правила переписывания для пометок

Зафиксируем множество $U = \{u_i\}$, $U \cap (X \cup F \cup P) = \emptyset$, элементы которого будем называть параметрами. Правилом (переписывания пометок) будем называть всякое выражение вида

$$\text{IF } t_1 = t_1' \,\&\, t_2 = t_2' \,\&\, \ldots \,\&\, t_n = t_n' \text{ THEN } t = t' \text{ FI}, \qquad (1)$$

где $t_1, t_1', \ldots, t_n, t_n', t, t' \in \mathrm{Term}(U)$, $n \geq 0$. Здесь $t_1 = t_1' \,\&\, \ldots \,\&\, t_n = t_n'$ – условие правила, $t = t'$ – его уточнение, t – левая, а t' – правая части уточнения. При $n=0$ для правила IF THEN $t=t'$FI будем использовать краткую форму правила $t = t'$. Мы требуем, чтобы всякий параметр, встречающийся в правой части уточнения, встречался также либо в левой части уточнения, либо в условии этого правила.

Приведем несколько примеров правил:

$u + u = 2 * u$, $u + 0 = u$, IF EQ(u,0) = FALSE THEN $1/(1/u) = u$ FI,

$8 + 13 = 21$, $u_1 + u_2 = u_2 + u_1$, NOT(NOT(u)) = u, $u - u = 0$,

u OR FALSE = u, IF EQ(u_1,u_2) = TRUE THEN $u_1 = u_2$ FI,

IF $u_1 + u_2 = u_1$ THEN $u_2 = 0$ FI.

Для $e \in M$ ($e \neq \mathbb{1}$) и конечного множества Y ($Y \subset U$) параметров через MAPS(Y,e) будем обозначать множество всех отображений из Y в V_e; var(R) будет означать множество параметров, встречающихся в правиле R. Если $m \in$ MAPS(var(R),e), то e^m будет означать пометку, получающуюся из пометки $e \in M$ добавлением нового элемента c, $\Psi(c) = u$, к вершине $m(u)$ для всех $u \in$ var(R).

Пусть R – правило вида (I), $e \in M$, $e \neq \mathbb{1}$, а $m \in$ MAPS(var(R),e) и var(R) = $\{y_1, \ldots, y_k\}$. Правило R называется m-применимым к пометке e, если $\langle t_1 \uparrow t_1', \ldots, t_n \uparrow t_n', t \uparrow t' \rangle e^m = e^m$. В этом случае его m-применение к e состоит в замене пометки e на пометку fold($\langle y_1, \ldots, y_k - \rangle \langle t \uparrow t' \rangle e^m$). Правило R применимо к e, если оно m-применимо к e с некоторым $m \in$ MAPS(var(R),e); в этом случае его применение к e состоит в m-применении R к e.

Пусть фиксировано некоторое конечное множество CS правил переписывания пометок (такие множества будем называть системами пополнения). Будем говорит, что пометка $e \in M$ редуцируется к пометке e' (обозначение: $e \to e'$), если e' получается из e (возможно, пустой) последовательностью применений правил из CS. Пометка $e \in M$ называется нередуцируемой, если применение всякого правила из CS не меняет e.

ЛЕММА 8. Отношение редукции пометок является конфлюентным [4], т.е.

$\forall e,e_1,e_2 \in M\ \exists e_3 \in M:\quad e \to e_1 \,\&\, e \to e_2 \Rightarrow e_1 \to e_3 \,\&\, e_2 \to e_3$.

ДОКАЗАТЕЛЬСТВО вытекает из того факта, что всякое применение правила не нарушает условий применимости других (или того же самого) правил.

Нормальной цепью с началом e_0 будем называть всякую максимальную по длине последовательность пометок $\{e_i\}, e_i \in M$, такую, что $\forall i(i \geq 0)\ e_i \to e_{i+1} \,\&\, e_i \neq e_{i+1}$. Система пополнения CS и индуцируемое ею отношение редукции пометок называются нётеровыми, если все нормальные цепи конечны; допустимыми, если для всякой пометки $e \in M$ существует конечная нормальная цепь с началом e, и недопустимыми в противном случае.

Например,

$\{f(u_1,u_2) = f(u_2,u_1)\}$, $\{g(h,u) = u\}$, $\{f(f(u)) = f(u)\}$, $\{f(f(u)) = u\}$

и

$\{EQ(u,u) = TRUE,\ EQ(u_1,u_2) = EQ(u_2,u_1),\ IF\ EQ(u_1,u_2) = TRUE\ \&$
$\&EQ(u_2,u_3) = TRUE\ THEN\ EQ(u_1,u_3) = TRUE\ FI,\ IF\ EQ\ (u_1,u_2) =$
$= TRUE\ THEN\ u_1 = u_2\ FI\}$ - нётеровы системы,

$\{f(u) = f(g(u)),\ g(u) = u\}$

и

$\{f(u_1,u_2) = f(u_2,u_1),\ f(f(u_1,u_2),u_3) = f(u_1,f(u_2,u_3)),\ f(u,h) = u,$
$IF\ f(u_1,u_2) = h\ THEN\ u_1 = h\ FI,\ IF\ f(u_1,u_2) = u_1\ THEN\ u_2 = h\ FI\}$ -
допустимые, а

$\{f(u) = f(g(u))\}$, $\{f(f(u_1,u_2),u_3) = f(u_1,f(u_2,u_3))\}$ и $\{f(g(u)) =$
$= g(f(u))\}$ - недопустимые системы.

Несмотря на то, что по форме правила переписывания пометок мало чем отличаются от правил переписывания термов [4], их свойства существенно различаются. Например, система $\{f(u_1,u_2)=f(u_2,u_1)\}$ является нётеровой системой переписывания пометок, и не нётеровой в качестве системы переписывания термов. Наоборот, $\{f(g(u)) = g(f(u))\}$ - нётерова система переписывания термов, а как система переписывания пометок она недопустима. Тем не менее свойства нётеровости и допустимости систем пополнения пометок оказываются алгоритмически неразрешимыми. Мы приведем здесь простые достаточные условия нётеровости системы пополнения пометок.

ЛЕММА 9. Система пополнения нётерова, если всякое ее правило R вида (I) таково, что либо t' - параметр, либо $t' = f(tt_1, \ldots, tt_k)$, $k \geq 0$, где для всякого i $(1 \leq i \leq k)$ терм tt_i либо не содержит вхождений параметров, либо встречается в качестве подтерма в левой части уточнения или в условии правила R.

ТЕОРЕМА 2. При допустимом отношении редукции для любой пометки $e \in M$ существует единственная нередуцируемая пометка norm(e) такая, что $e \to norm(e)$; пометку norm(e) можно эффективно строить по e.

В дальнейшем изложении мы ограничимся рассмотрением допустимых систем пополнения. Пометку norm(e) будем называть нормальной формой пометки e. Пример нормализации пометки для случая CS = $= \{f(f(u)) = f(u)\ f(f(u)) = u\}$ показан на рис.6.

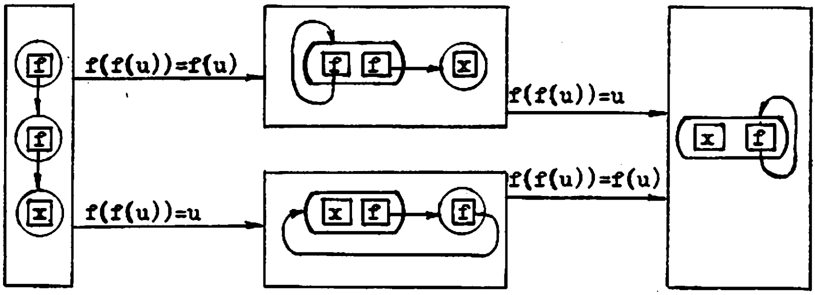


Рис. 6

ЛЕММА 10. Если $e \to e'$, то $e' \geq e$ и $norm(e) \geq e$.

4. Семейство алгоритмов глобального анализа

Рассмотрим класс схем программ, в которых имеются только распознаватели (с тестами $\pi \in Pterm(X)$ и двумя выходящими дугами) и операторы (совместного) присваивания $(x_1, \ldots, x_n := t_1, \ldots, t_n)$ с одним выходом, где $t_1, \ldots, t_n \in Fterm(X)$, а $x_1, \ldots, x_n$ – попарно различные переменные из X. Чтобы сформулировать алгоритм, решающий описанную в разделе I задачу глобального анализа и использующий полурешетку приведенных пометок в качестве полурешетки анализируемых свойств, достаточно описать преобразователи пометок для присваивания и распознавателя.

Для присваивания $s=(x_1, \ldots, x_n := t_1, \ldots, t_n)$ положим $\Phi^{CS}_{s,1}(e) =$
$= norm(\langle y_1, \ldots, y_n \rightarrow\rangle \langle x_1 \uparrow t'_1, \ldots, x_n \uparrow t'_n\rangle \langle x_1, \ldots, x_n \rightarrow\rangle \langle y_1 \uparrow x_1, \ldots, y_n \uparrow x_n\rangle\ e)$,
где $y_1, \ldots, y_n$ – новые попарно различные переменные, не встречающиеся в пометке e и операторе s, а $t'_i = t_i[y_1/x_1, \ldots, y_n/x_n]$ для $i = 1, \ldots, n$.

Для теста $\pi \in Pterm(X)$ положим:

$\Phi^{CS}_{\pi,1}(e) = norm(\langle \pi \uparrow TRUE \rangle\ e)$ – для выхода с результатом TRUE ;

$\Phi^{CS}_{\pi,2}(e) = norm(\langle \pi \uparrow FALSE \rangle\ e)$ – для выхода с результатом FALSE .

В силу леммы 7 все эти преобразователи пометок являются монотонными (но, вообще говоря, не дистрибутивными).

Таким образом, если в описанном в разделе I алгоритме в качестве полурешетки свойств использовать полурешетку нормальных форм приведенных пометок, а в качестве преобразователей свойств – определенные выше преобразователи пометок, то мы получим семейство алгоритмов $\mathfrak{A}(CS)$ глобального анализа схем программ с частично интерпретированными символами базисных функций. Эта частичная интерпретация определяется выбранной системой пополнения CS .

ТЕОРЕМА 3. *Пусть для схемы S и допустимой системы пополнения CS существует класс пометок K, $K \subset M$, удовлетворяющий следующим условиям:*

1) $\Phi^{CS}_{s,i}(e) \in K$ для всех операторов и тестов s из S, для всех номеров i выходных дуг s и для всех $e \in K$;

2) $\langle K, \wedge, 0 \rangle$ – ограниченная полурешетка.

Тогда алгоритм α(CS) заканчивается на S при любой пометке $e \in K$ на входе схемы и дает надежное решение задачи глобального анализа.

Пусть AM означает класс ациклических приведенных пометок, т.е. таких пометок $e = \langle V, \Psi, \Gamma \rangle$, что

$$\neg(\exists v_1,\ldots,v_n \in V \forall i (1 \le i \le n) \exists c_i \in v_i \exists j_i \Gamma(c_i, j_i) = v_{(i+1) \bmod n}).$$

Пусть, далее, ACS означает класс таких систем пополнения, отношения редукции которых сохраняют свойство ацикличности пометок. Тогда справедлива следующая

ТЕОРЕМА 4. Для CS $\in$ ACS алгоритм α(CS) заканчивается на любой схеме с ациклической пометкой на входе.

ДОКАЗАТЕЛЬСТВО следует из теоремы 3, поскольку полурешетка ациклических пометок удовлетворяет условию ограниченности [5], а при CS $\in$ ACS все преобразователи пометок также сохраняют ацикличность пометок.

Заключение

Описанный подход можно распространить на схемы программ с рекурсивными процедурами. Предлагаемый алгоритм может быть использован и для верификации программ. Например, если мы хотим доказать правильность приведенной ниже программы POWER возведения в целую степень со свойством $\{a = 1 * f(x,n)\}$ на входе, то достаточно применить к ней алгоритм α(CS) со следующей системой пополнения CS, описывающей свойства операций f и $*$:

CS= $\{u_1 * (u_2 * u_3) = (u_1 * u_2) * u_3,\ u * 1 = u,\ f(u,0) = 1,$

IF EQ(n,0)=FALSE & ODD(n)=TRUE THEN f(x,n)=x* f(x,n-1) FI,

IF EQ(n,0)=FALSE & ODD(n)=FALSE THEN f(x,n)=f(x*x,n div 2) FI,

IF EQ(u_1,u_2)=TRUE THEN u_1= u_2 FI}.

Алгоритм строит следующую стационарную разметку программы:

```
POWER: {a=1* f(x,n)}
y:= 1; {y=1,a=y*f(x,n)}
1: {a=y*f(x,n)}
IF EQ(n,0) THEN {EQ(n,0)=TRUE,n=0,a=y*f(x,n),f(x,n)=1,a=y}
```

```
GOTO 2; {EQ(n,0)=FALSE, a=y*f(x,n)}
IF ODD(n) THEN {EQ(n,0)=FALSE,ODD(n)=TRUE,a=y*f(x,n),
                f(x,n)=x*f(x,n-1),a=(y*x)*f(x,n-1)}
   BEGIN
   y:= y*x; {EQ(n,0)=FALSE,ODD(n)=TRUE,a=y*f(x,n-1),
            f(x,n)=x*f(x,n-1)}
   n:= n-1 {a=y*f(x,n)}
   END ELSE
   BEGIN {EQ(n,0)=FALSE,ODD(n)=FALSE,a=y*f(x,n),
         f(x,n)=f(x*x,n div 2)}
   x:= x*x; {EQ(n,0)=FALSE,ODD(n)=FALSE,a=y*f(x,n div 2)}
   n:= n div 2 {a=y*f(x,n)}
GOTO 1; {1}
2: {EQ(n,0)=TRUE,n=0,a=y*f(x,n),f(x n)=1,a=y}
STOP
```

При записи базисов пометок здесь были опущены тривиальные равенства t = t .

Литература

I. KILDALL G.A. A unified approach to global program optimization.- In: Conf.Records on ACM Symp. on Principles of Progr. Lang., Boston, MA, October 1-3,1973,p.194-206.

2. KAM J.B., ULLMAN J.D. Monotone data flow analysis frameworks. - Acta Informatica,1977,v.7, fasc 3, p.305-318.

3. SABELFELD V.K. The logic-termal equivalence is polynomial-time decidable.- IPL,1980,v.10,N 2,p.57-62.

4. HUET G., OPPEN D.C. Equations and rewrite rules: a survey.- In: Formal Languages: Perspectives and Open Problems (Book R.,ed.). Academic Press,1980,p.349-405.

5. САБЕЛЬФЕЛЬД В.К. Полиномиальная оценка сложности распознавания логико-термальной эквивалентности. -Докл. АН СССР, 1979, т. 249, № 4, с.793-796.

6. БУЛЬОНКОВ М.А. Итеративные алгоритмы разметки в трансформационной машине. -В кн.: Программное обеспечение задач информатики.Новосибирск, 1982, с. 38-52.

Поступила в ред.-изд.отд.
21 мая 1986 года