УДК 519.6:539.3

А. Н. Максимец

Поиск программных инвариантов в виде полиномов

(Представлено членом-корреспондентом НАН Украины С. И. Ляшко)

Представлено решение проблемы поиска инвариантов программ в виде полиномиальных зависимостей методом верхней аппроксимации. Этот итерационный метод, с успехом примененный к программам над абсолютно свободными алгебрами и векторными пространствами данных, адаптирован для кольца полиномов. Множество инвариантов в этом случае представляется в виде идеала кольца полиномов. Решены задачи о соотношениях и о пересечении множеств инвариантов с использованием базисов Гребнера при условии невырожденности оператора присваивания.

1. После прорыва в теории верификации, связанного с появлением теории индуктивных утверждений Флойда–Хоара–Дейкстры [1] в 70-х гг. (Вегбрейт, 1974, 1975; Герман и Вегбрейт, 1975; Кац и Манна, 1976; Куазот и Куазот, 1976; Сузуки и Ишихата, 1977; Дершовицем и Манна, 1978), наступил тихий период в этой области. Оживление последовало с изучением средств автоматического доказательства теорем (САДТ) и верификации моделей программ вместо исходных программ. Все перечисленные подходы формальных доказательств свойств программы используют утверждения, которые истинны во время выполнения программы в качестве входных данных. Проблема поиска таких увтерждений (инвариантов) остается особенно актуальной и по сей день.

Исследования в этой области велись и в СССР, особенно в Киеве и Новосибирске, в 70-х и 80-х гг. прошлого столетия. В результате были разработаны эффективные алгоритмы генерации инвариантов для программ. Проводились исследования следующих алгебр данных: абсолютно свободная алгебра данных [4, 12], группы, полугруппы, абелевы группы и абелевы полугруппы, векторное пространство [3], кольцо многочленов [9, 10].

Программа представлена в виде U–Y-схемы на алгебре многочленов. В данной работе итерационные алгоритмы, удачно примененные для абсолютно свободных алгебр и векторных пространств, адаптипированы для полиномиальных пространств [7].

Инвариант U–Y-схемы в состоянии — это утверждение, которое верно для любого достижимого состояния памяти U–Y-схемы. Термин "состояние" далее обозначает состояние (узел) U–Y-схемы. Рассматриваемый подход строит базис инвариантых соотношений для каждого состояния программы, учитывая переданные соотношения на вход U–Y-схемы.

Эта работа была вызвана схожими работами по генерации инвариантов в виде полиноминомов с использованием базисов Гребнера (Мюллер-Олм и Зайдель, 2004; Санкаранараянан, 2004; Родригес-Карбонелл и Капур, 2007). Рассмотренный итерационный метод позволяет генерировать более широкое множество инвариантов, что аргументирует его применение.

2. Обозначения. Пусть A — U–Y-схема над памятью [5] и $R = \{r1, \ldots, r_m\}$ — множество переменных, которое определено на алгебре данных (D, Ω); $K(\Omega, Eq)$ — класс алгебр, который включает в себя алгебру (D, Ω) [2]. Рассмотрим кольцо многочленов (D, Ω) и алгебру термов $T(\Omega, R)$ на R из класса $K(\Omega, Eq)$.

© А. Н. Максимец, 2013

Определение 1 (алгебраическое соотношение). Алгебраическим соотношением Ψ называется соотношение вида $\bigwedge\limits_i p_i(r_1, \ldots, r_m) = 0$, где каждый $p_i \in \Re[r1, \ldots, r_m]$. Степень соотношения — это максимальная степень среди степеней многочленов, которые составляют соотношение.

Определение 2 (идеал). Множество $I \subseteq \Re[r1, \ldots, r_n]$ называется идеалом тогда и только, когда:

1) $0 \in I$;

2) если $p_1,\ p_2 \in I$, то $p_1 + p_2 \in I$;

3) если $p_1 \in I$ и $p_2 \in \Re[r_1, \ldots, r_n]$, то $p_1 p_2 \in \Re[r_1, \ldots, r_n]$.

Идеалом, порожденным множеством многочленов N (обозначается $((N))$), называется минимальный идеал, содержащий N. Это равносильно $((N)) = \{g_1 p_1 + \cdots + g_m p_m \mid g_1, \ldots, \ldots, g_m \in R[r_1, \ldots, r_n], p_1, \ldots, p_m \in P\}$. Идеал I считается конечно порожденным, если существует конечное множество N, так, что $I = ((N))$. Знаменитая теорема Гильберта утверждает, что все идеалы в $\Re[r1, \ldots, r_n]$ являются конечнопорожденными. В результате алгебраические соотношения можно рассматривать как порождающие идеалы и наоборот. Любой идеал определяет многообразие, которое является множеством общих нулей всех полиномов, которые он содержит.

Определение 3 (пересечение идеалов). Множество K является пересечением идеалов $I = \{f_1, \ldots, f_l\}$ и $J = \{g_1, \ldots, g_m\}$, если

$$K = \left\{ s(r_1, \ldots, r_n) \mid s(r_1, \ldots, r_n) = \sum_{i=1}^{l} p_i f_i = \sum_{j=1}^{m} q_j g_j,\ p_1, \ldots, p_l, q_1, \ldots, q_m \in \right.$$

$$\left. \in \Re[r_1, \ldots, r_n] \right\}.$$

Теорема 1 (пересечение идеалов). *Пусть I и J — идеалы в $R[r_1, \ldots, r_n]$*

$$I \bigcap J = (tI + (1-t)J) \bigcap \Re[r_1, \ldots, r_n], \tag{1}$$

где $t \in R$ — новая введенная переменная [6].

Доказательство. Заметим, что $tI + (1-t)J$ является идеалом в $\Re[r_1, \ldots, r_n, t]$. Для доказательства истинности равенства мы докажем включение в обе стороны.

Пусть $f \in I \bigcap J$. Поскольку $f \in I$, то $tf \in tI$. Аналогично $f \in J$ следует $(1-t)f \in (1-t)J$. Получаем, что $f = tf + (1-t)f \in tI + (1-t)J$ и $I,\ J \subset \Re[r_1, \ldots, r_n]$.

Чтобы доказать включение в обратную сторону, допустим $f \in (tI + (1-t)J) \bigcap \Re[r_1, \ldots, r_n]$. Тогда $f(r) = g(r,t) + h(r,t)$, где $g(r,t) \in tI$ и $h(r,t) \in (1-t)J$. Сначала рассмотрим вариант $t = 0$. Исходя из того, что каждый элемент из tI умножается на t, $g(r,0) = 0$, $f(r) = h(r,0)$ и $f(r) \in J$. Рассмотрим вариант $t = 1$, тогда f можно представить в виде $f(r) = g(r,t) + h(r,t)$. Каждый элемент $(1-t)J$ умножается на $1-t$ и в результате $h(r,1) = 0$. В этом случае $f(r) = g(r,1)$ и $f(r) \in I$. Исходя из того, что f принадлежит I и J, $f \in I \bigcap J$. Получаем $I \bigcap J \supset (tI + (1-t)J) \bigcap \Re[r_1, \ldots, r_n]$, что и завершает доказательство.

Пусть $A = \{a_0, a_1, \ldots, a_*\}$ — множество вершин U–Y-схемы; N_{a_i} — базис соотношений в вершине a_i на текущем шаге метода; N_{a_0}, $N_{a_1}, \ldots, N_{a_*}$ — базисы соотношений в вершинах U–Y-схемы; U — множество условий в виде $u = (p(r_1, \ldots, r_n) = 0)$, где $p(r_1, \ldots, r_n) \in \in \Re[r_1, \ldots, r_n]$; Y — множество операторов присваивания в виде $r_i := p(r_1, \ldots, r_n)$, где $p(r_1, \ldots, r_n) \in \Re[r_1, \ldots, r_n]$.

3. Метод верхней аппроксимации. Рассмотрим псевдокод метода верхней аппроксимации(МВА) с [2].

Вход: U–Y-схема A, N_0 — базис инвариантов в вершине a_0.

Выход: N_{a_0}, $N_{a_1}, \ldots, N_{a_*}$ — базисы инвариантов для каждой вершины.

$N_{a_0} := N_0$
ToVisit.push(a_0)
Visited := {}
while ToVisit $\neq \varnothing$ **do**
  c := ToVisit.pop()
  Visited := Visited $\bigcup$ c
  for all (c, y, a') **do**
    if Not a' **in** Visited **then**
      $N_{a'} := ef(N_{a'}, y)$
      ToVisit.push(a')
    end if
  end for
end while
ToVisit := $A/\{a_0\}$
while ToVisit $\neq \varnothing$ **do**
  $c := take\ from$ ToVisit
  if $\mathrm{N_c} \neq \varnothing$ **then**
    $N := N_c$
    forall (a', y, c) **do**
      $N := N \bigcap ef(N_{a'}, y)$
    end for
    if $(N \neq N_c)$ **then**
      $N_c := N$
      ToVisit := ToVisit $\bigcup \{a \mid for\ every\ (c, y, a)\}$
    end if
  end if
end while

Для применения метода для алгебры данных в виде кольца полиномов нужно решить три задачи: задача о соотношениях ($ef(\ldots)$), пересечении ($\bigcap$) и стабилизации.

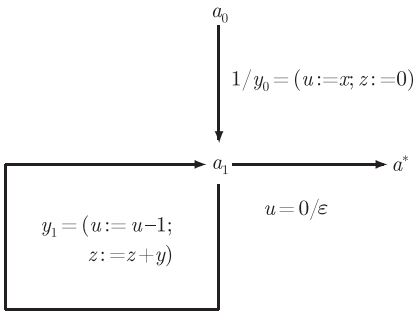

Рис. 1. U–Y-программа УМН (x, y)

Задача о соотношениях. Для данного базиса соотношений M и оператора присваивания $y \in Y$ построить базис соотношений $ef(M, y)$, которые истинны после выполнения оператора присваивания.

В работе рассмотрен частный случай, когда оператор присваивания обратимый. В этом случае равенство, которое представляет присваивание $r_i' = p(r_1, \ldots, r_n)$, можно представить в виде $r_i = p'(r_1, \ldots, r_i', \ldots, r_n)$, где r_i' — новое значение переменной. Заменив старую переменную ее значением относительно нового значения переменной, получим базис соотношений, справедливый после выполнения оператора присваивания.

Задача пересечения. По заданным базисам множеств соотношений I и J построить базис множества соотношений $I \bigcap J$. Согласно теореме 1 о пересечении идеалов, пересечение можно построить, используя формулу (1).

Задача стабилизации. Доказать, что процесс работы метода построения базисов соотношений конечен. Исследование этой задачи не представлено в данной работе.

4. Реализация. Рассмотрим U–Y-программу из рис. 1. На этой программе мы продемонстрируем роботу алгоритма, считая алгебру данных кольцом полиномов [8]. Применяя алгоритм МВА к этой U–Y-программе, получаем нижеприведенную цепочку вычислений и инвариантные соотношения.

На первом этапе алгоритма МВА генерируем начальные множества соотношений. Сначала множествам соотношений для всех состояний программы присваивается значение 1 $(\forall P\colon P \overset{*}{\bigcap} 1 = P)$, при этом ToVisit$= \{a_0\}$.

Далее рассматриваем состояние a_0 из ToVisit, фиксируя Visited$= \{a_0\}$. Из состояния a_0 есть один переход $(a_0, 1, y = (u := x; z := 0), a_1)$.

Анализируя этот переход, множеству соотношений P_{a_1} присваиваем

$$P_{a_1} := ef(P_{a_0}, y) = ef(1, (u := x, z := 0)) = \{u - x = 0, z = 0\}.$$

При этом ToVisit $= \{a_1\}$.

Переходим к состоянию a_1 из ToVisit, фиксируя Visited $= \{a_0, a_1\}$. Из состояния a_1 есть два перехода $(a_1, (u \neq 0), (u := u - 1; z := z + y), a_1)$, $(a_1, (u = 0), \varepsilon, a^*)$. Исходя из $a_1 \in$ Visited, рассмотрению подлежит только переход из a_1 в a^*.

Анализируя переход $(a_1, (u = 0), \varepsilon, a^*)$, находим целесообразным заменить этот переход следующим переходом $(a_1, 1, u := 0, a^*)$. То есть заменить обязательное условие перехода оператором присваивания. Множеству соотношений P_{a^*} присваиваем:

$$P_{a^*} := ef(P_{a_1}, y) = ef(\{u - x = 0, z = 0\}, \{u := 0\}) = \{x = 0, z = 0\}.$$

При этом ToVisit $= \{a^*\}$.

Переходим к состоянию a^* из ToVisit, фиксируя Visited $= \{a_0, a_1, a^*\}$. Нет переходов из состояния a^*. Исходя из ToVisit $= \varnothing$, первый этап МВА заканчиваем. Переходим ко второму этапу алгоритма.

На втором этапе алгоритма мы на каждом шаге итерации генерируем соотношения для состояний. После пересекаем множества соотношений с полученными на предыдущем шаге. При инициализации ToVisit $= A/\{a_0\} = \{a_1, a^*\}$.

Переходим к состоянию a_1 из ToVisit $P = P_{a_1} = \{u - x = 0, z = 0\}$. В состояние a_1 есть два перехода $(a_0, 1, (u := x; z := 0), a_1)$, $(a_1, (u \neq 0), (u := u - 1; z := z + y), a_1)$.

Анализируя переход $(a_0, 1, (u := x; z := 0), a_1)$, множеству соотношений P присваиваем

$$P := P\overset{*}{\bigcap} ef(P_{a_0}, y) := P\overset{*}{\bigcap} ef(1, (u := x; z := 0)) =$$

$$= \{u - x = 0, z = 0\}\overset{*}{\bigcap}\{u - x = 0, z = 0\} = \{u - x = 0, z = 0\}.$$

Анализируя переход $(a_1, (u \neq 0), (u := u - 1; z := z + y), a_1)$, множеству соотношений P присваиваем

$$P := P\overset{*}{\bigcap} ef(P_{a_1}, y) := P\overset{*}{\bigcap} ef(\{u - x = 0, z = 0\}, (u := u - 1; z := z + y)) =$$

$$= \{u - x = 0, z = 0\}\overset{*}{\bigcap}\{u + 1 - x = 0, z - y = 0\} =$$

$$= \{z + (u - x)y = 0, z(y - z) = 0, zx - zu - z = 0, (u - x)^2 + u - x = 0\}.$$

Исходя из $P \neq P_{a_1}$, присваиваем

$$P_{a_1} := \{z + (u - x)y = 0, z(y - z) = 0, zx - zu - z = 0, (u - x)^2 + u - x = 0\}$$

и $\quad$ ToVisit := ToVisit $\bigcup\{a_1, a^*\} = \{a_1, a^*\}$.

Переходим к состоянию a^* из ToVisit, фиксируя Visited $= \{a_1\}$ и $P = P_{a^*} = \{x = 0, z = 0\}$. В состояние a^* есть один переход $(a_1, 1, u := 0, a^*)$.

Анализируя переход $(a_1, 1, u := 0, a^*)$, множеству соотношений P присваиваем:

$$P := P\overset{*}{\bigcap} ef(P_{a_1}, y) = P\overset{*}{\bigcap} ef(\{z + (u - x)y = 0, z(y - z) = 0, zx - zu - z = 0,$$

$$(u - x)^2 + u - x = 0\}, u := 0) = P\overset{*}{\bigcap}\{xy - z = 0, zy - z^2 = 0, zx - z = 0, x^2 - x = 0\} =$$

$$= \{x = 0, z = 0\}\overset{*}{\bigcap}\{xy - z = 0, zy - z^2 = 0, zx - z = 0, x^2 - x = 0\} =$$

$$= \{z(x - 1) = 0, z(y - z) = 0, (x^2 - x)u = 0, (xy - z)u = 0\}.$$

Исходя из $P \neq P_{a^*}$, присваиваем $P_{a^*} := \{z(x - 1) = 0, z(y - z) = 0, (x^2 - x)u = 0, (xy - z)u = 0\}$.

Переходим к состоянию a_1 из ToVisit, фиксируя Visited $= \varnothing$ и $P = P_{a_1} = \{z+(u-x)y = 0, z(y-z) = 0, zx - zu - z = 0, (u-x)^2 + u - x = 0\}$. В состояние a_1 есть два перехода $(a_0, 1, (u := x; z := 0), a_1)$, $(a_1, (u \neq 0), (u := u-1; z := z+y), a_1)$.

Анализируя переход $(a_0, 1, (u := x; z := 0), a_1)$, множеству соотношений P присваиваем:

$$P := P\overset{*}{\bigcap} ef(P_{a_0}, y) := P\overset{*}{\bigcap}\{u - x = 0, z = 0\} = \{z + (u-x)y = 0, z(y-z) = 0,$$

$$zx - zu - z = 0, (u-x)^2 + u - x = 0\}\overset{*}{\bigcap}\{u - x = 0, z = 0\} =$$

$$= \{(x-u)y - z = 0, zy - z^2 = 0, zx - zu - z = 0, (u-x)^2 + u - x = 0\}.$$

Анализируя переход $(a_1, (u \neq 0), (u := u-1; z := z+y), a_1)$, множеству соотношений P присваиваем

$$P := P\overset{*}{\bigcap} ef(P_{a_1}, y) := P\overset{*}{\bigcap} ef(P_{a_1}, (u := u-1; z := z+y)) = \{xy - uy - z = 0, \ldots\}.$$

Исходя из $P \neq P_{a_1}$, присваиваем

$$P_{a_1} := \{xy - uy - z = 0, \ldots\} \qquad \text{и} \qquad \text{ToVisit} := \text{ToVisit}\bigcup\{a_1, a^*\} = \{a_1, a^*\}.$$

Переходим к состоянию a^* из ToVisit, фиксируя Visited $= \{a_1\}$ и $P = P_{a^*} = \{z(x-1) = 0, z(y-z) = 0, (x^2 - x)u = 0, (xy - z)u = 0\}$. В состояние a^* есть один переход $(a_1, 1, u := 0, a^*)$.

Анализируя этот переход, множеству соотношений P присваиваем: $P := P\overset{*}{\bigcap} ef(P_{a_1}, u := 0) := P\overset{*}{\bigcap}\{xy - z = 0, \ldots\} = \{xy - z = 0, \ldots\}$.

Исходя из $P \neq P_{a^*}$, присваиваем $P_{a^*} := \{xy - z = 0, \ldots\}$.

Выполняя действия по алгоритму, мы заметили, что с каждой новой итерацией размерность базисов P_{a_1} и P_{a^*} остается неизменной, так же, как и один из многочленов этих базисов. Это многочлен $xy - uy - z = 0$ в a_1 и $xy - z = 0$ в a_*. Как мы видим из постановки задачи, инвариант $z = xy$ подтверждает правильность программы. В этом примере работа автоматического доказчика не понадобилась.

Таким образом, в работе представлены теоретические обоснования и реализация метода верхней аппроксимации для программ над кольцами полиномов. Инварианты программ приведены в виде идеалов. Идеалы, в свою очередь, задаются базисами Гребнера, операции над которыми удовлетворяют требованиям по применению МВА, изложенным в [2]. Метод реализован в программном коплексе Maple, который содержит мощные инструменты для символьных вычислений. Приведен пример функционирования разработанного метода.

1. *Hoare T.* The verifying compiler: A Grand challenge for computing research // J. of the ACM. – 2003. – No 50(1) – P. 63–69.
2. *Годлевский А. Б., Капитонова Ю. В., Кривой С. Л., Летичевский А. А.* Итеративные методы анализа программ. Равенства и неравенства // Кибернетика. – 1990. – № 3. – С. 1–9.
3. *Кривой С. Л.* Об одном алгоритме поиска инвариантных соотношений в программах // Там же. – 1981. – № 5. – С. 12–18.
4. *Кривой С. Л.* О поиске инвариантных соотношений в программах // Математическая теория проектирования вычислительных машин. – Киев: Изд. Ин-та кибернетики АН УССР, 1978. – С. 35–51.

5. *Летичевский А. А.* О поиске инвариантных соотношений в программах // Алгоритмы в современной математике – 1981. – № 122. – С. 304–314.
6. *Gröbner* bases and applications / Eds. B. Buchberger, F. Winkler. – Cambridge: Cambridge Univ. Press, 1998. – 552 p.
7. *Максимец А. Н.* Поиск инвариантов U–Y-программ итерационным алгоритмом над совершенно свободными алгебрами данных // Пробл. програмування. – 2012. – № 2–3. – С. 228–333.
8. *Maksymets O. M.* Upper approximation method for polynomial invariants // Theoretical and Applied Aspects of Cybernetics. – Proceedings of the 2nd Intern. Scientific Conf. of Students and Young Scientists. Computer Science Section. – Kyiv, 2012. – P. 45–47.
9. *Lvov M. S.* About one algorithm of program polynomial invariants generation // Proc. Workshop on Invariant Generation: (Techn. rep.) Univ. of Linz / Eds. M. Giese, T. Jebelean. No 0707. – (RISC Report Series). Linz (Austria), 2007. – P. 85–99 (electronic).
10. *Львов М. С., Крекнін В. А.* Нелінійні інваріанти лінійних циклів та власні поліноми лінійних операторів // Кибернетика и системный анализ. – 2012. – № 2. – С. 126–140.
11. *Годлевский А. Б., Капитонова Ю. В., Кривой С. Л., Летичевский А. А.* Итеративные методы анализа программ. Равенства и неравенства // Кибернетика. – 1990. – № 3. – С. 1–9.
12. *Sabelfeld V. K.* Equivalente Transformationen für Flussdiagramme // Acta Informatica. – 1978. – **10**. – P. 127–155.

Киевский национальный университет им. Тараса Шевченко

Поступило в редакцию 28.01.2013

О. М. Максимець

Пошук програмних інваріантів у вигляді поліномів

Наведено рішення проблеми пошуку інваріантів програм у вигляді поліноміальних залежностей методом верхньої апроксимації. Цей ітераційнний метод, вдало застосований для програм з абсолютно вільними алгебрами і векторними просторами даних, адаптований для кільця поліномів. Множина інваріантів в цьому випадку являє собою ідеал кільця поліномів. Розв'язані задачі про співвідношення і про перетин множин інваріантів з використанням базисів Грьобнера при умові невиродженості оператора присвоювання.

O. M. Maksymets

Polynomial invariants generation of programs

A solution of the polynomial invariant generation problem for programs is presented. The iteration upper approximation method which was successfully applied to free algebras is adopted for a polynomial ring. The set of invariants is interpreted as an ideal over a polynomial ring. The solutions of the relationship and intersection problems are proposed. An intersection of Gröbner bases is applied to solve the intersection problem. The inverse obligatory is applied to solve the relationship problem.