УДК 519.6

Об одном подходе к реализации алгебраических вычислений: вычисления в алгебре высказываний

М. С. Львов

Херсонский государственный университет, Украина

В процессе проектирования математических систем учебного назначения был разработан достаточно общий подход к проектированию и реализации вычислений в многосортных алгебрах. Он использует наследование, расширения и морфизмы многосортных алгебр. В настоящей работе рассмотрены основные принципы этого подхода на примере реализации вычислений в алгебре высказываний. Это позволило кратко, но полно представить основные идеи. Материал работы может быть использован в качестве хорошего учебного примера в курсах «Специальные языки программирования», «Математическая логика», «Компьютерная алгебра и символьные вычисления» для иллюстрации глубоких связей между алгеброй, логикой и практикой программирования.

Ключевые слова: *проектирование систем, многосортная алгебра, наследование, расширения, морфизмы, алгебра высказываний, учебный пример.*

В процесі проектування математичних систем навчального призначення був розроблений достатньо загальний підхід до проектування та реалізації обчислень у багатосортних алгебрах. Він використовує наслідування, розширення та морфізми багатосортних алгебр. В даній роботі розглянуті основні принципи цього підходу на прикладі реалізації обчислень в алгебрі висловлень. Це дозволило стисло і разом з цим повно представити основні ідеї. Матеріал роботи може бути використаний у якості цікавого навчального прикладу у курсах «Спеціальні мови програмування», «Математична логіка», «Комп'ютерна алгебра і символьні обчислення» для ілюстрації глибоких зв'язків між алгеброю, логікою та практикою програмування.

Ключові слова: *проектування систем, багатосортна алгебра, наслідування, розширення, морфізми, алгебра висловлень, навчальний приклад.*

In the process of design of educational mathematical software based on symbolic transformations quite general approach to design and computations realization in multisorted algebras has been developed. It uses inheritance, extensions and morphisms of multisorted algebras. In the paper main principles of this approach on example of computations realization in propositional algebra are considered. This choice allows briefly but completely present basic ideas. Material of the paper can be used as good educational example in courses "Special Programming Languages", "Mathematical Logic", "Computer Algebra and Symbolic Computations" to illustrate the deep relationships between algebra, logic and practice of programming.

Keywords: *designing systems, polysort algebra, inheritance, extension, morphisms, algebra statements, case studies*

Введение

Разработка алгоритмов выполнения алгебраических вычислений - одна из основных задач, возникающих при реализации математических систем, основанных на символьных преобразованиях. Математическая модель этой задачи - многосортные алгебраические системы (МАС). Практика разработки даже достаточно простых математических систем учебного назначения [1-4] показала, что реализация алгебраических вычислений нуждается в тщательном предварительном проектировании МАС путем разработки иерархий сортов МАС и спецификаций интерпретаторов многосортных алгебраических операций [6]. В [5] подробно изложен общий подход (далее ІЕМ) к реализации

интерпретаторов многосортных алгебраических операций по их спецификациям, основанный на конструктивном уточнении понятия расширения в МАС, понятии морфизма в МАС, а также понятии наследования в МАС.

Использование подхода IEM при разработке математических систем учебного назначения показала его эффективность и даже универсальность. Подход IEM проиллюстрирован в [5] многочисленными примерами. Основные принципы и идеи IEM достаточно просты и хорошо известны как в математике, так и программировании. Поэтому, кроме подробного изложения, важно проиллюстрировать применение IEM на простом, хорошо известном и вместе с тем интересном примере, не требующем точных определений, связанных с МАС. В качестве такого примера выбрана алгебра высказываний.

В силу целого ряда причин [7, 8] для реализации вычислений, основанных на символьных преобразованиях, мы используем систему алгебраического программирования APS [9-12], адаптированную В.Песчаненко [7] для наших целей. APS использует технологии алгебраического программирования, основанные на системах правил переписываний и стратегиях переписываний. Таким образом, интерпретатор алгебраической операции определяется системой правил переписываний термов (rewriting rules system). Мы рассмотрим решения задач проектирования, синтеза и верификации интерпретаторов операций алгебры высказываний.

Постановка задачи

Стандартная формулировка задачи вычислений в алгебре высказываний состоит в следующем:

Вычислить значение $Val(F(x_1,...,x_n))$ логического выражения $F(x_1,...,x_n)$ в сигнатуре логических связок $<\&,\vee,\neg,0,1>$.

Под значением логического выражения $F(x_1,...,x_n)$ подразумевается т.н. каноническая форма выражения $F(x_1,...,x_n)$, т.е. такое выражение $Can(F)(x_1,...,x_n)$, что

1. $F(x_1,...,x_n) \equiv Can(F)(x_1,...,x_n)$ (1)
2. $F(x_1,...,x_n) \equiv G(x_1,...,x_n) \Rightarrow Can(F)(x_1,...,x_n) = Can(G)(x_1,...,x_n)$ (2)

Здесь знак «=» означает синтаксическое равенство, т.е. равенство в свободной алгебре термов. Определение (1)-(2) иллюстрируется диаграммой рис.1:

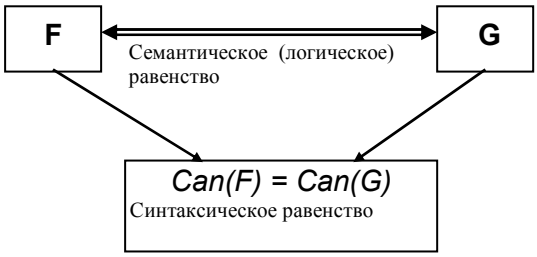

Рис. 1. Диаграмма определения канонической формы.

. В традиционных курсах математической логики изучаются две такие форме – СДНФ и СКНФ. Мы будем использовать т.н. рекурсивную каноническую форму (РНФ), которая несколько отличается от этих. Однако, из определения РНФ будет очевидно, что эти формы легко сводятся друг к другу (в смысле сложности вычислений).

Основная идея, рассматриваемая нами, иллюстрируется на рис.2.

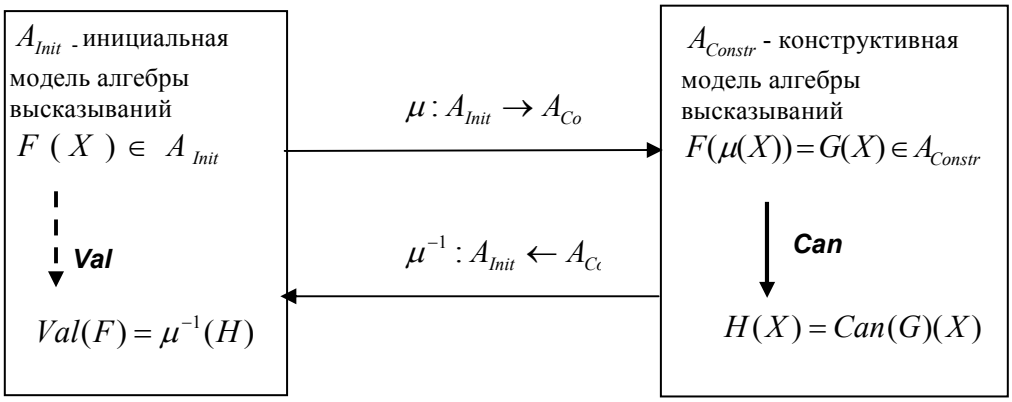

Рис.2. Иллюстрация парадигмы алгебраических вычислений определения канонической формы.

Для того, чтобы вычислить значение логического выражения $F(x_1,...,x_n)$, заданного в инициальной модели $\boldsymbol{A_{Init}}$ алгебры высказываний, алгоритм отображает атомы $x_1,...,x_n$ в конструктивное представление $\boldsymbol{A_{Constr}}$ алгебры высказываний. Затем в этом представлении выполняются вычисления (построение канонической формы), и их результат отображается обратно в инициальную модель алгебры высказываний (3). Отображения μ, μ^{-1} - суть изоморфизмы.

$$Val(F(X)) = \mu^{-1}(Can(F(\mu(X))) \quad (3)$$

Метод наследования

В алгебраических вычислениях практически всегда используются либо классические (абстрактные или конструктивные) алгебраические структуры, либо специально определяемые на высоком уровне абстракции алгебраические структуры. Для программирования это обстоятельство играет важную роль: в этих случаях можно повторно использовать программный код. Таким образом, проектирование алгебраических вычислений заключается в определении **иерархии наследования алгебр**,

Иерархия наследования для аксиоматического (абстрактного) определения алгебры высказываний изображена на рис. 3.

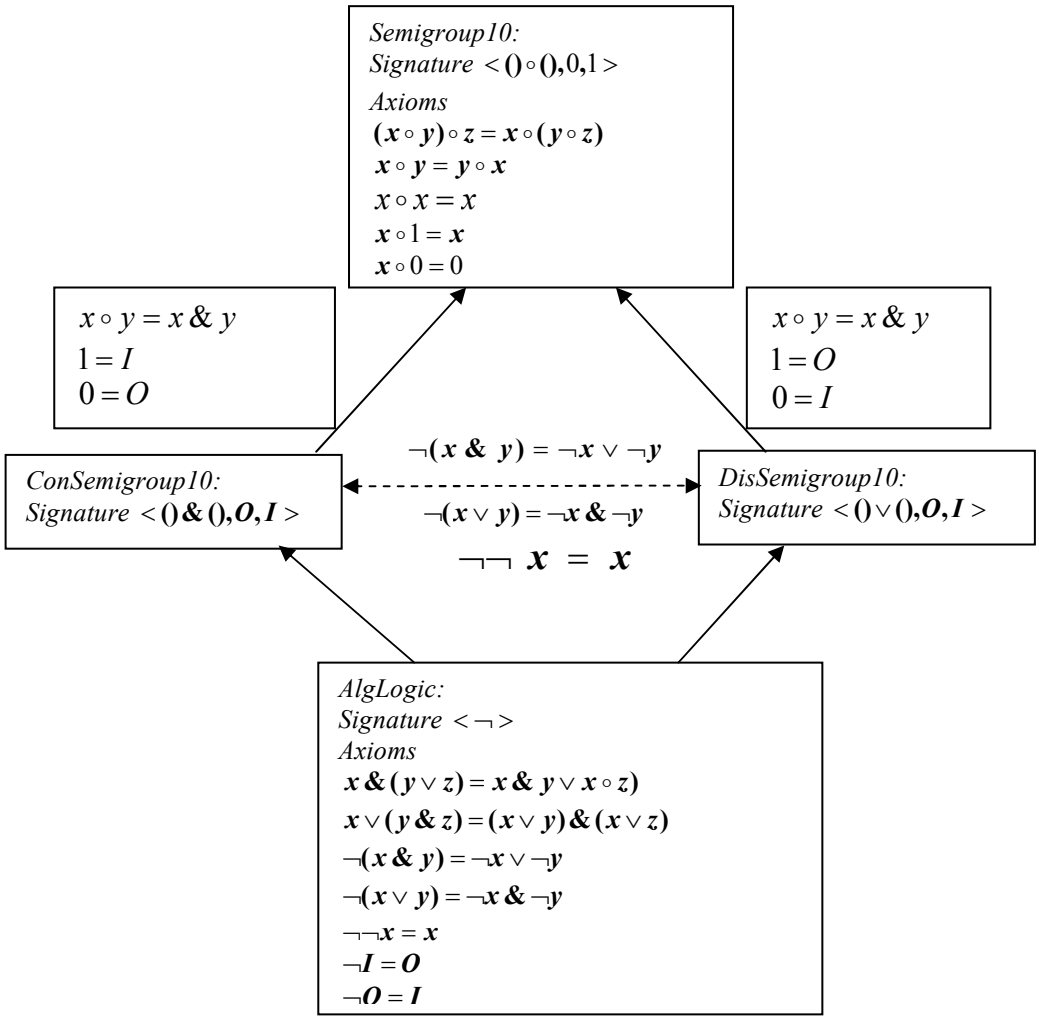

Рис.3. Диаграмма наследования алгебры высказываний.

Наследования в верхней части диаграммы рис.3. **Semigroup10::ConSemigroup10, Semigroup10::DisSemigroup10** специфичны: они определяют сигнатуры и аксиомы алгебр-наследников через переобозначения сигнатуры алгебры-родителя. Полугруппы **ConSemigroup10, DisSemigroup10** определяются через абстрактную полугруппу **Semigroup10** как коммутативные идемпотентные полугруппы с единицей и нулем. В классических объектных языках программирования переопределения имен не допускаются. Тем не менее, это отношение, на наш взгляд, является отношением наследования, поскольку свойства полугруппы **Semigroup10** переносятся на полугруппы **ConSemigroup10, DisSemigroup10.** Для математики как предметной области это – стандартный прием. В идеале спецификации алгебр должны каждое свойство описывать ровно один раз. Этого можно достичь только с помощью переобозначений сигнатур.

Наследование используется для определения интерпретаторов алгебраических операций. Дело в том, что аксиоматические описания играют еще и конструктивную роль. Они определяют имена констант и вычисления с константами. Т.о. уже в **Semigroup10** определяется интерпретатор полугрупповой операции $()\circ()$:

```
SGOperation := rs(a)
{
a°1=a, 1°a=a,
a°0=0, 0°a=0
};
```

Для полугрупп **DisSemigroup10, ConSemigroup10** тем самым определяются интерпретаторы дизъюнкции и конъюнкции

```
Dis := rs(a)
{
a∨O = a, O∨a = a,
a∨I = 0, I∨a = I
};
Con := rs(a)
{
A&I = a, I&a = a,
A&O = 0, O&a = O
};
```

Наследование может быть использовано, например, для определения алгоритмов решения прикладных задач в данной предметной области, если эти алгоритмы можно сформулировать в абстрактных терминах, т.е. без использования свойств носителя. Например, алгоритм Евклида может быть сформулирован в (абстрактном) евклидовом кольце.

Нижняя часть диаграммы наследования рис.3 определяет алгебру высказываний через множественное наследование дизъюнктивной и конъюнктивной полугрупп. Операции дизъюнкции и конъюнкции связываются аксиомами дистрибутивности. Кроме того, синтаксически, аксиоматически и конструктивно определяется логическая операция отрицания.

```
Not := rs()
{
¬O=I, ¬I=O
};
```

Замечание. На диаграмме рис. 3 показано, что операцию отрицания можно интерпретировать как прямой и обратный изоморфизмы полугрупп **DisSemigroup10, ConSemigroup10.** Этот факт обоснован законами де Моргана и законом двойного отрицания. Множественное наследование превращает этот изоморфизм в автоморфизм алгебры высказываний **AlgLogic,** известный как свойство двойственности.

Метод расширений

Алгебра **AlgLogic** определена аксиоматически диаграммой наследования рис.3. Это определение является исходным для дальнейшего. Мы будем использовать три конструктивных модели алгебры **AlgLogic,** изначально определяемые диаграммой наследования и расширений (рис. 4):

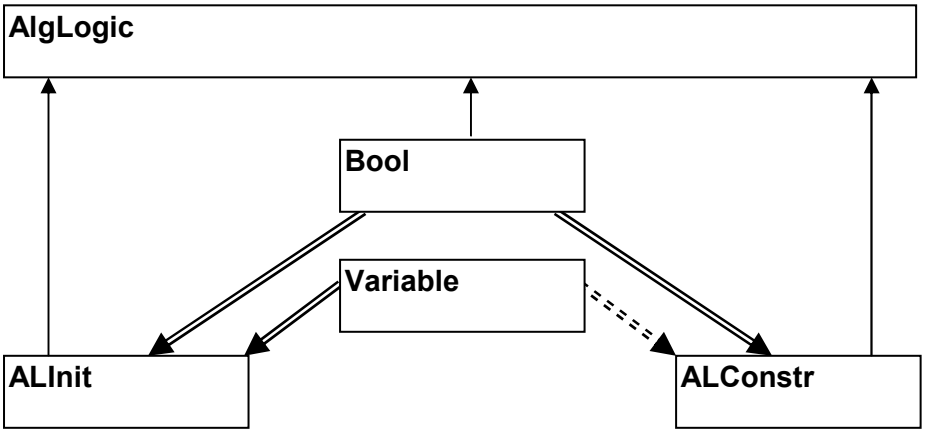

Рис. 4. Исходная диаграмма метода расширений определения алгебры высказываний.

Конструктивное определение алгебры заключается в определении ее носителя и интерпретаторов операций.

Алгебра **Bool** определена на носителе {**O, I**}. Интерпретаторы операций в этой алгебре определены полностью. Выполнение аксиом алгебры высказываний **AlgLogic** проверяется непосредственными вычислениями: все аксиомы **AlgLogic** являются тождествами **Bool.**

Алгебра **Variable** определена на носителе, содержащем некоторое множество переменных. Можно считать, что этот носитель содержит прописные буквы латинского алфавита и буквы с натуральными числами в качестве нижних индексов. На носителе алгебры определено отношение линейного порядка. Это означает, что **Variable** наследует абстрактную алгебру **LinearOrder.** Обычно линейный порядок в **Variable** основан на лексикографическом порядке и порядке на множестве **Nat** натуральных чисел.

Алгебры **ALInit** и **ALConstr,** в соответствии с диаграммой, являются расширением алгебр **Bool** и **Variable.** Т.о. носители этих алгебр содержат как логические значения **O, I,** так и переменные. На диаграмме рис. 4 мы используем двойные стрелки для обозначений отношения расширения.

Алгебра **ALInit** – инициальная алгебра многообразия, определенного сигнатурой логических связок и системой аксиом алгебры высказываний. Носитель **ALInit** – множество термов от переменных **Variable** в сигнатуре <&,∨,¬,O,I>.

Определение алгебры **ALConstr** – наша следующая непосредственная цель. В этом определении алгебры **Bool** и **Variable** играют роль базовых алгебр.

Определение

Пусть $Bool = \{O, I\}$, $Variable = \{x_1, x_2, ...\}$. Определим бесконечную возрастающую последовательность алгебр $A^{(0)} \subset A^{(1)} \subset ... \subset A^{(j)} \subset A^{(j+1)} \subset ...$ следующим образом:

$$1.\ A^{(0)} = Bool \tag{4}$$

2. $A^{(j+1)} = \{F : F = L(F^{(j)}, G^{(j)}, x_{j+1})\}$, где $F^{(j)}, G^{(j)} \in A^{(j)}$, $\boldsymbol{L}$-специальный функциональный символ, называемый конструктором алгебры ***ALConstr***. Мы будем использовать обозначение $A(x)$ для обозначения алгебры, построенной на основе алгебры $\boldsymbol{A}$ присоединением символа $\boldsymbol{x}$.

$$A(x) = \{F : F = L(F_1, G_1, x), F_1, G_1 \in A, x \notin A\} \tag{5}$$

Подчеркнем, что элементы алгебры $A(x)$ имеют вид $L(F, G, x)$. Введем т.н. селекторы (функции доступа) элемента алгебры следующими равенствами:

$$Arg_1(L(F,G,x)) = F,\ Arg_2(L(F,G,x)) = G,\ Var(L(F,G,x)) = x \tag{6}$$

3. На множестве ***ALConstr*** выполняются соотношения вложений:

$$L(F, F, x) = F \tag{7}$$

- вложение алгебры A в алгебру $A(x)$.

$$L(I, O, x) = x \tag{8}$$

- вложение алгебры элемента $x \in Variable$ в алгебру $A(x)$

Алгебра A_{Constr} теперь определяется как прямой предел возрастающей последовательности алгебр $A^{(j)}$:

$$\boldsymbol{A_{Constr}} = \bigcup_{\boldsymbol{j}=0}^{\infty} \boldsymbol{A^{(j)}} \tag{9}$$

Определим на A_{Constr} логические операции. Пусть

$L(A_1, B_1, x)$, $L(A_2, B_2, x) \in A(x)$, A_1, B_1, A_2, $B_2 \in A$.

Положим по определению

$$L(A1,B1,x) \vee L(A2,B2,x) = L(A1 \vee A2, B1 \vee B2, x), \tag{10}$$

$$L(A1,B1,x) \,\&\, L(A2,B2,x) = L(A1 \,\&\, A2, B1 \,\&\, B2, x), \tag{11}$$

$$\neg L(A1, B1, x) = L(\neg A1, \neg B1, x). \tag{12}$$

Правила (10) - (12) назовем основными правилами выполнения операций в алгебре $ALConstr$. Выведем т.н. дополнительные правила, применяя соотношение вложения (7) к основным правилам. В правиле (10) для дизъюнкции положим $A_1 = B_1$, подготавливая применение к нему соотношения вложения (7). Получим:

$A1 \vee L(A2,B2,x) = L(A1,A1,x) \vee L(A2,B2,x) = L(A1 \vee A2, A1 \vee B2, x)$.

Это равенство определяет условное переписывающее правило

$$Var(A1) < x \rightarrow A1 \vee L(A2,B2,x) = L(A1 \vee A2, A1 \vee B2, x). \tag{10a}$$

Аналогично

$L(A1,B1,x) \vee A2 = L(A1,B1,x) \vee L(A2,A2,x) = L(A1 \vee A2, B1 \vee A2, x)$.

т.е. соответствующее условное переписывающее правило имеет вид

$$Var(A2)<x \rightarrow L(A1,B1,x) \vee A2=L(A1 \vee A2,B1 \vee A2,x) \quad (10b)$$

Равенства (10), (10a), (10b) образуют систему переписывающих правил интерпретатора дизъюнкции для операндов вида L(A, B, x).

```
Dis := rs(A1, B1, A2, B2, x){
  L(A1,B1,x)∨ L(A2,B2,x)=L(A1∨A2,B1∨B2,x),
  Var(A1)<x → A1∨L(A2,B2,x)=L(A1∨A2,A1∨B2,x),
  Var(A2)<x → L(A1,B1,x)∨A2=L(A1∨A2,B1∨A2,x)
};
```

Вполне аналогично выводится система переписывающих правил для конъюнкции:

```
Con := rs(A1, B1, A2, B2, x)
{
  L(A1,B1,x)& L(A2,B2,x) = L(A1&A2,B1&B2,x),
  Var(A1)<x → A1&L(A2,B2,x)=L(A1&A2,A1&B2,x),
  Var(A2)<x → L(A1,B1,x)&A2=L(A1&A2,B1&A2,x)
};
```

Система переписывающих правил для отрицания состоит из одного основного правила, поскольку применение к основному правилу соотношения вложения (7) приводит к тождеству.

```
Not := rs(A1, B1, x)
{
   ¬L(A1,B1,x) = L(¬A1,¬B1,x)
};
```

Математическая сущность вычислений, описанных выше, заключается в следующем: для того, чтобы вычислить в алгебре A_{Constr}, построенной в соответствии с (9), результат бинарной операции $a \circ b$, где $a \in A^{(j)}$, $b \in A^{(k)}$, необходимо рассмотреть 3 случая: $j = k$, $j < k$, $j > k$. Случай $j = k$ - основной. Правила для случаев $j < k$, $j > k$ выводятся из основного правила «поднятием» операнда из алгебры с меньшим индексом в алгебру с большим индексом: если $j < k$, то $A = L(a, a, x_k)$. Далее применяется основное правило. Заметим, что результат, вообще говоря, нужно упростить, «опустив» его в алгебру с наименьшим возможным индексом.

```
Reduce := rs(A, x)
{
L(A, A, x) = A
};
```

Технически удобно использовать т.н. интерпретируемую отметку конструктора L(A,B,x) с функцией Reduce в качестве интерпретатора.

Таким образом, определение (4)-(7), (9) алгебры *ALConstr* как прямого предела возрастающей последовательности алгебр по сути определило алгоритм вывода (синтеза) интерпретаторов алгебраических операций. Это и есть основной результат метода расширений.

Теперь, если учесть систему правил, унаследованную *ALConstr* от *AlgLogic,* можно получить полные системы переписывающих правил

интерпретаторов логических операций $ALConstr$. Приведем интерпретатор операции дизъюнкции:

```
Dis := rs(a, A1, B1, A2, B2, x)
{
 a∨O = a, O∨a = a,
  a∨I = 0, I∨a = I,
 L(A1,B1,x)∨L(A2,B2 ,x)=L(A1∨A2,B1∨B2,x),
 Var(A1)<x → A1∨ L(A2,B2,x)= L(A1∨A2, A1∨B2,x),
 Var(A2)<x → L(A1,B1,x)∨A2 = L(A1∨A2, B1∨A2,x)
};
```

Метод морфизмов

Возвратимся к рассмотрению диаграммы рис.1, иллюстрирующей парадигму алгебраических вычислений. Ее правая часть – алгебра A_{Constr} , в нашем примере – алгебра $ALConstr$ полностью определена. Как видно из диаграммы, вычисления в алгебре A_{Init} определяются изоморфизмами

$$\mu : A_{Init} \to A_{Constr},\ \mu^{-1} : A_{Constr} \to A_{Init}$$

Для рассматриваемой нами алгебры в соответствии с (3) изоморфизм μ на элементах $Variable \subset A_{Init}$ определим равенством (8), прочитанным справа налево:

$$\boldsymbol{x = L(I,O,x)} \tag{13}$$

Обратный изоморфизм μ^{-1} определим равенством:

$$\boldsymbol{L(F,G,x) = x\,\&\,F \vee \neg x\,\&\,G} \tag{14}$$

Отметим, что основные правила (10)-(12) могут быть выведены из (14) равносильными преобразованиями в алгебре высказываний. Формула (14) представляет так называемую *рекурсивную нормальную форму* (РНФ) формулы алгебры высказываний.

```
ALInitToConstr := rs(x)
{
   IsVar(x)→ x = L(I,O,x)
};
```

В систему правил для обратного изоморфизма, кроме основного правила, можно включить еще и дополнительные правила, упрощающие вычисления с константами:

```
ALConstToInit := rs(x, A, B)
{
L(I,O,x) = x,
L(O,I,x) = ¬x,
L(A,O,x) = x&A,
L(O,B,x) = ¬x&B,
```

```
L(A,B,x) = x&A ∨ ¬x&B
};
```

Результирующее выражение представляет собой РНФ исходного выражения, атомами которого являются переменные и их отрицания, а также константы I, O. Для упрощения результата осталось применить интерпретаторы логических связок, определяющие вычисления с константами. Тогда результирующее выражение будет либо логической константой, либо содержать либо только переменные.

Верификация вычислений в алгебре высказываний

Включение аксиом в спецификации алгебры можно использовать для верификации системы интерпретаторов алгебраических операций. В случае, когда алгебра реализована методом расширений (4)-(8), схема рассуждений состоит в следующем:

Предположим, что $F(x,y,..)=G(x,y,...)$ - одна из аксиом алгебры. Тогда она должна быть тождеством алгебры A_{Init}, а ее образ $F(\mu x,\mu y,..)=G(\mu x,\mu y,...)$ - тождеством алгебры $A_{Constr.}$. Через $L(a,j+1)$ обозначим конструктор алгебры $A_{Constr.}$. Элементы $\mu x,\mu y,...$ можно считать принадлежащими алгебре $A^{(j+1)}$ для некоторого j. Подставив в равенство $F(x,y,..)=G(x,y,...)$ конструктивные представления $L(x,j+1),L(y,j+1),...$ элементов $x,y,...$ «в общем виде» как элементов $A^{(j+1)}$, получаем равенство в алгебре $A^{(j+1)}$. Значения левой и правой частей этого равенства можно вычислить, используя основные правила интерпретации операций алгебры. Результат вычислений будет иметь вид

$$\boldsymbol{L(F_x,F_y,....,j+1)=L(G_x,G_y,....,j+1)} \tag{15}$$

Ввиду каноничности представления элементов в $A_{Constr.}$ из (15) должно следовать

$$\boldsymbol{F_x=G_x,F_y=G_y,...} \tag{16}$$

Равенства (16) должны выполняться в алгебре $A^{(j)}$.

Таким образом, доказательство аксиом как тождеств алгебры $A_{Constr.}$ может быть осуществлено методом математической индукции по j - индексу алгебры в возрастающей последовательности $A^{(0)}\subset A^{(1)}\subset ...\subset A^{(j)}\subset A^{(j+1)}\subset ....$

Приведем пример: В качестве аксиомы выберем один из законов де Моргана:

$$\boldsymbol{\neg(x\,\&\,y)=\neg x\vee\neg y} \tag{17}$$

Положим x = L(A1, B1, u), y = L(A2, B2, u). Подставив значения x, y в (17), получим:

```
¬(L(A1,B1,u)&L(A2,B2,u)= ¬L(A1,B1,u)∨ ¬L(A2,B2,u)
```

Вычислив значения левой и правой части этого равенства, получим:

```
L(¬(A1&A2),¬(B1&B2),u)=L(¬A1∨¬A2,¬B1∨¬B2,u)
```

Приравнивая аргументы левой и правой части, получим:

$$\neg(A1\&A2)=\neg A1\vee\neg A2,\ \neg(B1\&B2)=\neg B1\vee\neg B2 \qquad (18)$$

Т.о. выполнимость равенства (17) в алгебре $A(u)$ свелась к выполнимости этого равенства в алгебре A. Осталось проверить выполнимость (17) в базовой алгебре $Bool$.

Если производные правила интерпретации операций синтезированы правильно, доказательство выполнимости равенства (17) в алгебре A_{Constr} закончено.

Отметим, что рассматриваемые нами вычисления в алгебре высказываний верифицируются очень легко, поскольку основные правила интерпретации (10)-(12) действуют покоординатно: алгебра $A(x)$ оказывается изоморфной прямому произведению $A\times A$. В других алгебрах все не так просто. Тем не менее, этот подход к верификации алгебраических вычислений весьма перспективен как с практической, так и теоретической точек зрения.

Заключение

В настоящей работе мы показали, что понятия *наследования*, конструктивного *расширения* и *морфизмов* МАС являются ключевыми для проектирования и реализации символьных вычислений. Мы сделали это на простом и наглядном примере вычислений в алгебре высказываний. Однако все конструкции без труда переносятся и на другие алгебраические системы. Прежде всего, это касается символьных вычислений в математических системах учебного назначения [13], где используются классические алгебры и алгебраические системы, хорошо описанные как аксиоматически, так и конструктивно.

На самом деле конструктивный подход, наряду с аксиоматическим подходом в алгебре общеизвестен. Идея конструктивного определения элемента носителя алгебры через элементы базовых алгебр систематически используется в алгебраических исследованиях. С другой стороны, механизм перегрузки алгебраических операций является стандартным средством программирования математических систем.

Таким образом, основным теоретическим результатом работы является идея систематического применения наследования, конструкций расширения и морфизмов в программировании сигнатур МАС как перегруженных сигнатур.

Практика использования этого подхода при разработке математических систем учебного назначения показала его эффективность и даже универсальность. Это и есть основной практический результат работы.

ЛИТЕРАТУРА

1. Lvov M., Kuprienko A., Volkov V. Applied Computer Support of Mathematical Training Proc. of Internal Work Shop in Computer Algebra Applications, Kiev. – 1993. – pp. 25-26.
2. Lvov M. AIST: Applied Computer Algebra System Proc. of ICCTE’93. Kiev. – pp. 25-26.

3. Львов М.С. Терм VII – шкільна система комп'ютерної алгебри Комп'ютер у школі та сім'ї. – 2004. – №7.- С. 27-30.
4. М.Львов. Концепция информационной поддержки учебного процесса и ее реализация в педагогических программных средах. Управляющие системы и машины.- 2009.-N2 N 2. — С. 52-57, 72.
5. Львов М.С. Синтез інтерпретаторів алгебраїчних операцій в розширеннях багатосортних алгебр. Вістник Харьківського національного университету. Серія «Математичне моделювання. Інформаційні технології. Автоматизовані системи управління» №847, 2009, с.221-238.
6. Goguen J., Meseguer J. Ordered-Sorted Algebra I: Partial and Overloaded Operations. Errors and Inheritance. SRI International, Computer Science Lab., 1987.
7. Песчаненко В.С. Розширення стандартних модулів системи алгебраїчного програмування APS для використання у системах навчального призначення // Науковий часопис НПУ імені М.П. Драгоманова Серія №2. Комп'ютерно-орієнтовані системи навчання: Зб. наук. Пр./Редкол.- К.:НПУ ім.М.П.Драгоманова, - №3 (10), 2005. - С.206-215.
8. Песчаненко В.С. Об одном подходе к проектированию алгебраических типов данных // Проблеми програмування. - 2006.- №2-3.-С. 626-634.
9. 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.
10. 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, Ukraine, 1998.
11. Капитонова Ю.В., Летичевский А.А., Волков В.А. Дедуктивные средства системы алгебраического программирования, Кибернетика и системный анализ, 1, 2000, 17-35.
12. Kapitonova J., Letichevsky A., Lvov M., Volkov V. Tools for solving problems in the scope of algebraic programming. Lectures Notes in Computer Sciences. –№ 958. – 1995. – pp. 31-46.
13. Львов М.С. Основные принципы построения педагогических программных средств поддержки практических занятий. Управляющие системы и машины.- 2006.-N6. с. 70-75.

Надійшла у першій редакції 28.07.2009, в останній - 14.10.2009.