УДК 51.681.3

С.М. Львов

О ТЕХНОЛОГИЯХ ПОСТРОЕНИЯ И ОБРАБОТКИ МАТЕМАТИЧЕСКИХ МОДЕЛЕЙ ПРОГРАММ

В работе рассматриваются проблемы, связанные с построением математических моделей программ, предназначенных для использования в системах автоматизации решения задач анализа и преобразования программ. В качестве такой системы рассматривается система автоматического поиска и доказательства инвариантных равенств в программах. Методы поиска программных инвариантов основаны на анализе свойств предметной области. Определена основная функциональность программного модуля Транслятор этой системы.

Введение

В теоретических исследованиях, посвященных задачам анализа и преобразования программ, объект исследования, как правило, представлен математической моделью программы. При этом используются либо графовые модели программ (пример: U-Y схема программы), либо алгебраические модели программ (пример: выражение в алгоритмической алгебре В.М.Глушкова). Программные системы, проектируемые для решения задач анализа и преобразования программ с целью проведения вычислительных экспериментов или для производственных целей, должны оперировать с исходными текстами программ, написанных на одном из языков программирования высокого уровня. Таким образом, такие системы должны содержать специальные программные модули, осуществляющие трансляцию исходного текста программы в математическую модель. Во многих задачах преобразования и синтеза программ важной является также задача обратной трансляции – преобразования математической модели в исходный код программы. С точки зрения теории программирования задачи трансляции

исходный код программы $\leftarrow\rightarrow$ математическая модель программы

не являются существенно важными, их решение предполагается полученным «по умолчанию». Однако, с точки зрения Software Engineering, т.е. проектных решений и технологий реализации соответствующих программных систем здесь есть проблемы и темы для обсуждения. Основные из них:

- выбор исходного языка программирования и вычислительной платформы;
- выбор математической модели (класса математических моделей);
- методы и технологии трансляции;
- функциональные требования к программному модулю;
- выбор стратегии развития программного модуля.

В работе описан наш первоначальный опыт решения этих проблем, полученный при проектировании и реализации программного модуля Транслятор системы автоматической генерации инвариантов программ. Достаточно полная постановка задачи, решаемой этой системой, дана в [4].

1. Выбор исходного языка программирования и вычислительной платформы

Как правило, в первой версии такого программного модуля, как Транслятор , решается та конкретная задача трансляции, которая необходима для постановки соответствующего вычислительного эксперимента. Поэтому выбор объектного языка программирования и вычислительной среды играет подчиненную роль.

Наш выбор языка Паскаль обусловлен:

- близостью средств управления вычислениями языка к сигнатуре выбранной нами теоретической модели (формулы программной динамической логики);

- наличием описания языка в БНФ-нотации, что позволило использовать технологии автоматизации процедур синтаксического анализа.

Первая версия ПМ Транслятор реализована в среде Win32, хотя нет практически никаких препятствий для переноса ее в Linux/Unix.

Требования к коммерческой версии значительно шире: ПМ Транслятор должен поддерживать, кроме языков с паскалевским синтаксисом (Паскаль, Модула), еще и языки с С-подобным синтаксисом (C, C++, C#, Java).

В некоторых экспериментах нужно иметь дело со специализированными языками (например, с языком охраняемых команд Дейкстры, языком Лисп и т.п.). Поэтому коммерческий программный модуль Транслятор в идеале должен поддерживать описание синтаксиса исходного языка программирования самим пользователем.

2. Выбор математической модели

В теоретических работах, посвященных задаче автоматической генерации инвариантов программ, на которые мы опираемся, используются как графовые модели программ [1, 2], так и алгебраические модели программ [3].

Общими понятиями для этих моделей являются:

- понятие памяти X программы P как конечного множества (вектора) переменных $(x_1,...,x_n)$. $X=(x_1,...,x_n)$;
- понятие алгебры данных D как области определения всех переменных (памяти) программы. Алгебра данных определяет множество операций над данными, допустимыми при программировании. Вообще, D следует рассматривать как многосортную алгебру. Мы полагаем, что одним из сортов D является *Boolean*. В наших задачах, кроме *Boolean*, определен еще только один сорт – сорт, над которым с помощью операторов присваивания определены вычисления;
- понятие состояния $\bar{d}$ памяти программы P как вектора $\bar{d}=(d_1,...,d_n)$. Если память программы находится в состоянии $\bar{d}$, это означает, что $x_1=d_1,...,x_n=d_n$. Таким образом, мы говорим о пространстве состояний памяти как о множестве D^X;
- операторы присваивания в наших задачах интерпретируются как векторные.

Оператор $y=(x_1:=F_1(X),...,x_n:=:=F_n(X))$ преобразует пространство состояний памяти: y: $D^X \rightarrow D^X$. Это означает, что их компоненты выполняются одновременно (параллельно). Множество операторов присваивания обозначим *Y*. Отметим, что в некоторых задачах анализа программ следует рассматривать последовательное выполнение компонент;

- условия преобразуют терм *F* алгебры *D* в (*True, False*). Множество условий обозначим *U*.

2.1. Определение U-Y – программы

U-Y – схемой программы над памятью (*U-Y* – программой) называют инициальный связный граф *P* с выделенным подмножеством заключительных вершин (состояний), дуги которого отмечены парами (условие-оператор).

$$P=<S,E,s_0S^*,\delta>,\ \delta:E\rightarrow(U,Y).$$

Вершины графа *P* – суть контрольные точки программы. Переход от одной точки к другой сопровождается выполнением оператора y, если u в текущем состоянии памяти программы принимает значение *True* (рис.1).

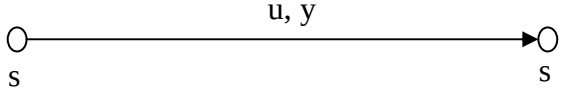

Рис.1

Выполнение программы P начинается в состоянии s_0, осуществляется недетерминировано и заканчивается в одном из заключительных состояний S^*.

2.2. Алгебраические модели программ

Алгебраические модели императивных структурированных программ исполь-

зуют представление программ формулами специальной программной алгебры, операциями которой являются операторы управления: последовательное выполнение, ветвление, повторение. Например: $P;Q$ – последовательное выполнение; $P \underset{u}{\vee} Q$ – ветвление с условием; $\{P\}_u$ – повторение с предусловием; $\{P\}_u$ – повторение с постусловием.

Пример программы и ее алгебраической модели

Рассмотрим в качестве примера программы и ее алгебраической модели расширенный алгоритм Евклида, который находит $b = GCD(x, y)$ и такие числа c, d, что $c*x + d*y = b$.

$U = (u_1, u_2)$ – множество условий программы;

$$u_1 = (r >= b);$$

$$u_2 = (r <> 0);$$

$Y = (y_1, y_2, y_3)$ – множество операторов присваивания программы;

$$y_1 = (a := x, b := y, c := 0, d := 1, u := 1, v := 0, q := 0, r := x);$$

$$y_2 = (r := r - b,\ q := q + 1);$$

$$y_3 = (a := b, b := r, c := u - q * c, d := v - q * d, u := c, v := d,\ q := 0, r := b).$$

Алгебраическая модель программы теперь определяется формулой

$$P = y_1; \{y_2\}_{u_1}; \{y_3; \{y_2\}_{u_1}\}_{u_2}.$$

```
Procedure ExtGCD(x, y: Integer; var b, c, d: Integer);
Var
  a, u, v, q, r : Integer;
Begin
  a:=x; b:=y;
  c:=0; d:=1;
  u:=1; v:=0;
  q:=0; r:=x
  While r >= b do begin
    r:=r-b;
    q:=q+1
  End;
  While r<>0 do begin
    a:=b; b:=r;
    c:=u-q*c; d:=v-q*d;
    u:=c; v:=d;
    q:=0; r:=b
    While r >= b do begin
      r:=r-b;
      q:=q+1
    End
  End
End;
```

Для задач анализа программ из текста программы нужно сформировать:

$E = (x, y, b, c, d)$ – множество входных параметров программы;

$X = (a, b, c, d, q, r, u, v, x, y)$ – память программы;

2.3. Инварианты программ

Поскольку программный модуль Транслятор в первой версии разрабатывается для реализации алгоритмов поиска инвариантов некоторого класса программ, сформулируем понятие программного инварианта более точно.

Алгеброй данных анализируемых программ является поле F. Если все правые части операторов присваивания программы – многочлены, т.е. не содержат операции деления на выражения, зависящие от переменных, такую программу называют интерпретированной над полиномиальным кольцом $F[X]$. Если же операции деления на выражения, зависящие от переменных, в программе используются, такую программу называют интерпретированной над полем $F(X)$.

Многочлен $i(X) \in F[X]$ называется полиномиальным инвариантом программы P, если $\square \{True\}\ P\ \{i(X)=0\}$.

Это означает, что при любом начальном состоянии памяти $\boldsymbol{a} = (a_1, \ldots, a_n)$, если программа P завершает работу, для заключительного состояния $\boldsymbol{b}$ (т.е. такого, что $\boldsymbol{a}\{P\}\boldsymbol{b}$) выполняется равенство $i(b) = 0$.

Множество всех полиномиальных инвариантов программ, интерпретированных либо над кольцом $F[X]$, либо над полем $F(X)$, образует радикальный идеал этого кольца, который будем обозначать I_p. В дальнейшем будем рассматривать множество всех элементов I_p , степени которых ограничены некоторой константой M. Множество таких многочленов образует векторное пространство, которое обозначается $I_p^{(M)}$. Понятно, что $I_p^{(M)} \subset I_p$.

Основные задачи исследования – построение и вычислительные эксперименты с программной системой, которая доказывает инвариантность заданного полинома $i(X) \in F[X]$, а также строит базис векторного пространства $I_p^{(M)}$ для заданного значения M.

Отметим, что проблема построения базиса идеала I_p алгоритмически неразрешима [3]. В доказательстве этого утверждения существенную роль играет то, что в программах используются условия типа равенств и их отрицания. Поэтому условия программ приходится игнорировать, считая, что программные циклы завершаются недетерминировано. Даже при этих ограничениях проблема построения базиса I_p остается открытой.

Анализ методов решения задачи автоматического доказательства и автоматической генерации программных инвариантов для программ, интерпретированных над конкретными предметными областями показал, что:

1) алгоритмы решения основных задач формулируются в терминах предметной области и, по существу, не зависят от типа модели программы. Графовая модель должна использоваться только для управления вычислениями инвариантов;

2) построение графовой модели в виде U-Y схемы программы требует преобразования операторов присваивания в охраняемые команды Дейкстры. Эти преобразования несущественны для теории, но требуют дополнительных ресурсов для реализации;

3) алгебраическая модель программы в гораздо большей степени адекватна исходному тексту программы, написанному на языке структурного программирования. Ее легче и получать, и обрабатывать. Разумеется, программы, написанные с нарушением правил структурирования, нуждаются в дополнительной обработке.

Кроме того, это наиболее важно, алгоритмы решения основных задач и общий алгоритм управления вычислениями (в наших задачах автоматического доказательства и автоматической генерации программных инвариантов) могут быть реализованы в средах алгебраического программирования (APS, Obj). Выбор алгебраической модели программы приводит к тому, что весь основной модуль системы может быть реализован в среде алгебраического программирования, что существенно ускоряет построение функционального прототипа и окончательной версии системы.

3. Методы и технологии трансляции

Различные классы задач анализа и преобразования программ используют различные сведения об объектной программе. Так, классическая задача оптимизации памяти программы использует сведения о

вхождении переменных в левые и правые части операторов присваивания объектной программы. Задача оптимизации программы на основе анализа ее линейных участков кода требует сведений обо всех операторах присваивания линейных участков. Задачи преобразований программ основаны на преобразованиях условных операторов управления. В задаче автоматической генерации инвариантов программ на основе свойств предметной области, как уже отмечалось, условия в операторах управления приходится игнорировать, так как их учет приводит к алгоритмической неразрешимости.

Эти обстоятельства приводят к следующему решению: результат трансляции исходного кода программы в ее алгебраическую модель должен, как минимум, содержать следующие компоненты: список входных переменных программы; список всех переменных (память) программы; алгебраическое выражение, описывающее структуру управления программы с точностью до условий и операторов управления; словарь условий; словарь операторов присваивания.

Все эти компоненты модели представлены в нашем примере.

В работе [4], теорема 1, а, определена общая схема вычислений для решения задачи доказательства инвариантности полинома и сформулированы основные задачи для реализации этой схемы. Реализация общей схемы вычислений требует:

1) процедуры элиминации всех условий из формулы *P*. В нашем примере элиминацией условий выражение $P = y_1;\{y_2\};\{y_3;\{y_2\}\}$ редуцируется до $P = y_1;\{y_2\};\{y_3;\}y_2\}\};$

2) представления всех операторов присваивания программы в векторном виде. В нашем примере все операторы из *Y* уже представлены в этом виде;

3) полиномиальной канонизации правых частей всех операторов присваивания из *Y*. Это представление связано с выбором и фиксацией некоторого лексикографического упорядочения на множестве мономов и представления полиномов в виде суммы мономов. В нашем примере это уже сделано;

4) процедуры анализа формулы *P* программы, заключающейся в выделении главной операции программы, рекурсивной обработке ее операндов и последующей обработке главной операции.

Все эти процедуры легко реализуются в системах алгебраического программирования, в частности, в системе APS-1 [5].

Алгоритмы решения основных задач основаны на построении базисов Гребнера полиномиальных идеалов. Таким образом, все идеалы, которые строятся в алгоритме, должны быть представлены своими базисами Гребнера. В дальнейшем мы будем обозначать тот факт, что идеал **I** представлен базисом Гребнера $\mathbf{f_1, \ldots, f_k}$ равенством $\mathbf{I = (f_1, \ldots, f_k)_{Gr}}$. Базис Гребнера идеала **I** будем обозначать **Gr(I)**.

Наш алгоритм должен решать следующие задачи:

1. Для $\mathbf{I = (f_1, \ldots, f_k)_{Gr}}$, $\mathbf{J = (g_1, \ldots, g_l)_{Gr}}$ найти $\mathbf{I \cap J = (h_1, \ldots, h_m)_{Gr}}$.
2. Для $\mathbf{I = (f_1, \ldots, f_k)_{Gr}}$, оператора присваивания **X := F(X)**, найти $\mathbf{J = (g_1, \ldots, g_l)_{Gr}}$, т.е. найти $\mathbf{Gr((f_1(H(X)), \ldots, f_k(H(X))))}$.
3. Для $\mathbf{I = (f_1, \ldots, f_k)_{Gr}}$, $\mathbf{J = (g_1, \ldots, g_l)_{Gr}}$ распознать $\mathbf{I \subseteq J}$.

Задачи 1 и 3 – классические задачи теории полиномиальных идеалов, решение которых в терминах базисов Гребнера хорошо известно [6]. Для решения задачи 2 можно воспользоваться общим алгоритмом пополнения, который строит базис Гребнера по произвольному конечному множеству полиномов. Таким образом, в системе алгебраического программирования должны быть реализованы вышеотмеченные вычисления в базисах Гребнера.

В нашем примере следует доказать инвариантность полинома $c*x + d*y - b$.

Рассмотрим теперь вопросы, связанные с реализацией алгоритма вычисления базиса векторного пространства $I_p^{(M)}$ ([С. Львов], теорема 1, б). Там предлагается подход, связанный с заданием общего

вида инварианта в виде полиномиальной формы, т.е. многочлена «специального» вида с неопределенными коэффициентами. Например, можно определить общий вид искомых инвариантов как линейную комбинацию нескольких многочленов с неопределенными коэффициентами:

$$I(X) = a_1*P_1(X)+\ldots+a_k*P_k(X).$$

Если, например, требуется искать линейные инварианты, следует положить

$$P_1= x_1, \ldots, P_n = x_n, P_{n+1} = 1.$$

Можно, к примеру, искать все инварианты вида $I(X) = a_1*x_1^2 +\ldots+a_n * x_n^2$ и т.п.

Таким образом, в проектируемой системе следует реализовать процедуру ввода и редактирования этой формы. Если в нашем примере ограничиться поиском инвариантов 2-ой степени, в качестве этой формы нужно выбрать многочлен 2-ой степени от всех переменных программы. В результате вычислений получаем базис $I_p^{(2)}$:

$$i_1 = c*x + d*y - b;\ i_2 = u*x + v*y - a.$$

Следовательно, если нас интересуют инварианты 2-ой степени в терминах входных параметров программы, в качестве этой формы нужно выбрать многочлен 2-ой степени от параметров программы, т.е множества $E = (x, y, b, c, d)$. В результате вычислений получаем

$$i_1 = c*x + d*y - b.$$

Проведение вычислительных экспериментов такого рода с целью уточнения функциональности системы представляет определенный интерес.

4. Функциональные требования

Исходя из вышеприведенных рассмотрений, сформулируем общие функциональные требования к системе.

Данные о задаче, обрабатываемой системой, содержат следующие компоненты:

- исходный код программы;
- математическую модель программы;
- результат вычислительного эксперимента.

Программная система, проектируемая для выполнения выше описанного вычислительного эксперимента, должна поддерживать следующие функции:

1) хранение библиотеки исходных кодов, математических моделей программ, а также результатов вычислительного эксперимента (функции Save, Load). Предполагается, что исходный код программы синтаксически и семантически правельный, а также удовлетворяет тем ограничениям, которые неизбежно возникают при постановке задачи теоретического исследования. Эта библиотека используется как для тестирования программной системы, так и для проведения вычислительного эксперимента;

2) трансляция исходного кода в математическую модель (функция Translate) с возвратом результатов трансляции;

3) редактирование исходного кода с клавиатуры (функция EditCode) для обработки исходного кода перед построением его математической модели;

4) редактирование математической модели с клавиатуры (функция EditModel) для обработки модели независимо от исходного кода;

5) интерфейс к среде алгебраического программирования с передачей на исполнение математической модели программы и возвратом результатов (функция Execute);

6) ввод дополнительной информации о задаче для вычислительного эксперимента и настройках математической модели (выбор компонентов математической модели, передаваемых на исполнение функцией Execute). Пример: в задаче автоматического доказательства инвариантности равенства такой дополнительной информацией является равенство, инвариантность которого доказывается;

7) стандартные сервисы ОС, такие, как возможность распечатки результатов и т.п. (рис. 2, 3).

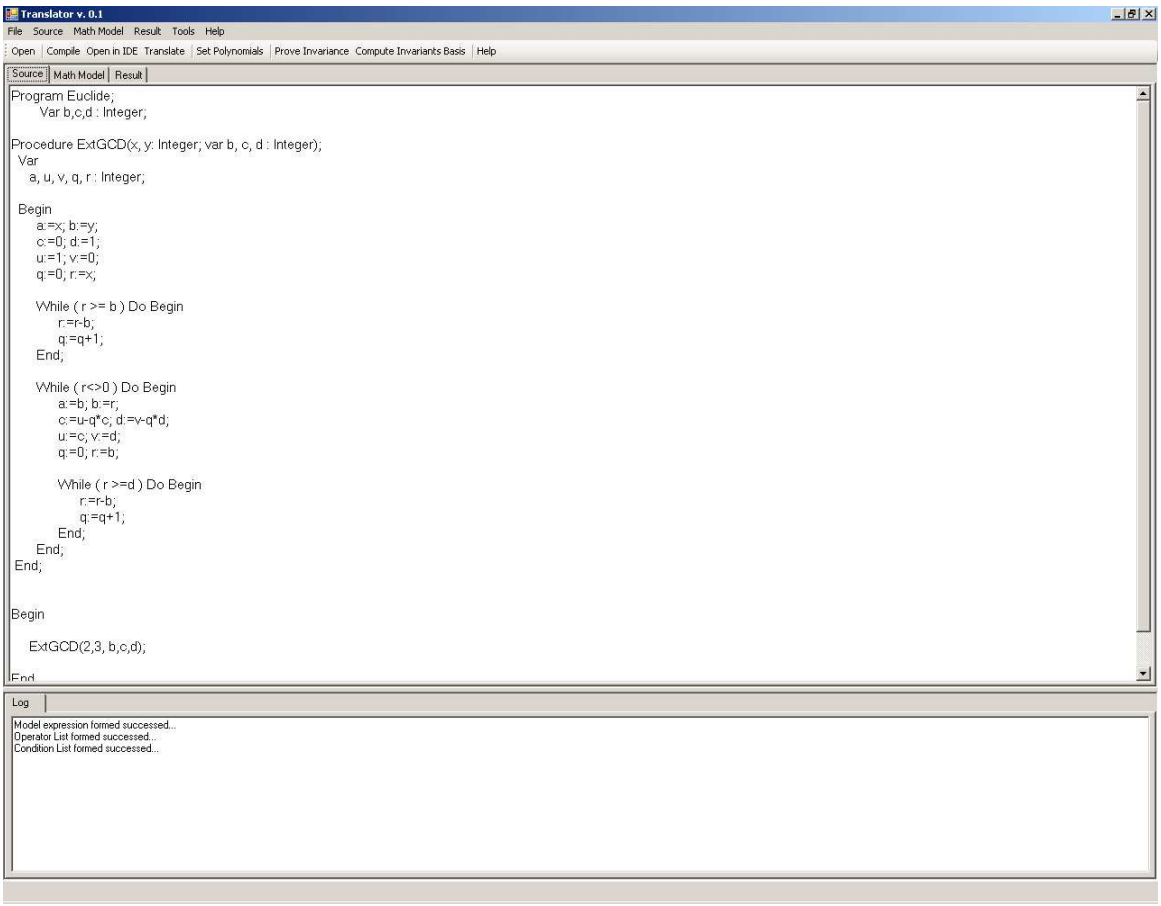

Рис. 2. Общий вид главного окна системы с примером исходного кода программы

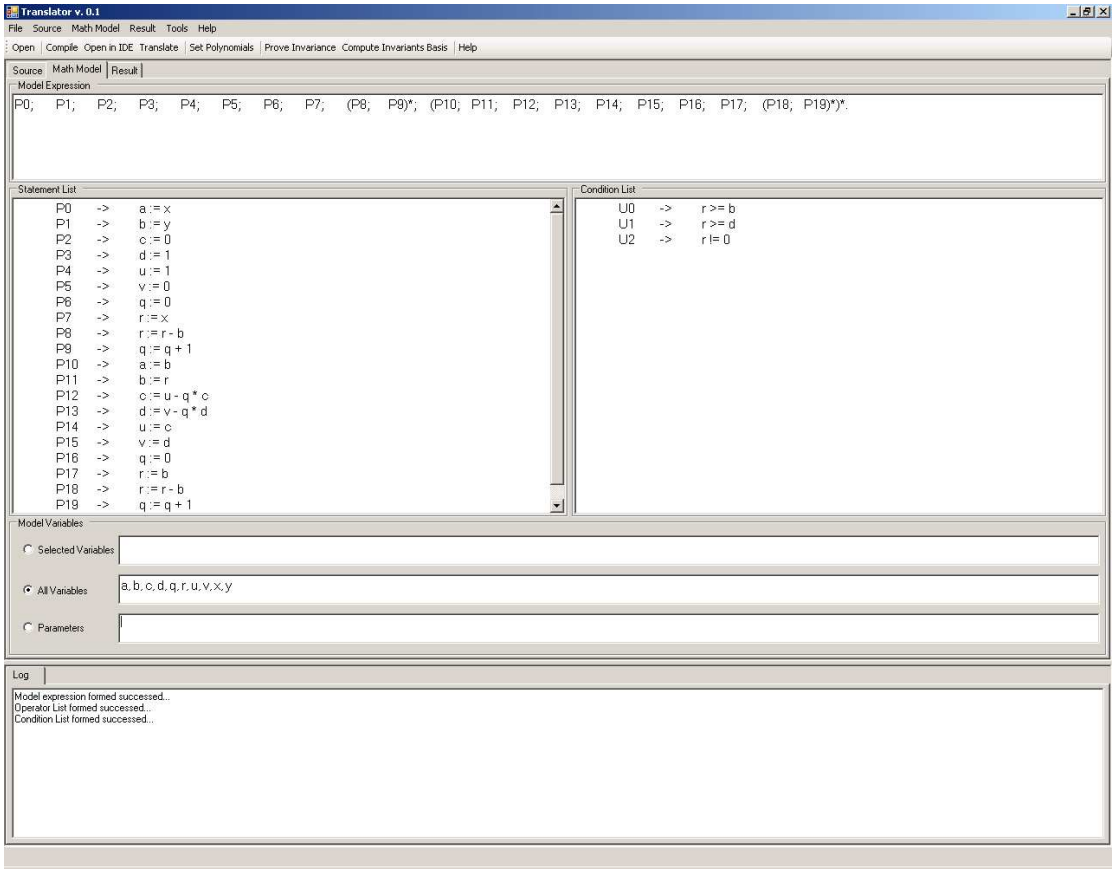

Рис. 3. Общий вид главного окна системы с примером математической модели

Выводы

Использование ПМ Транслятор и программной системы, основная функциональность которой описана в работе, позволяет сократить время и минимизировать подготовительную работу к проведениям вычислительных экспериментов в задачах анализа и преобразования программ. Выбор алгебраической модели программы обусловлен возможностью использования систем алгебраического программирования и, для многих классов задач, методов компьютерной алгебры. Развитие систем такого рода предполагает как расширение и универсализацию модуля Транслятор, так и развитие систем алгебраического программирования, которые являются главным инструментом исследования.

1. *Летичевский А.А.* Об одном подходе к анализу пограмм // Кибернетика. – 1979. – № 6. – С. 1 – 8.
2. *Годлевский А.Б., Капитонова Ю.В., Кривой С.Л., Летичевский А.А.* Итеративные методы анализа программ // Кибернетика. – 1984. – № 2. – С. 23 – 28.
3. *Львов М.С.* Инвариантные равенства малых степеней в программах, опередеенных над полем // Кибернетика. – 1988. – № 1. – С. 108 – 110.
4. *Львов С.М.* О реализации вычислений в задачах анализа программ, определенных над векторными пространствами//Проблемы программирования. – 2004. – № 2 – 3. Спец. вып. С. 95 – 101.
5. *Letichevsky A., Kapitonova J., Volkov V., Chugajenko A., Chomenko V.* Algebraic programming system APS (user manual). Glushkov Institute of Cybernetics, National Acad. of Sciences of Ukraine. – Kiev. – 1998. – 50 p.
6. *Бухбергер Б., Калме Ж., Калтофен Э.* Компьютерная алгебра: символьные и алгебраические вычисления: Пер. с англ. – М.: Мир, 1986. – 392 с.

Получено 23.01.2007

Об авторе:
Львов Сергей Михайлович,
аспирант Института кибернетики
им. В.М. Глушкова НАН Украины.

Место работы автора:
Институт кибернетики им. В.М. Глушкова
НАН Украины,
Киев 187,
проспект Академика Глушкова, 40.
Тел.: 424 7804,
моб. +38(095)1751522,
email: aksserg@yandex.ru.