О некорректности ветвящейся программы и цикла

С. А. Жуков

***Аннотация* — В рамках хоаровского подхода к программной правильности изучается некорректность определенного вида программ с ветвлениями, они могут моделировать таблицы решений с ограниченным входом, и цикла, не имеющего подциклов. Изучаются связи между структурой кода такой программы с ветвлениями и постусловиями, приводящие к некорректности. Даны выполнимые постусловия, при которых любые указанные программы с ветвлениями некорректны. Определены достаточные свойства постусловия, при которых цикл указанного выше вида будет некорректным, а также логическое выражение, связывающее инварианты цикла с постусловием в случае некорректности цикла. Некорректность программы можно выразить как ложность определенного утверждения, описывающего свойства постусловия программы в случае, когда функция программы задается рекурсивно. А именно, был выделен класс рекурсивных определений D, так что некорректность программы вычисления функции, заданной D, эквивалентна нарушению некоторых свойств, которые описываются логическим утверждением, построенным по D. Приведен пример определений двух взаимно рекурсивных функций, некорректность программ вычисления которых анализируется через построение логических утверждений, описывающих свойства этих функций. Отрицание построенных утверждений выражает некорректность соответствующих программ.**

***Ключевые слова*— ветвящаяся программа, диофантово представление экспоненты, инвариант цикла, логическая спецификация постусловия, негативное высказывание, тройка Хоара, элементарные по Кальмару функции.**

I. Введение

Созидательная природа программирования оправдывает важность позитивных утверждений о корректности программ, в то время как негативные утверждения фиксируют проблемы в коде или в спецификациях, или и в том, и в другом. Обычная математическая установка – доказать утверждение, либо найти для него контрпример применяется при модельной проверке [1] и нет причин ее неприменения при дедуктивной верификации. Однако негативные утверждения вносят определенные трудности в попытках формального описания или анализа математических объектов. Известно, что существуют негативные недоказуемые утверждения, например, в формальной арифметике [2]. Другой пример – множество предваренных логических формул с префиксом ∃∀∃∀, бескванторная часть которых является хорновским предложением, в нем все или почти все дизъюнкты являются негативными утверждениями, является редукционным классом для неразрешимой проблемы логической выполнимости [3]. Приведенные выше соображения и примеры показывают необходимость изучения свойств негативных утверждений, в частности, относящихся к области дедуктивной верификации программ.

Статья получена 10 февраля 2026

Жуков С. А. – кандидат физико-математических наук, доцент кафедры вычислительных технологий Кубанского госуниверситета, Краснодар, Россия (e-mail: szhucov@fpm.kubsu.ru)

В статье проводится логический анализ ложности утверждения <φ>P<ψ> о тотальной правильности программы P относительно ее спецификации <φ, ψ>, где φ – предусловие, ψ – постусловие для P, так что ψ описывает свойства, которым должен удовлетворять результат P, формируемый в переменной r. В рамках этого анализа формулируются достаточные условия ложности <φ>P<ψ> и ее следствия. Предполагается, что анализируемые программы P останавливаются на любых входах для $\bar{x}$, удовлетворяющих предусловию φ($\bar{x}$), где $\bar{x}$ – набор входных переменных P. С учетом этого, утверждение о тотальной правильности <φ>P<ψ> равносильно утверждению о частичной правильности {φ}P{ψ}, и в дальнейшем используется последняя нотация.

Объектами исследования являются программы двух видов. Первый – это программы if ($Cond_1$) P_1 else if($Cond_2$) P_2 else …, являющиеся композициями условных операторов, где $Cond_i$ – логическое выражение, образованное из логических переменных и операций: ¬, &, ∨; P_i – последовательность операторов присваивания. Обычные таблицы решений с ограниченным входом преобразуются в такие программы [4, 5]. Формат рассматриваемых таблиц решений и соответствующих им программ уточняются далее. Второй вид – циклы while Cond do P. Предметной областью является Z – множество целых чисел с обычными арифметическими операциями и сравнениями. Следующие символы обозначают: ′=′ – отношение равенства значений, ′≡′ – отношение быть определенным, т.е. объект слева от символа определяется выражением справа от символа, ′⇔′ – отношение логической эквивалентности.

II. Некорректность ветвящихся программ

Таблицу решений с ограниченным входом [4, 5], известную форму наглядного представления ветвящегося алгоритма, можно задать следующим образом. Условия представляются логическими переменными u_i, 1≤ i ≤ n, где n – количество условий.

Пусть выбору j-го правила, $2 \le j \le m$, $m \le 2^n$, отвечает комбинация значений $v_1, v_2, \ldots, v_n$, $v_i \in${false, true, –}, так, что '–' обозначает независимость выбора правила от v_i. Выбор этого правила запускает выполнение P_j – последовательности операторов, которые определяют то, что называется неветвящимся вычислением или "straight-line program" [6]. Указанное действие программируется как

if(z_1& z_2 & …& z_n) P_j, где $z_i = \begin{cases} u_i, \text{если } v_i = true \\ \neg u_i, \text{если } v_i = false \\ true, \text{если } v_i = \text{'–'} \end{cases}$

или в сокращенном виде $Cond_j \to P_j$, где $Cond_j = z_{i_1}$& z_{i_2}& …& z_{i_k} и никакое z_{i_p} не равно true. Код P_j – это конечный список операторов присваивания t_i:= $exp_i(w_1, \ldots, w_{k_i})$, $1 \le i \le p_j$, где t_i, $1 \le i < p_j$, являются различными вспомогательными переменными, отличными от входных переменных, однако t_{p_j} совпадает с результирующей переменной r. Переменная w_l, $1 \le l \le k_i$, – это либо некоторая t_k, k < i, либо одна из входных переменных набора $\bar{x}$. Выражение $exp_i(w_1, \ldots, w_{k_i})$ формируется из переменных $w_1, \ldots, w_{k_i}$, операций набора {+, –, *}, который для этого вида программ является обычным [6, 7], а также подходящих констант. Эффект выполнения P_j можно выразить "сверткой" последовательности присваиваний t_i := $exp_i(w_1, \ldots, w_{k_i})$ в одно результирующее выражение $res_j(\bar{x})$, определяющее результат выполнения P_j на входе $\bar{x}$. Выражение $res_j(\bar{x})$ формируется по выражению $exp_{p_j}(w_1, \ldots, w_{k_{p_j}})$ – правой части последнего присваивания для r в P_j, в котором все вхождения вспомогательных переменных w_k заменены выражениями – правыми частям присваиваний для w_k. Например, если $\bar{x}$ = <x_1,x_2> и $P_j \equiv$ u := 2*$x_1 + 1$; $v := u^2$+ 1; w := x_2– 1; r := v * w, то $res_j(\bar{x}) \equiv$ ((2*$x_1 + 1$)^2 + 1)*(x_2– 1). Вся таблица решений программируется как

$Cond_1 \to P_1 | Cond_2 \to P_2 | \ldots | Cond_m \to P_m$, (1)

где | является сокращением для else.

Так, что $Cond_1 \to P_1 | Cond_2 \to P_2 | Cond_3 \to P_3$ кодирует if($Cond_1$)P_1elseif($Cond_2$)P_2elseif ($Cond_3$)P_3. Если $Cond_i$ при некотором j является пустым условием, то в программе оно заменяется на true. Все ветви вида true → P_i, при их наличии, группируются вместе в конце программы так, что код P_i предшествует коду P_k, если j < k. Условия $Cond_j$ таковы, что выполнено $\varphi \supset Cond_1 \vee Cond_2 \vee \ldots \vee Cond_m$ иначе таблица решений не является полной относительно φ. В таблицах решений и, соответственно, в программах вида (1) логические переменные u_i могут быть входными данными или обозначать сравнения $x_i \otimes x_j$ или $x_i \otimes$ c, где ⊗ обозначает сравнение: =, ≠, >, ≥, <, ≤ ; x_i, x_j – элементы входа $\bar{x}$, c – константа. Естественным обобщением рассмотренного формата таблицы решений является использование в разделе условий вычислимых предикатов $\pi_i(\bar{x})$ вместо логических переменных u_i.

Возможность задания рассмотренных таблиц решений программами вида (1) показывает, что этот вид программ достаточно представителен. Кроме того, деревья решений, используемые в задачах классификации и анализе больших данных, легко представимы таблицами решений [5], а следовательно и программами вида (1).

Относительно заданной спецификации <φ, ψ> утверждение

$\neg(\{\varphi\}Cond_1 \to P_1 | Cond_2 \to P_2 | \ldots | Cond_m \to P_m\{\psi\})$

описывает некорректность ветвящейся программы рассмотренного вида.

Используя известные [8] правила вывода

$\frac{\{\varphi\}P\{\tau\},\ \{\tau\}Q\{\psi\}}{\{\varphi\}P;Q\{\psi\}}$ и $\frac{\{\varphi\ \&Cond\}P\{\psi\},\ \{\varphi\ \&\neg Cond\}Q\{\psi\}}{\{\varphi\}if(Cond)P\ else\ Q\{\psi\}}$,

можно обосновать производное правило вывода

$$\frac{\{\varphi\ \&\ Cond_1\}P_1\{\psi\}, \ldots, \{\varphi\ \&\neg Cond_1\ \&\neg Cond_2\ \&\ \ldots\ \&\ Cond_m\}P_m\{\psi\}}{\{\varphi\}Cond_1 \to P_1 | Cond_2 \to P_2 | \ldots | Cond_m \to P_m\{\psi\}} \quad (2)$$

Теорема 1. $\neg(\{\varphi\}Cond_1 \to P_1 | Cond_2 \to P_2 | \ldots | Cond_m \to P_m\{\psi\}) \Leftrightarrow \neg(\{\varphi\ \&\ Cond_1\}P_1\{\psi\}) \vee \neg(\{\varphi\ \&\ \neg Cond_1\ \&\ Cond_2\}P_2\{\psi\}) \vee \ldots \vee \neg(\{\varphi\ \&\ \neg Cond_1\ \&\ \ldots\ \&\ \neg Cond_{m-1}\ \&\ Cond_m\}P_m\{\psi\})$.

Доказательство необходимости проводится применением закона о контрапозиции к заключению и посылке правила (2) и закона де Моргана. Обратно, если выполнено $\neg(\{\varphi\ \&\ Cond_1\}P_1\{\psi\}) \vee \neg(\{\varphi\ \&\ \neg Cond_1\ \&\ Cond_2\}P_2\{\psi\}) \vee \ldots \vee \neg(\{\varphi\ \&\ \neg Cond_1\ \&\ \ldots\ \&\ \neg Cond_{m-1}\ \&\ Cond_m\}P_m\{\psi\})$, то существует j такое, что $\neg(\{\varphi\ \&\ \neg Cond_1\ \&\ \ldots\ \&\ \neg\ Cond_{j-1}\ \&\ Cond_j\}P_j\{\psi\})$. Последнее утверждение означает наличие такого $\bar{x}_0$, что выполнено $\varphi(\bar{x}_0)\ \&\ \neg Cond_1(\bar{x}_0)\ \&\ \ldots\ \&\ \neg\ Cond_{j-1}(\bar{x}_0)\ \&\ Cond_j(\bar{x}_0)\ \&\ \neg\psi(\bar{x}_0,\ P_j(\bar{x}_0))$. Иными словами, в программе $Cond_1 \to P_1 | Cond_2 \to P_2 | \ldots | Cond_m \to P_m$ на входе $\bar{x}_0$ будет выбрана и выполнена ветвь $Cond_j \to P_j$ так, что $\neg\psi(\bar{x}_0,\ P_j(\bar{x}_0))$. Следовательно, утверждение $\{\varphi\}Cond_1 \to P_1 | Cond_2 \to P_2 | \ldots | Cond_m \to P_m\{\psi\}$ ложно, поскольку в противном случае на входе $\bar{x}_0$ в этой программе выбирается ветвь $Cond_j \to P_j$, по завершении которой выполнилось бы $\psi(\bar{x}_0, P_j(\bar{x}_0))$.

Пример 1. Рассмотрим проявление некорректности программы вида (1), которая реализует следующую таблицу решений, использующую входные логические переменные x, y, z и результирующую переменную r

правила	1	2	3	4	5	6
x	0	–	0	1	–	1
y	0	0	–	1	1	–
z	–	0	0	–	1	1
r :=	0	0	0	1	1	1

Приведенная таблица задает f – функцию голосования, в которой результат r – это преобладающее значение ее аргументов (0 кодирует false, 1 кодирует true). Несложно установить, что r = f(x, y, z), где f(x, y, z) ≡ x&y ∨ x&z ∨ y&z.

Предметно рассуждать о некорректности реализации вышеприведенной таблицы решений можно по отношению к программам P вида

if(¬x & ¬y) r :=v_1 else if(¬y & ¬z) r :=v_2 else if(¬x & ¬z) r :=v_3 else if(x & y) r:=v_4 else if(y & z) r :=$v_5$else if(x & z) r

:=v_6, в которых условия $Cond_j$ конкретизируются логическими выражениями – охранами срабатывания P_j, где P_j – это r :=v_j, а v_j- константное выражение со значением 0 или 1. В программах P ошибки – это неправильный выбор значений v_1– v_6, в правильной программе должно быть v_1= v_2= v_3= 0, v_4= v_5= v_6= 1.

По отношению к спецификации <φ, ψ>, где φ(x, y, z) ≡ true, ψ(x, y, z, r) ≡ (r = f(x, y, z)), если P некорректна, то, по Теореме 1, примененной слева направо, учитывая, что ¬({φ & Cond}r:= v{ψ}) и φ ≡ true эквивалентно Cond & r = ¬ r′, где r′ – надлежащее значение для r согласно данной таблице решений, истинным является выражение

¬x & ¬y & r ∨ (x ∨ y) & ¬y & ¬z & r ∨ (x ∨ y) & (y ∨ z) & ¬x & ¬z & r ∨ (x ∨ y) & (y ∨ z) & (x ∨ z) & x & y & ¬ r ∨ (x ∨ y) & (y ∨ z) & (x ∨ z) & (¬x ∨ ¬y) & y & z & ¬ r ∨ (x ∨ y) & (y ∨ z) & (x ∨ z) & (¬x ∨ ¬y) & (¬y ∨ ¬z) & x & z & ¬r. Истинность этого выражения логически эквивалентна r = ¬(x & y ∨ x & z ∨ y & z) или r ≠ f(x, y, z).

Теорема 2. Для каждой программы вида (1) существует спецификация, относительно которой эта программа не является правильной.

Доказательство. Пусть P – произвольная программа вида (1), эта программа завершается на любом входном наборе $\bar{x}$. Рассмотрим спецификацию <φ, ψ>, где ψ($\bar{x}$, r) ≡ (r = $\prod_{k=1}^{m}(res_k^2(\bar{x})+1)$). Если в P срабатывает j-ое правило, то результат P($\bar{x}$) = $res_j(\bar{x})$. Но для любого $\bar{x}$ $res_j(\bar{x}) < res_j^2(\bar{x})+1$ и $res_k^2(\bar{x})+1 \geq 1$ при k ≠ j. Поэтому P($\bar{x}$) < $\prod_{k=1}^{m}(res_k^2(\bar{x})+1)$ для любого набора $\bar{x}$, следовательно ψ($\bar{x}$, P($\bar{x}$)) ложно.

Теорема 3. Существует спецификация <φ, ψ>, относительно которой любая программа вида (1) не является правильной.

Доказательство. Выше установлено, что P($\bar{x}$) < $\prod_{k=1}^{m}(res_k^2(\bar{x})+1)$ для любого $\bar{x}$. Выражение R($\bar{x}$) ≡ $\prod_{k=1}^{m}(res_k^2(\bar{x})+1)$ является многочленом. Рассмотрим отношение экспоненцирования c = a^b. Было доказано, например, в [9] и , по другому, в [10], что оно является диофантовым. Это означает существование такого многочлена D(a, b, c, y_1, …, y_p) с целыми коэффициентами, что c = a^b ⇔ ∃y_1…∃y_p(D(a, b, c, y_1, …, y_p) = 0). Теперь определим требуемую спецификацию <φ, ψ>, полагая φ($\bar{x}$) ≡ true для любого целочисленного набора $\bar{x}$ и ψ($\bar{x}$, r) ≡ ∃y_1…∃y_p(D(2, $\sum_{i=1}^{n} x_i^2$, r, y_1, …, y_p) = 0). Поскольку P($\bar{x}$) < R($\bar{x}$) при любом $\bar{x}$, и, начиная с некоторого v, R($\bar{t}$) < $2^{\sum_{i=1}^{n} t_i^2}$

для всех наборов $\bar{t}$ таких, что v < $\sum_{i=1}^{n} t_i^2$, справедливо P($\bar{t}$) < $2^{\sum_{i=1}^{n} t_i^2}$. Таким образом, для бесконечно многих целочисленных наборов $\bar{x}$ утверждение ψ($\bar{x}$, P($\bar{x}$)) является ложным.

III. О НЕКОРРЕКТНОСТИ ОПЕРАТОРА ЦИКЛА

Рассмотрим оператор цикла S ≡ while Cond do P, тело которого не содержит как явно, так и неявно (с помощью операторов перехода и меток) циклов. Назовем такие циклы плоскими. Это базовый вариант цикла при анализе его некорректности в данной работе и, например, в [11]. Известно, что произвольный структурный цикл [12] можно преобразовать в функционально эквивалентный плоский цикл [12 глава 4, 11], так, что плоские циклы вполне выразительны. Кроме того, в неплоском цикле внешний цикл может быть правильным, а вложенный – неправильным относительно их спецификаций. При этом вложенный цикл использует данные, общие с внешним циклом, а их постусловия не тождественно ложны. Вот примеры такого рода.

Пример 2.

{φ_1(i, j) ≡ (i = 2) & (j = 0)}
while (i > 0) do { //цикл1
{φ_2(i, j) ≡ (1 ≤ i ≤ 2) & (j = 0)}
while j < i do j := j + 2; //цикл2
{ψ_2(i, j) ≡ (i = j)}
i := i – 1; j := 0
}
{ψ_1(i, j) ≡ (i + j = 0)}

Спецификация цикла1 – <φ_1, ψ_1>, спецификация цикла2 – <φ_2, ψ_2>, сильнейший инвариант цикла1 I_1(i, j) ≡ (0 ≤ i ≤ 2) & (j = 0), сильнейший инвариант цикла2 I_2(i, j) ≡ (1 ≤ i ≤ 2) & (j = 0) ∨ (j = 2). Постусловие ψ_2 выполнено при первом исполнении цикла2, но не выполнено впоследствии, постусловие ψ_1 выполнено.

Пример 3.

{φ_1(i, j) ≡ (i = 2) & (j = 0)}
while (i > 0) do { //цикл1
{φ_2(i, j) ≡ (1 ≤ i ≤ 2) & (j = 0)}
while j < i do j := j + 2; //цикл2
{ψ_2(i, j) ≡ (i = j)}
{φ_3(i, j) ≡ (1 ≤ i ≤ 2) & (j = 2)}
while j ≥ i do j := j – 2; //цикл3
{ψ_3(i, j) ≡ (i % 2 = 0 ⇔ j % 2 = 0)}
i := i – 1;
}
{ψ_1(i, j) ≡ (i + j = 0)}

Спецификация цикла1 – <φ_1, ψ_1>, спецификация цикла2 – <φ_2, ψ_2>, спецификация цикла3 – <φ_3, ψ_3>, сильнейший инвариант цикла1 I_1(i, j) ≡ (0 ≤ i ≤ 2) & (j = 0), сильнейший инвариант цикла2 I_2(i, j) ≡ ((j = 0) ∨ (j = 2)) & (1 ≤ i ≤ 2), сильнейший инвариант цикла3 I_3(i, j) ≡ (j % 2 = 0) & (1 ≤ i ≤ 2).

Постусловие ψ_2выполнено при первом исполнении цикла2, но не выполнено впоследствии, постусловие ψ_3 выполнено при первом исполнении цикла3, но не выполнено впоследствии, постусловие ψ_1 выполнено. Как видно, статус правильности циклов в неплоском цикле может быть неоднородным.

Семантика цикла S определяется стандартным образом [8, 12] так, что множество {I_j} инвариантов цикла [8, 12] непусто, каково бы ни было тело цикла P. Как обычно, совокупность программных переменных образует память программы, состояние памяти, т.е. текущие значения этих переменных, обозначим как v, так, что r(v) – значение r в состоянии v.

Теорема 4. Если I_1, …, I_k – произвольная конечная совокупность инвариантов S и постусловие ψ – для S таково, что для любого входа $\bar{x}$ такого, что φ($\bar{x}$), и состояния памяти v выполнено ψ($\bar{x}$, r(v)) ⊃ ¬ (I_1($\bar{x}$, v) & … & I_k($\bar{x}$, v)), то ¬(<φ> S <ψ>).

Иначе говоря, некорректность S, относительно <φ, ψ>, гарантирована, если у постусловия ψ указанное свойство.

Доказательство. Предположим противное, т.е. <φ> S <ψ>, тогда, так как указанный цикл завершается, по его завершению будут выполнены постусловие ψ и, в силу выбора ψ, ¬ (I_1& … & I_k) ≡ ¬I_1∨ … ∨ ¬I_k также. Истинность последнего условия означает одновременную выполнимость ¬I_j и I_j, как инварианта, при некотором j, 1≤ j ≤ k. Но это невозможно. Полученное противоречие показывает ложность сделанного предположения.

Пример 4. Пусть φ(x, y, q, r) ≡ (x≥ 0) & (y > 0) & (q = 0) & (r = x), S – оператор вида while r ≥ y do { r := r – y; q := q + 1 }, для которого инвариантом является I(x, y, q, r) ≡ x = q*y + r. Тогда для ψ(x, y, q, r) ≡ (q = 0) & (r = x +1) будет ¬(<φ> S <ψ>).

Пример 5. Пусть φ(x, r) ≡ (x_0≥ 0) & (x = x_0) & (r = 1), S – оператор вида while x > 0 do {r := r*x; x := x – 1 }, для которого инвариантом является I(x_0, x, r) ≡ x_0! = r*x!. Тогда для ψ(x, r) ≡ r = 0 будет ¬(<φ> S <ψ>).

Теорема 5. Если S и спецификация <φ, ψ> таковы, что ¬(<φ> S <ψ>), то какова бы ни была конечная совокупность инвариантов I_1, …, I_k для S существует такой вход $\bar{x}_0$, удовлетворяющий φ, при котором S завершается и в финальном состоянии памяти v выполнено ψ($\bar{x}_0$, r(v)) ⇔ ¬ I_1($\bar{x}_0$, v) ∨ … ∨ ¬ I_k($\bar{x}_0$, v).

Доказательство. По предположению, программа S завершается на любых входах, удовлетворяющих предусловию φ. Тогда из ¬(<φ> S <ψ>) следует существование такой входа $\bar{x}_0$ и состояния v, что ¬Cond($\bar{x}_0$, v) и ¬ψ($\bar{x}_0$, r(v)). Кроме того, истинно ¬ψ($\bar{x}_0$, r(v)) & I_j($\bar{x}_0$,v) и ложно ψ($\bar{x}_0$, y(v)) & ¬I_j($\bar{x}_0$,v), где I_j($\bar{x}_0$,v) – произвольный инвариант, 1 ≤ j ≤ k . Дизъюнкция двух последних утверждений ¬ψ($\bar{x}_0$, r(v)) & I_j($\bar{x}_0$,v) ∨ ψ($\bar{x}_0$, r(v)) & ¬I_j($\bar{x}_0$,v) равносильна ψ($\bar{x}_0$, r(v)) ⇔ ¬I_j($\bar{x}_0$,v). Логически сложив левую и правые части последней эквивалентности по всем 1 ≤ j ≤ k, получим утверждение теоремы.

Замечание. Теоремы 4, 5 справедливы для произвольного аннотированного цикла. Кроме того, далее в качестве предметной области рассматривается N – множество натуральных чисел вместе с нулем и обычными арифметическими операциями и сравнениями.

Возвращаясь к плоским циклам, заметим, что в теле такого цикла может программироваться лишь функция, вычисление которой цикла не требует. Класс таких функций, в рамках фиксированного языка, обозначим LFF. Состав LFF зависит от базовых функций, операций и команд языка. Например, класс E^3 функций, элементарных по Кальмару, определяется классом программ, по существу ассемблерного уровня, с глубиной вложенности циклов не более 2 [13]. Тот же класс функций реализуется классом программ с дополнительными средствами управления, но глубиной вложенности циклов не более 1 [14]. В ряде исследований были предложены различные варианты базисов для класса E^3 относительно суперпозиции – единственной операции построения новой функции из уже имеющихся. Например, если базис B_1= {x + y, x ∸ y, [x/y], 2^x}из [15] для E^3 входит в состав базовых функций и операций языка или задается подходящими выражениями языка, то E^3 ⊆ LFF. Множество L_1 функций, определенных программами, заданными на языке ассемблерного уровня из [13] с глубиной вложенности циклов не более 1, задается суперпозициями функций из базиса B_2= { x + 1, x ∸ 1, o(x)= 0, x + y, $U_i^n(x_1, x_2, ..., x_n) = x_i$, (y = 0 → x | y > 0 → 0)} ∪ {[x/k], x % k | k ∈N} [16, 17]. В языке, базовые функции и операции которого позволяют выразить элементы из B_2, функции из L_1 перестают быть циклическими, т.е. L_1 ⊆ LFF.

IV. Анализ некорректности на основе рекурсии

Помимо логической спецификации эффект программы можно описать рекурсивными уравнениями по аналогии с рекурсивными описаниями программных функций, вычислимых блок-схемой, сделанными в [18]. Рассмотрим определение следующей функции f типа N^2 → N по схеме возвратной рекурсии порядка k:

$$\begin{cases} f(x,0) = a_1(x), f(x,1) = a_2(x), ..., f(x,k-1) = a_k(x), \\ f(x,n+1) = g\big(x, f(x,n), f(x,n-1), ..., f(x,n-k+1)\big), \text{для } n \geq k-1, \end{cases} \quad (3)$$

где функция g ∈ LFF типа N^{k+1} → N задается подходящим выражением, a_1, …, a_k – функции типа N → N, принадлежат LFF и заданы подходящими выражениями языка, x – входное данное. Пусть ψ(x, n, r) ≡ (r = f(x, n)) – постусловие программы вычисления функции f.

Вычисление функции f для всех n, кроме начальных значений, дает следующий вариант аннотированного простого цикла с кратным присваиванием.

{n > k & i = k + 1 & r_1 = a_1(x) & … & r_k = a_k(x)}
while i ≤ n do { r :=g(x, r_k, r_{k-1}, …, r_1); r_1, r_2, …, r_{k-1}, r_k:= r_2, r_3, …, r_k, r ;
i := i + 1
}
{ψ(x, n, r)}

По схеме (3) можно определить требования, которым должно удовлетворять ψ, в виде следующей формулы:

R(ψ) ≡ ψ(x, 0, a_1(x)) & ψ(x, 1, a_2(x)) & … & ψ(x, k – 1, a_k(x)) & ∀n∀r_1 ∀r_2…∀r_k∃t(n ≥ k & ψ(x, n – k, r_1) & ψ(x, n – k + 1, r_2) & … & ψ(x, n – 1, r_k) ⊃ t = g(x, r_k, r_{k-1}, …, r_1) & ψ(x, n, t)).

Теорема 6. ∀n∀x∃r ψ(x, n, r) ⇔ ∀x R(ψ).

Доказательство. Пусть R_1(ψ) ≡ ψ(x, 0, a_1(x)) & ψ(x, 1, a_2(x)) & … & ψ(x, k – 1, a_k(x)), R_2(ψ) ≡

$\forall n \forall r_1 \forall r_2 \ldots \forall r_k \exists t(n \geq k \,\&\, \psi(x, n-k, r_1) \,\&\, \psi(x, n-k+1, r_2) \,\&\, \ldots \,\&\, \psi(x, n-1, r_k) \supset t = g(x, r_k, r_{k-1}, \ldots, r_1) \,\&\, \psi(x, n, t))$, так что $\forall x\ \mathcal{R}(\psi) \Leftrightarrow \forall x\ \mathcal{R}_1(\psi) \,\&\, \forall x\ \mathcal{R}_2(\psi)$. Достаточность. Из $\forall n \forall x \exists r\ \psi(x, n, r)$ следует истинность $\forall x \exists r\ \psi(x, n, r)$ при n = 0, 1,…, k –1 и, так как f – функция, определенная по схеме (3), то выполнено $\forall x\ \mathcal{R}_1(\psi)$. Для $\forall x\ \mathcal{R}_2(\psi)$ конъюнкт $\psi(x, n, t)$ в заключении импликации верен при любом n и некотором t, таком что t = f(x, n), это следует из истинности $\forall n \forall x \exists r\ \psi(x, n, r)$. При n ≥ k, результат t для функции f возможен, когда $t = g(x, r_k, r_{k-1}, \ldots, r_1)$ и $r_k, r_{k-1}, \ldots, r_1$ – значения f при предыдущих значениях n, т.е. $\psi(x, n-k, r_1) \,\&\, \psi(x, n-k+1, r_2) \,\&\, \ldots \,\&\, \psi(x, n-1, r_k)$ выполнено. Следовательно $\forall x\ \mathcal{R}_2(\psi)$ истинно.

Необходимость. Индукцией по n можно доказать $\forall n \forall x \exists r\ \psi(x, n, r)$. База индукции следует из $\forall x\ \mathcal{R}_1(\psi)$, а индуктивный переход – из $\forall x\ \mathcal{R}_2(\psi)$.

Пусть функция f(x, n) определена согласно (3) и $\varphi(x, n) \equiv$ true, $\psi(x, n, r) \equiv (r = f(x, n))$.

Теорема 7. Если $\neg(\{\varphi\}S\{\psi\})$ то

$\exists i(1 \leq i \leq k \,\&\, \neg\psi(x_0, i-1, a_i(x_0))) \vee \exists r_1 \exists r_2 \ldots \exists r_k \forall t(n_0 \geq k \,\&\, \psi(x_0, n_0-k, r_1) \,\&\, \psi(x_0, n_0-k+1, r_2) \,\&\, \ldots \,\&\, \psi(x_0, n_0-1, r_k) \,\&\, (t \neq g(x_0, r_k, r_{k-1}, \ldots, r_1) \vee \neg\psi(x_0, n_0, t)))$

для некоторых входов x_0, n_0 программы S.

Доказательство. Из тотальной завершимости программы S следует существование таких x_0, n_0, для которых $f(x_0, n_0) \neq S(x_0, n_0)$. С учетом теоремы 6, последнее неравенство означает нарушение требований $\mathcal{R}(\psi)$, то есть: $\exists i(1 \leq i \leq k \,\&\, \neg\psi(x_0, i-1, a_i(x_0))) \vee \exists r_1 \exists r_2 \ldots \exists r_k \forall t(n_0 \geq k \,\&\, \psi(x_0, n_0-k, r_1) \,\&\, \psi(x_0, n_0-k+1, r_2) \,\&\, \ldots \,\&\, \psi(x_0, n_0-1, r_k) \,\&\, (t \neq g(x_0, r_k, r_{k-1}, \ldots, r_1) \vee \neg\psi(x_0, n_0, t)))$.

Можно сказать, что программа S некорректна относительно $\langle\varphi, \psi\rangle$ потому, что для некоторого входа x_0 она неправильно вычисляет какое-то из начальных значений $a_i(x_0)$ или на каком-то этапе n_0 неправильно вычисляется значение функции g.

Возможны и взаимно рекурсивные определения нескольких программных функций. В таком случае и требования, которым должны удовлетворять постусловия соответствующих программ, задаются совместно. Вот пример:

$$even(n) = \begin{cases} true, & \text{если } n = 0 \\ odd(n-1), & \text{если } n > 0 \end{cases}$$

$$odd(n) = \begin{cases} false, & \text{если } n = 0 \\ even(n-1), & \text{если } n > 0 \end{cases} \quad (4)$$

Для программ вычисления указанных функций предусловие – это $\varphi(n) \equiv n \in N$. Пусть $\psi_f(n, r) \equiv (r = f(n))$, f –логическая функция от натурального аргумента, r – логическая переменная. Тогда постусловием программы для функции even является $\psi_{even}(n, r) \equiv (r = even(n))$, постусловием программы для функции odd является $\psi_{odd}(n, r) \equiv (r = odd(n))$. По схемам (4) для функций even и odd и виду определений ψ_{even}, ψ_{odd} определим поведенческие свойства, которые должны иметь ψ_{even} и ψ_{odd}, используя квантифицированную логическую переменную r:

$\mathcal{R}0(\psi_{even}, \psi_{odd}) \equiv \psi_{even}(0, \text{true}) \,\&\, \psi_{odd}(0, \text{false})$;

$\mathcal{R}1(\psi_{even}) \equiv \forall r(\psi_{even}(0, r) \Leftrightarrow \neg\psi_{even}(0, \neg r))$;

$\mathcal{R}2(\psi_{odd}) \equiv \forall r(\psi_{odd}(0, r) \Leftrightarrow \neg\psi_{odd}(0, \neg r))$;

$\mathcal{R}3(\psi_{even}, \psi_{odd}) \equiv \forall n \forall r\ (n > 0 \supset (\psi_{even}(n, r) \Leftrightarrow \psi_{odd}(n-1, r)))$;

$\mathcal{R}4(\psi_{even}, \psi_{odd}) \equiv \forall n \forall r\ (n > 0 \supset (\psi_{odd}(n, r) \Leftrightarrow \psi_{even}(n-1, r)))$.

Формула $\mathcal{R}0(\psi_{even}, \psi_{odd})$ определяет какие результаты считаются правильными при вычислении функций even и odd на аргументе 0. Формулы $\mathcal{R}1(\psi_{even})$, $\mathcal{R}2(\psi_{odd})$ выражают чувствительность ψ_{even} и ψ_{odd} к изменению r, когда n = 0. Формулы $\mathcal{R}3(\psi_{even}, \psi_{odd})$ и $\mathcal{R}4(\psi_{even}, \psi_{odd})$ выражают идентичность проверок четность–нечетность для двух соседних чисел, что позволяет свести такую проверку к начальным значениям n. Пусть $\mathcal{R}(\psi_{even}, \psi_{odd}) \equiv \mathcal{R}0(\psi_{even}, \psi_{odd}) \,\&\, \mathcal{R}1(\psi_{even}) \,\&\, \mathcal{R}2(\psi_{odd}) \,\&\, \mathcal{R}3(\psi_{even}, \psi_{odd}) \,\&\, \mathcal{R}4(\psi_{even}, \psi_{odd})$. Истинность постусловий $\psi_{even}(n, r)$ и $\psi_{odd}(n, r)$ в зависимости от четности-нечетности n также характеризуется выражением $PnP(\psi_{even}, \psi_{odd}) \equiv Par(\psi_{even}) \,\&\, nPar(\psi_{odd})$, где $Par(\psi_f) \equiv \forall n((n \,\%\, 2 = 0 \Leftrightarrow \psi_f(n, \text{true}))$, $nPar(\psi_f) \equiv (n \,\%\, 2 = 1 \Leftrightarrow \psi_f(n, \text{true}))$.

Теорема 8. $\mathcal{R}(\psi_{even}, \psi_{odd}) \Leftrightarrow PnP(\psi_{even}, \psi_{odd})$.

Доказательство. Достаточность (правое утверждение влечет левое). Очевидно, что функции

$$e(n) = \begin{cases} true, \text{если } n - \text{четно} \\ false, \text{если } n - \text{нечетно} \end{cases}$$

и

$$o(n) = \begin{cases} true, \text{если } n - \text{нечетно} \\ false, \text{если } n - \text{четно} \end{cases}$$

удовлетворяют рекурсивным уравнениям (4) и, следовательно, применяя индукцию по рекурсии [18], для любого n выполнено: e(n) = even(n), o(n) = odd(n). Кроме того, из определений e(n) и o(n), а также правого утверждения теоремы следует, что $\forall n((e(n) = \text{true}) \Leftrightarrow \psi_{even}(n, \text{true})) \,\&\, ((o(n) = \text{true}) \Leftrightarrow \psi_{odd}(n, \text{true})))$. Равносильность e(n) = r и ψ_{even}, а также o(n) = r и ψ_{odd} означает, что в формулах из $\mathcal{R}(\psi_{even}, \psi_{odd})$ постусловия ψ_{even} и ψ_{odd} можно заменять соответственно на e(n) = r и o(n) = r и обратно. Из указанной равносильности и того, что e(n) и o(n) – решения уравнений (4), следует выполнимость $\mathcal{R}(\psi_{even}, \psi_{odd})$, так как формулы $\mathcal{R}0(\psi_{even}, \psi_{odd})$, $\mathcal{R}3(\psi_{even}, \psi_{odd})$, $\mathcal{R}4(\psi_{even}, \psi_{odd})$ – следствия уравнений (4), а $\mathcal{R}1(\psi_{even})$, $\mathcal{R}2(\psi_{odd})$ – следствия инъективности $\lambda r(e(0) = r)$ и $\lambda r(o(0) = r)$.

Необходимость (левое утверждение влечет правое). Применяя $\mathcal{R}3(\psi_{even}, \psi_{odd})$ и $\mathcal{R}4(\psi_{even}, \psi_{odd})$ подходящее число раз, получаем:

$$\psi_{even}(n, r) \Leftrightarrow \psi_{even}(n-2, r) \Leftrightarrow \ldots \Leftrightarrow \psi_{even}(m, r)$$
$$\psi_{odd}(n, r) \Leftrightarrow \psi_{odd}(n-2, r) \Leftrightarrow \ldots \Leftrightarrow \psi_{odd}(m, r), \quad (5)$$

где m = 0, если n – четно, и m = 1, если n – нечетно. Из $\mathcal{R}0(\psi_{even}, \psi_{odd})$ следует, что выполнено ψ_{even}(m, true) и, в силу $\mathcal{R}1(\psi_{even})$, не выполнено ψ_{even}(m, false), когда m = 0. Поскольку ψ_{even}(1, false) ⇔ ψ_{odd}(0, false) согласно $\mathcal{R}3(\psi_{even}, \psi_{odd})$ и ψ_{odd}(0, false) выполнено, согласно $\mathcal{R}0(\psi_{even}, \psi_{odd})$, а ψ_{odd}(0, true) не выполнено, согласно $\mathcal{R}2(\psi_{odd})$, получаем, что ψ_{even}(m, false) выполнено и ψ_{even}(m, false) не выполнено, когда m = 1. Применяя аналогичные рассуждения к ψ_{odd}, получаем, что ψ_{odd}(m, false) выполнено и ψ_{odd}(m, true) не выполнено, когда m = 0, и ψ_{odd}(m, true) выполнено и ψ_{odd}(m, false) не выполнено, когда m = 1. Анализ выполнимости ψ_{even} и ψ_{odd} при начальных значениях n и (5) доказывает истинность формулы PnP(ψ_{even}, ψ_{odd}).

Лемма. ∀n∀r(ψ_f(n, r) ⇔ ¬ψ_f(n, ¬r)).

Доказательство. По определению, ψ_f(n, r) = (r = f(n)) ⇔ (¬r ≠ f(n))⇔ ¬(¬r = f(n)) ⇔ ¬ψ_f(n, ¬r).

Из определений Par, nPar и леммы следует, что Par(ψ_{even}) и nPar(ψ_{odd}) характеризуют функции e(n) и o(n), что выражается следующими эквивалентностями:

Par(ψ_{even}) ⇔ ∀n($\psi_{even}(n, e(n))$),

nPar(ψ_{odd}) ⇔ ∀n($\psi_{odd}(n, o(n))$). (6)

Теорема 9. Если ¬({φ}E{ψ_{even}}) ∨ ¬({φ}O{ψ_{odd}}), то ¬ $\mathcal{R}0(\psi_E, \psi_O)$ ∨ ¬ $\mathcal{R}3(\psi_E, \psi_O)$ ∨ ¬ $\mathcal{R}4(\psi_E, \psi_O)$.

Доказательство. Условия теоремы и завершимость E и O влекут истинность ∃n(¬ψ_{even}(n, E(n))) ∨ ∃m(¬ψ_{odd}(m, O(m))) или, раскрывая определения ψ_{even}, ψ_{odd}, ¬(∀n(E(n) ⇔ even(n))) ∨ ¬(∀m(O(m) ⇔ odd(m))).

Рассмотрим один из вариантов истинности последнего утверждения. В остальных случаях рассуждения аналогичны. Пусть ¬(∀n(E(n) ⇔ even(n))) истинно. Очевидно, что

∀n(E(n) ⇔ even(n)) ⇔ ∀n(ψ_E(n, true) ⇔ ψ_{even}(n, true)) (7)

Предположим, что Par(ψ_E) истинно, т.е.

∀n((ψ_E(n, true) ⇔ n % 2 = 0) (8)

Согласно (6),

∀n((n % 2 = 0 ⇔ ψ_{even}(n, true)), (9)

поскольку функции even и e совпадают, что показано в теореме 8, и следовательно ∀n(ψ_{even}(n, e(n))) = true. Тогда из (8) и (9) следует ∀n(ψ_E(n, true) ⇔ ψ_{even}(n, true)), что противоречит предположению об истинности ¬(∀n(E(n) ⇔ even(n))), если учесть (7). В итоге, из условия теоремы следует, что Par(ψ_E) ложно или nPar(ψ_O) ложно т.е. ¬ PnP(ψ_E, ψ_O) истинно. По теореме 8, ¬ PnP(ψ_E, ψ_O) эквивалентно

¬$\mathcal{R}0(\psi_E, \psi_O)$ ∨ ¬$\mathcal{R}1(\psi_E)$ ∨ ¬$\mathcal{R}2(\psi_O)$ ∨ ¬$\mathcal{R}3(\psi_E, \psi_O)$ ∨ ¬$\mathcal{R}4(\psi_E, \psi_O)$ (10)

Но, по лемме, $\mathcal{R}1(\psi_E)$ и $\mathcal{R}2(\psi_O)$ – истинные утверждения, с учетом этого из (10) следует заключение теоремы.

Содержательно теорема утверждает, что если правильность какой-то из программ или обеих ложна, то причина этого в ошибке назначения результирующего значения при нулевом входе или в ошибке определения четности-нечетности для некоторых последовательных аргументов. Иначе говоря, если указанных ошибок программы не имеют, то они правильны.

V. Заключение

Некорректность программ – сложное явление, требующее изучения. Практическое значение такого изучения может состоять в выявлении образцов кода или образцов логических спецификаций, при наличии которых устанавливается некорректность программы. На этом пути было сделано следующее:

1. выделен класс программ, использующих ветвления и присваивания, сформулирован и обоснован критерий некорректности программ этого класса;
2. для любой программы из отмеченного класса построено выполнимое постусловие, относительно которого программа некорректна;
3. определено выполнимое постусловие, относительно которого никакая программа из отмеченного класса не является корректной;
4. выделен класс рекурсивных определений D так что некорректность программы вычисления функции, заданной D, эквивалентна нарушению некоторых свойств, которые описываются логическим утверждением, построенным по D;
5. приведен пример определений двух взаимно рекурсивных функций, некорректность программ вычисления которых анализируется через построение логических утверждений, описывающих свойства этих функций.

Библиография

[1] Baier C., Katoen J. P. Principles of model checking.– The MIT Press, 2008.– 975 p.

[2] Глушков В.М. Теорема о неполноте формальных теорий с позиции программиста //Кибернетика, 1979, № 2, с. 1–5.

[3] Borger E., Gradel E., Gurevich Y. The classical decision problem.– Springer-Verlag, 2001.– 482 p.

[4] Хамби Э. Программирование таблиц решений. – М.: Мир, 1976. – 86 с.

[5] Фрайтаг Г. и др. Введение в технику работы с таблицами решений. – М.: Энергия, 1979. – 88 с.

[6] Григорьев Д.Ю. Алгебраическая сложность вычисления билинейных форм //Ж. вычисл. матем. и матем. физ., 1979, т.19, № 3, с. 563 – 580.

[7] Ben-Or M. Lower bounds for algebraic computation trees //STOC'83: Proc. of the fifteenth annual ACM symposium on theory computing, 1983, p. 80 –86.

[8] Apt K.R., de Boer F.S., Olderog E.R. Verification of sequential and concurrent programs, 3rd Edition. – N.Y.:Springer, 2009.– 502 p.

[9] Matiyasevich Y. On Hilbert's tenth problem. – Calgary : PIMS, 2000.–71 p.

[10] Pak K. The Matiyasevich theorem. Preliminaries. //Formalized mathematics, 2017, v. 25, № 4, p. 315 – 322.

[11] S.-W. Lin, J. Sun, H. Xiao, Y. Liu, D. Sanán, H. Hansen Fib: Squeezing loop invariants by interpolation between forward/backward predicate transformers // in 2017 32nd IEEE/ACM International Conference on Automated Software Engineering (ASE). IEEE, 2017, pp. 793– 803.

[12] Лингер Р., Миллс Х., Уитт Б. Теория и практика структурного программирования. – М.: Мир, 1982. – 408 с.

[13] Meyer A.R., Ritchie D.M. The complexity of loop programs // Proceedings 22nd National ACM Conference, 1967, pp. 465 – 470.

[14] Constable R.L., Borodin A.B. Subrecursive programming languages, Part I: Efficiency and program structure //Journal of ACM, 1972, v. 19, № 3, pp. 526 – 568.

[15] Mazzanti S. Plain bases for classes of primitive recursive functions //Mathematical logic quarterly, 2002, v. 48, № 1, pp. 93 – 104.
[16] Tsichritzis D. A note on comparison of subrecursive hierarchies // Information processing letters, 1971, v. 1, pp. 42 – 44.
[17] Tsichritzis D. The equivalence problem of simple programs // Journal of ACM, 1970, v. 17, № 4, pp. 729 – 738.
[18] McCarthy J. Towards a mathematical science of computation // Proc. IFIP Congress, 1962, ed. C.M. Popplewell, Ams.:Norh-Holland, pp. 21 – 28.

On the incorrectness of a branching program and loop

S. A. Zhucov

Abstract — Within the Hoare approach to program correctness, the incorrectness of a certain type of branching program, which can model decision tables with bounded input, and a loop without subloops is studied. The relationships between the code structure of such a branching program and the postconditions that lead to incorrectness are studied. Satisfiable postconditions are given under which any given branching program is incorrect. Sufficient properties of the postcondition are determined under which a loop of the above type will be incorrect, as well as a logical expression linking loop invariants to the postcondition in the case of loop incorrectness. Program incorrectness can be expressed as the falsity of a certain statement describing the properties of the program postcondition in the case where the program function is defined recursively. Specifically, a class of recursive definitions D was identified such that the incorrectness of a program for computing a function defined by D is equivalent to the violation of certain properties described by a logical assertion constructed from D. An example of the definitions of two mutually recursive functions is given, the incorrectness of whose programs is analyzed through the construction of logical assertions describing the properties of these functions. The negation of the assertions constructed expresses the incorrectness of the corresponding programs.

Keywords - branching program, Diophantine representation of the exponential, loop invariant, logical specification of a postcondition, negative proposition, Hoare triple, Kalmar elementary functions

References:

[1] Baier C., Katoen J. P. Principles of model checking.□ The MIT Press, 2008.□ 975 p.
[2] Glushkov V.M. Teorema o nepolnote formal'nyh teorij s pozicii programmista //Kibernetika, 1979, # 2, s. 1□5.
[3] Borger E., Gradel E., Gurevich Y. The classical decision problem.□ Springer-Verlag, 2001.□ 482 p.
[4] Hambi Je. Programmirovanie tablic reshenij. □ M.: Mir, 1976. □ 86 s.
[5] Frajtag G. i dr. Vvedenie v tehniku raboty s tablicami reshenij. □ M.: Jenergija, 1979. – 88 s.
[6] Grigor'ev D.Ju. Algebraicheskaja slozhnost' vychislenija bilinejnyh form //Zh. vychisl. matem. i matem. fiz., 1979, t.19, # 3, s. 563 – 580.
[7] Ben-Or M. Lower bounds for algebraic computation trees //STOC□83: Proc. of the fifteenth annual ACM symposium on theory computing, 1983, p. 80 □86.
[8] Apt K.R., de Boer F.S., Olderog E.R. Verification of sequential and concurrent programs, 3rd Edition. – N.Y.:Springer, 2009.□ 502 p.
[9] Matiyasevich Y. On Hilbert's tenth problem. □ Calgary : PIMS, 2000.□71 p.
[10] Pak K. The Matiyasevich theorem. Preliminaries. //Formalized mathematics, 2017, v. 25, # 4, p. 315 – 322.
[11] S.-W. Lin, J. Sun, H. Xiao, Y. Liu, D. Sanán, H. Hansen Fib: Squeezing loop invariants by interpolation between forward/backward predicate transformers // in 2017 32nd IEEE/ACM International Conference on Automated Software Engineering (ASE). IEEE, 2017, pp. 793– 803.
[12] Linger R., Mills H., Uitt B. Teorija i praktika strukturnogo programmirovanija. – M.: Mir, 1982. □ 408 s.
[13] Meyer A.R., Ritchie D.M. The complexity of loop programs // Proceedings 22nd National ACM Conference, 1967, pp. 465 – 470.
[14] Constable R.L., Borodin A.B. Subrecursive programming languages, Part I: Efficiency and program structure //Journal of ACM, 1972, v. 19, # 3, pp. 526 – 568.
[15] Mazzanti S. Plain bases for classes of primitive recursive functions //Mathematical logic quarterly, 2002, v. 48, # 1, pp. 93 – 104.
[16] Tsichritzis D. A note on comparison of subrecursive hierarchies // Information processing letters, 1971, v. 1, pp. 42 – 44.
[17] Tsichritzis D. The equivalence problem of simple programs // Journal of ACM, 1970, v. 17, # 4, pp. 729 – 738.
[18] McCarthy J. Towards a mathematical science of computation // Proc. IFIP Congress, 1962, ed. C.M. Popplewell, Ams.:Norh-Holland, pp. 21 – 28.