нашей работы *. В результате получим совокупность проективных подцепочек (в нашем случае их четыре). Если исходное сообщение проективно, то этот этап анализа является единственным.

На втором этапе первая и вторая подцепочки упорядочиваются согласно условиям теоремы 2 и ищутся синтагмы *. Если синтагма не найдена, то вторая подцепочка удаляется. Далее процедура повторяется.

На последующих этапах после получения единственной проективной подцепочки поочередно включаются удаленные подцепочки. После каждого включения осуществляется переупорядочение слов сообщения для получения подцепочек без гнездования. Завершается процесс анализа после получения единственной синтагматической структуры.

**Алгоритм синтаксического анализа.** На входе алгоритма — непроективное или слабопроективное сообщение $a_1a_2\ldots a_n$ $(n \geqslant 2)$, состоящее из проективных подцепочек $a_i$ $(i = \overline{1, n})$, полученных в результате синтаксического анализа в соответствии с приведенным алгоритмом *, на выходе — синтагматическая структура или композиция. Алгоритм состоит из следующих шагов.

1. $i:=1$, $j:=2$, $a:=a_i$, $b:=a_j$, $C=\varnothing$.
2. Упорядочить цепочку $ab$ (в соответствии с достаточными условиями проективности из теоремы 2).
3. Применить для анализа цепочки $ab$ алгоритм *. При успешном завершении процедуры анализа перейти к п. 4, в противном случае — к п. 5.
4. $a:=ab$. Если $j<n$, то $j:=j+1$, $b:=a_j$, перейти к п. 2, иначе — к п. 6.
5. $C:=C\cup\{a_j\}$, $j:=j+1$, $b:=a_j$, перейти к п. 2.
6. $i:=2$.
7. Если $C=\varnothing$, то упорядочить $a$. КОНЕЦ.
8. Если $a_i\in C$, то $b:=a_i$, перейти к п. 9, иначе — к п. 10.
9. Упорядочить $ab$ и применить алгоритм из указанной работы С. Ф. Липницкого, В. С. Яковишина. При успешном завершении процедуры анализа $C:=C\setminus\{a_i\}$, $a:=ab$, перейти к п. 7. В противном случае перейти к п. 10.
10. Если $i<n$, то $i:=i+1$, перейти к п. 8, иначе — к п. 6.

Очевидно, что алгоритм заканчивает свою работу не более чем за $n-1$ проход шагов 2—5 и $(n-2)$ — кратное выполнение шагов 6—9.

Поступила в редакцию 23.05.88.

*УДК 681.3.06*

*К. А. ЗУБОВИЧ*

# ПОСТРОЕНИЕ ПРЕОБРАЗОВАТЕЛЕЙ ПРЕДИКАТОВ И ГЕНЕРАЦИЯ МИНИМАЛЬНЫХ РЕШЕНИЙ ЛИНЕЙНЫХ ФУНКЦИОНАЛЬНЫХ УРАВНЕНИЙ В СИСТЕМЕ ФУНКЦИОНАЛЬНОГО ПРОГРАММИРОВАНИЯ

Одним из свойств широко обсуждаемых в настоящее время систем функционального программирования (СФП) [1] является упрощение процесса доказательства правильности программ [2—5]. Однако существует необходимость в разработке программных средств поддержки этого процесса [6]. В данной статье предлагается математическое обоснование и описывается программная реализация одного из этапов доказательства правильности программ в СФП — построения минимальных решений функциональных уравнений [2, 4, 6].

*Определение 1.* Функциональным уравнением в системе функционального программирования называется выражение вида $X_i=E(X_i)$, где $X_i$ —

* С. Ф. Липницкий, В. С. Яковишин // Весці АН БССР. Сер. фіз.-тэхн. навук. 1987. № 4.

функциональная переменная базиса β, $E(X_i)$ — терм, построенный в базисе β.

*Определение 2.* Базисом β называется множество, представляющее собой объединение следующих множеств: 1) $X=\{X_1, \ldots, X_n\}$ — функциональных переменных; 2) $L=\{a, b, c, \ldots, x, y, z\}$ — латинских букв; 3) $S=\{=, ., (,)\}$ — специальных символов; 4) $C=\{\text{"}\circ\text{"}, [, ], if, \rightarrow, ;\}$ — символов функциональных форм; 5) $T$ — термов (функциональных выражений), которое определяеся рекурсивно: $E$ есть терм относительно переменной $X_i$ ($E(X_i)\in T$) тогда и только тогда, когда имеет место ровно одно из следующих предложений: а) $E(X_i)\in L$; б) $E(X_i)=X_i$; в) существуют термы $E_1, E_2, \ldots, E_k$ такие, что $E(X_i)=E_1(X_i)\circ\ldots\circ E_k(X_i)$, или $E(X_i)=[E_1(X_i), \ldots, E_k(X_i)]$, или $E(X_i)=if\ E_1(X_i)\rightarrow E_2(X_i); E_3(X_i)$.

Замечание. Неформально базис и функциональное уравнение интерпретируются так (формальная интерпретация дана в [3]). Под символами из множества $L$ понимаются функции системы функционального программирования [1, 4] — базисные и ранее определенные. Символы $X_i$ соответствуют определяемым функциям, а элементы множества $C$ — функциональным формам «композиция», «конструкция» и «условие» [1] соответственно. Уравнения используются для задания новых функций, причем все функции определяются над некоторым множеством так называемых объектов $D=\{d_1, d_2, \ldots, d_t\}$ [1, 4].

*Определение 3.* Терм $E(X_i)$ называется линейным относительно переменной $X_i$, если существует терм (называемый преобразователем предиката) $E_t(X_i)$ такой, что имеет место: 1) для любых функций СФП $a$, $b$, $c$ $E(if\ a\rightarrow b; c)=if\ E_t(a)\rightarrow E(b); E(c)$, 2) для любого объекта $d\in D$, если $E_t(w)\neq w$, то для любой функции а СФП $E_t(a)$: $d=true$, где $true$ — объект, определяющий значение «истина», $w$ — объект «неопределенность», «:» определяет результат применения функции к аргументу.

*Определение 4.* Уравнение вида $X_i=if\ p\rightarrow g; E(X_i)$ называется линейным относительно переменной $X_i$, если терм $E(X_i)$ линеен по $X_i$.

**Теорема 1.** Линейное уравнение $X_i=if\ p\rightarrow g; E(X_i)$ имеет наименьшее решение, представимое посредством следующего разложения:

$$X_{i_{\min}}=if\ p\rightarrow g;\ if\ E_t(p)\rightarrow E(g);\ if\ E_t^2(p)\rightarrow E^2(g);\ \ldots;\ if\ E_t^n(p)\rightarrow E^n(g);\ \ldots,$$

где $E^2(X_i)=E(E(X_i))$, ... , $E^n(X_i)=E(E(\ldots E(X_i)\ldots))$; $E_t$ — преобразователь предиката.

В процессе построения минимальных решений функциональных уравнений встает задача выделения классов уравнений, для которых этот процесс может быть упрощен. Далее расматриваются линейные уравнения с одним включением переменной (неизвестной) в их правых частях.

Имеют место следующие утверждения.

1) Терм $E(X_i)=a$ линеен с $E_t(X_i)=true$.
2) Терм $E(X_i)=X_i\circ a$ линеен с $E_t(X_i)=X_i\circ a$.
3) Терм $E(X_i)=a\circ X_i$ линеен с $E_t(X_i)=X_i$.
4) Терм $E(X_i)=[a, \ldots, X_i, \ldots, z]$ линеен с $E_t(X_i)=X_i$.
5) Терм $E(X_i)=if\ a\rightarrow b; X_i$ линеен с $E_t(X_i)=if\ a\rightarrow true; X_i$.
6) Терм $E(X_i)=if\ a\rightarrow X_i; b$ линеен с $E_t(X_i)=if\ a\rightarrow X_i; true$.
7) Терм $E(X_i)=if\ X_i\rightarrow a; b$ линеен с $E_t(X_i)=X_i$.

**Теорема 2.** Если термы $E_1(X)$ и $E_2(X)$ линейны с соответствующими $E_{1_t}$ и $E_{2_t}$ преобразователями предиката, то и их композиции $E_1(E_2(X))$ и $E_2(E_1(X))$ линейны, причем $(E_1(E_2(X)))_t=E_{1_t}(E_{2_t}(X))$ и $(E_2(E_1(X)))=E_{2_t}(E_{1_t}(X))$.

**Пример.** Функциональное уравнение $X=if\ (d\circ e)\rightarrow h; (a\circ[b, (X\circ c)])$ является линейным в силу теоремы 2, так как справедливо следующее: $p=(d\circ e)$, $g=h$, $E(X)=(a\circ[b, (X\circ c)])=E_1(E_2(E_3(X)))$, где $E_3(X)=(X\circ c)$, $E_2(X)=[b, X]$, $E_1(X)=(a\circ X)$. $E_1, E_2, E_3$ линейны с $E_{3_t}(X)=$

$= (X \circ c)_t = X \circ c$, $E_{2_t}(X) = [b, X]_t = X$, $E_{1_t}(X) = (a \circ X)_t = X$. Согласно теореме 2, имеет место: $E_t(X) = E_{1_t}(E_{2_t}(E_{3_t}(X))) = E_{1_t}(E_{2_t}(X \circ c)) = E_{1_t}(X \circ c) = X \circ c$.

**Алгоритм** построения преобразователей предикатов и минимальных решений линейных функциональных уравнений.

1. Первый просмотр.

1.1. Выделение левой части уравнения (символа функциональной переменной).

1.2. Выделение в правой части уравнения функциональных выражений, соответствующих символам $p$ и $g$.

1.3. Формирование массива символов функциональных форм MASDOX, используемых в терме $E$ и встречающихся в нем до символа функциональной переменной.

**Пример.** Для рассмотренного выше уравнения после первого просмотра имеет место: функциональный символ — $X$; $p = d \circ e$; $g = h$; $masdox_1 = (;\ masdox_2 = \ddot{\bigcirc}\ ;\ masdox_3 = [;\ masdox_4 = (.$

Замечание. Контроль синтаксической правильности функционального уравнения при первом просмотре не производится.

2. Второй просмотр. Просмотр терма $E$ и формирование предиката $E_t$.

2.1. $i :=$ количество символов в $MASDOX$.

2.2. Пока не все символы функциональных форм слева от $X$ в терме $E$ обработаны (пока $i > 0$), делать следующее.

2.2.1. Взять $i$-й символ из $MASDOX$.

а) Если $masdox_i = '\circ'$, то в соответствии с тем, что $(a \circ X \circ b)_t = (X \circ b)$ (где $X$ соответствует обработанной части терма $E$, а в самом начале обработки $X$ — символ функциональной переменной), необходимо в терме $E$ просмотреть слева от $X$ все символы, стоящие до символа $'('$, учитывая при этом уровень вложенности скобок. После этого осуществляется выделение в терме $E$ выражения, соответствующего $b$, и формирование предиката путем переноса выражения «$\circ\, b$» в область формирования предиката. При просмотре влево проводится удаление (за счет изменения $i$) просмотренных символов функциональных форм из массива $MASDOX$.

б) Если $masdox_i = '['$, то, следуя тому, что $([\ldots, X, \ldots])_t = X$, необходимо, не изменяя области формирования предиката, осуществить просмотр терма $E$ влево и вправо от $X$, учитывая вложенность квадратных скобок в $E$ и проводя удаление из $MASDOX$ просматриваемых символов функциональных форм.

в) Если $masdox_i = '('$, то проводится обработка терма $E$, аналогичная обработке пункта а), так как $(X \circ a \circ \ldots)_t = (X \circ a \circ \ldots)$.

г) Если $masdox_i = ';'$, то терм имеет вид: $if\ a \to b;\ X$. В этом случае формируется предикат вида $if\ a \to true;\ X$ и осуществляется просмотр в исходном терме функций $a$ и $b$. Изменение сосояния $MASDOX$ состоит в удалении из него символов функциональных форм, образующих функции $a$ и $b$, символов $';'$, $'\to'$, $if$.

д) Если $masdox_i = '\to'$, то формируется предикат вида $if\ a \to X;\ true$ (так как исходный терм содержит композицию $if\ a \to X;\ b$). При этом осуществляется просмотр терма $E$ влево и вправо с целью выделения выражений, соответствующих $a$ и $b$, и проводится «чистка» массива MASDOX.

е) Если $masdox_i = 'if'$, то, согласно тому, что $(if\ X \to a;\ b)_t = X$, текущее состояние предиката не меняется, а проводится лишь выделение функций $a$ и $b$ с соответствующим изменением в состоянии $MASDOX$ за счет удаления из этого массива символа функциональной формы $if$.

Замечание. Параллельно с разбором терма и формированием предиката проводится синтаксический контроль терма на правильность.

**Пример.** Процесс работы алгоритма при втором просмотре и состояния используемых при формировании предиката областей для ранее рассмотренного уравнения представлен в таблице.

Второй просмотр терма алгоритма построения преобразователя предиката и минимального решения

| Шаг просмотра | Состояние *MASDOX* | Значения | | Состояние терма | Состояние предиката |
|---|---|---|---|---|---|
| | | $i$ | $masdox_i$ | | |
| Вход для 2.2.1.(в) | (∘[( | 4 | ( | $(a \circ [b,\ (X \circ c)])$ ↑↑ | $X$ ↑ ↑ |
| Выход для 2.2.1.(в) Вход для 2.2.1.(б) | (∘[ | 3 | [ | $(a \circ [b,\ (X \circ c)])$ ↑ ↑ | $(X \circ c)$ ↑ ↑ |
| Выход для 2.2.1.(б) Вход для 2.2.1.(а) | (∘ | 2 | ∘ | $(a \circ [b,\ (X \circ c)])$ ↑ ↑ | $(X \circ c)$ ↑ ↑ |
| Выход для 2.2.1.(а) Выход 2-го просмотра | | 0 | | $(a \circ [b,\ (X \circ c)])$ ↑ ↑ | $(X \circ c)$ ↑ ↑ |

3. Генерация минимального решения состоит в подстановке в общий вид линейного разложения выражений, соответствующих выделенному терму и построенному преобразователю предиката.

Замечание. Нетрудно видеть, что вычислительная трудоемкость представленного алгоритма оценивается значением $O(n)$, где $n$ — число символов функциональных форм (элементов множества $C$), образующих терм $E$ в правой части уравнения.

Описанный алгоритм реализован на языке СР/ТРАН [7]. Программа состоит из 528 строк, записанных на этом языке. С помощью программы проведено построение предикатов и минимальных решений 2058 различных, наиболее часто используемых на практике в СФП уравнений, термы в правых частях которых представляли собой композиции термов базиса β с различными уровнями их вложенности. Для обработки указанного количества уравнений потребовалось 184 мин 52 с машинного времени ЭВМ ЕС 1022. В силу того, что при построении предикатов и линейных разложений вручную на каждое такое уравнение уходит около 10 мин, с очевидностью следует, что использование разработанной программы приводит к значительному сокращению времени нахождения минимального решения. А это, в свою очередь, позволяет ускорить процесс доказательства правильности программ в СФП, так как построение минимального решения является одним из наиболее трудоемких шагов этого процесса [2, 4]. При этом следует отметить, что практически отсутствуют ограничения на вложенность термов в реализации алгоритма — они определяются лишь размерностью массива MASDOX, которая при необходимости может быть увеличена в рамках оперативной памяти используемой ЭВМ.

## Список литературы


1. Дробушевич Г. А., Зубович К. А. // Вестн. Белорус. ун-та. Сер. 1: Физ. Мат. Мех. 1988. № 1. С. 46.
2. Зубович К. А., Мелешко А. Н. Там же. 1987. № 1. С. 31.
3. Кутепов В. П. // Программирование. 1975. № 4. С. 3; 1976. № 6. С. 3.
4. Backus J. // Lecture Notes in Computer Science. 1981. V. 107. P. 1.
5. Sun G., Ge Z., Geh R. T. // 1st Int. Conf. Computer and Appl. (Beijing, 20—22 June, 1984). Md.: Silver Spring. 1984. P. 848.
6. Дробушевич Г. А., Зубович К. А. // Программирование. 1987. № 2. С. 8.
7. Дробушевич Г. А., То Туан, Зубович К. А. Там же. 1984. № 3. С. 84.


Поступила в редакцию 19.10.88.