УДК 004.415.28

М.С. Львов

ОБ ОДНОМ ПОДХОДЕ К ВЕРИФИКАЦИИ АЛГЕБРАИЧЕСКИХ ВЫЧИСЛЕНИЙ

Рассмотрен подход к верификации интерпретаторов многосортных алгебраических операций по их спецификациям, основанный на конструктивном уточнении понятия расширения многосортной алгебраической системы и доказательстве аксиом алгебраической системы. Этот подход иллюстрируется примерами верификации интерпретаторов операций поля рациональных чисел, многочленов одной переменной и алгебры высказываний. Подход использован при разработке систем компьютерной математики учебного назначения.

Введение

Разработка алгоритмов алгебраических вычислений – одна из основных задач, возникающих при реализации программных систем, основанных на символьных преобразованиях [1, 2]. Математическая модель этой задачи – многосортные алгебраические системы (МАС) [3]. Практика разработки даже достаточно простых математических систем учебного назначения [4–6] показала, что реализация алгебраических вычислений нуждается в тщательном предыдущем проектировании МАС в виде иерархий сортов и спецификаций интерпретаторов многосортных алгебраических операций [7–8].

Для реализации вычислений, основанных на символьных преобразованиях, используется система алгебраического программирования APS [1, 8, 9], адаптированная В. Песчаненко [10]. APS использует технологии алгебраического программирования, основанные на системах правил переписываний и стратегиях переписываний. Таким образом, интерпретаторы алгебраических операций определяются системой правил переписываний термов (rewriting rules system).

В данной работе рассматривается подход к верификации интерпретаторов многосортных алгебраических операций по их спецификациям, основанный, во-первых, на конструктивном уточнении понятия расширения многосортной алгебраической системы и, во-вторых, на алоритме доказательства аксиом МАС как тождеств. Основная идея заключается в следующем: если интерпретаторы операций МАС специфицированы и каждая аксиома – равенство или условное равенство, она является тождеством или условным тождеством в конструктивной реализации МАС, т. е. ее можно доказывать.

Этот подход проиллюстрирован примерами реализации интерпретаторов операций поля рациональных чисел, многочленов одной переменной и алгебры высказываний. Отметим, что эта работа – прямое продолжение [11].

1. Многосортные алгебры как модель алгебраических вычислений

Определение 1. Пусть $U = \{u_1,...,u_k\}$ – конечное множество символов, называемое сигнатурой сортов. Символы u_l, $l \in \{1,..., k\}$, называются именами сортов или сортами.

Перечислим имена стандартных сортов:
Variable – сорт переменных,
Bool – сорт логических значений,
Nat – сорт полукольца натуральных чисел,
Int – сорт кольца целых чисел,
Real – сорта поля действительных чисел.

Другие имена сортов будем вводить при определении соответствующих алгебраических понятий.

Определение 2. Пусть $S = \{S_{u_1}, ..., S_{u_k}\}$ – конечное семейство множеств, индексированных именами сортов, называемых носителями соответствующих сортов.

$S_{Variable}$ – множество переменных,

S_{Bool} – множество {False, True},

S_{Nat} – множество натуральных чисел,

S_{Int} – множество целых чисел,

$S_{\mathrm{Re}al}$ – множество действительных чисел.

Определение 3. Многосортной операцией f над S называется отображение $f : S_{u_1} \times ... \times S_{u_m} \to S_v$, где $u_1, ..., u_m, v \in U$ – сорта аргументов и результата операции f соответственно, а m – арность f.

Тип операции определяется списком имен сортов ее аргументов и значения и обозначается $(u_1, ..., u_m) \to v$. Сигнатурой Σ операций называется конечное множество символов операций вместе с отображением, которое каждому символу $\varphi \in \Sigma$ ставит в соответствие многосортную операцию f_φ вместе с ее типом. Запись $\varphi : (u_1, ..., u_m) \to v$ означает, что символу φ соотвтествует операция типа

$$(u_1, ..., u_m) \to v.$$

Многосортной операцией является, например, операция умножения в векторном пространстве. Если *Vectorspace* – имя векторного пространства над полем *Real* действительных чисел, то операция умножения *Mult* задается спецификацией

$$Mult : \mathrm{Re}al \times VectorSpace \to VectorSpace.$$

В дальнейшем будем пользоваться обычными математическими обозначениями операций. Поскольку умножение вектора на скаляр – бинарная операция с инфиксной формой записи, ее спецификация имеет вид

$$\mathrm{Re}al * VectorSpace \to VectorSpace.$$

Определение 4. Многосортным предикатом P называется отображение $P : S_{u_1} \times ... \times S_{u_m} \to S_{Bool}$, где $u_1, ..., u_m \in U$. Последовательность $u_1, ..., u_m$ определяет тип предиката, а число m – его арность.

Сигнатура Π многосортных предикатов определяется аналогично сигнатуре операций.

Определение 5. Многосортной алгебраической системой A называется четверка $A = \langle S, U, \Sigma, \Pi \rangle$, где S – множество сортов с именами из U, $\Sigma = \{\varphi_1, ..., \varphi_l\}$ – сигнатура многосортных операций, $\Pi = \{\pi_1, ..., \pi_p\}$ – сигнатура многосортных предикатов.

Замечание 1. Поскольку сорт *Bool* можно включить во множество сортов, предикаты можно рассматривать как многосортные операции. Поэтому сигнатуры операций и предикатов можно объединить, рассматривая многосортные алгебры.

Определение 6. Пусть $A = \langle S, U, \Sigma \rangle$ – многосортная алгебра и $u, v \in U$ – символы ее сортов. Будем говорить, что сорт v зависит от сорта u, если одна из операций Σ имеет тип $u_1 \times ... \times u \times ... \times u_m \to v$. U_v обозначим подмножество сортов, от которых зависят сорт v. Подмножество элементов Σ, которые имеют тип $u_1 \times ... \times u \times ... \times u_m \to v$, обозначим Σ_v, а семейство областей значений сортов U_v – S_v. Ограничением A_v алгебры A на сорт v называется МАС $A_v = \langle S_v, U_v, \Sigma_v \rangle$.

Таким образом, МАС A может быть представлена набором ограничений (алгебр) A_v, $v \in U$, т. е. $A = \langle A_{u_1}, ..., A_{u_k} \rangle$.

Пример 1. Задача состоит в том, чтобы построить программную систему, которая поддерживает упрощение целых алгебраических и тригонометрических выражений. Модуль алгебраических вычислений системы должен поддерживать вычисления в кольце полиномов и кольце тригонометрических выражений многих переменных над полем рациональных чисел Rat. Спецификациям подлежат алгебры – ограничения на следующие сорта:

Rat – поле рациональных чисел,

$MultiPolynom$ – кольцо полиномов многих переменных,

$Lincomb$ – векторное пространство линейных комбинаций многих переменных (аргументы тригоном. полиномов),

$MultiTPolynom$ – кольцо целых тригонометрических выражений многих переменных.

Отношение зависимости сортов порождает структуру зависимости на множестве алгебр A_u, $u \in U$: алгебра A_v зависит от алгебры A_u, если сорт v зависит от сорта u. Итак, многосортная алгебра представляет собой иерархическую структуру (рис.1), которую можно реализовывать инкрементным образом.

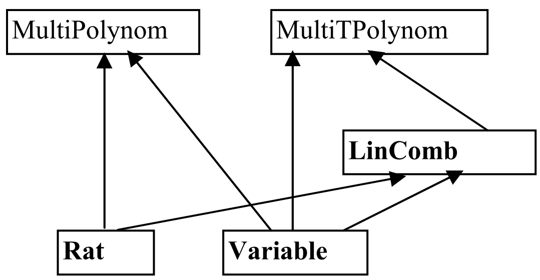

Рис. 1. Диаграмма отношения зависимости алгебр примера 1

Аксиомы и конструкции МАС. Для построения алгебр A_u будем использовать их аксиоматические и конструктивные описания (спецификации).

Определение 7. *Аксиомой* алгебры A_u называется равенство или условное равенство в сигнатуре Σ_u. *Аксиоматическим описанием* алгебры A_u, по определению, является конечная система аксиом алгебры A_u.

Замечание 2. Алгебры с аксиомами типа равенств называют многообразиями. Алгебры с аксиомами типа равенств или условных равенств называют квазимногообразиями. Будем пользоваться алгебраической терминологией и системами аксиом из [11].

Для определения конкретной алгебры вместе с аксиомами часто используют так называемые постулаты минимальности. Например, поле рациональных чисел можно определить как минимальное поле, содержащее множество натуральных чисел. Аналогично, кольцо многочленов $F[x]$ над полем F – минимальное коммутативное кольцо с единицей, содержащее поле F и x.

Конструктивное описание алгебры A_u состоит в определении конструктора сорта S_u и спецификаций интерпретаторов операций Σ_u.

Определение 8. *Сигнатурой* конструкторов T называется конечное множество символов конструкторов вместе с отображением, которое каждому символу $\tau_u \in T$ ставит в соответствие символ сорта u вместе со списком символов сортов его аргументов. Если τ – символ конструктора сорта u, то выражение $u = \tau(u_1,..., u_m)$ означает, что τ соответствует символ сорта u и символы сортов его аргументов $u_1,..., u_m$.

Конструктором сорта S_u называется система равенств**,** которая определяет синтаксически элементы сорта S_u в виде термов вида $\tau(u_1,..., u_m)$ в сигнатуре T.

Определение 8 – ключевое в нашем подходе к спецификации алгебраических вычислений.

Пример 2. *Поле Rat рациональных чисел.* Элементы этого поля – рациональные числа, представленные в виде обычных дробей. Конструктор сорта определяет стандартную форму представления элемента этого сорта. Чаще всего это *каноническая форма*. Таким образом,

$$S_{Rat} = \{\frac{p}{q} : p \in S_{Int}, q \in S_{Nat}, GCD(p,q) = 1\}.$$

Роль конструктора сорта играет символ “–” (горизонтальная черта). Этот же символ математики используют для обозначения операции деления, в частности, в *Rat.* Это – математическая традиция. Такую же роль играет, например, символ корня, символы сложения и умножения в представлении ($a+b*\sqrt{2}$). Однако, для задачи спецификаций алгебраических вычислений это не совсем удобно. Конструкторы сортов должны иметь свои обозначения. Поэтому отдельно вводится понятие сигнатуры операций Σ и сигнатуры конструкторов T. В частности, для конструктора сорта *Rat* используется *двойная наклонная черта*:

$$S_{Rat} = \{p // q : p \in S_{Int}, q \in S_{Nat}, GCD(p,q) = 1\}.$$

В стандартных формах представления элементов сортов синтаксические аспекты, определяемые символами конструкторов, практически всегда дополняется семантическими аспектами, задаваемыми в виде контекстных условий. В нашем примере – это равенство $GCD(p,q)=1$.

Пример 3. Кольцо *Polynom* полиномов одной переменной над полем *Rat*. Элементы этого поля – полиномы, представляемые в виде суммы мономов, упорядоченных в порядке убывания степеней.

$$S_{Polynom} = \{M_0 + M_1 + ... + M_k : M_i \in S_{Monom}, \deg(M_i) > \deg(M_{i+1})\}.$$

В спецификациях сорта *Polynom* это определение рекурсивно. При таком подходе отдельно нужно определить понятие степени полинома.

$$S_{Polynom} = \{Q : Q = M ++ P, M \in S_{Monom}, P \in S_{Polynom}, \deg Q \overset{df}{=} \deg M; \deg(M) > \deg(P)\} \cup S_{Monom}$$

Для определения носителей сортов будем использовать специальный язык спецификаций, который допускает нерекурсивные и рекурсивные синтаксические определения элементов сортов, определения функций доступа и контекстных условий. Приведем определения сортов:

```
Rat r ={
  (Int a)//(Nat b);
                        // Конструктор сорта
   Num(r) = a, Den(r) = b;
                  // Функции доступа (селекторы)
  GCD(a, b) = 1 //Контекстное условие
};
Monom M ={
(Rat  c)$(Const  Variable  x)^(Nat
n);   // Конструктор сорта
    Coef(M) = c, // Функции доступа
   Var(M) = x,
   Deg(M) = n
};
Polynom P = {
   (Monom M)++(Polynom Q);
                //Конструктор сорта
   LeadMonom(P) = M,                //
Функции доступа
  LeadCoef(P) = Coef(M),
  Deg(P) = Deg(M);
Deg(P) > Deg(Q)
                // Контекстное условие
};
```

Реализовать вычисления в алгебре $A_v, v \in U$ означает реализовать алгоритмы выполнения каждой ее операции так, чтобы выполнялись аксиомы этой алгебры.

Определение 9. *Интерпретатором* операции сигнатуры Σ_u называется функция, реализующая алгоритм выполнения этой операции.

Интерпретаторы операций описываются в языке *APLAN* системы APS.

Итак, для аксиоматического и конструктивного описания алгебры A_v в ее определение включается конечное множество аксиом Ax_v и конеченое множество интерпретаторов I_v. Многосортная алгебра A_v определяется следующим образом:

$$A_v = \langle S_v, U_v, T_v, \Sigma_v, Ax_v, I_v \rangle.$$

2. Расширения многосортных алгебр

Определение 10. Пусть A_u и A_v – многосортные алгебры. A_v называется *расширением* A_u, если $S_u \subset S_v$ и для любой пары операций f_1 и f_2 типов $f_1 : (u_1,..., u_m) \to u$, $f_2 : (v_1,..., v_m) \to v$, если $S_{u_1} \subseteq S_{v_1},..., S_{u_m} \subseteq S_{v_m}$, то

$$\forall (a_1,..., a_m) \in S_{u_1} \times ... \times S_{u_m} \to$$
$$\to f_1(a_1,..., a_m) = f_2(a_1,..., a_m).$$

Вложением называется изоморфное отображение $\mathrm{Re}\,d : S_u \to S'_v$, которое отображает S_u на подмножество $S'_v \subset S_v$. Ограничение алгебры A_u на подмножество S'_v, изоморфное A_u, определяется системой условных тождеств $E_1(x),..., E_k(x)$:

$$S'_v = \{a \in S_v \mid E_1(a),..., E_k(a)\}.$$

Применение системы $E_1(x),..., E_k(x)$ как системы переписываний «редуцирует» терм $a \in S'_v$ к терму $a' \in S_u$: $\mathrm{Re}\,d^{-1}(a) = a'$. Конструктивное описание A_v как расширения A_u состоит в описании конструктора A_v и вложения A_u в алгебру A_v.

Замечание 3. Понятие конструктивного расширения многосортных алгебр – основное в данной работе. Основы теории упорядочено-сортных алгебр в ее применении к теории программирования изложены в [12, 13].

Пример 4. Рассмотрим конструктор поля *Rat* (пример 2). В соответствии с определением, он определяет конструкцию *Rat,* аргументы которой – сорта *Int* и *Nat*. Дополним спецификации сорта *Rat* вложением $Red : Rat \to Int$, определеным равенством $\mathrm{Re}\,d(a // 1) = a$. Тем самым сорт *Rat* определен как расширение сорта *Int.*

Рассмотрим конструктор кольца *Polynom* (пример 3). Он определяет рекурсивно сорт *Polynom* через сорт *Monom.* Дополним спецификации сорта *Polynom* вложением $Red : Polynom \to Monon$, определенным равенством $\mathrm{Re}\,d(M + +0) = M$. Тем самым сорт *Polynom* определен как расширение сорта *Monom.*

В свою очередь, сорт *Monom* – расширение сорта *Degree* с функцией *Red*, определенной равенством $1\$x \wedge k = x \wedge k$, расширение сорта *LinMonom* с функцией *Red*, определенной равенством $a\$x \wedge 1 = a\$x$ и расширение сорта *Rat* с функцией *Red*, определенным равенством $a\$x \wedge 0 = a$.

Сорта *Degree* и *LinMonom*, в свою очередь – расширения сорта *Variable* с редукциями $x \wedge 1 = x$ и $1\$x = x$. Диаграмма расширений примера показана на рис. 2.

Использование расширений – один из основных методов спецификаций многосортных алгебр. В частности, он позволяет определить перегруженные алгебраические операции и функции приведения алгебраических типов.

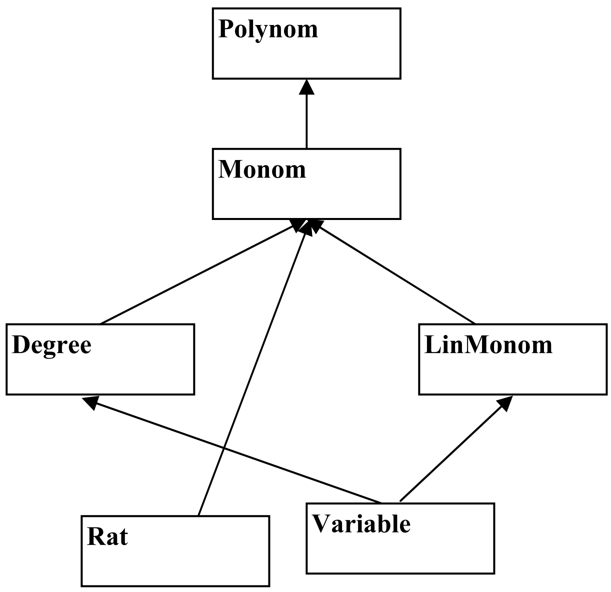

Рис. 2. Диаграмма расширений примера 4

Статические и динамические расширения

Определение 11. Расширение B алгебры A называется статическим (нерекурсивным), если в его конструкторе $B = \tau(A_1,...,A,...,A_n)$ ни один из аргументов не совпадает с B.

Примеры статических расширений:

1) поле *Rat* – статическое расширение кольца *Int*, поскольку
Rat R = (Int A) // (Nat B);

2) полугруппа мономов Monom одной образующей – статическое расширение поля коэффициентов *Coef*, поскольку

$$Monom\ M = (Coef\ a)\$(Var\ x)\hat{}\,(Nat\ N).$$

Определение 12. Расширение B алгебры A называется динамическим (рекурсивным), если в конструкторе $B = \tau(A_1,...,A,...,A_n)$ по меньшей мере один из аргументов совпадает с B.

Конструкторы динамических расширений определяются рекурсивно. Например:

3) векторное пространство *Lincomb* линейных комбинаций многих переменных над полем *Coef* – линейное динамическое расширение одномерного пространства *Linmonom*, элемент которого имеет вид *a*$*x*. Элемент $w \in LinComb$ имеет вид $w = a_1\$x_1 + +... + +a_m\$x_m$ с конструктором

$$LinComb\ w = (LinMonom\ u) + +(LinComb\, w),$$
$$u + +0 = u;$$

4) кольцо *Polynom* над полем *Coef* – линейное динамическое расширение *Monom:*

$$Polynom\ \ w = (Monom\ M) + +(Polynom\ w),$$
$$M\ + +0 = u.$$

Практика использования динамических расширений показала полезность их дальнейшей классификации как линейных и бинарных расширений.

Определение 13. Динамическое расширение B алгебры A называется линейным, если в ее конструкторе $B = \tau(A_1,...,A,...,A_n)$ в точности один из аргументов совпадает с B.

Динамическое расширение B алгебры A называется бинарным, если в ее конструкторе $B = \tau(A_1,...,A,...,A_n)$ в точности два аргументы совпадают с B.

В примерах 3), 4) рассмотрены линейные динамические расширения. Рассмотрим пример бинарного динамического расширения:

5) числовое поле *Rad*, элементы которого – суть линейные комбинации квадратных корней натуральных чисел, свободных от квадратов, с рациональными коэффициентами, можно представить как бинарное расширение поля *Rat* следующей конструкцией. Пусть $p_1, p_2,..., p_n,...$ – последовательность всех простых чисел, расположенных в порядке возрастания. Введем следующие обозначения:

$$Rad_0 = Rat,\ Rad_n = \{r : r = a + b * \sqrt{p_n},$$
$$a, b \in Rad_{n-1}, n = 1,2,...\};$$

Поле *Rad* можно представить в виде бесконечного объединения возрастающей последовательности полей Rad_n.

$$Rad = \bigcup_{n=0}^{\infty} Rad_n,$$
$$Rat = Rad_0 \subset Rad_1 \subset ... \subset Rad_n \subset ... \qquad (1)$$

Конструктор *Rad* имеет вид

$$Rad\ r = (Rat\ q)\ |$$
$$(Rad\ a) + (Rad\ b) * \sqrt{Nat\ p}. \qquad (2)$$

Заметим, что (1) – последовательность конечных алгебраических расширений полей корнями полиномов $x^2 - p_n = 0$.

В представление (2) включены как описание элемента базовой алгебры $Rat\ q$, так и описание механизма расширения – конструктор $(Rad\ a) + (Rad\ b) * \sqrt{Nat\ p}$.

Такая спецификация в точности соответствует определению (1). Формулы (1) непосредственно обобщаются на произвольные динамические расширения. Если алгебра *B* – динамическое расширение алгебры *A с конструктором* $B = \tau(A_1,...,B,...,A_n)$, возрастающую последовательность $B_0 \subset B_1 \subset ...B_n \subset ...$ определим таким образом:

$$B_{(0)} = A,$$

$$B_{(n+1)} = \tau(A_1,...,B_{(n)},...,A_n).$$

Вложение $Red : A \to B$ определяет вложение $Red_i : B_{i+1} \to B_i$, откуда непосредственно следует представление *A* в виде объединения возрастающей последовательности алгебр, каждая из которых – статическое расширение предыдущей.

$$B = \bigcup_{n=0}^{\infty} B_n,\ B_0 \subset B_1 \subset ... \subset B_n \subset ... \ .$$

Таким образом, динамические расширения алгебр – суть последовательности статических расширений.

3. Верификация вычислений в расширениях многосортных алгебр

В работе [11] понятие расширения алгебр используется для синтеза интерпретирующих правил операций и их оптимизации. При этом специфицируются только правила «общего случая». Производные правила выводятся из общих редуцированием – применением функции *Red*. Таким образом, *задача верификации алгебраических вычислений сводится к задаче обоснования правильности правил «общего случая».* Какие свойства присущи «правильным» спецификациям операций? Как отличать «правильные» определения от «неправильных»? Для ответа на эти вопросы нужно проверить выполнение следующих условий:

1) *правила спецификации алгебраических операций, определенных конструктивно в спецификациях алгебры, должны удовлетворять системе аксиом этой алгебры;*

2) *результат выполнения операции должны удовлетворять синтаксическим определениям и контекстным условиям конструктора сорта*.

Пример 10. Верификация сорта Boolalg.

Сорта *Boolalg* специфицирует алгебру высказываний многих переменных. Пусть $F(x_1, x_2,..., x_n)$ – произвольная формула формулы логики высказываний *n* переменных. Обозначим O, I – соответственно *истина* и *ложь*. Тогда

$$F(x_1, x_2,..., x_n) = x_n \ \&\ F(x_1 x_2,..., x_{n-1}, I) \vee$$
$$\vee \neg x_n \ \&\ F(x_1, x_2,..., x_{n-1}, O).$$

Если обозначить

$$A(x_1,..., x_{n-1}) = F(x_1,..., x_{n-1}, I),$$
$$B(x_1,..., x_{n-1}) = F(x_1,..., x_{n-1}, O),$$

получим представление

$$F(x_1, x_2,..., x_n) = x_1 \ \&\ A(x_1,..., x_{n-1}) \vee$$
$$\vee \neg x_1 \ \&\ B(x_1,..., x_{n-1}). \quad (3)$$

Выполним последовательно эти преобразования для переменных $x_{n-1},..., x_1$. В результате получим рекурсивное представление формулы логики высказываний. Через $BoolA\lg_m$ обозначим множество формул логики высказываний от переменных $x_1,..., x_m$. Тогда

$$BoolA\lg_m = \{F : F = x_m \ \&\ A \vee \neg x_m \ \&\ B,$$
$$A, B \in BoolA\lg_{m-1}\}.$$

Итак, алгебра *Boolalg* – объединение возрастающей последовательности алгебр $BoolA\lg_m$:

$$BoolA\lg_0 = Bool,$$
$$BoolA\lg_0 \subset BoolA\lg_1 \subset ... \subset BoolA\lg_m \subset ... .$$

$$BoolA\lg = \bigcup_{m=0}^{\infty} BoolA\lg_m. \quad (4)$$

Формула (3) задает рекурсивную каноническую форму формул алгебры высказываний. Обозначим

$$BF(A,B,x) \overset{df}{=} x \,\&\, A \vee \neg x \,\&\, B\,.$$

$BF(A,B,x)$ – конструктор сорта *Boolalg,* причем выполнены контекстные условия $x > Arg(A),\ x > Arg(b).$ Интерпретаторы логических операций определяются формулами:

$$BF(A_1,B_1,x) \,\&\, BF(A_2,B_2,x) = \\ = BF(A_1 \,\&\, A_2, B_1 \,\&\, B_2, x), \qquad (5)$$

$$BF(A_1,B_1,x) \vee BF(A_2,B_2,x) = \\ = BF(A_1 \vee A_2, B_1 \vee B_2, x), \qquad (6)$$

$$\neg BF(A,B,x) = BF(\neg A, \neg B, x)\,. \qquad (7)$$

Это означает, что основные логические операции выполняются поаргументно. Легко проверить, что функция вложения определяется равенством

$$BF(A,A,x) = A\,.$$

Рассмотрим одну из аксиом сорта *Bool*, например, правило де Моргана

$$\neg(A \,\&\, B) = \neg A \vee \neg B\,.$$

Подставив вместо *A, B* их определения $A = BF(A_1,B_1,x), B = BF(A_2,B_2,x)$, получаем:

$$\neg(BF(A_1,B_1,x) \,\&\, BF(A_2,B_2,x)) = \\ = \neg BF(A_1,B_1,x) \vee \neg BF(A_2,B_2,x).$$

Выполним вычисления в левой и правой частях равенства:

$$BF(\neg(A_1 \,\&\, A_2), \neg(B_1 \,\&\, B_2), x) = \\ = BF(\neg A_1 \vee \neg A_2, \neg B_1 \vee \neg B_2, x)). \qquad (8)$$

Выпишем соответствующие контекстные условия:

$$x > Arg(\neg(A_1 \,\&\, A_2)), x > Arg(\neg(B_1 \,\&\, B_2)),$$
$$x > Arg(\neg A_1 \vee \neg A_2), x > Arg(\neg B_1 \vee \neg B_2)\,. \quad (9)$$

Равенство (8) означает, что доказательство правила де Моргана сводится к доказательству этого правила на аргументах:

$$\neg(A_1 \,\&\, A_2) = \neg A_1 \vee \neg A_2, \neg(B_1 \,\&\, B_2) = \neg B_1 \vee \neg B_2.$$

Проверка контекстных условий сводится к доказательству импликации

$$(x > Arg(A_1)) \,\&\, (x > Arg(A_2)) \to \\ \to (x > Arg(\neg(A_1 \,\&\, A_2)).$$

Вывод: Доказательство аксиом алгебры высказываний осуществляется индукцией по индексам последовательности (4). Поскольку формулы (5)–(7) сводят вычисления логических операций к вычислению этих операций на аргументах, индуктивные рассуждения примера тривиальны.

Пример 11. Верификация сорта *Polynom*.

Рассмотрим одну из аксиом сорта *Polynom*, например, аксиому дистрибутивности

$$x(y+z) = xy + xz\,.$$

Как и в предыдущем примере, подставим вместо x, y, z их определения:

$$x = a + +A, y = b + +B, z = c + +C.$$

Получаем

$$(a + +A)((b + +B) + (c + +C)) = \\ = (a + +A)(b + +B) + (a + +A)(c + +C).$$

Вычислим левую и правую части равенства по правилам сорта *Polynom:*

$$a(b+c) + +(a(B+C) + A(b+c) + A(B+C)) = \\ = (ab + ac) + +(((Ab + aB) + AB) + \\ + ((Ac + aC) + AC)).$$

Контекстные условия имеют вид:

$$Deg(b) = Deg(c), Deg(a) > Deg(A), \\ Deg(b) > Deg(B), Deg(c) > Deg(C).$$

Требуется доказать

$$a(b+c) = ab + ac\,. \qquad (10)$$

$$a(B+C) + A(b+c) + A(B+C) = \\ = ((Ab + aB) + AB) + ((Ac + aC) + AC), \quad (11)$$

$$Deg(a(b+c)) >$$
$$> Deg(a(B+C)+A(b+c)+A(B+C). \quad (12)$$

Рассмотрим задачу доказательства (12). Это неравенство – следствие правил выполнения операций сложения и умножения. Если эти правила правильно оперируют со степенями, оно выполняется автоматически. Поэтому задача доказательства неравенств – контекстных условий аксиом, сводится к задачам доказательства правильности условий в правых частях правил интерпретации операций. В нашем случае нужно доказать:

1) для правила операции сложения:

$$(Deg(a) = Deg(b))\,\&\,((Deg(a) > Deg(A))\,\&$$
$$\&\,(Deg(b) > Deg(B)) \rightarrow$$
$$\rightarrow (Deg(a+b) > Deg(A+B));$$

2) для правила операции умножения:

$$(Deg(a) > Deg(A))\,\&\,(Deg(b) > Deg(B)) \rightarrow$$
$$\rightarrow Deg(ab) > Deg((aB + Ab) + AB).$$

Первая из импликаций – следствие общего правила:

$$LeadMon(P+Q) \neq 0 \rightarrow$$
$$\rightarrow Deg(P+Q) = Max(Deg(P), Deg(Q)).$$

Вторая – следствие правила

$$Deg(PQ) = Deg(P) + Deg(Q)\ .$$

Вывод: доказательство правильности контекстных условий в отдельной операции сводится к доказательству условных неравенств, формулируюмых для данной аксиомы на основании правил вычисления степеней полиномов – результатов операций. При этом нужно проверять не только основные правила интерпретации, но и производные.

Рассмотрим доказательства равенств (10), (11). Как и в примере 10, они имеют место с точностью до соотношений вложения сорта: $M++0 = M$, $0++P = P$, т. е. являются тождествами алгебры многочленов. Но, в отличие от примера 10, равенство (11) не является аксиомой. Его доказательство требует использования аксиом дистрибутивности, комутативности и ассоциативности. Таким образом, доказательство по индукции осуществляется такой формулировке:

1) *базис индукции*: для элементов a,b,c,A,B,C сорта *Monom* выполняются все аксиомы сорта *Polynom;*

2) *предположение индукции*: для элементов A,B,C сорта *Polynom* и элементов a,b,c сорта *Monom* выполняются все аксиомы сорта *Polynom;*

3) *шаг индукции: для* элементов $X = a++A, Y = b++B, Z = c++C$ сорта *Polynom* также выполняются все аксиомы сорта *Polynom;*

4) *вывод индукции*: на сорте *Polynom* выполняются все аксиомы сорта *Polynom*.

Вывод: проверка выполнения отдельной аксиомы (например, дистрибутивности) на элементах сорта *Polynom* сводится к проверке выполнения всей системы аксиом на аргументах конструктора сорта. Поэтому доказательство осуществляется в вышеуказанной форме только для основных правил интерпретации операций.

Пример 12. Верификация сорта *Rat*. Рассмотрим задачу проверки выполнения аксиомы дистрибутивности для сорта *Rat*.

На множестве *Int*×*Nat* должна быть определена конгруэнтность, выделяющая равные пары аргументов (числитель, знаменатель) конструктора сорта *Rat*. Эта конгруэнтность имеет вид

$$(a = p_1 // q_1,\ b = p_2 // q_2,\ a \sim b \leftrightarrow p_1 q_2 = p_2 q_1)\ .$$

Для элементов сорта *Rat* имеет место соотношение

$$(p / q = r // s)\,\&\,(ps = rq)\,\&$$
$$\&\,(GCD(r,s) = 1) \qquad (13)$$

Рассмотрим доказательство дистрибутивности. Пусть $a(b+c) = ab + ac$ и $a = p_1 // q_1,\ b = p_2 // q_2,\ c = p_3 // q_3$.

Подставив значения a, b, c в аксиому дистрибутивности и вычислив левую и правую части равенства по правилам из спецификаций *Rat,* получаем:

$$p_1(p_2q_3 + q_2p_3)/(q_1q_2q_3) =$$
$$= (q_1q_3p_1p_2 + q_1q_2p_1p_3)/(q_1q_2q_1q_3).$$

Для упрощения выражений введем обозначения:

$$A = p_1(p_2q_3 + q_2p_3),\ B = q_1q_2q_3,$$
$$C = q_1q_3p_1p_2 + q_1q_2p_1p_3, D = q_1q_2q_1q_3.$$

Верификация аксиомы дистрибутивности сводится к проверке тождества $AC = BD$ над сортом *Int* и доказательстве (13). Доказательство тождества верифицирует данную аксиому, а доказательство (13) верифцирует интерпретатор конструктора сорта *Rat*.

Алгоритмы верификации алгебраических вычислений в расширениях алгебр. Рассмотрим алгебры

$$A{=}{<}S_A, U_A, T_A, \Sigma_A, \mathrm{Ax}_A, I_A>, \qquad (14)$$
$$B{=}{<}S_B, U_B, T_B, \Sigma_B, \mathrm{Ax}_B, I_B>.$$

Верификация в статических расширениях. Обобщим метод верификации системы алгебраических программ – интерпретаторов операций сорта v из его спецификаций при условии, что алгебра B сорта v – *статическое расширение* алгебры A сорта u: $A \overset{df}{=} A_u,\ B \overset{df}{=} A_v, A \subset B.$

Спецификации сорта v определяют:

Конструктор сорта: $v = \tau(u, W)$. Обязательным аргументом τ является сорт u.

Контекстное условие: $\pi(u, W)$.

Условие вложения: $\rho(u, W)$.

Функция вложения: $\rho(u, W) \to \tau(u, W) = u$.

Предикат $\rho(u, W)$ функционален: для каждого значения u существует единственное значение P такое, что $\rho(u, W)$. Эту функцию будем обозначать F_ρ: $\rho(u, W) = (F_p(u) = W)$.

Интерпретатор конструктора сорта: $!\tau(x, P) = y$. Вызов интерпретатора $!\tau(x, P)$, обозначенный восклицательным знаком как префиксом имени функции, делает истинным предикат $\pi(x, P)$.

Конгруэнция конструктора сорта: Конгруэнция Θ на множестве

$$S_u \times S_{w_1} \times ... \times S_{w_k} : (x, p_1, ..., p_k) \overset{\Theta}{\approx} (x', p_1', ..., p_k')$$

определяет условие на аргументах конструктора сорта:

если $\tau(x, P) = \tau(x', P')$, то $(x, P) \overset{\Theta}{\approx} (x', P')$.

Нетрудно видеть, что конструкция элемента сорта является канонической формой. Поэтому эта конгруэнция определяет классы эквивалентности элементов $S_u \times S_{w_1} \times ... \times S_{w_k}$, имеющих одну и ту же каноническую форму.

Сигнатуры операций сортов:

сорта *u:* $\Sigma_u = <\varphi_1, ... \varphi_m>$;

сорта v: $\Sigma_v = <\varphi_1, ... \varphi_m, \psi_1, ..., \psi_l>$.

Мы считаем, что сигнатура операций расширения B – расширение сигнатуры операций базовой алгебры A.

Аксиомы сортов

сорта *u:* $AX_u = <A_1, ... A_s>$;

сорта v: $AX_v = <B_1, ..., B_{s'}>$.

Мы считаем, что система аксиом B – расширение системы аксиом алгебры A.

Спецификации операции $\varphi_1, ... \varphi_m$ заданы системами равенств.

Верификация интерпретаторов операций алгебры B заключается в:

доказательстве тотальной корректности интерпретатора функции $!\tau(x, P)$;

доказательстве тождеств над алгеброй A – следствий аксиом AX_B и конгруэнции Θ конструктора алгебры B.

Доказательство частичной корректности интерпретатора $!\tau(x, P)$ опирается на его представление в виде суперпозиции функций построения канонической формы аргумента (x, P) как элемента носителя алгебры B и применении вложения к этой форме: $\rho(x, P) \to (!\tau(x, P) = x)$.

Обозначим τ_{can} функцию, которая возвращает аргумент (x,P) как элемент носителя алгебры B, а τ_{red} – функцию вложения $\rho(x,P) \to (!\tau(x,P) = x)$.
Тогда

$$!\tau(x,P) = \tau_{red}(\tau_{can}(x,P)).$$

Частичную корректность $!\tau(x,P)$ можно доказать, доказав импликации

$$(\tau_{can}(x,P) = (x',P')) \to \to ((x,P)\overset{\Theta}{\sim}(x',P')) \,\&\, \pi(x',P')), \quad (15)$$

$$\rho(x,P) \,\&\, (\tau_{red}(x,P) = x) \to \to ((x,P)\overset{\Theta}{\sim}(x,F_\rho(x)) \,\&\, \pi(x,F_\rho(x))). \quad (16)$$

Определение 14. Расширение алгебры $A \subset B$ называется прямым, если функция τ_{can} является тождественной.

Элемент носителя прямого расширения B – произвольный элемент $S_u \times S_{w_1} \times \ldots \times S_{w_k}$, а конгруэнция Θ единична. Таким образом, при прямых расширениях доказывать нужно только корректность вложения $A \to B$. Расширение $A_i \subset A_{i+1}$ примеров 10, 11 прямые. Расширение $Int \to Rat$ (пример 12) не является прямым.

Рассмотрим *доказательста тождеств*. Тождества над A, являющиеся следствиями аксиом AX_B и конгруэнции Θ, получаются следующим образом: если

$$B_i \overset{df}{=} (L(x_1,\ldots,x_n,P) = R(x_1,\ldots,x_n,P)) \in AX_B,$$

причем переменные $x_1,\ldots,x_x$ определены над B, подстановки $\tau(u_i,P_i)$ вместо x_i определяют равенства над $S_u \times S_{w_1} \times \ldots \times S_{w_k}$. Формальное выполнение интерпретаторов операций алгебры B в левой и правой частях равенства

$$L(\tau(u_1,P_1),\ldots,\tau(u_n,P_n),P) = = R(\tau(u_1,P_1),\ldots,\tau(u_n,P_n),P),$$

приводит к равенству

$$\tau(f,G_1,,,G_k) = \tau(f',G_1',\ldots,G_k'),$$

где

$$f = f(u_1,..,u_n,P_1,..,P_n,P),$$
$$f' = f'(u_1,..,u_n,P_1,..,P_n,P),$$
$$G_i = G_i(u_1,..,u_n,P_1,..,P_n,P),$$
$$G_i' = G_i'(u_1,..,u_n,P_1,..,P_n,P),\ i = 1,..,n.$$

Итак, имеет место отношение эквивалентности

$$(f,G_1,,,G_k)\overset{\Theta}{=}(f',G_1',\ldots,G_k'). \quad (17)$$

Это отношение зависит от переменных $u_1,..,u_n,P_1,..,P_n,P$. Будем считать, что конгруэнтность Θ задана системой тождеств $V_j = W_j$, $j = 1,..,e.$ Тогда верификация интерпретаторов операций алгебры B состоит в доказательстве тождеств

$$V_j(u_1,..,u_n,P_1,..,P_n,P) = = W_j(u_1,..,u_n,P_1,..,P_n,P),\ j = 1,..,e, \quad (18)$$

над базовыми сортами $u,w_1,\ldots,w_k$. Алгоритм доказательства (18) должен опираться на аксиомы базовых сортов. Поэтому этот алгоритм существует, если проблема тождества над базовыми алгебрами алгоритмически разрешима.

Если расширение $A \subset B$ – прямое, тождества (18) “покоординатные”. Поэтому конгруэнтность (17) определяет тождества

$$f = f',\ G_i = G_i',\ i = 1,..,k. \quad (19)$$

Верификация в динамических расширениях. Обобщим метод верификации системы интерпретаторов операций сорта v при условии, что алгебра A_v – динамическое расширение алгебры A. Пусть

$$A_0 = A,\ A_{i+1} = \tau(A_0,A_i),\ B = \bigcup_{i=0}^{\infty} A_i$$

– спецификация конструкции линейного динамического расширения $A \subset B$.

$$x \in A_0,\ X \in A_i,\ \rho(x,X) \to \tau_{red}(x,X) = X$$

– вложения, определенные для элементов A_i этого расширения.

Как и для статических расширений, система аксиом B – расширение системы аксиом базовой алгебры A. Это означает, что этой системе должны удовлетворять все алгебры A_i . Тогда:

– верификация функции $!\tau$ проводится индукцией по индексу i и состоит в доказательстве импликаций (15), (16);

– верификация интерпретаторов операций на B с помощью аксиом осуществляется индукцией по индексу i и состоит в доказательстве тождеств (19) над алгебрами A_{i+1} в индуктивном предположении, что все аксиомы доказаны как тождества алгебры A_i. Если алгоритм доказательства на базовой алгебре $A_0 = A$ использует только аксиомы A, индукция позволяет расширить этот алгоритм на B .

Метод расширений используется, как правило, для пределения новых операций. Например, сорт *Int* расширяется до *Rat* с целью определения операции деления (пример 12).

Сорт *Polynom* расширяется как векторное пространство. Для этого сорт *Monom* рассматривается как одномерное векторное пространство, а шаг расширения состоит в добавлении новой координаты. В этом примере A_i – i-мерное векторное пространство. Операции умножения на скаляр, сложение и вычитание, а также все аксиомы векторного пространства удовлетворяют рассмотренному методу. Однако операция умножения многочленов и все аксиомы, содержащие умножение, должны быть рассмотрены отдельно.

Пусть $P, Q,\ R = P * Q \in Polynom$, $\deg P = i, \deg Q = j$, $A_k = \{p : \deg p \le k\}$.

Тогда $\deg R = i + j$. Итак, если $P \in A_i,\ Q \in A_j,\ P * Q \in A_{i+j}$.

Верификация операции умножения методом математической индукции, вышерассмотренная, сводит доказательство аксиом на A_{i+j} к доказательству тождеств на A_{i+j-1} . Поэтому базис индукции должен содержать доказательство аксиом как тождеств на базовых алгебрах, т. е. на мономах. Например, в (12) нужно считать, что $a, A, b, B, c, C \in Monom$. .

Обобщим эти соображения на произвольные алгебры. Метод индукции применим, поскольку степень произведения является монотонно возрастающей функцией степеней множителей. Сформулируем это в общем для бинарных операций. Пусть

$$A_0 = A,\ A_{i+1} = \tau(A_0, A_i),\ B = \bigcup_{i=0}^{\infty} A_i$$

и $\varphi(a,b)$ - бинарная операция, определенная на алгебре B . Тогда, если существует такая монотонно возрастающая по каждому из аргументов функция $s(i, j)$, что для произвольных $a \in A_i, b \in A_j$, $\varphi(a,b) \in A_{s(i,j)}$ метод индукции сводит доказательство аксиом алгебры B, содержащих операцию φ, к доказательству этих аксиом для переменных, определенных на алгебре A.

Выводы

Наш подход к верификации алгебраических вычислений характеризуется такими особенностями.

1. Определение иерархии зависимостей сортов позволяет упорядочить процесс построения и верифиции новых сортов с целью расширения функциональности системы компьютерной математики.

2. Включение в спецификации сортов аксиом и интерпретаторов операций позволяет использовать аксиомы для верификации интерпретаторов операций.

3. Определения сортов как конструктивных расширений позволяет сводить задачи верификации сорта к верификации вычислений в базовых сортах.

4. Классификация расширений как статических и динамических позволяет

определять и использовать либо прямой, либо индуктивный метод верификации.

5. Данный подход особенно эффективен при реализации вычислений в алгебрах, аксиоматика которых хорошо известна и полна.

1. *Kapitonova J., Letichevsky A., Lvov M., Volkov V.* Tools for solving problems in the scope of algebraic programming // Lectures Notes in Computer Sciences. – 1995. – N 958. – P. 31–46.
2. *Львов М.С.* Основные принципы построения педагогических программных средств поддержки практических занятий // Управляющие системы и машины. – 2006. – № 6. – С. 70–75.
3. *Песчаненко В.С.* Об одном подходе к проектированию алгебраических типов данных // Проблеми програмування. – 2006. – № 2–3. – С. 626–634.
4. *Lvov M., Kuprienko A., Volkov* V. Applied Computer Support of Mathematical Training // Proc. of Internal Work Shop in Computer Algebra Applications, Kiev. – 1993. – P. 25–26.
5. *Lvov M.* AIST: Applied Computer Algebra System // Proc. of ICCTE'93. – Kiev. – P. 25–26.
6. *Львов М.С.* Терм VII – школьная система компьютерной алгебры // Компьютер в школе и семье. – 2004. – № 7. – С. 27–30.
7. *Песчаненко В.С.* Расширение стандартных модулей системы алгебраического программирования APS для использования в системах учебного назначения // Компьютерно-ориентированные системы обучения: – Киев: НПУ им. М.П. Драгоманова.– 2005. –№ 3(10). – С. 206–215.
8. *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.
9. *Капитонова Ю.В., Летичевский А.А., Волков В.А.* Дедуктивные средства системы алгебраического программирования // Кибернетика и системный анализ. – 2000. – № 1. – С. 17–35.
10. *Песчаненко В.С.* Использование системы алгебраического программирования APS для построения систем поддержки изучения алгебры в школе // Управляющие системы и машины. – 2006. – № 4. – С. 86–94.
11. *Львов М.С.* Синтез интерпретаторов алгебраических операций в расширениях многосортных алгебр // Вісник Харківського національного університету. – 2009. – № 847. – С. 221–238.
12. *Ван дер Варден Б.Л.* Алгебра. Изд. 2-ое. – М.: ГРФМЛ, 1979. – 624 с.
13. *Goguen J., Meseguer J.* Ordered-Sorted Algebra I: Partial and Overloaded Operations. Errors and Inheritance // Theoretical Computer Science. – Oxford: Elsevier, 1992. – Vol. 105 (N 2). – P. 217–273.
14. *Дж. А. Гоген, Ж. Мезегер*. Модели и равенство в логическом программировании // Математическая логика в программировании. – М.: Мир, 1991. – С. 274–310.

Получено 16.05.2011

Об авторе:

Львов Михаил Сергеевич,
кандидат физико-математических наук,
доцент, профессор кафедры информатики.

Место работы автора:

Херсонский государственный университет,
ул. 40-лет Октября, 27.
Херсон, Украина,73000.
Тел.: (0552) 32 6781.
Факс.: (0552) 32 6785.
lvov@ksu.ks.ua