О некорректности безусловных программ

С.А. Жуков

Аннотация — В логическом анализе императивной программы ситуации, требующие формулировки негативного высказывания о ней, мало исследованы. Логика некорректности – исключение из этого положения, поскольку позволяет формулировать утверждения о возможных неправильных, относительно логической спецификации, результатах программы. Она послужила поводом для исследования негативных утверждений о правильности завершающихся программ, но уже в рамках традиционного подхода Хоара. В рамках императивного безпроцедурного языка исследуются условия, при которых в тройках Хоара применительно к безусловным программам, т.е. программам, в которых логические условия не используются или не учитываются как средство управления, должно использоваться отрицание. В частности, установлены условия, при которых отрицание постусловия в тройке Хоара влечет отрицание самой тройки Хоара и, наоборот, отрицание тройки Хоара влечет отрицание ее постусловия. Сформулированы условия, при которых невыполнима тройка Хоара для оператора присваивания. В случае последовательной композиции двух программ выявлены достаточные условия ее некорректности, а также показано, что последовательная композиция двух некорректных программ может быть корректной. Доказывается алгоритмическая неразрешимость тотальной правильности в классе императивных безпроцедурных программ.

***Ключевые слова*— верификация программы, логика некорректности, неразрешимость проверки тотальной правильности, ошибка кода, ошибка спецификации, тройка Хоара.**

I. Введение

Успешное доказательство правильности программы – общепринятая ориентация исследований в области программной верификации, основанных на хоаровской логике [1, 2]. Но что делать, если это доказательство невозможно, и что это значит? Пусть {φ}P{ψ} – тройка Хоара, представляющая утверждение $(\forall\bar{x})(\varphi(\bar{x}) \to P(\bar{x})\downarrow \,\&\, \psi(\bar{x}, P(\bar{x})))$ о тотальной правильности программы P относительно предусловия $\varphi(\bar{x})$ и постусловия $\psi(\bar{x}, y)$, где $\bar{x}$ – набор входных переменных P, y – результирующая переменная, $P(\bar{x})\downarrow$ – утверждение о завершении запуска P на входе $\bar{x}$, при этом $P(\bar{x})$ – значение результата. Программа P правильна относительно спецификации <φ, ψ> , если {φ}P{ψ} истинно. Препятствием к доказательству {φ}P{ψ} может быть: неполная спецификация <φ, ψ>, например, φ или ψ не полностью описывают множества требуемых входов или результатов, нереализуемая спецификация ψ, ошибки в коде P [3, 4].

Шагом к более полному исследованию программного поведения логическими средствами стала статья о логике некорректности, в которой признается, что программа, по ее завершении, может формировать состояния, оцениваемые разработчиком или заказчиком как ошибочные [5]. Под состоянием программы понимается установление значений всем определенным в программе переменным. Учитывая это, можно выделить множество ok:ψ_1 финальных состояний программы, каждое из которых определяет правильный или требуемый результат счета, а также множество er:ψ_2 финальных состояний, каждое из которых признается как ошибочное. Предикаты ψ_1 и ψ_2, помечаемые маркерами ok и er соответственно, неявно определяются этими множествами правильных и ошибочных состояний. Изучение программных свойств относительно ok:ψ_1 и er:ψ_2 основано на аппроксимации снизу (в первоисточнике используется термин “under-approximation”) [5]. Эта аппроксимация предполагает, что ok:ψ_1 (er:ψ_2) описывает подмножество правильных (ошибочных) состояний, которые являются предметом интереса.

II. Тройки Хоара с отрицанием

В логике некорректности выделим еще один вид утверждений. Высказывание [π]P[ok:σ] означает, что каждое значение y, удовлетворяющее σ, подтверждается программой P в качестве правильного результата ее выполнения на подходящем входе $\bar{x}$, удовлетворяющем условию π. Формально, в терминах входов-выхода P, определим его как $[\pi]P[ok:\sigma] = (\forall y\exists\bar{x})(\sigma(\bar{x},y) \to (P(\bar{x})\downarrow \,\&\, y = P(\bar{x}) \,\&\, \pi(\bar{x})))$. Аналогично можно определить высказывание [π]P[er:ε] о том, что значения $\bar{x}$, y, удовлетворяющие $\varepsilon(\bar{x}, y)$, оцениваются как ошибочный результат $y = P(\bar{x})$ при запуске программы P на $\bar{x}$, удовлетворяющем $\pi(\bar{x})$. Поскольку ε описывает ошибочные результаты, то $(\forall\bar{x}\forall y)(\sigma(\bar{x},y) \,\&\, \varepsilon(\bar{x}, y) \equiv false)$, т.к. $(\forall\bar{x}\forall y)(\varepsilon(\bar{x}, y) \to \neg\sigma(\bar{x}, y))$. Обратное

Статья получена 1 сентября 2025

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

утверждение $(\forall\bar{x}\forall y)(\neg\sigma(\bar{x}, y) \to \varepsilon(\bar{x}, y))$ в общем случае неверно, поскольку ε может определять лишь часть ошибок, касающихся программы P, специфицированной <φ, ψ>. Из определений {φ}P{ψ} и [φ]P[ok:σ] вытекает справедливость высказывания {φ}P{ψ}&[φ]P[ok:σ] → (σ → ψ), которое называется принципом согласия [5]. Фактически этот принцип описывает смысл аппроксимации снизу, которую выражает высказывание [φ]P[ok:σ] о том, что P правильна. Этот же принцип, но заданный в негативной форме [φ]P[ok:σ] & ¬(σ → ψ) → ¬({φ}P{ψ}), называется принципом отрицания [5]. Он утверждает некорректность P относительно <φ, ψ> в предположении истинности его посылок, непротиворечивость логики некорректности установлена в [5].

Вернемся к тройкам Хоара. Учитывая определение {φ}P{ψ}, данное вначале, докажем, что {φ}P{¬ψ}→ ¬({φ}P{ψ}), если $\varphi(\bar{x}) \not\equiv false$, т.е. $\varphi(\bar{x})$ не является тождественно ложным. Пусть, наоборот, истинны оба утверждения: {φ}P{¬ψ} и {φ}P{ψ}. Тогда, применив правило вывода для конъюнкции, (см., например, [2], с. 98):

$$\frac{\{\varphi_1\}P\{\psi_1\}, \{\varphi_2\}P\{\psi_2\}}{\{\varphi_1\&\varphi_2\}P\{\psi_1\&\psi_2\}}$$

получаем истинность утверждения {φ}P{¬ψ & ψ}⇔ {φ}P{false}. Последнее, однако, не может быть истинным, т.к. постусловие false исключает любое финальное состояние для P, которое обязательно существует, если $(\forall\bar{x})(\varphi(\bar{x}) \to P(\bar{x})\downarrow)$. С другой стороны, если P завершима относительно φ, т.е. выполнено $(\forall\bar{x})(\varphi(\bar{x}) \to P(\bar{x})\downarrow)$, то ¬({φ}P{ψ}) означает, что $(\exists\bar{x})(\varphi(\bar{x})\ \&\ \neg\psi(\bar{x}, P(\bar{x})))$. Определим непустое множество $E = \{\bar{x} \mid \varphi(\bar{x})\ \&\ \neg\psi(\bar{x}, P(\bar{x}))\}$ и условие $\varepsilon(\bar{x}) \Leftrightarrow \bar{x} \in E$. Тогда $(\forall\bar{x})(\varepsilon(\bar{x}) \to P(\bar{x})\downarrow\ \&\ \neg\psi(\bar{x}, P(\bar{x}))) \equiv$ {ε}P{¬ψ}. Подводя итог, получаем

Теорема 1. Если $\varphi(\bar{x}) \not\equiv false$ и $\forall\bar{x}\ (\varphi(\bar{x}) \to P(\bar{x})\downarrow)$, то

1. {φ}P{¬ψ} → ¬({φ}P{ψ});
2. ¬({φ}P{ψ}) → {ε}P{¬ψ} для некоторого ε такого, что $\varepsilon(\bar{x}) \not\equiv false$ и ε → φ, т.е. в тройке Хоара {ε}P{¬ψ} постусловие ψ не выполняется на тех результатах, которые P вычисляет по входным данным, удовлетворяющим ε.

Некорректность программы P относительно ее спецификации <φ, ψ> установить проще, когда больше известно о связях между P и вычисляемой ею функцией f. Пусть предусловие $\varphi(\bar{x})$ задает область определения функции f, которую призвана вычислить P. Определим постусловие $\psi(\bar{x}, y) \equiv (y = f(\bar{x}))$ и предикат $\rho(y) \equiv (\exists\bar{x})(\varphi(\bar{x})\ \&\ y = f(\bar{x}))$.

Теорема 2. Если программа P такова, что $(\exists\bar{x})(\varphi(\bar{x})\ \&\ \neg\rho(P(\bar{x})))$, то ¬({φ}P{ψ}).

Доказательство. Предположим, что {φ}P{ψ} истинно. Тогда, учитывая вид ψ и что область определения f описывается φ, заключаем, что все результаты P принадлежат множеству значений функции f, однако это противоречит свойству программы P, заданному в условии теоремы.

Рассмотрим нарушения тотальной правильности применительно к присваиванию. Пусть φ – произвольный предикат, выраженный традиционными логическими средствами, все переменные которого определены в программе, φ(E/x) – предикат, полученный из предиката φ путем замены каждого свободного вхождения x в записи предиката φ (если оно встречается в φ) на выражение E.

Теорема 3. Если $\varphi(E/x) \not\equiv false$, то утверждение {φ(E/x)}x:=E{¬φ(x)} ложно.

Доказательство. Очевидно φ(E/x) & ¬φ(x) = false, если x= E. Последнее условие обеспечивается присваиванием x:=E. Присваивание – завершимое действие, которое в {φ(E/x)}x:=E{¬φ(x)}создает комбинацию контрарных предусловия φ(r) и постусловия ¬φ(r), где r – значение E и переменной x после присваивания, так что {φ(E/x)}x:=E{¬φ(x)}⇔(φ(r)→¬φ(r))⇔¬φ(r)=false, если $\varphi(E/x) \not\equiv false$.

Запись ¬φ(x) в постусловии вместо φ(x) в тройке {φ(E/x)}x:=E{¬φ(x)}можно назвать ошибкой спецификации результата присваивания. Например, {x < 0}x:= –x{x ≤ 0}– ложно, где φ(x) ≡ x > 0.

Следствие. Утверждение {φ(E/x)}x:=E′{φ(x)} ложно, если выражения E, E′ таковы, что E ≠ E′ и φ(E′/x) = ¬φ(E/x) при некотором наборе $\bar{y}$ значений свободных переменных, входящих в φ(E/x) или в φ(E′/x).

Действительно, пусть ψ(x) ≡ ¬φ(x). Тогда {φ(E/x)}x:=E′{φ(x)}⇔{¬φ(E′/x)}x:=E′{φ(x)}⇔ {ψ(E′/x)}x := E′{¬ψ(x)}.

Запись E′ вместо E в тройке{φ(E/x)}x:=E′{φ(x)} можно назвать ошибкой кода в присваивании. Например, утверждение {φ(11)}x:=9{φ(x)}, где φ(x)≡(x > 10), ложно. Здесь E=11, E′=9. Утверждение {φ(2)}x:=x^2 + 3{φ(x)}, где φ(x) ≡ (x^2 –3*x + 2 =0), ложно. Здесь E=2, E′ = x^2 + 3. Ошибка спецификации результата присваивания приводит к ложному утверждению, противоречащему аксиоме присваивания. Его можно синтаксически выявить. Влияние ошибки кода не постоянно, если варьировать пару выражений E, E′ даже при фиксированном φ. Поэтому возможность автоматического выявления ложности утверждения из-за ошибки кода в присваивании требует отдельного исследования.

III. Нарушения тотальной правильности в последовательной композиции

Пусть F, G – одноаргументные программы, завершающиеся на любых входах, т.е. (∀x)(φ(x) →F(x)↓ & G(x)↓), где предусловие φ(x) ≡ (x∈T), T – некоторый числовой тип, значения которого имеют конечное представление, с арифметическими операциями, сравнениями и константами 0, 1. В частности, N – тип целых неотрицательных чисел. При этом F предназначена для вычисления функции f : T → T, а G – для вычисления функции g : T → T. С учетом этого, для F постусловие ψ_f определяется как ψ_f(x, y) ≡ (y = f(x)), а для G постусловие ψ_g определяется как ψ_g(x, y) ≡ (y = g(x)).

Теорема 4. Если ¬({φ}F{ψ_f}) и {φ}G{ψ_g} для инъективной функции g, то ¬({φ}F;G{$\psi_{g \bullet f}$}), где $\psi_{g \bullet f}$(x, y) ≡ (y = g(f(x))).

Доказательство. Если ¬({φ}F{ψ_f}), то, по теореме 1, ¬ψ_f, т.е. существует такое натуральное z, что F(z) ≠ f(z). Из тотальной завершимости G следует, что она вычислит результат G(F(z)). Поскольку G тотально правильна, то G(F(z)) = g(F(z)) ≠ g(f(z)) в силу инъективности g. Поскольку G(F(z)) ≠ g(f(z)) для некоторого z, то ¬$\psi_{g \bullet f}$ и следовательно, по определению тотальной правильности, ¬({φ}F;G{$\psi_{g \bullet f}$}).

Похожие рассуждения приводят к следующей теореме.

Теорема 5. Если f – сюрьективная функция, {φ}F{ψ_f} и ¬{φ}G{ψ_g}, то ¬({φ}F;G{$\psi_{g \bullet f}$}).

Замечание к теореме 4. Когда функция g является константной (условие инъективности нарушено), композиция F;G тотально правильна относительно <φ, $\psi_{g \bullet f}$>. Таким образом, условие инъективности g в теореме 4 существенно.

Замечание к теореме 5. Если выполнены следующие условия:

1. f является ограниченной функцией, f: N → 0..fmax, где fmax – максимальное значение f, т.е. f – несюрьективная функция;
2. g является неограниченной функцией;
3. программа F тотально правильна относительно <φ, ψ_f>, т.е. {φ}F{ψ_f};
4. G является программой, определенной как G(x) ≡ if x ≤ fmax then g(x) else fmax;

тогда ¬ψ_g(x, G(x)) = (G(x) ≠ g(x)) для некоторых x > fmax, поскольку G, построенная по таблице значений функции g, реализует ограниченную функцию, а g – неограниченная функция. Поэтому ¬ψ_g(x, G(x)) влечет ¬({φ}G{ψ_g}), однако {φ}F;G{$\psi_{g \bullet f}$}, поскольку F(x) = f(x) и область определения функции G(x) в композиции G(F(x)) ограничена диапазоном 0..fmax, на котором G(F(x)) = G(f(x)) = g(f(x)). Таким образом, условие сюръективности f в теореме 5 существенно.

Теорема 6. Если выполнены следующие условия:

1. f – линейная функция вида f(x) = a_f*x + b_f при заданных константах a_f, b_f ∈T, x ∈ T;
2. программа F реализует некоторую линейную функцию F(x) = a_F*x + b_F, где a_F, b_F – возможно, явно незаданные константы такие, что a_F, b_F ∈T, x ∈T;
3. g – линейная функция вида g(x) = a_g*x + b_g, при заданных константах a_g, b_g ∈ T, x ∈ T;
4. программа G реализует некоторую линейную функцию G(x) = a_G*x + b_G, где a_G, b_G – возможно, явно незаданные константы такие, что a_G, b_G ∈T, x ∈T;
5. $a_f * a_g \neq a_F * a_G \vee a_g * b_f + b_g \neq a_G * b_F + b_G$;

тогда ¬({φ}F;G{$\psi_{g \bullet f}$}).

Доказательство “от противного”. Предположим, что {φ}F;G{$\psi_{g \bullet f}$} истинно. Поскольку все функции f(x), g(x), F(x), G(x) всюду определены, то программы F и G завершимы на любом входе. Тогда, по определению тотальной правильности, из истинности {φ}F;G{$\psi_{g \bullet f}$} следует истинность $\psi_{g \bullet f}$ = (G(F(x)) = g(f(x))) для любого x ∈ T. Найдем g(f(x)) и G(F(x)):

g(f(x)) = a_g*(a_f*x + b_f) + b_g = a_f*a_g*x + a_g*b_f + b_g, G(F(x))= a_G*(a_F*x + b_F) + b_G = a_F*a_G*x + a_G*b_F + b_G. Из g(f(x)) = G(F(x)) следует, что a_f*a_g = a_F*a_G и a_g*b_f + b_g = a_G*b_F + b_G. Одновременное выполнение последних равенств противоречит условию 5.

Замечание к теореме 6. Хотя какие-то из a_F, b_F, a_G, b_G могут быть явно не заданы, доступ к программам F и G предполагается. Поэтому неизвестные константы можно вычислить, например, b_F = F(0), a_F = F(1) – F(0).

Следствие. Если функции f(x), g(x), F(x), G(x) имеют вид f(x) = x + b_f , g(x) = x + b_g, F(x) = x + b_F, G(x) = x + b_G и $b_f + b_g \neq b_F + b_G$, то ¬({φ}F;G{$\psi_{g \bullet f}$}).

Подтверждающими теорему 6 парами функций могут быть, например, такие: f(x) = x, g(x) = –x и F(x) = G(x) = x; f(x) = g(x) = x и F(x) = x, G(x) = x + 1. В каждом случае условие теоремы выполнено, а неравенство g(f(x)) ≠ G(Fx)) очевидно.

Последовательная композиция программ, неправильно реализующих свои функции, может корректно реализовать их функциональную композицию. Например, пусть для x ∈ N, f(x) = g(x) = 3*x, тогда g(f(x)) = 9*x. С другой стороны, пусть программы F и G таковы, что

$$F(x) = \begin{cases} 3 * x + 1, & \text{если } x \text{ четное} \\ 3 * x - 1, & \text{если } x \text{ нечетное,} \end{cases}$$

$$G(x) = \begin{cases} 3 * (x - 1), & \text{если } x \text{ нечетное} \\ 3 * (x + 1), & \text{если } x \text{ четное.} \end{cases}$$

Неравенства f(x) ≠ F(x), g(x) ≠ G(x) при любом x ∈ N следуют из определений функций. Однако g(f(x)) = G(F(x)).

IV. Неразрешимость тотальной правильности

Рассмотрим теперь структурированные императивные программы с одним входом, в которых возможны условные операторы и операторы цикла, но нет подпрограмм. Формальным уточнением таких программ являются while-программы [2]. Рассмотрим те while-программы с одним входом, которые работают лишь с переменными и выражениями типа N. Класс таких программ, назовем их 1whileN-программами, является алгоритмически полным [6]. Известно, что проблема остановки сводится к проблеме частичной правильности с помощью проверки утверждений вида {true}P{false}, которые истинны, если и только если программа P не останавливается на любых входах [1]. Однако такой метод сведения неприменим в отношении тотально завершимых программ. Поэтому используется другая схема сведения между массовыми проблемами, основанная на диафантовости рекурсивно перечислимых множеств.

Теорема 7. Проблема проверки тотальной правильности алгоритмически неразрешима в классе 1whileN-программ.

Доказательство. Пусть A – некоторое нерекурсивное рекурсивно перечислимое множество, A ⊂ N. Согласно ДПРМ-теореме [7], существует диофантов полином D с некоторым числом d неизвестных и параметром a со значениями из N, такой, что

$$(\forall a)(a \in A \Leftrightarrow \exists x_1 \exists x_2 \ldots \exists x_d (D(a, x_1, x_2, \ldots, x_d) = 0)) \quad (1)$$

Вместо полинома D, описание которого заранее неизвестно, удобнее рассмотреть фиксированный универсальный диофантовый полином U(p, q, y_1, ..., y_m), обладающий, согласно [7], свойством:

$$\forall a (\exists x_1 \ldots \exists x_d (D(a, x_1, \ldots, x_d) = 0) \Leftrightarrow$$
$$\exists y_1 \ldots \exists y_m (U(a, k_D, y_1, \ldots, y_m) = 0)) \quad (2)$$

для некоторого значения $k_D \in$ N. Пусть P_a – программа с параметром a и входом x со значениями из N, а P_U(p, q, y_1, ..., y_m) – программа вычисления U(p, q, y_1, ..., y_m) в соответствии с определенной вычислительной схемой. Схема может основываться на прямом определении U(p, q, y_1, ..., y_m) как алгебраической сумме мономов или на обобщенной схеме Горнера [8]. В программе P_a на вход подается канторовский номер c(y_1, ..., y_m) значений y_1, ..., y_m, так что ее код имеет вид: t_1:= $l^{(m)}$(x); t_2:= r($l^{(m-1)}$(x)); …; t_{m-1}:= r(l(x)); t_m:= r(x); P_U(a, k_D, t_1, ..., t_m), где l(c) и r(c) – примитивно рекурсивные функции вычисления левой и правой компонент пары по ее канторовскому номеру c [9], а $l^{(k)}$ – это k-кратная итерация применения l. Из строения P_a следует ее тотальная завершимость.

При постусловии ψ(x, y) ≡ y^2 > 0, с учетом (2), справедливо {x ∈N}P_a {$P_a^2(x)$ > 0}⇔ ¬($\exists x_1 \ldots \exists x_d$)(D(a, x_1, …, x_d) = 0). Из этой эквивалентности следует, что при наличии разрешающего алгоритма для тотальной правильности его можно применять, в частности, к утверждениям {x∈N}P_a {$P_a^2(x) > 0$} и таким образом проверять истинность утверждения ($\exists x_1 \exists x_2 \ldots \exists x_d$)(D(a, x_1, …,x_d) = 0). Но, учитывая (1), алгоритмическая проверка a ∈ A невозможна.

В силу неразрешимости проблемы тотальной правильности, алгоритм проверки истинных утверждений {φ}P{ψ} может быть, в лучшем случае, полуразрешимым. Однако и возможности перечисления всех истинных утверждений указанного вида тоже ограничены. Известно [10], что непротиворечивой и относительно полной в смысле Кука аксиоматической системы Хоара не существует в стандартной интерпретации. В работе [11], например, сформулирован относительный, т.е. работающий в связке с оракулом, проверяющим истинность произвольного утверждения в заданной интерпретации I, алгоритм перечисления всех истинных утверждений тотальной правильности в I. Но этот алгоритм применим только для специальных интерпретаций [11]. Идея Хоара о проекте верифицирующего компилятора и сопутствующий ей проект репозитария верифицированного программного обеспечения предполагает автоматизацию проверок истинности утверждений о тотальной правильности для кода из этого репозитария, который задуман как открытое пополняемое, поучительное хранилище кодов, спецификаций и инструментов анализа программ [12].

V. Заключение

В работе проведено исследование причин и следствий ложности утверждения {φ}P{ψ}. Установлено, что причиной ложности {φ}P{ψ} может быть такая ошибка спецификации, как конфликт между результатом работы программы – P($\bar{x}$) и требованием к нему, заданному в постусловии ψ (теоремы 1, 3). С другой стороны, следствием ¬({φ}P{ψ}) является наличие входов $\bar{x}$, для которых постусловие ψ($\bar{x}$, P($\bar{x}$)) ложно (теорема 1). Другими причинами того, что ¬({φ}P{ψ}) могут быть ошибки в коде P (теорема 2 и следствие теоремы 3).

При рассмотрении последовательной композиции оказалось, что если только одна из программ не является правильной, то, при наличии у программной функции правильной программы свойства инъективности/сюрьективности, будет неправильной композиция этих программ (теоремы 4, 5). Показано, что свойство инъективности/сюрьективности существенно. Возможны неправильные программы композиция которых правильна. В теореме 6 для программ, вычисляющих линейные функции, сформулированы условия, при которых их композиция не является правильной.

Доказана алгоритмическая неразрешимость проблемы проверки утверждений о тотальной правильности. Это оказалось возможным для программ с одним входом из алгоритмически полного класса программ, хотя достаточно ограничиться программами, обеспечивающими вычисление многочленов над кольцом целых чисел и функций [√x], [x/2].

Полученные результаты могут использоваться в теории и практике дедуктивной верификации.

Библиография

[1] Apt K.R. Ten years of Hoar’s logic: A survey – Part 1 // ACM Transactions on programming languages and systems, 1984, v.3, № 4, pp. 431-483.

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

[3] Kemmerer R.A. Testing fomal specification to detect design errors // IEEE Transactions on software engineering, 1985, v. SE-11, № 1, pp. 32-43.

[4] Konighofer R., Hofferek G., Bloem R. Debugging formal specifications using simple counterstrategies // IEEE: Formal Methods in Computer-Aided Design, 2009, pp. 152–159.

[5] O’Hearn P.W. Incorrectness logic // Proceedings ACM Programming languages, 2020, v. 4, POPL, Article 10, pp. 10.1-10.32.

[6] Mills H. D. The new math of computing programming // Communications of the ACM, 1975, v.18, № 1, pp. 43 – 48.

[7] Matiyasevich Y. On Hilbert's tenth problem. – Calgary: PIMS, 2000. 71 p.
[8] Carnicer J., Gasca M. Evaluation of multivariate polynomials and their derivations // Mathematics of computation, 1990, v. 54, № 189, pp. 231-243.
[9] Мальцев А. И. Алгоритмы и рекурсивные функции. М: Наука, 1986. 368 с.
[10] Grabowski M. On relative incompleteness of logic for total correctness // Lecture notes in computer science, 1985, v. 195, pp. 118–127.
[11] Clarke E., German S., Halpern J. Effective axiomatizations of Hoare logics // JACM, 1983, v. 30, № 3, pp. 612–636.
[12] Bicarregui J., Hoare C.A.R., Woodcock J. The verified software repository: a step towards the verifying compiler // Formal Aspects of Computing, 2006, v. 18, № 2, pp. 143–151.

On the Incorrectness of Unconditional Programs

S.A. Zhucov

Abstract — **In the logical analysis of imperative programs, situations requiring the formulation of a negative statement about them have been little explored. Incorrectness logic is an exception to this rule, as it allows for the formulation of statements about possible incorrect program results relative to the logical specification. It has inspired the study of negative statements about the correctness of terminating programs, but within the framework of Hoare's traditional approach. Within the framework of an imperative, non-procedural language, the conditions under which negation must be used in Hoare triples for unconditional programs—that is, programs in which logical conditions are not used or not considered as a control mechanism—are investigated. Specifically, conditions are established under which the negation of a postcondition in a Hoare triple entails the negation of the Hoare triple itself, and, conversely, the negation of a Hoare triple entails the negation of its postcondition. Conditions are formulated under which a Hoare triple for the assignment operator is unsatisfiable. In the case of sequential composition of two programs, sufficient conditions for its incorrectness are identified, and it is also shown that sequential composition of two incorrect programs can be correct. The algorithmic undecidability of total correctness in the class of imperative non-procedural programs is proven.**

Keywords— **program verification, incorrectness logic, undecidability of total correctness checking, code error, specification error, Hoare triple**.

REFERENCES

[1]Apt K.R. Ten years of Hoar's logic: A survey – Part 1 // ACM Transactions on programming languages and systems, 1984, v. 3, # 4, pp. 431-483.
[2]Apt K.R., de Boer F.S., Olderog E.R. Verification of sequential and concurrent programs, 3rd Edition. N.Y.:Springer, 2009. 502 p.
[3]Kemmerer R.A. Testing formal specification to detect design errors // IEEE Transactions on software engineering, 1985, v. SE-11, # 1, pp. 32-43.
[4]Konighofer R., Hofferek G., Bloem R. Debugging formal specifications using simple counterstrategies // IEEE: Formal Methods in Computer-Aided Design, 2009, pp. 152–159.
[5]O'Hearn P.W. Incorrectness logic // Proceedings ACM Programming languages, 2020, v. 4, POPL, Article 10, pp. 10.1-10.32.
[6]Mills H. D. The new math of computing programming // Communications of the ACM, 1975, v. 18, # 1, pp. 43 – 48.
[7]Matiyasevich Y. On Hilbert's tenth problem. Calgary: PIMS, 2000. 71 p.
[8]Carnicer J., Gasca M. Evaluation of multivariate polynomials and their derivations // Mathematics of computation, 1990, v. 54, # 189, pp. 231-243.
[9]Mal'cev A. I. Algoritmy i rekursivnye funkcii. M: Nauka, 1986. 368 s.
[10] Grabowski M. On relative incompleteness of logic for total correctness // Lecture notes in computer science, 1985, v. 195, pp. 118-127.
[11] Clarke E., German S., Halpern J. Effective axiomatizations of Hoare logics // JACM, 1983, v. 30, # 3, pp. 612-636.
[12] Bicarregui J., Hoare C.A.R., Woodcock J. The verified software repository: a step towards the verifying compiler // Formal Aspects of Computing, 2006, v. 18, # 2, pp. 143-151.