Математические методы и модели в системах учебного назначения

УДК 004.421.6

М.С. Львов

Реализация вычислений в алгебрах числовых множеств в математических системах учебного назначения

Рассмотрены методы реализации вычислений – основных при программировании алгоритмов решений стандартных задач школьного курса алгебры: задач решения алгебраических и трансцендентных уравнений и неравенств в алгебрах множеств на числовой оси.

The methods of implementation of computations in algebras of sets on the number axis are considered. These computations are basic for the programming of solutions algorithms for standard algebraic school type tasks: the solution of algebraic and transcendental equations and inequalities.

Розглянуто методи реалізації обчислень – основних у програмуванні розв'язання стандартних задач шкільного курсу алгебри: задач розв'язання алгебраїчних та трансцендентних рівнянь та нерівностей в алгебрах множин на числовій осі.

Введение Разработка математических систем учебного назначения [1–4] показала, что реализация алгебраических вычислений в системах алгебраического программирования [5–9] требует предварительного проектирования многосортных алгебраических систем (МАС), которое состоит в разработке иерархий сортов МАС и спецификаций интерпретаторов многосортных алгебраических операций [10, 11].

Школьный курс алгебры использует несколько алгебр числовых множеств (АЧМ). Так, решением алгебраического уравнения является конечное числовое множество. Решение тригонометрического уравнения – периодическое числовое множество. Решением алгебраического неравенства является объединение числовых интервалов. Решение уравнения $f(x)*g(x)=0$ – объединение решений уравнений $f(x)=0$ и $g(x)=0$. И этот перечень можно продолжить. Поэтому и математические системы учебного назначения должны поддерживать вычисления в АЧМ.

Ключевые слова: математические системы учебного назначения, методы компьютерной алгебры, символьные преобразования, алгебраические и трансцендентные уравнения и неравенства.

Абстрактные свойства алгебр числовых множеств

Абстрактные (аксиоматические) свойства АЧМ определяются иерархией наследования, определяющей сигнатуры и аксиомы абстрактных алгебр [12] (рис. 1).

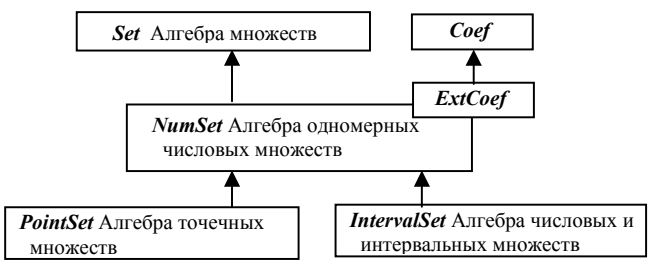

Рис. 1. Иерархия наследования одномерных числовых множеств

Set – абстрактная алгебра множеств, в которой определены сигнатуры и аксиомы теоретико-множественных операций.

ExtCoef – расширенная числовая ось – расширение линейно-упорядоченного числового поля *Coef* константами *PlusInf*, *MinusInf*, обозначающими плюс и минус бесконечность. Основные свойства:

```
a ∈ Coef → {PlusInf + a = PlusInf,
MinusInf + a = MinusInf;
  PlusInf – a = PlusInf,
```

MinusInf – a = MinusInf;
a - PlusInf = MinusInf,
a - MinusInf = PlusInf;
a > 0 → PlusInf*a = PlusInf,
a < 0 → PlusInf*a = MinusInf,
a < 0 → MinusInf*a = PlusInf,
a > 0 → MinusInf*a = MinusInf;
a > 0 → PlusInf/a = PlusInf,
a < 0 → PlusInf/a = MinusInf,
a > 0 → MinusInf/a = MinusInf,
a < 0 → MinusInf/a = PlusInf,
a/PlusInf = 0, a/MinusInf = 0
MinusInf < a, a < Plusnf };

NumSet – абстрактная алгебра одномерных числовых множеств, параметризованная *ExtCoef* как базовым множеством. Операция ***In*** принадлежности элемента множеству определена спецификацией ***In***:*ExtCoef*×*NumSet* → *Bool*.

Сигнатура *NumSet* включает операции *Inf*: *NumSet* → *Coef*, *Sup*: *NumSet* → *Coef*, *Min*: *NumSet* → *Coef*, *Max*:*NumSet* → *Coef* и другие операции, специфические для числовых множеств на числовой оси. Кроме теоретико-множественных операций, в алгебре *NumSet* определены арифметические операции *Add, Subsrtuct, Mult, UnionMult*:

Add(2): *Coef*×*NumSet* → *NumSet*, *ac*;
Subsrtuct(2): *Coef*×*NumSet* → *NumSet*, *NumSet*×*Coef* → *NumSet*;
Mult(2): *Coef*×*NumSet* → *NumSet*, *ac*;
UnionMult(2),()°(): *Nat*×*NumSet* → → *NumSet*, *ac*;

Операции *Add, Subsrtuct, Mult* используются для решения задач типа $ax+b \in S$, где $S \in NumSet$ (числовое множество). В терминах этих операций эта задача решается одним правилом:

$$ax+b \in S = x \in a^{-1} * (S-b). \qquad (1)$$

Семантика операции *UnionMult* более сложна:

$$k \circ S = S \cup 2 * S \cup ... \cup k * S. \qquad (2)$$

Эта операция является производной от операций *Union, Mult:*

UnionMult:{1°*S* = *S*, *k*°*S* = *k***S* ∪ ∪ (*k*-1)°*S*}.

PointSet – алгебра конструктивных точечных множеств. Элементами конструктивно определенного точечного множества может быть либо конечное множество точек, либо бесконечное конструктивно определенное точечное множество. Конструктивные определения бесконечных множеств реализованы в терминах операций

Element:*Function*×*Nat*→*Coef*, *NextElement*:*Function*×*PointSet*→*ExtCoef*.

Операция *Element* задает числовое множество определением $a_n = f(n)$, где функция f задается в спецификации каждой конкретной алгебры.

Операция *NextElement* задает числовое множество определением $a_{n+1} = f(a_n)$, где функция f задается в спецификации каждой конкретной алгебры.

Допускается также определение числовых множеств любыми конструктивными средствами, т.е. алгоритмами перечисления элементов множеств. Это рекомендуется делать путем определения алгебр, наследующих *NumSet* или *PointSet*.

IntervalSet – алгебра конструктивных числовых интервальных множеств. Элементы этой алгебры – суть конечные или бесконечные конструктивно определенные объединения числовых промежутков.

$$S = A_1 \cup A_2 \cup ... \cup A_k, \text{ или}$$
$$S = A_1 \cup A_2 \cup ... \cup A_k \cup ...,$$
$$A_i = (a,b), \text{ или } A_i = [a,b),$$
$$\text{или } A_i = (a,b] \text{ или } A_i = [a,b]. \qquad (3)$$

Как и для точечных множеств, определяем операции

Element(1):*Function*×*Nat*→*Interval*,
NextElement(2):*Function*×*PointSet*→ →*Interval*.

Структура расширений алгебр числовых множеств

Конструктивные реализации АЧМ заключаются в конструктивных определениях элементов алгебр и интерпретаторов операций в виде

систем переписывающих правил. Наш подход состоит в определении следующей структуры расширений АЧМ (рис. 2).

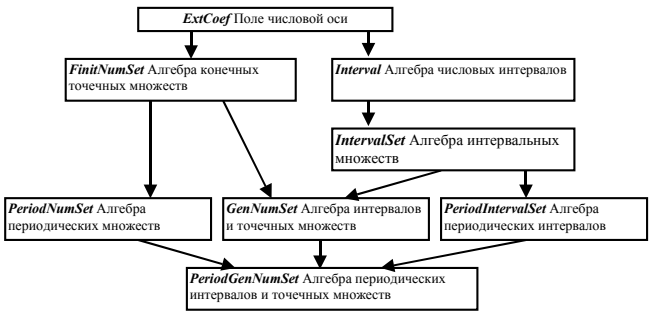

Рис. 2. Структура расширений одномерных числовых множеств

Алгебра *FinitNumSet* определяется спецификациями:

Sort FinitNumSet*::*PointSet;
Parameter *Field Coef*;
Constructor
{*FinitNumSet P* = {*Coef M*, *FinitNumSet Q*}
// Конструктор сорта
{***Empty,*** *P*} = {*P*}; {*M*, ***Empty***} = {*M*};
// Функции вложения
LeadElement(*P*) = *M*, *Deg*(*P*) = *M*, *Deg*(*M*) = *M*; // Функции доступа
M < *Deg*(*Q*);
// Контекстное условие
Form: { *M* ∈ *Coef*, Q ∈ *FinitNumSet*,
{***Empty,*** *P*} = {*P*}, {*M*, ***Empty***} = {*M*};
Deg(*M*) > *Deg*(*Q*) };
Operations
Union: {*M*, *P*} ∪ {*M*, *Q*} = (*M*, *P* ∪ *Q*};
Intersect: {*M*, *P*} ∩ {*M*, *Q*} = (*M*, *P* ∩ *Q*};
Substruct: {*M*, *P*} – {*M*, *Q*} = (*P* – *Q*};
In: {*M* ({*M*, *P*} = *True*,
M < *N* (*N* ({*M*, *P*} = *N* ({*P*},
M > *N* (*N* ({*M*, *P*} = *False*,
M ({*M*} = *True*,
M <> *N* → *N* ∈ {*M*} = *False*, // Или *N* ∈ {M} =*False*
M (*Empty* = *False*};
Add: *a* + (*M*, *P*) = (*a* + *M*, *a* + *P*);
Substruct:{*a* – (*M*, *P*) = (*a* – *M*, *a* – *P*), *M*, *P*)–*a* = = (*M* – *a*, *P* – *a*)};
Mult: *a**(*M*, *P*) = (*a**M, *a**P).

Отметим, что в спецификациях теоретико-множественных операций приведены лишь основные правила вычислений операций *Union, Intersect, Substract*. Таким образом, алгебру *FinitNumSet* определяют как линейное динамическое расширение *Coef* [9]. Поэтому полные системы интерпретирующих правил операций *Union, Intersect, Substract* можно вывести методами [10]. В спецификациях приведена полная система правил лишь для операции `In`.

Сорт *Interval*. Элементы алгебры ***Interval*** – числовые интервалы $A=(a,b)$, $A=[a,b)$, $A=(a,b]$, $A=[a,b]$. Рассматриваем и бесконечные, и пустые интервалы. Операция *Union* определена частично. Ее полное определение будет предоставлено в алгебре *IntervalSet*. Основная проблема при реализации этой алгебры – проблема частных случаев. Подсчитано, что система интерпретирующих правил для бинарной теоретико-множественной операции, написанная на основе полного перебора возможных вариантов взаимного расположения интервалов, имеет 96 правил. Для решения этой проблемы в спецификациях определяем ***Interval*** как статическое расширения базового сорта *OpenInterval*.

Сорт *OpenInterval*. Алгебра *OpenInterval* – алгебра открытых интервалов в числовой вехе. Операции алгебры – *Intersect, Union, Substract, In*. Заметим, что операция *Union* определена частично – для пересекающихся открытых интервалов. Операция *Substract* не является монотонной: тип результата не совпадает с типами аргументов. Поэтому она будет интерпретирована частично в алгебре *Interval* как производная от операций *Intersect, Union*. Полная интерпретация операций *Union, Substract* возможна лишь в алгебре *IntervalSet*. Кроме теоретико-множественных операций, определены операции частичного порядка «меньше», «больше» – соответственно *Les* и *Gre*, определенные для непересекающихся интервалов.

```
Intersect:{
Max(a,c) < Min(b,d)((a,b)((c,d) =
=(Max(a,c), Min(b,d)),
(a,b)∩(c,d) = Empty  // Max(a,c) >=
>=Min(b,d)};
Union:{Max(a,c) < Min(b,d)( (a,b) (
(c,d) = (Min(a,c), Max(b,d))};
```

```
In: {x ∈ (a, b) = (x > a)&(x < b)};
Les: {(a, b) < (c, d) = (b <= c) };
Gre: {(a, b) > (c, d) = (d >= a) }.
```

Сорт ***Interval*** определяется следующей спецификацией:

```
Sort Interval::IntervalSet;
Parameter Field ExtCoef;
Constructor
Interval P =  {// Конструктор сорта
((ExtCoef L, Bool LP), (ExtCoef R,
Bool RP))|Empty;
(L, False), (R, False)) = (L, R);
          // Вложение в OpenUnterval
(L, True), (L, True)) = {L};
          // Вложение в PointSet
LeftPoint(P) = L, RightPoint(P) =
= R,          // Селекторы
LeftClosed(P) = LP, RightClosed(P)=
= RP,
(L < R) or(L = R)&LP&RP    // Контекстное условие
Form:{L, R (Coef, LP, RP (Bool, L < P}.
```

Интерпретация монотонных операций в *Interval* отличается от их интерпретации в *OpenInterval* интерпретаторами функций *Min, Max.* Рассмотрим алгебру *PointBool* с носителем `PointBool A = (ExtField A, Bool LA)` и операциями

```
Min(2): PointBool ( PointBool (
PointBool,
Max(2): PointBool (PointBool (
PointBool,
Rev(1): PointBool ( PointBool;
Max:{a > b → Max((a, u),(b, v))=
=(a, u),
     a < b (Max((a, u),(b, v))=(b, v),
Max((a, u),(a, v))=(a, Max(u,v))};
Min:{a < b → Min((a, u),(b, v))=
=(a, u),
     a > b ( Min((a, u),(b, v))=(b, v),
Min((a, u),(a, v))=(a, Min(u,v))};
Rev:{Rev(a, u) = (a, Not(u))}.
```

Спецификации теоретико-множественных операций алгебры *Interval* имеют вид:

```
Les: {((a,u1), (b, v1)) < ((b,u2),
(d, v2)) = not(v1 | u2),
 ((a,u1), (b, v1)) < ((c,u2), (a,
v2)) = not(v2 | u1),
 ((a,u1), (b, v1)) < ((c,u2), (d,
v2)) = b < c };
Gre: ((a,u1), (b, v1) > ((c,u2),
(d, v2)) =
  ((c,u2), (d, v2)) < ((a,u1), (b, v1).
```

// Если интервалы не пересекаются, каждый из них лежит по одну сторону от другого.

```
Intersect:{Max(a,b)<Min(c,d)→(a,b)
∩(c,d)=(Max(a,b), Min(c,d)),
     (a,b)∩(c,d) = Empty
          // Max(a,b) >= Min(c,d)};
Union:{Max(a,c)<Min(b,d)→(a,b)∪(c,
d)=(Min(a,c), Max(b,d)),
v1∨u2 → ((a,u1), (b,v1))∪((b,u2),
(d,v2))=((a,u1),(d,v2))};
Substract:{
(a,b) - (c,d) = (a,b) ∩ (MinusInf,
Rev(c)) ∪ (a,b) ∩ (Rev(d), PlusInf)};
In: {a ∈ ((a,True), (b,v)) = True,
b ∈ ((a,u), (b,True)) = True,
x ∈ ((a,u), (b,v)) = (x > a)&(x < b)}.
```

Сорт *IntervalSet.* Элементы *IntervalSet* – суть объединения конечной упорядоченной последовательности числовых интервалов.

$$S = A_1 \cup ... \cup A_k,\ A_i = [(a_i, b_i)],$$
$$A_i \cap A_j = \varnothing,\ b_i \le a_{i+1}. \qquad (4)$$

Двойные скобки [(,)] обозначают или круглые, или прямоугольные скобки. Таким образом, A_i – попарно непересекающиеся элементы алгебры *Interval*, а алгебра *IntervalSet* – линейное динамическое расширение алгебры *Interval.*

```
Sort IntervalSet::NumSet;;
Parameter
Field ExtCoef;
Constructor
    {IntervalSet P = Interval S ++
IntervalSet Q;
Empty ++ P = P,     M ++ Empty = M;
LeadInterval(P) = S,
          // Функции доступа
LeftClosed(P) = LeftClosed(S),
          // Контекстное условие
 (RightPoint(S) < Inf(Q)) |
(RightPoint(S)=Inf(Q)&
Not(RightClosed(S)|LeftClosed(Q));
```

Form: *S* (*Interval*, *Q* (*Interval-Set*,
0 ++ *P* = *P*, *M* ++ 0 = *M*, *S* <> *Empty*};

Operations

Intersect:{(*a*∩*b*<>∅)→(*a*++*A*)∩(*b*++*B*)= =*a*∩*b*++((*a*∩*B*)∪(*b*∩*A*))++*A*∩*B*,
a < *b* ((*a*++*A*)((*b*++*B*)= *A* ((*b*++*B*),
a > *b* ((*a*++*A*)((*b*++*B*)=(*a*++*A*)(*B* }.

// Другие правила выводятся методами линейного динамического расширения

Union:{ (*a*∩*b*<>∅)→(*a*++*A*)∪(*b*++*B*)= =((*a*∪*b*)∪*A*)∪*B*,
a < *b* ((*a*++*A*)((*b*++*B*)= *a*++*A*((*b*++*B*),
a > *b* ((*a*++*A*)((*b*++*B*)= *b*++(*a*++*A*)(*B* }.

// Другие правила выводятся методами линейного динамического расширения

Substract:{(*a*∩*b*<>∅)→(*a*++*A*)−(*b*++*B*)= =(*a*−*b*)++((*A*−*b*)∪(*a*−*B*)∪(*A*−*B*)),
a < *b* ((*a*++*A*)−(*b*++*B*)= *a*++(*A*−*b*)((*A*−*B*),
a > *b* ((*a*++*A*)−(*b*++*B*)= (*a*++*A*)−*B*};
In:{*x* (*a*++*A* = (*x* (*a*) | (*x* (*A*)}.

Сорт *GenNumSet.* Элементы алгебры *GenNumSet* – суть объединения конечных упорядоченных последовательностей числовых множеств, каждое из которых – либо числовой интервал, либо конечное точечное множество:

$$S = A_1 \cup ... \cup A_k,\ A_i = [(a_i, b_i)] \vee A_i = \\ = \{a_{i1}, ..., a_{im}\},\ A_i \cap A_j = \varnothing. \qquad (5)$$

Рассмотрим алгебру числовых множеств *BaseNumSet* с носителем – объединением носителей алгебр *Interval* и *FinitNumSet.*

BaseNumSet = *Interval* ∪ *FinitNumSet.*

Частичный порядок на этом объединении определяется следующим образом:

$$A < B \Leftrightarrow \forall a \in A\ \forall b \in B\ \ a < b\,.$$

Если алгебра *BaseNumSet* определена, алгебру *GenNumSet* можно определить как линейное динамическое расширение алгебры *BaseNumSet.* Итак, спецификации операций алгебры *GenNumSet* не отличаются от спецификаций операций алгебры *IntervalSet*. Различие заключается в интерпретации этих операций в алгебре *BaseNumSet.* Таким образом, необходимо перегрузить теоретико-множественные операции на такие типы аргументов:

Intersect: *Interval* × *FiniteNumSet* → → *FinitNumSet*, *ac*;
Union: *Interval* × *FiniteNumSet* → → *GenNumSet*, *ac*;
Substract: *FiniteNumSet* × *Interval* → → *FinitNumSet*;
Substract: *Interval* × *FiniteNumSet* → → *GenNumSet*.

Продемонстрируем такое расширение на примере операции пересечения:

// Пересечение конечного числового множества и числового интервала

Intersect:{*a* ∈ *B* → {*a*,*A*}∩*B* ={*a*, *A*∩*B*), {*a*,*A*}∩*B* = *A*∩*B*}.

Алгебры периодических точечных множеств

Элементами алгебры периодических точечных множеств являются множества типа арифметических прогрессий. Необходимость в реализации периодических числовых множеств возникает, прежде всего, при решении тригонометрических уравнений и неравенств. В структуре расширений АЧМ (рис. 2) они представлены алгебрами *PeriodicNumSet* и *PeriodicIntervalSet.*

Определение 1. Пусть $S \subset F$ – числовое множество и t – положительный элемент действительного поля F (т.е. *ExtCoef*). Периодическим числовым множеством (S, t) с периодом t называется множество

$$(S, t) = \{s + kt:\ s \in S,\ k \in Z\}\,. \qquad (6)$$

Если множества S являются элементами АЧМ A, определяемая алгебра называется периодическим расширением A и обозначается через (A,t). Алгебра A называется базовой алгеброй периодического расширения (A,t). Примеры базовых алгебр периодических расширений *PeriodicNumSet* и *PeriodicIntervalSet* – соответственно алгебры *FinitNumSet, IntervalSet.*

Вычисление в периодических АЧМ с фиксированным периодом t по сути не отличаются от вычислений в базовой алгебре. Новые чер-

ты появляются в алгебрах, элементы которых являются арифметическими прогрессиями с разными периодами.

Определение 2. Пусть A – (базовая) АЧМ и T – подмножество действительного поля F: $T \subset F$**.** Периодическим расширением A с (множественным) периодом T называется алгебра

$$(A, T) = \{s + kt: s \in S \in A, k \in Z, t \in T\}. \quad (7)$$

В частности, в школьных тригонометрических задачах на уравнения и неравенства периоды, как правило, кратны π

Абстрактный сорт *PeriodicSet* является статическим расширением базовой алгебры *NumSet*. Однако не будем использовать вложения *NumSet* в *PeriodicSet* при условии $T = \{0\}$, за исключением вложений пустых множеств. Имеет смысл считать, что пустое множество является периодическим с периодом $t = 0$.

Определение 3. Представление элемента (S,t) алгебры (A,t) называется приведенным, если элемент t – наименьший положительный период и

$$\forall s \in S \in A \;\; 0 \le s < t. \quad (8)$$

Приведенное представление элемента алгебры (A,t) – каноническая форма. Приведение элемента (S,t) к его канонической форме по сути является операцией базовой алгебры. Далее рассмотрим алгоритмы приведения для алгебр *FinitNumSet, IntervalSet.*

При реализации теоретико-множественных операций основным является алгоритм приведения двух элементов алгебры *PeriodicSet* к общему периоду. Если два элемента имеют общий период, теоретико-множественные операции выполняются над базовыми множествами:

$$(A, t) \cap (B, t) = (A \cap B, t), (A, t) \cup (B, t) = (A \cup B, t), (A, t) - (B, t) = (A - B, t).$$

После выполнения операций результат надо приводить.

Определение 4. Наименьшим общим периодом периодов $t_1, t_2 \in T$ называется такой наименьший элемент $t \in T$, что для некоторых натуральных m,n выполняется равенство $t = nt_1 = mt_2$.

Пусть $t_1 = \alpha t_b, t_2 = \beta t_b, \alpha, \beta \in Q$, где t_b – базисный элемент T. Вычислим дробь $\frac{m}{n} = \frac{\alpha}{\beta}$. Числитель m и знаменатель n этой дроби суть искомые числа. Назовем их дополнительными множителями периодов t_2, t_1. Процедуру вычисления наименьшего общего периода t_1, t_2 обозначим через $LCP(t_1,t_2)$. Эта процедура возвращает m, n, $t = nt_1 = mt_2$. Таким образом,

$$LCP(t_1,t_2) = (t,m,n). \quad (9)$$

Сорт *PeriodNumSet.* Рассмотрим процедуру приведения элемента (S,t), $S \in FinitNumSet$. Она основана на операции $Frac(s,t)$ вычисления дробной части и операции $Entier(s,t)$ целочисленного деления двух положительных чисел s и t. Эти операции задаются соотношениями

$$s = Entier(s,t) \cdot t + Frac(s,t),$$
$$Entier(s,t) \in N,\ 0 \le Frac(s,t) < t. \quad (10)$$

Соотношение (10) можно обобщить на случай отрицательного числа s:

$$s = Entier(s,t) \cdot t + Frac(s,t),$$
$$Entier(s,t) \in Z,\ 0 \le Frac(s,t) < t. \quad (11)$$

Нетрудно увидеть, что $(s,t) = (Frac(s,t),t)$. Итак, для элемента $S \in FinitNumSet$ приведенным представлением является элемент $(S',t) = \{(s',t): s' = Frac(s,t)\}$, который обозначим через $(Frac(S,t),t)$. Отметим, что при приведении множества S некоторые из приведенных элементов S могут совпасть: если $s_1, s_2 \in S$ и $Frac(s_1 - s_2, t) = 0$, т.е. $s_1 - s_2 = kt, k \in Z$, $Frac(s_1,t) = Frac(s_2,t)$.

В результате приведения элемента (S,t) к виду $(Frac(S,t),t)$ каноническая форма (S,t) еще не построена. Дело в том, что множество $S' = Frac(S,t)$ может быть периодическим. Обозначим через $S + t$ множество $\{q: q = s + t, s \in S\}$. Предположим, что

$$S' = S_1 \cup S_2 \cup \ldots \cup S_k,\ k > 1,$$
$$Si \cap Sj = \varnothing,\ S_i + t = S_{i+1},\ i = 1,\ldots k-1.$$

Тогда множество S' соответственно назовем периодическим. В этом случае $(S',t)=(S_1,t/k)$ и построено представление (S,t) с периодом меньшим, чем t. Однако на практике для сравнения элементов $(S_1,t_1)=(S_2,t_2)$ проще использовать алгоритм приведения этих элементов к наименьшему общему периоду.

Предположим, что элементы $(S_1,t_1),(S_2,t_2)$ приведены: $S_1=Frac(S_1,t_1), S_2=Frac(S_2,t_2)$. Через $k\circ S$ обозначим множество $S\cup 2\cdot S\cup\ldots$ $\ldots\cup k\cdot S$. Тогда, если $(t,m,n)=LCP(t_1,t_2)$, то $(S_1,t_1)=(n\circ S_1,t),\ (S_2,t_2)=(m\circ S_2,t)$.

Итак, $(S_1,t_1)\equiv(S_2,t_2)\Leftrightarrow(n\circ S_1,t)=(m\circ S_2,t)$, где знак "≡" означает равенство элементов алгебры (A,t) (семантическое равенство), а знак "=" – синтаксическое равенство.

Сорт *PeriodIntervalSet.* Процедура приведения элемента (S,t), $S\in IntervalSet$ также использует операции $Entier(s,t), Frac(s,t)$. Рассмотрим числовой интервал – элемент алгебры $Interval$ как бесконечное множество точек числовой оси, к каждой из которых нужно применить процедуру приведения. Пусть $S=[(a,b)]$. Тогда периодическое множество определяется формулой $(S,t)=\bigcup_{k=0}^{\infty}[(a+kt,b+kt)]$. Если $|b-a|>t,\ (S,t)=(-\infty,\infty)=R$.

При $|b-a|=t,\ S=[a,b)\vee S=(a,b]\vee S=$ $=[a,b]\ (S,t)=(-\infty,\infty)=R$. Лишь при $|b-a|=$ $S=(a,b)$ или $|b-a|<t$ процедура дает нетривиальный результат. В этих случаях имеет место соотношения $(S,t)=([(a+kt,b+kt)],t)$, $k\in Z$). Итак, при $frac(a,t)<frac(b,t)$ $(S,t)=$ $=([(Frac(a,t),Frac(b,t)],t)$. Как и в алгебре *PeriodNumSet,* приведение базового множества S еще не определяет каноническую форму, поскольку приведенное множество может быть периодическим. При $frac(a,t)>frac(b,t)$ приведенное множество является объединением двух числовых интервалов: $(S,t)=[0,Frac(b,t))]\cup$ $\cup[(Frac(a,t),t),t)$.

```
Sort PeriodicSet::NumSet;
Parameter
NumSet BaseSet, Field PeriodSet;
Constructor{
PeriodicSet S = (BaseSet A, Period-
Set t);
(A, 0) = A, (Empty, t) = Empty;
BSet(A, t) = A,      // Функции доступа
Period(A, t) = t;   T > 0;     // Кон-
текстное условие
Form: A ∈ BaseSet, t ∈ PeriodSet,
t > 0};
Signature
LCP(2;3): PeriodSet×PeriodSet →
→ Int×Int×PeriodSet;
  Entier(2): PeriodSet×PeriodSet →
→ Int;
  Frac(2): PeriodSet×PeriodSet →
→ PeriodSet;
Operations
Intersect:{ (A, tA)∩(B, tB) =((A, B),
LCP(tA, tB)),
(A,B), (m,n,t))= (n°A∩m°B, t) }.
// Другие правила выводятся методами статического расширения
Union:{(A, tA)∪(B, tB) =((A, B),
LCP(tA, tB)),
(A,B), (m,n,t))=(n°A∪m°B, t) }.
// Другие правила выводятся методами статического расширения
Substract:{(A, tA)- (B, tB)=((A, B),
LCP(tA, tB)),
(A,B), (m,n,t))=(n°A-m°B, t)};
In:{a in (A, tA)=Frac(a, tA) in
Frac(A, tA)}.
```

Заключение. Предложенные методы реализации вычислений в алгебрах множеств на числовой оси используются при программировании алгоритмов решения стандартных задач школьного курса алгебры. Они легли в основу создания ряда математических систем учебного назначения, разработанных в НИИ информационных технологий Херсонского государственного университета.

1. *Lvov M., Kuprienko A., Volkov V.* Applied Computer Support of Mathematical Training // Proc. of Internal Work Shop in Comp. Algebra Appl., Kiev, 1993. – P. 25–26.

2. *Lvov M.* AIST: Applied Computer Algebra System // Proc. of ICCTE'93, 1993. – P. 25–26.
3. *Львов М.С.* Терм VII – школьная система компьютерной алгебры // Компьютер в школе и семье. – 2004. – № 7. – С. 27–30.
4. *Львов М.С.* Основные принципы построения педагогических программных средств поддержки практических занятий // УСиМ. – 2006. – № 6.– С. 70–75.
5. *Algebraic* programming system APS (user manual) / A. Letichevsky, Yu. Kapitonova, V. Volkov et al. – Kiev: Glushkov Inst. of Cybernetics, 1998. – 44 p.
6. *Tools* for solving problems in the scope of algebraic programming / Yu. Kapitonova, A. Letichevsky, M. Lvov et al. // Lectures Notes in Comp. Sci. – 1995. – № 958. – P. 31–46.
7. *Песчаненко В.С.* Расширение стандартных модулей системы алгебраического программирования *APS* для использования в системах учебного назначения // Комп'ютерно-ориентированные системы обучения: Зб. наук. пр. – К.: НПУ им. М.П. Драгоманова, 2005. – № 3 (10). – С. 206–215.
8. *Песчаненко В.С.* Об одном подходе к проектированию алгебраических типов данных // Проблемы программирования. – 2006. – № 2–3. – С. 626–634.
9. *Песчаненко В.С.* Использование системы алгебраического программирования *APS* для построения систем поддержки изучения алгебры в школе // УСиМ. – 2006. – № 4. – С. 86–94.
10. *Львов М.С.* Синтез интерпретаторов алгебраических операций в расширениях многосортных алгебр // Вестн. Харьк. нац. ун-та. Сер. Математическое моделирование. Информационные технологии. Автоматизированные системы управления. – 2009. – № 847. – С. 221–238.
11. *Львов М.С.* Об одном подходе к реализации алгебраических вычислений: вычисления в алгебре высказываний // Там же. – **12**, № 863. – С. 157–168.
12. *Львов М.С.*Метод спадкування при реалізації алгебраїчних обчислень в математичних системах навчального призначення // Системи управління, навігації та зв'язку. – 2009. – **3** (11). – С. 120–130.

E-mail: *lvov@ksu.ks.ua*

Окончание статьи Т.Л. Мазурок

Практическая целесообразность определяется построением и обучением нейронной сети, на основе которой возможен выбор управляющего воздействия индивидуально для каждого обучаемого, что является базовым элементом для формирования индивидуальной траектории обучения. Дальнейшее перспективное развитие данного подхода, на наш взгляд, – синтез нейронной сети, связывающей индивидуальные характеристики обучаемого с прогнозируемыми значениями уровней достижения компетенций, что позволит осуществлять интеллектуальную поддержку процесса управления обучением.

1. *Gritsenko V.* Higher education in information epoch: challenges of globalization // Proc. of the Fourth International Conf. «New Information Technologies in Education for All: e-education». – Kiev: IRTC, 2009. – P. 11–23.
2. *Грищенко В.И.* Информационно-коммуникационные технологии в образовании для всех – в ракурсе проблем общества знаний. – Киев: МНУЦИТиС, 2007. – 28 с.
3. *Мазурок Т.*Л. Модель обучаемого как объекта автоматизированного управления // Proc. of the Fourth International Conference «New Information Technologies in Education for All: e-education». – Kiev: IRTC, 2009. – P. 112–121.
4. *Александров Г.Н.*, *Молчанова И.А.* Использование кибернетических и синергетических аналогий при разработке новых информационных технологий обучения. – http://ito.edu.ru/2001/ito/II/4/II–4–19.html
5. *Белова Л.А.*, *Метешкин К.А., Уваров О.В.* Логико-математические основы управления учебными процессами высших учебных заведений. – Харьков: Вост.-рег. центр гуманит.-образ. инициатив, 2001. – 272 с.
6. *Ясінський В.В.* Системне моделювання процесів накопичення і дисипації знань // Системні дослідження та інформаційні технології. – 2007. – № 3. – С. 111–121.
7. *Орлов А.И.* Менеджмент: Учебник. – М.: Изумруд, 2003. – 298 с.
8. *Дружинин В.Н.* Структура психометрического интеллекта и прогноз индивидуальных достижений // Интеллект и творчество: Сб. науч. тр. / Ин-т психологии РАН. – Москва, 1999. – С. 5–29.
9. *Скороход А.В.* Элементы теории вероятностей и случайных процессов. – К.: Вища шк., 1980. – 218 с.
10. *Терехов В.А.* Нейросетевые системы управления. – М.: ВШ, 2002. – 183 с.
11. *Колесников А.А.* Синергетические методы управления сложными системами: теория системного синтеза. – М.: УРСС, 2006. – 240 с.
12. *Медведев В.С.*, *Потёмкин В.Г.* Нейронные сети. Matlab 6. – М.: ДИАЛОГ–МИФИ, 2002. – 496 с.

E-mail: *Mazurok62@mail.ru*