# Математика и механика

*УДК 681.322.06.519.86*

*К. А. ЗУБОВИЧ, А. Н. МЕЛЕШКО*

## ОБ АВТОМАТИЧЕСКОМ ДОКАЗАТЕЛЬСТВЕ ПРАВИЛЬНОСТИ ПРОГРАММ В СИСТЕМЕ ФУНКЦИОНАЛЬНОГО ПРОГРАММИРОВАНИЯ

В последнее время много внимания уделяется проблемам функционального (аппликативного) программирования. Одно из наиболее обсуждаемых направлений этого подхода к программированию — *FP*-функциональный стиль написания программ [1]. Наряду с такими свойствами, как наглядность, прозрачность, компактность программ, *FP*-системы характеризуются простотой верификации создаваемого программного обеспечения. Однако сам процесс верификации, будучи простым и требуя лишь выполнения механических преобразований, отличается громоздкостью математических выкладок и необходимостью постоянно держать в уме теоремы и законы алгебры программ (особенно при доказательстве правильности нетривиальных программ). Это приводит к увеличению времени верификации и появлению ошибок, что, в свою очередь, ставит вопрос о создании инструментальной системы поддержки доказательства. В данной статье предлагаются основанные на схематизации процесс и система автоматизации доказательства правильности *FP*-функциональных программ, написанных в рамках реализованной системы функционального программирования *SFP* [2], которая, за исключением незначительных деталей, является аналогом *FP*-системы Бэкуса [1].

*Определение 1.* Назовем базисом β множество, представляющее собой объединение следующих множеств: 1) множества $X$ функциональных символов: $X = \{X_1, X_2, \ldots, X_n\}$; 2) множества $S$ специальных символов: $S = \{=, ;, (,)\}$; 3) множества $L$ латинских букв для обозначения функций языка; 4) множества $C$ символов функциональных форм: $C = \{«o», «[\ldots]», «\ldots \rightarrow \ldots ; \ldots »\}$; 5) множества термов (функциональных выражений), которое определяется рекурсивно; $E$ есть терм относительно переменных $X_1, X_2, \ldots, X_n$, если имеет место только одно из следующих предложений: а) $E(X_1, \ldots, X_n) = B \in L$; б) $E(X_1, \ldots, X_n) = X_i \in X$; в) существуют термы $E_1, E_2, \ldots, E_k$ и функциональная форма $\gamma$ с $k$ параметрами, построенная посредством композиции функциональных форм из множества $C$, такая, что $E(X_1, \ldots, X_n) = \gamma(E_1, \ldots, E_k)$.

*Определение 2.* Пусть задан базис β. Функциональным *SFP*-уравнением назовем выражение $e_i$ вида: (I) $X_i = E_i(X_1, \ldots, X_n)$, где $E_i(X_1, \ldots, X_n)$ — терм, $X_i$ — функциональная переменная.

*Определение 3.* Функциональной *SFP*-схемой называется последовательность (система) функциональных уравнений $(e_1, e_2, \ldots, e_m)$ (I″) такая, что все $X_i$ в левых частях различны и любой неопределенный ранее символ $X_j$, встречающийся в правой части некоторого уравнения, входит в левую часть этого или другого уравнения.

Введем понятие интерпретации базиса β в области интерпретации $D$ — множестве последовательностей-кортежей $<d_1, \ldots, d_n>$, т. е. $D = (D_1^+)^n$, где $D_1$ — произвольное множество, $D_1^+ = D_1 \cup w$, $(D_1^+)^n = D_1^+ \times D_1^+ \times \ldots \times D_1^+$, $w$ — специальный элемент для представления величины «не определено», который служит для достижения всюду определенности функций, получаемых в результате интерпретации схем.

*Определение 4.* Интерпретацией базиса β в области интерпретации $D$ называется функция $I$, которая сопоставляет: 1) каждому $X_i$ — всюду определенную функцию $f_i = I(X_i)\colon D \to D$; 2) каждому $x \in L$ — некоторую всюду определенную функцию $f = I(x)\colon D \to D$, построенную по правилам *SFP*-языка; 3) каждому символу $c \in C$ — некоторую всюду определенную функцию $G = I(c)\colon F \to F$, где $F$ — множество допустимых *SFP*-функций, причем символы функциональных форм интерпретируются следующим образом. Если $E_1$, $E_2$, $E_3$ — термы, а $g_1$, $g_2$, $g_3$ — функции, полученные в результате интерпретации $E_1$, $E_2$, $E_3$ соответственно, то интерпретация терма $(E_1 o E_2)$ определяется так: $(g_1 o g_2) : d \equiv g_1 : (g_2 : d)$; терм вида $[E_1, E_2]$ интерпретируется как результат применения функций $g_1$ и $g_2$ к $d \in D$ с получением кортежа $<g_1 : d, g_2 : d>$; а форма $E_1 \to E_2; E_3$ при ее интерпретации определяет функцию вида: для любого $d \in D$, если $g_1 : d = T$, то $g_2 : d$; иначе $g_3 : d$, где символ $T$ используется для обозначения логического значения «истина».

На множестве $D$ по аналогии с [3] введем частичный порядок $\sqsubseteq$ следующим образом: для любого $d \in D_1^+$ справедливо $w \sqsubseteq d$ и $d \sqsubseteq d$, а для $(D_1^+)^n$ имеет место $<a_1, \ldots, a_n> \sqsubseteq <b_1, \ldots, b_n>$ тогда и только тогда, когда для каждого $i$, $1 \leqslant i \leqslant n$, $a_i \sqsubseteq b_i$. Далее, следуя [3] и вводя понятия монотонности и непрерывности функций над $D$ и термов над $F$, можно показать, что каждый терм $E(X_1, \ldots, X_n)$, построенный в базисе β, является непрерывным относительно любой из переменных. Согласно теореме о наименьшей неподвижной точке [3], *SFP*-схема (I″) всегда имеет наименьшее решение вида $<f_{1\min}, \ldots, f_{m\min}>$. Эффективная процедура его построения подобна методу исключения неизвестных решения систем линейных алгебраических уравнений и использует результаты, аналогичные полученным в теории линейных форм [4]. Для замкнутости изложения сформулируем основные из них.

**Теорема 1.** Уравнение вида $X_i = p \to g; E_t(X_1, \ldots, X_i, \ldots, X_n)$ в случае линейности терма $E$ относительно $X_i$ (уравнение будем называть линейным) имеет наименьшее решение, представимое в виде условия: $X_{i\,\min} = p \to g; E_t(X_1, \ldots, p, \ldots, X_n) \to E(X_1, \ldots, g, \ldots, X_n); \ldots;$ $E_t^n(X_1, \ldots, p, \ldots, X_n) \to E^n(X_1, \ldots, g, \ldots, X_n); \ldots$, где $E_t$ — так называемый преобразователь предиката.

*Определение 5.* Терм $E_i(X_1, \ldots, X_i, \ldots, X_n)$ называется линейным относительно переменной $X_i$, если существует терм $E_{t_i}(X_1, \ldots, X_i, \ldots, X_n)$, называемый преобразователем предиката, такой, что имеет место следующее:

1) Для любых функций $a$, $b$, $c$ $E_i(X_1, \ldots, a \to b; c, \ldots, X_n) = E_{t_i}(X_1, \ldots, a, \ldots, X_n) \to E(X_1, \ldots, b, \ldots, X_n); E(X_1, \ldots, c, \ldots, X_n)$.

2) Для любого объекта $d \in D$, если $E_{t_i}(X_1, \ldots, w, \ldots, X_n) \neq w$, то для любой функции $a$ $E_{t_i}(X_1, \ldots, a, \ldots, X_n) : d = T$.

**Теорема 2.** Любой терм, построенный в базисе β, линейный, если переменные $X_i$ встречаются в нем только один раз.

**Теорема 3.** Любая функция, построенная в *SFP*-системе [2], может быть представлена посредством композиции функциональных форм базиса β и рекурсивных определений.

Сформулированные выше определения и теоремы являются математической основой процесса и системы верификации достаточно большого класса *SFP*-программ, где под верификацией понимается установление

формальными методами соответствия исследуемой программы своим спецификациям.

Процесс верификации функциональных *SFP*-программ предполагает выполнение следующей последовательности действий: 1) схематизацию исходной программы; 2) исключение свободных переменных; 3) линеаризацию уравнений полученной системы; 4) разложение линейных уравнений и построение наименьшего решения; 5) интерпретацию полученного наименьшего решения функциями исходной программы; 6) символическое выполнение построенного и проинтерпретированного минимального решения.

При схематизации определяемые функции заменяются символами множества $X$, функции, ранее определенные и базисные,— символами множества $L$, а функциональные формы, используемые в *SFP*-системе, следуя теореме 3, заменяются рекурсивными определениями, построенными из форм базиса β. Исключение свободных переменных состоит в удалении уравнений вида $X_i = A$, $A \in L$, и замене всех вхождений в правую часть уравнений таких $X_i$ на символы множества $L$. Под линеаризацией понимается приведение *SFP*-схемы к линейному виду. Разложение линейных уравнений с построением наименьшего решения осуществляется, согласно теореме 1. Интерпретация решения предполагает обратную замену его символов функциями исходной программы. Под символическим выполнением понимается такое выполнение, когда исходная информация для анализируемого решения и результат представляются не в виде конкретных данных, а в виде их символьных обозначений. На основании полученных выходных можно судить о правильности исходной программы.

Описанные подход к схематизации *SFP*-программ и процесс верификации являются теоретической основой подсистемы автоматизированного доказательства правильности программ. В состав подсистемы входят: анализатор, схематизатор, линеаризатор, минимизатор, интерпретатор, исполнитель. Задача анализатора — проверить правильность записи *SFP*-программ. Результат его работы представляет собой либо перечень ошибок, либо (если ошибок не обнаружено) программу, преобразованную к виду, удобному для схематизации. Смысл работы схематизатора состоит в преобразовании исходной программы в *SFP*-схему. Побочным результатом работы схематизатора является таблица соответствия символов и функций. Основным блоком системы (как с точки зрения выполняемых функций, так и с точки зрения их сложности) служит линеаризатор, осуществляющий свертку *SFP*-схемы и линеаризацию уравнений. Для этого организован ряд библиотек и таблиц, наиболее важные из них: библиотека целей линеаризации, библиотека трансформаций *SFP*-схем, таблица линейных термов и их преобразователей предиката. Один из принципов работы линеаризатора — постоянное, в процессе проведения доказательств, пополнение указанных библиотек. Библиотека целей линеаризации наряду с шаблонами схем содержит ссылки на способ построения решения. Минимизатор, используя эту библиотеку и полученные ранее преобразователи предикатов, осуществляет построение модели минимального решения. Замена символов функциями исходной программы в минимальном решении осуществляется интерпретатором в соответствии с таблицей замен. После этого исполнитель выполняет действия, необходимые для вычисления функции, полученной интерпретатором. Пользователю системы необходимо лишь сравнить полученную в результате символического выполнения цепочку с ожидаемой и сделать заключение о правильности исходной программы.

Реализована первая версия описанной системы. Реализация осуществлена по СР-технологии с использованием языка СР/ТРАН [5], причем свойства СР-технологии и языка СР/ТРАН позволили сократить время создания комплекса. Сокращению времени способствовал и тот факт, что реализация одной из версий системы *SFP* [2] также осуществлена на языке СР/ТРАН. Это позволило, взяв за основу описание структуры

*SFP*-системы и изменив лишь семантические действия, определяющие алгоритмы преобразования входной информации, просто и быстро построить такие подсистемы, как анализатор, схематизатор и линеаризатор. Средства работы с рекурсивными функциями языка СР/ТРАН также упростили процесс реализации системы.

## Список литературы


1. Backus J.// Communications of the ACM. 1978. V. 21. № 8. P. 613.
2. Зубович К. А.// Проблемы создания и совершенствования технических и программных средств э. в. м. широкого применения. Минск. 1982. С. 54.
3. Манна З.// Кибернетический сб. Новая серия. М. 1978. Вып. 15. С. 38.
4. Backus J.// Lecture Notes in Computer Science. 1981. V. 107. P. 1.
5. Дробушевич Г. А., То Туан, Зубович К. А.// Программирование. 1984. № 3. С. 84.


Поступила в редакцию 11.09.85.

*УДК 517.938*

*Б. С. КАЛИТИН*

# ОТНОСИТЕЛЬНАЯ УСТОЙЧИВОСТЬ КОМПАКТНЫХ МНОЖЕСТВ

**1.** Пусть $(X, \boldsymbol{R}, \pi)$ — динамическая система [1, 2], заданная на локально компактном метрическом пространстве $(X, d)$ с расстоянием $d: X \times X \to \boldsymbol{R}$. Пусть $M$ и $Y$ — два положительно инвариантных подмножества $X$, из которых $M$ компактно, а $Y$ замкнуто и содержит $M$. Положим $B(M, \varepsilon) = \{x \in X \mid d(M, x) < \varepsilon\}$; $\bar{B}(M, \varepsilon)$ — замыкание $B(M, \varepsilon)$ и $S(M, \varepsilon) = \bar{B}(M, \varepsilon) \setminus B(M, \varepsilon)$. Обозначим через $\gamma^+(x)$, $\gamma^-(x)$ и $\gamma(x)$ соответственно положительную полутраекторию, отрицательную полутраекторию и траекторию движения $xt$, $x: t \mapsto xt$, $\forall\, x \in X$, и пусть $L^+(x)$ означает множество всех $\omega$-, а $L^-(x)$ — множество всех $\alpha$-предельных точек для $x \in X$. Напомним (см. [1, 2]), что множество $N$ из $X$ называют положительно инвариантным, отрицательно инвариантным или инвариантным, если соответственно выполняются равенства: $N\boldsymbol{R}_+ = N$, $N\boldsymbol{R}_- = N$ или $N\boldsymbol{R} = N$. Кроме того, $M$ называется [2]:

(а)-устойчивым относительно $Y$, если

$$(\forall\, \varepsilon > 0)\,(\exists\, \delta = \delta(\varepsilon) > 0) : \gamma^+(B(M, \delta) \cap Y) \subset B(M, \varepsilon);$$

(б)-асимптотически устойчивым относительно $Y$, если оно устойчиво относительно $Y$ и область его притяжения $A_Y(M) = \{x \in Y \mid L^+(x) \subset M\}$ является окрестностью $M$ в $Y$;

(в)-глобально асимптотически устойчивым относительно $Y$, если оно асимптотически устойчиво относительно $Y$ и $A_Y(M) = Y$. (В современной литературе устойчивость относительно $Y$ иногда называют $Y$-устойчивостью.)

**2.** Приведем теперь ряд утверждений, касающихся качественной теории проблем относительной устойчивости компактных множеств. Соответствующие этому результаты являются продолжением исследований устойчивости компактных множеств, выполненных в [1, 3, 4].

**Лемма 1.** Если $M$ устойчиво относительно $Y$, то $\forall\, \gamma(x) \subset Y$ — пересечение $L^-(x) \cap M = \varnothing$. (Здесь и всюду в дальнейшем $\varnothing$ — пустое множество.)

**Лемма 2.** Если $M$ неустойчиво относительно $Y$, то всякая окрестность $U$ множества $M$ в $Y$ обладает тем свойством, что $U \setminus M$ содержит отрицательную полутраекторию. Более точно, существует компактная окрестность $\bar{B}(M, \varepsilon)$, $\varepsilon > 0$, для которой $\forall\, \varepsilon_1 \in ]0, \varepsilon]$ существуют последовательности $(x_n)$, $x_n \in Y$, и $(t_n)$, $t_n \geqslant 0$, такие, что $x_n \to x \in M$, $x_n t_n \to y \in S(M, \varepsilon) \cap Y$ при $n \to +\infty$, причем $\gamma^-(y) \subset (\bar{B}(M, \varepsilon_1) \setminus M) \cap Y$.

Доказательство. Пусть $M$ не является $Y$-устойчивым. Тогда