Вісник Харківського національного університету
Серія «Математичне моделювання. Інформаційні технології. Автоматизовані системи управління»
№ 847, 2009, с.221-238

УДК 519.6

Синтез інтерпретаторів алгебраїчних операцій в розширеннях багатосортних алгебр

М. С. Львов
Херсонський державний університет, Україна

Development of algorithms of algebraic computations is one of the main problems, which arises with realization of mathematical software based on symbolic transformations. Multisorted algebraic systems are mathematical model for this problem. Present paper deals with some solution of this problem. We propose the approach to realization of interpreters of multisorted algebraic operations by its specifications, based on constructive improvement of notion of extension of multisorted algebraic system. This approach is illustrated by examples of realization of interpreters of operations in the field of rational numbers, ring of one variable polynomials over the field, algebra of Boolean functions. Practice of this approach using for development of mathematical educational software shows its effectiveness and even universality.

Вступ

Розроблення алгоритмів виконання алгебраїчних обчислень є однією з основних задач, що виникають при реалізації математичних систем, основаних на символьних перетвореннях. Математичною моделлю цієї задачі є багатосортні алгебраїчні системи (надалі БАС). Практика розроблення навіть достатньо простих математичних систем навчального призначення [1, 2, 3] показала, що реалізація алгебраїчних обчислень потребує ретельного попереднього проектування БАС шляхом розроблення ієрархій сортів БАС та специфікацій інтерпретаторів багатосортних алгебраїчних операцій [8]. В силу цілого ряду причин [9] для реалізації обчислень, основаних на символьних перетвореннях, ми використовуємо систему алгебраїчного програмування APS [4, 5, 6], адаптовану В. Песчаненко [10] для наших цілей.

APS використовує технології алгебраїчного програмування, основані на системах правил переписувань та стратегіях переписувань. Таким чином, інтерпретатор алгебраїчної операції визначається системою правил переписувань термів (rewriting rules system).

Постановка задачі та структура роботи.

У даній роботі ми пропонуємо підхід до реалізації інтерпретаторів багатосортних алгебраїчних операцій за їх специфікаціями, оснований на конструктивному уточненні поняття розширення багатосортної алгебраїчної системи. Означення БАС та означення (конструктивного) розширення БАС надані у п.1. Типовим прикладом конструктивного розширення є приклад поля раціональних чисел як розширення кільця цілих чисел (приклад 2).

Конструктивні розширення БАС типізовані як статичні розширення, лінійні або бінарні динамічні розширення (означення 1.3.1, 2.1.).

Ми показуємо, що інтерпретатор алгебраїчної операції у конструктивному розширенні може бути синтезований автоматично з її специфікації, яка визначає

по-перше, правила інтерпретації операції у розширенні, по-друге, умови вкладення основної алгебри у її розширення.

Алгоритм синтезу інтерпретатора операції визначається типом розширення. Тому у п.2.2. ми наводимо приклади реалізації інтерпретаторів операцій поля раціональних чисел (статичні розширення), поля квадратичних радикалів (бінарні динамічні розширення), кільця багаточленів однієї змінної (лінійні динамічні розширення) та алгебри висловлень(бінарні динамічні розширення).

1. Багатосортні алгебри як математична модель алгебраїчних обчислень

Означення 1.1 Нехай **U**=$\{u_1,\ldots,u_k\}$ - скінчена множина символів, яка називається сигнатурою сортів. Символи $u_l, l \in \{1,...,k\}$ називаються іменами сортів або просто сортами.

Далі ми будемо користуватися, зокрема, такими сортами – елементами сигнатури сортів:

Variable – ім'я сорту – множини змінних,

Bool - ім'я сорту – множини логічних значень,

Nat - ім'я сорту – множини натуральних чисел,

Int - ім'я сорту – множини цілих чисел,

Real - ім'я сорту – множини дійсних чисел.

Інші імена сортів ми будемо вводити при означенні відповідних алгебраїчних понять.

Означення 1.2 Нехай, далі, $\mathbf{S} = \{S_{u_1},...,S_{u_k}\}$ - скінчене сімейство множин, індексованих іменами сортів, які називаються областями значень відповідних сортів.

$S_{Variable}$ - множина змінних,

S_{Bool} - множина {False, True},

S_{Nat} - множина натуральних чисел,

S_{Int} - множина цілих чисел,

$S_{\mathrm{Re}al}$ - множина дійсних чисел.

Означення 1.3 Багатосортною операцією f над сімейством **S** називається відображення $f : S_{u_1} \times S_{u_2} \times ... \times S_{u_m} \to S_v$, де $u_1,..., u_m, v \in \mathbf{U}$ - сорти аргументів та області значень операції f, відповідно, а m – арність операції f.

Тип операції визначається переліком імен сортів її аргументів та іменем сорту області її значень. Тип операції f будемо позначати через $(u_1,..., u_m) \to v$. Сигнатурою Σ операцій називається скінчена множина символів операцій разом з відображенням, яке кожному символу $\varphi \in \Sigma$ співставляє багатосортну операцію f_φ разом з її типом (якщо φ символ операції, то вираз $\varphi : (u_1,..., u_m) \to v$ означає, що цьому символу співставлено операцію типу $(u_1,..., u_m) \to v$).

Багатосортною операцією, є, наприклад, операція множення у векторному просторі. Якщо *VectorSpace* – ім'я сорту-множини векторів на полем *Real* дійсних чисел, то операція множення *Mult* "*" задає відображення

$$\texttt{Mult : Real} \times \texttt{VectorSpace} \rightarrow \texttt{VectorSpace}$$

Надалі ми будемо користуватися більш звичними, тобто традиційними математичними позначеннями операцій. Оскільки множення вектора на скаляр э бінарною операцією з інфіксною формою запису, маємо:

$$Real * VectorSpace \rightarrow VectorSpace$$

Означення 1.4. Визначимо сорт *Bool* з областю значень $S_{Bool} = \{True, False\}$. Багатосортним предикатом P називається відображення $P : S_{u_1} \times ... \times S_{u_m} \rightarrow S_{Bool}$, де $u_1, ..., u_m \in \mathbf{U}$, послідовність $u_1, ..., u_m$ визначає тип предикату, а число m – його арність. Сигнатура Π багатосортних предикатів визначається аналогічно сигнатурі операцій як множина операцій символів предикатів, яким поставлені у відповідність багатосортні предикати разом з їх типами.

Означення 1.5. Багатосортною алгебраїчною системою **A** називається четвірка $\mathbf{A} =< \mathbf{S}, \mathbf{U}, \Sigma, \Pi >$, де **S** – множина сортів, індексованих символами множини **U**, $\Sigma = \{\varphi_1, ..., \varphi_l\}$ - сигнатура багатосортних операцій, $\Pi = \{\pi_1, ..., \pi_p\}$ - сигнатура багатосортних предикатів.

Зауваження. Оскільки сорт *Bool* можна включити до множини сортів, предикати можна розглядати як багатосортні операції. Тому, об'єднавши сигнатури операцій та предикатів, замість розгляду багатосортних алгебраїчних систем далі ми будемо розглядати багатосортні алгебри.

Означення 1.6. Нехай $\mathbf{A} =< \mathbf{S}, \mathbf{U}, \Sigma >$ багатосортна алгебра та $u, v \in \mathbf{U}$ символи сортів. Будемо казати, що сорт *v* залежить від сорту *u*, якщо одна з операцій сигнатури Σ має тип $u_1 \times ... \times u \times ... \times u_m \rightarrow v$. Через $\mathbf{U}_v$ позначимо підмножину сортів, від яких залежать сорт *v*. Підмножину елементів Σ, що мають тип $u_1 \times ... \times u \times ... \times u_m \rightarrow v$ позначимо через Σ_v, а сімейство областей значень сортів $\mathbf{U}_v$ позначимо через $\mathbf{S}_v$. Обмеженням $\mathbf{A}_v$ багатосортної алгебри **A** на сорт *v* називається багатосортна алгебра $\mathbf{A}_v =< \mathbf{S}_v, \mathbf{U}_v, \Sigma_v >$.

Таким чином, багатосортна алгебра **A** може бути представлена набором обмежень (алгебр) $\mathbf{A}_v, v \in \mathbf{U}$, тобто $\mathbf{A} =< A_{u_1}, ..., A_{u_k} >$.

Приклад 1

Розглянемо програмну систему, яка підтримує спрощення цілих алгебраїчних та тригонометричних виразів. Ядро цієї системи має підтримувати обчислення у кільці поліномів та кільці тригонометричних виразів багатьох змінних над полем раціональних чисел. Специфікаціям підлягають такі алгебри – обмеження на указані сорти:

MultiPolynom – кільце поліномів багатьох змінних.

MultiPolynom + MultiPolynom → MultiPolynom

MultiPolynom - MultiPolynom → MultiPolynom
*MultiPolynom * MultiPolynom → MultiPolynom*
MultiPolynom ^ Int → MultiPolynom
*Rat * MultiPolynom → MultiPolynom*
MultiPolynom / Rat → MultiPolynom

MultiTrig – кільце тригонометричних поліномів багатьох змінних.

Sin(LinComb) → MultiTrig
Cos(LinComb) → MultiTrig
MultiTrig + MultiTrig → MultiTrig
MultiTrig - MultiTrig → MultiTrig
*MultiTrig * MultiTrig → MultiTrig*
MultiTrig ^ Int → MultiTrig
*Rat * MultiTrig → MultiTrig*
MultiTrig / Rat → MultiTrig

LinComb – векторний простір лінійних комбінацій багатьох змінних (аргументи тригонометричних поліномів)

Pi
LinComb + LinComb → LinComb
LinComb - LinComb → LinComb
*Rat * LinComb → LinComb*
LinComb / Rat → LinComb

Rat – поле раціональних чисел (коефіцієнти поліномів та тригонометричних поліномів)

Rat + Rat → Rat
Rat - Rat → Rat
*Rat * Rat → Rat*
Rat ^ Int → Rat
Rat / Rat → Rat *//Знаменник не дорівнює нулю*
Rat = Rat → Bool
Rat < Rat → Bool
Rat > Rat → Bool
Rat <= Rat → Bool
Rat => Rat → Bool

Відношення залежності сортів породжує структуру залежності на множині алгебр $\mathbf{A}_u, u \in \mathbf{U}$: алгебра $\mathbf{A}_v$ залежить від алгебри $\mathbf{A}_u$ якщо сорт *v* залежить від сорту *u*. Якщо відношення залежності не має циклів, то багатосортну алгебру можна будувати крок за кроком (інкрементно), будуючи алгебру $\mathbf{A}_v$, якщо алгебри, від яких $\mathbf{A}_u$ залежить, вже побудовані.

1.2. Аксіоми та конструкції багатосортної алгебри

Для побудування алгебр $\mathbf{A}_u$ ми використовуємо їх аксіоматичні та конструктивні описи (означення).

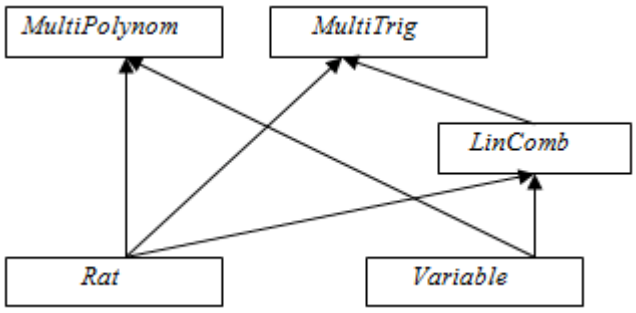

Рис. 1. Діаграма відношення залежності алгебр прикладу 1.

Означення 1.7. *Аксіомою* алгебри $\mathbf{A}_u$ називається тотожність або умовна тотожність у сигнатурі Σ_u. *Аксіоматичним описом* алгебри $\mathbf{A}_u$ є, за означенням, є скінчений набір аксіом (система аксіом) алгебри $\mathbf{A}_u$.

Ми будемо користуватися алгебраїчною термінологією та відповідними системами аксіом з підручника [11].

Конструктивний опис алгебри $\mathbf{A}_u$ - це означення конструктору сорту $\mathbf{S}_u$ (тобто визначення термів сорту $\mathbf{S}_u$) та множини інтерпретаторів операцій Σ_u.

Означення 1.8. *Сигнатурою* конструкторів $\mathbf{T}_u$ називається скінчена множина символів операцій разом з відображенням, яке кожному символу $\tau \in \mathbf{T}_u$ співставляє символ сорту u разом з переліком символів сортів його аргументів (якщо τ символ операції, то вираз $u = \tau(u_1,...,u_m)$ означає, що цьому символу співставлено символ сорту u та символи сортів його аргументів $u_1,...,u_m$.)

Конструктором сорту $\mathbf{S}_u$ алгебри $\mathbf{A}_u$ називається система рівностей**,** яка визначає синтаксично елементи сорту $\mathbf{S}_u$ як терми у сигнатурі T_u. Отже, сорт $\mathbf{S}_u$ – це множина термів у (власній) сигнатурі T_u конструктора сорту $\mathbf{S}_u$.

Означення 1.8 є ключовим у нашому підході до специфікації алгебраїчних обчислень. Тому ми надамо необхідні приклади та пояснення.

Приклад 2. Поле *Rat* раціональних чисел

Раціональні числа представлено у вигляді звичайних дробів. Конструктор сорту визначає стандартну форму представлення елементу цього сорту. Частіше за все це *канонічна форма.* Таким чином,

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

Роль конструктора сорту грає символ - горизонтальна риска. Цей же символ математики використовують для позначення операції ділення, зокрема, у *Rat.* Для задачі специфікацій алгебраїчних обчислень це не зовсім зручно. Тому ми вводимо окремо поняття сигнатури операцій Σ та сигнатури конструкторів T.

Зокрема, для конструктора сорту *Rat* ми використовуємо *подвійну нахильну риску*:

$$\boldsymbol{S_{Rat}} = \{\boldsymbol{p} // \boldsymbol{q} : \boldsymbol{p} \in \boldsymbol{S_{Int}}, \boldsymbol{q} \in \boldsymbol{S_{Nat}}, \boldsymbol{GCD(p,q)} = 1\} \quad (2)$$

Ще однією важливою обставиною є те, що у стандартних формах представлення елементів сортів синтаксичні аспекти означення, які визначаються символами конструкторів, практично завжди поєднуються з семантичними аспектами, що задаються у вигляді контекстних умов, тобто предикатів. У нашому прикладі таким предикатом є рівність $GCD(p,q) = 1$.

Приклад 3. Кільце *Polynom* поліномів однієї змінної над полем *Rat*.

Елементи цього поля - поліноми, які представлені у вигляді суми мономів, упорядкованих у порядку спадання степенів. Це означення має бути рекурсивним, причому окремо треба визначити поняття степеня полінома.

$$S_{Polynom} = \{Q : Q = M + + P, M \in S_{Monom}, P \in S_{Polynom}, \deg Q \overset{df}{=} \deg M; \deg(M) > \deg(P)\} \cup S_{Monom} \quad (3)$$

Для визначення носіїв сортів ми будемо користуватися спеціальною мовою специфікацій, яка допускає нерекурсивні та рекурсивні синтаксичні визначення елементів сортів, визначення функцій доступу та контекстних умов. Наприклад:

```
Rat r ={(Int a)//(Nat b);                    // Конструктор сорту
  Num(r) = a, Den(r) = b;                    // Функції доступу
  GCD(a, b) = 1                              // Контекстна умова
};
Monom M =    {(Rat c)$(Const Variable x)^^(Nat n);   // Конструктор сорту
    Coef(M) = c,                             // Функції доступу
    Var(M) = x,
    Deg(M) = n
};
Polynom P = {(Monom M)++(Polynom Q);         // Конструктор сорту
  LeadMon(P) = M,                            // Функції доступу
  LeadCoef(P) = Coef(M),
  Deg(P) = Deg(M);
  Deg(P) > Deg(Q)                            // Контекстна умова
 };
```

Для того, щоб реалізувати обчислення в деякій алгебрі $\mathbf{A}_v, v \in \mathbf{U}$, потрібно реалізувати алгоритми виконання кожної її операції таким чином, щоб виконувались аксіоми цієї алгебри.

Означення 1.9. *Інтерпретатором* операції сигнатури Σ_u називається функція, яку реалізовано алгоритмом виконання відповідної операції.

Інтерпретатори операцій визначають засобами мови програмування. Для наших цілей застосовується мова *APLAN*. Отже, ми включаємо цю мову у мову специфікацій.

Таким чином, для аксіоматичного та конструктивного опису алгебри $\mathbf{A}_v$ до її означення ми включаємо скінчену множину аксіом Ax_v та скінчену множину

інтерпретаторів I_v. Тоді багатосортна алгебра $\mathbf{A}_v$ визначається наступним чином: $\mathbf{A}_v = <\mathbf{S}_v, \mathbf{U}_v, \mathrm{T}_v, \Sigma_v, \mathrm{Ax}_v, \mathrm{I}_v>$.

1.3. Методи побудови багатосортної алгебри

Побудова структури багатосортних алгебр полягає у специфікуванні, прототипуванні та реалізації алгебраїчних обчислень. Специфікації структури багатосортних алгебр здійснюється у термінах розширень, гомоморфізмів, ізоморфізмів та спадкування багатосортних алгебр. Таким чином, разом з діаграмами залежності, які є графічними моделями специфікацій сигнатур операцій та конструкторів, будуються діаграми розширень, діаграми морфізмів (тобто ізоморфізмі та гомоморфізмів) та діаграми спадкування. Ми розглянемо метод розширень. Методи морфізмів та спадкування виходить за рамки даної роботи.

1.3.1 Метод розширення алгебр

Означення 1.10. Нехай $\mathbf{A}_u$ та $\mathbf{A}_v$ – багатосортні алгебри. Багатосортна алгебра $\mathbf{A}_v$ називається *розширенням* багатосортної алгебри $\mathbf{A}_u$, якщо $S_u \subseteq S_v$ та для будь-якої пари операцій f_1 та f_2 типів відповідно $\varphi : (u_1,...,u_m) \to u$ та $\varphi : (v_1,...,v_m) \to v$, якщо виконуються входження $S_{u_1} \subseteq S_{v_1},..., S_{u_m} \subseteq S_{v_m}$, то $\forall (a_1,...,a_m) \in S_{u_1} \times ... \times S_{u_m}$ має місце рівність $f_1(a_1,...,a_m) = f_2(a_1,...,a_m)$.

Вкладенням називається ізоморфне відображення $Red: S_u \to S'_v$, яке відображає S_u на підмножину $S'_v \subset S_v$. Обмеження алгебри $\mathbf{A}_v$ на підмножину S'_v, ізоморфне $\mathbf{A}_u$, визначається системою умовних тотожностей $E_1(x),...,E_k(x)$: $S'_v = \{a \in S_v \mid E_1(a),...,E_k(a)\}$. Застосування системи $E_1(x),...,E_k(x)$ як системи переписувань «спрощує» терм $a \in S'_v$ до терму $a' \in S_u : Red^{-1}(a) = a'$.

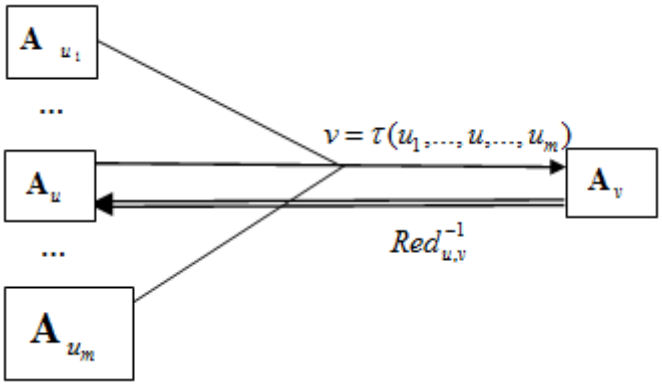

Рис 2. Фрагмент діаграми розширень як конструкцій та вкладень

Конструктивний опис розширення $\mathbf{A}_v$ полягає в описі конструктора $\mathbf{A}_v$ та вкладення $\mathbf{A}_u$ в алгебру $\mathbf{A}_v$. На рис. 2 подвійною стрілкою позначено той факт, що $\mathbf{A}_v$ є розширенням алгебри $\mathbf{A}_u$, в $\mathbf{A}_v$ визначено конструкцію $v = \tau(u_1,...,u,...,u_m)$ та вкладення $Red_{u,v}$.

Довідка. Відношення розширення БАС, (відношення «сорт-підсорт») є основним у даній роботі. Багатосортні алгебри, частково упорядковані цим відношенням, називають упорядковано-сортними. Основи теорії упорядковано-сортних алгебр у її застосуванні до теорії програмування закладені у [12]. Російською мовою вони викладені у [13].

Приклад 4. Розглянемо конструктор поля *Rat* (приклад 2). У відповідності до означення він визначає конструкцію *Rat,* аргументами якої є сорти *Int* та *Nat*. Доповнимо специфікації сорту *Rat* вкладенням $\mathrm{Re}\,d : Rat \to Int$, яке визначено рівністю $\mathrm{Re}\,d(a//1) = a$. Таким чином сорт *Rat* визначено конструктивно як розширення сорту *Int.*

Розглянемо тепер конструктор кільця *Polynom* (приклад 3). Він визначає рекурсивно конструкцію *Polynom*, аргументом якої є сорт *Monom.* Доповнимо специфікації сорту *Polynom* вкладенням $\mathrm{Re}\,d : Polynom \to Monom$, яке визначено рівністю $\mathrm{Re}\,d(M++0) = M$. Отже, сорт *Polynom* визначено як розширення сорту *Monom.*

У свою чергу, сорт *Monom* є розширенням сорту *Degree* з функцією *Red*, визначеною рівністю `1$x^^k = x^^k`, розширенням сорту LinMonom з функцією *Red*, визначеною рівністю `a$x^^1 = a$x` та розширенням сорту *Rat* з функцією *Red*, визначеною рівністю `a$x^^0 = a`. Сорти *Degree* та *LinMonom* є розширенням сорту *Variable* з функціями редукції, заданими відповідно рівностями `x^^1 = x` **та** `1$x = x`**.** Отже, діаграма розширень має вид

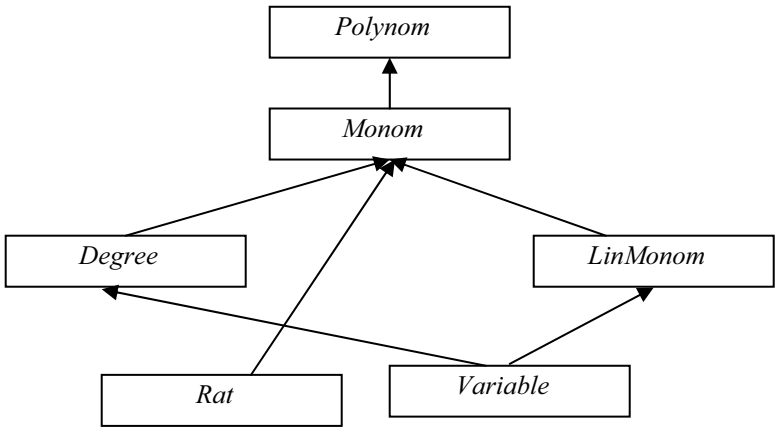

Рис 3. Діаграма розширень прикладу 4.

Механізм *розширень* є одним з основних методів специфікацій багатосортних алгебр. Зокрема, він дозволяє визначити перевантажені алгебраїчні операції та функції приведення алгебраїчних типів.

2. Методи специфікацій алгебраїчних обчислень. Синтез алгебраїчних програм.

2.1. Статичні та динамічні розширення

Означення 2.1. Розширення B алгебри A називається статичним (нерекурсивним), якщо у його конструкторі $B = \varphi(A_1,...,A,...,A_n)$ жоден з аргументів не співпадає з $\boldsymbol{B}$.

Приклади статичних розширень:

Поле Rat є статичним розширенням кільця Int. Дійсно, $RatR = (IntA)//(NatB)$

Півгрупа мономів `Monom` однієї твірної є статичним розширенням поля коефіцієнтів `Coef`, оскільки $MonomM = (Coef\ a)\$(Var\ x)\hat{}\ (Nat\ N)$.

Означення 2.2. Розширення B алгебри A називається динамічним (рекурсивним), якщо у його конструкторі $B = \varphi(A_1,...,A,...,A_n)$ щонайменше один з аргументів співпадає з B .

Конструктор динамічних розширень є рекурсивним визначенням, отже, місить як базову, так і рекурентну частини.

Приклади динамічних розширень:

Векторний простір *LinComb* лінійних комбінацій багатьох змінних над полем *Coef* є лінійним динамічним розширенням *LinMonom*, елемент якого має вигляд $a\$x$. Елемент $w \in LinComb$ має вид $w = a_1\$x_1 + +a_2\$x_2 + +... + +a_m\$x_m$.

$$LinComb\ w = (LinMonom\ u) + +(LinComb\ w)$$

Кільце многочленів однієї змінної *Polynom* над полем *Coef*. Це кільце в алгебрі традиційно позначається через $F[x]$.

$$Polynom\ w = (MonomM) + +(Polynom\ w)$$

Означення 2.3. Динамічне розширення B алгебри A називається лінійним, якщо у його конструкторі $B = \varphi(A_1,...,A,...,A_n)$ у точності один з аргументів співпадає з B .

Динамічне розширення B алгебри A називається бінарним, якщо у його конструкторі $B = \varphi(A_1,...,A,...,A_n)$ у точності два аргументи співпадають з B .

Приклад 5. Поле квадратних радикалів

Приклади 3, 4 є прикладами лінійних динамічних розширень. Розглянемо приклад бінарного динамічного розширення:

Числове поле *Rad*, елементами якого є лінійні комбінації квадратних коренів натуральних чисел, вільних від квадратів, з раціональними коефіцієнтами, можна представити як бінарне розширення поля *Rat* за допомогою наступної конструкції. Нехай $p_1, p_2,..., p_n,...$ - послідовність усіх простих чисел, розташованих у порядку зростання. Позначимо також через Q поле раціональних чисел. Введемо наступні позначення: $Rad_0 = Q,\ Rad_n = \{r : r = a + b * \sqrt{p_n}\ ,\ a,b \in Rad_{n-1}, n = 1,2,...\}$. Поле Rad є нескінченним об'єднанням зростаючої послідовності полів Rad_n .

$$Rad = \bigcup_{n=0}^{\infty} Rad_n,\ Rat = Rad_0 \subset Rad_1 \subset ... \subset Rad_n \subset ... \qquad (4)$$

Отже, конструктор Rad має вид

$$Rad\ r = (Rad\ a) + (Rad\ b) * \sqrt{Nat\ p}\ |(Rat\ q) \qquad (5)$$

Зауважимо, що послідовність розширень (4) є послідовністю скінчених алгебраїчних розширень полів коренями поліномів $x^2 - p_n = 0$.

У представлення (5) включені як опис елементу базової алгебри $Rat\ q$, так і опис механізму розширення - конструктор $(Rad\ a)+(Rad\ b)*\sqrt{Nat\ p}$. Така специфікація у точності відповідає означенню (4). З іншого боку, опис базового елементу є зайвим, якщо його можна отримати з співвідношення вкладення. Дійсно, специфікація векторного простору *LinComb* з включенням до неї опису елемента базової алгебри *LinMonom* має вигляд

$$LinComb\ w = (LinMonom\ u) + +(LinComb\ w)|(LinMonom\ u)$$

Однак, включення $LinMonom \subset LinComb$ визначено рівністю $u++0=u$. Тому окремий опис $LinMonom\ u$ є зайвим. Ми допускаємо обидва способи специфікацій.

Формули (4) безпосередньо узагальнюються на довільні динамічні розширення. Якщо алгебра B є динамічним розширенням алгебри A *з конструктором* $B=\varphi(A_1,...,B,...,A_n)$, зростаючу послідовність $B_0 \subset B_1 \subset ...B_n \subset ...$ визначимо таким чином:

1. $B_{(0)} = A,$ (6)

2. $B_{(n+1)} = \varphi(A_1,...,B_{(n)},...,A_n)$ (7)

Вкладення $\mathrm{Re}\, d : A \to B$ визначає вкладення $\mathrm{Re}\, d_i : B_{i+1} \to B_i$, звідки безпосередньо випливає представлення А у вигляді об'єднання зростаючої послідовності алгебр, кожна з яких є статичним розширенням попередньої.

$$B = \bigcup_{n=0}^{\infty} B_n,\ B_0 \subset B_1 \subset ... \subset B_n \subset ... \tag{8}$$

У практиці розроблення БАС виявилися і деякі узагальнення визначення (8). Так, замість послідовності алгебр $\{\boldsymbol{B_i}\}_{\boldsymbol{i}=0}^{\infty}$ розглянемо множину індексованих алгебр $\{A_i\}_{i\in I}$, де $\boldsymbol{I}$ - лінійно-упорядкована множина індексів. Алгебру $\boldsymbol{B_J}, \boldsymbol{J} \subset \boldsymbol{I}, |\boldsymbol{J}| < \infty$ визначимо як об'єднання алгебр $\boldsymbol{A_j}$, $\boldsymbol{j} \in \boldsymbol{J}$: $\boldsymbol{B_J} = \bigcup_{\boldsymbol{j}\in\boldsymbol{J}} \boldsymbol{A_j}$. Будемо вважати, що існує таке вкладення алгебр $\boldsymbol{B_J}$, при якому $\boldsymbol{B_{J_1}} \cup \boldsymbol{B_{J_2}} \subset \boldsymbol{B_{J_1 \cup J_2}}$. Тоді

$$B = \bigcup_{J\subset I, |J|<\infty} B_J \tag{9}$$

Прикладом такої алгебри є кільце $K[[x]]$, елементами якого є суми мономів з раціональними показниками степенів:

$$K[[x]] = \{P : P = \sum_{j\in J, J\subset Rat, |J|<\infty} a_j x^j \tag{10}$$

Динамічні розширення алгебр є послідовностями статичних розширень. Це дозволяє використовувати загальну схему реалізації динамічних розширень, виводячи відповідні системи переписуючи правил.

2.2 Синтез алгебраїчних програм

2.2.1 Приклад виводу алгебраїчної програми з специфікацій сорту

Приклад 7. Специфікації сорту Rat та вивід обчислень з раціональними числами. Специфікації сорту Rat визначають цей сорт як поле, лінійний порядок та статичне розширення Int.

```
Sort Rat:: Field, LinOrd;                    //Спадкування
Constructor
Rat r ={(Int a)//(Nat b);  // Конструктор сорту
  a//1 = a;                // Функція вкладення  RatToInt
  Num(r) = a, Den(r) = b;  // Функції доступу
  GCD(a, b) = 1            // Контекстна умова
  Form: Num(Form(r))∈ Int, Den(Form(r))∈ Nat,
  GCD(Num(Form(r), Den(Form(r)) = 1;
};
Operations
Add:  a//b + c//d = Form((a*d + b*c)//(b*d));
Sub:  a//b - c//d = Form((a*d - b*c)//(b*d));
Mult: a//b * c//d = Form((a*c)//(b*d));
Div:  a//b / c//d = Form((a*d)//(b*c));
Div:  a/b = Form(a//b),a/0=Exeption('Divison by zero');
Pow:  n >= 0 -> (a//b)^n  = (a^n//b^n),
n < 0 ->   (a//b)^n  = (b^-n//a^-n);
Predicates
Equ:  a//b == c//d = (a == c)&(b == d);
Gre:  a//b > c//d = (a*d > b*c);
Les:  a//b < c//d = (a*d < b*c);
UnLes: a//b >= c//d = (a//b > c//d)∨( a//b == c//d);
UnGre: a//b <= c//d = (a//b < c//d)∨( a//b == c//d);
```

Розглянемо вивід інтерпретатору операції `Add`. З специфікацій маємо:

```
a//b + c//d = Form((a*d + b*c)//(b*d));        (11)
a//1 = a;                                      (12)
```

Звідки отримуємо:

```
a + c//d = Form((a*d + 1*c)//(1*d));           (13)
```

Застосовуючи тотожність сорту *Int* `a*1 = 1*a = a` до (13), отримуємо:

```
a + c//d = Form((a*d + c)//d);                 (14)
```

Аналогічно, для другого операнду отримуємо:

```
a//b + c = Form((a + b*c)//b);
```

Виведені співвідношення є частковими випадками (12) – специфікації операції `Add` вкладенням RatToInt. Разом з загальним співвідношенням (11) вони визначають правила виконання додавання дробів:

```
Add:=rs{
 a//b + c//d = Form((a*d + b*c),(b*d)),
 a + c//d = Form((a*d + c),d),
 a//b + c = Form((a + b*c),b)
};
```

Наступні перетворення є оптимізуючими. Використовуючи контекстні умови неважко показати, що виклики `Form` у другому та третьому правилах можна видалити. Система переписувань інтерпретатору `Add` приймає вид

```
Add:=rs{
a//b + c//d = Form((a*d + b*c),(b*d)),
a + c//d = (a*d + c)//d,
a//b + c = (a + b*c)//b
};
```

Цілком аналогічно виводяться інтерпретатори операцій віднімання та множення на *Rat*. Виключенням є операція ділення, яка відсутня у сигнатурі сорту *Int*. Тому її треба визначити як багатосортну та специфікувати (див специфікації сорту *Rat*). Ще одним виключенням є операція піднесення до степеня, яка є похідною від множення. Для її інтерпретації визначення конструктору сорту не потрібно.

Звернемо тепер увагу на функцію *Form*. Визначення цієї функції зв'язує її з символом конструктору сорту. Роль цієї функції по суті полягає у канонізації елементу сорту. При виконанні правила загального виду ця функція викликається на результат операції. Тому функція *Form* є інтерпретатором символу конструктора сорту. Для символу // конструктора сорту *Rat* ми будемо користуватися позначенням !/. Загальне правило додавання прийме вигляд

```
a//b + c//d = a*d + b*c)!/(b*d).
```

Знак "!" завжди буде включатися до складу інфіксних позначень конструкторів сортів і завжди буде означати виклик інтерпретатора конструктора сорту. Таким чином, у системі правил

```
Add:=rs{
 a//b + c//d = (a*d + b*c)!/(b*d),
 a + c//d = (a*d + c)//d,
 a//b + c = (a + b*c)//b
};
```

Інтерпретатор конструктору сорту викликається тільки у першому правилі.

Метод виводу інтерпретатора операції сорту $\boldsymbol{v}$ з його специфікацій якщо алгебру A_v визначено як статичне розширення алгебри A_u можна узагальнити як алгоритм синтезу алгебраїчної програми.

Зауважимо, що цей метод можна реалізувати у вигляді алгебраїчної програми, оскільки він спирається тільки на екваціональний вивід. Для автоматизації елімінації функції *Form* треба використовувати більш складні методи та технології – системи доведення теорем над базовим сортом *u*.

2.2.2 Вивід інтерпретаторів у лінійних динамічних розширеннях

Приклад 8. Специфікації сорту Polynom та вивід обчислень з многочленами однієї змінної.

Специфікації сорту Polynom визначають цей сорт як евклідову область та лінійне динамічне розширення Monom.

```
Sort Polynom::EuclideDomain;
Parameter Field Coef, Const Variable Argument;
Constructor{
```

```
Polynom P = Monom M ++ Polynom Q // Конструктор сорту
0 ++ P = P,              //Функція вкладення  PolynomToPolynom
M ++ 0 = M;              // Функція вкладення  PolynomToMonom
LeadMon(P) = M,          // Функції доступу
LeadCoef(P)= Cf(M),
Arg(P)= Arg(M), Deg(P)= Deg(M);
Deg(M)> Deg(Q), Arg(M)= Arg(Q);     // Контекстна умова
Form: M∈Monom, Q∈Polynom,
 0 ++ P = P, M ++ 0 = M,
 Arg(M) = Arg(Q), Deg(M)> Deg(Q), Cf(M)<>0
};
Operations
Add: Deg(a)==Deg(b)→ (a++A)+(b++B)=(a+b) !+ (A+B); (17)
Sub: Deg(a)==Deg(b)→ (a++A)-(b++B)=(a-b) !+ (A-B);
Mult:        // Polynom * Polynom → Polynom; Commutative
(a++A)*(b++B)=(a*b)++((a*B+A*b)+A*B);
Mult:        // Coef * Polynom → Polynom; Commutative
 c*(b++B)= c*b ++ c*B;
 (b++B)*c = Form(c*b, c*B);
Div: (a++A)/b = a/b ++ A/b;
Pow: a^n=sqr(a^n div 2)*a^(n mod 2); // З сорту MiltSemiGroup
IntDiv:
 Deg(P)==Deg(Q) → P div Q = LeadCoef(P)/LeadCoef(Q),
 Deg(P)< Deg(Q) → P div Q = 0,
 Deg(P)> Deg(Q) → P div Q = LeadMon(P)div LeadMon(Q)++
 (P-(LeadMon(P)div LeadMon(Q))*Q div Q);
Mod: P mod Q = P - (P div Q)*Q; // З сорту EuclideDomain
```

На цьому прикладі ми покажемо, що методи виводу специфікацій, розглянуті вище, приводять до обґрунтованих з математичної точки зору систем інтерпретуючих правил.

Перш за все, звернемо увагу на два принципово різних метода визначення операцій. Операції *Add, Sub, Mult, Div* визначені у термінах конструкторів операндів. Такі визначення операцій ми будемо називати *конструктивними*. Це демонструє визначення операції *IntDiv* (ділення с остачею). Операцію *Mod* визначено у термінах операцій сигнатури сорту *Polynom*. Такі визначення операцій ми будемо називати *абстрактними* або *похідними*. Оскільки цю сигнатуру успадковано від абстрактного сорту *EuclideDomain*, специфікацію операції *Mod* наводиться саме у цьому сорті. У сорті *EuclideDomain* визначається і алгоритм Евкліда. Операцію *Pow* треба визначити ще раніше – у специфікаціях сорту *MiltSemiGroup*.

Конструктор сорту визначений рекурсивно. Отже, Алгебра *Polynom* є послідовністю вкладених алгебр, яка починається з алгебри *Monom* (мономи однієї змінної):

$$Mon = Pol_0 \subset Pol_1 \subset ... \subset Pol_k \subset ... \qquad (18)$$

Алгебра Pol_i є множиною поліномів степеня *i.* Тоді Pol_i є векторними просторами розмірності $i+1$. При такій інтерпретації степінь поліному визначає його індекс у послідовності. Тому операцію додавання *Add* (17) визначено трьома рівностями, перша з яких визначає правило додавання, якщо обидва операнди належать одній алгебрі, інші дві – різним:

$$a,b \in Pol_i; a \in Pol_i, b \in Pol_j, i < j; a \in Pol_i, b \in Pol_j, i > j$$

Таким чином, розширення (18) є розширенням векторних просторів. Правила інтерпретації векторних операцій виводяться з їх специфікацій цілком аналогічно вже розглянутому випадку статичного розширення алгебр:

$$\dim(Pol_i) = i+1,\ \dim(Pol_{i+1}) = i+2,\ Pol_i \subset Pol_{i+1},$$

$$V \in Pol_{i+1} \to V = a++A, A \in Pol_i; 0++A = A,\ a++0 = a\ .$$

`Deg(a)==Deg(b)`$\to$`(a++A)+(b++B)=(a+b)!+(A+B),` //основне правило

`Deg(A)< Deg(b)` $\to$ `A + (b++B)= b !+ (A+B),` // часткові випадки

`Deg(a)> Deg(B)` $\to$ `(a++A) + B = a !+ (A+B)`

Виведені правила ще не враховують другу з умов вкладення `M++0 = M`. Тому кожне з цих правил треба ще перетворити:

`Deg(a)== Deg(b)`$\to$ `a + (b++B)=(a+b) !+ B.`

`Deg(a)== Deg(b)`$\to$`(a++A) + b=(a+b) !+ A.`

`Deg(a)== Deg(b)`$\to$ `a + b = a + b.`

Останнє правило є тривіальним. Отже, для операції *Add* сорту *Polynom* отримуємо наступну систему правил інтерпретації:

`Deg(a)== Deg(b)`$\to$ `(a++A) + (b++B) = (a+b) !+ (A+B),`

`Deg(a)== Deg(b)`$\to$ `a + (b++B)=(a+b) !+ B,`

`Deg(a)== Deg(b)`$\to$ `(a++A) + b = (a+b) !+ A,`

`Deg(A)< Deg(b)` $\to$ `A + (b++B) = b !+ (A+B),`

`Deg(A)< Deg(b)` $\to$ `A + b = b ++ A,`

`Deg(a)> Deg(B)` $\to$ `(a++A) + B = a !+ (A+B),`

`Deg(a)> Deg(B)` $\to$ `a + B = a ++ B;`

Крім того, на сорті `Monom` визначаємо додатково часткову операцію `Add`:

`a$x + b$x = (a + b)$x.`

Ця система враховує обидві умови вкладення, тобто вкладення послідовності розширень $Monom = Pol_0 \subset Pol_1$, $Pol_i \subset Pol_{i+1}$.

Зробимо висновки. Специфікації сорту *Polynom* визначають динамічне розширення векторного простору. Похідні операції мають були виключені з специфікацій *Polynom* і віднесені до специфікацій відповідних абстрактних алгебр. Конструктивні операції сигнатури векторного простору визначаються основним випадком. Часткові випадки виводяться методами виводу статичних розширень.

Оскільки для сорту *Polynom* існує два співвідношення вкладення, вивід повної системи правил здійснюється послідовно: спочатку по першому співвідношенню, а потім – по другому.

Операції множення та неповного ділення специфікуються окремо як додаткові операції на векторному просторі *Polynom*.

```
Sort Polynom::EuclideDomain, VectorSpace;
Parameter Field Coef, Const Variable Argument;
Constructor
{Polynom P = Monom M ++ Polynom Q | Monom M
                    // Конструктор сорту
0 ++ P = P,         // Функція вкладення  PolynomToPolynom
M ++ 0 = M;         // Функція вкладення  PolynomToMonom
LeadMon(P) = M,  // Функції доступу
LeadCoef(P)= Cf(M),
Arg(P)= Arg(M),
Deg(P)= Deg(M);
Deg(M)> Deg(Q),  // Контекстна умова
Arg(M)= Arg(Q);
Form: M∈Monom, Q∈Polynom,
 Arg(M) = Arg(Q), Deg(M)> Deg(Q)
};
Operations
Add: Deg(a)==Deg(b)   → (a++A)+(b++B)= Form(a+b, A+B),
Sub: Deg(a)==Deg(b)   → (a++A)-(b++B)= Form(a-b, A-B),
Mult: S*(b++B)= Form(S*b, S*B);
(b++B)*S = Form(S*b, S*B);
Div: (a++A)/Scal = Form(a/S, A/S);
Mult: (a++A)*(b++B)=Form(a*b, (a*B+A*b)+ A*B);
IntDiv:Deg(P)==Deg(Q)→P div Q = LeadCoef(P)/LeadCoef(Q),
Deg(P)< Deg(Q)  → P div Q = 0,
Deg(P)> Deg(Q)  → P div Q = Form(LeadMon(P)div LeadMon(Q),
(P-(LeadMon(P)div LeadMon(Q))*Q div Q);
```

2.2.3 Приклад виводу алгебраїчної програми з специфікацій сорту. Бінарне динамічне розширення

Розглянемо приклад бінарного динамічного розширення алгебри *Bool* – алгебру логіки *BoolAlg*. Ця алгебра є також розширенням базового сорту *Variable,* оскільки елементи цього сорту – латинські букви інтерпретуються як логічні формули. Ми покажемо, що вивід інтерпретаторів логічних операції здійснюється тими ж методами.

Елементами сорту *BoolAlg* є формули логіки висловлювань багатьох змінних. Нехай $F(x_1, x_2, ..., x_n)$ - довільна формула формули логіки висловлювань від *n* змінних. Позначимо через O, I логічні значення відповідно *істина* та *хибність*. Тоді

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

Якщо позначити

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

отримаємо представлення

$$F(x_1, x_2, ..., x_n) = x_1 \,\&\, A(x_1, ..., x_{n-1}) \vee \neg x_1 \,\&\, B(x_1, ..., x_{n-1}). \quad (19)$$

Тепер здійснимо послідовно такі ж перетворення формул ***A***, ***B*** відносно змінних $x_{n-1}, ..., x_1$. В результаті отримаємо рекурсивне представлення формули логіки висловлювань. Насправді, через $BoolAlg_m$ позначимо множину формул логіки висловлювань від змінних $x_1, ..., x_m$. Тоді

$$BoolA\lg_n = \{F : F = x_n \,\&\, A \vee \neg x_n \,\&\, B, A, B \in BoolA\lg_{n-1}\}$$

Отже, сорт *BoolAlg* є об'єднанням зростаючої послідовності алгебр $BoolAlg_m$:

$$BoolAlg_0 = Bool, \; BoolAlg_0 \subset BoolAlg_1 \subset ... \subset BoolAlg_m \subset ... \quad (20)$$

$BoolAlg = \cup BoolAlg_m$

Зауважимо, що формула (19) задає канонічну форму формули алгебри висловлювань. Позначимо

$$BF(A, B, x) \overset{df}{=} x \,\&\, A \vee \neg x \,\&\, B. \quad (21)$$

Тоді

$$BF(A_1, B_1, x) \,\&\, BF(A_2, B_2, x) = BF(A_1 \,\&\, A_2, B_1 \,\&\, B_2, x), \quad (22)$$

$$BF(A_1, B_1, x) \vee BF(A_2, B_2, x) = BF(A_1 \vee A_2, B_1 \vee B_2, x), \quad (23)$$

$$\neg BF(A, B, x) = BF(\neg A, \neg B, x). \quad (24)$$

Таким чином, основні логічні операції виконуються поаргументно! Нарешті, легко перевірити, що функція вкладення визначається рівністю

$$BF(A, A, x) = A. \quad (25)$$

Цю канонічну форму будемо називати *рекурсивною нормальною формою* (РНФ). Наведемо далі специфікації сортів *Bool*, *BoolAlg*.

Приклад 9. Специфікації сорту *BoolAlg* та вивід обчислень логічних формул багатьох змінних.

Специфікації сорту *BoolAlg* визначають цей сорт як булеву алгебру – нащадок абстрактної алгебри *Bool* та бінарне динамічне розширення *Bool*.

```
Sort Bool; //Абстрактна алгебра висловлювань.
```

Axioms

```
~O = I,                    ~I = O,
A&O = O,                   A&I = A,
A|O = A,                   A|I = I,
A&~A  = O,                 A|~A = I,
(A&B)&C = A&(B&C),         (A|B)|C = A|(B|C),
A&B = B&A,                 A|B = B|A,
A&A = A,                   A|A= A,
A&(B|C) = A&B|A&C,         A|(B&C)= (A|B)&(A|C),
~(A&B) = ~A|~B,            ~(A|B) = ~A&~B, // правила де Моргана
                ~~A = A;
```

Operations

```
Con:   A&O = O, A&I = A;
Dis:   A|O = A, A|I = I;
Neg:   ~O = I, ~I = O;
```

```
Sort BoolAlg::Bool;
```

Constructor

```
{BoolAlg A=BF(BoolAlg A, BoolAlg B, Variable x)|Bool A;
```

BF(A, A, x) = A; *// Функція вкладення BoolAlgToBoolAlg*

Arg(BF(A, B, x)) = x,

Left(BF(A, B, x))= A, Right(BF(A, B, x))= B;

x > Arg(A), x > Arg(B); *// Контекстна умова*

Form: A, B ∈ BoolAlg, x > Arg(A), x > Arg(B)

};

Operations

```
Con: BF(A1,B1,x)& BF(A2,B2,x)= BF(A1&A2,B1&B2,x);
Dis: BF(A1,B1,x)|BF(A2,B2,x)= BF(A1|A2,B1|B2,x);
Neg: ~BF(A,B,x)= BF(~A,~B,x);
```

Для виводу системи правил інтерпретації, як і у попередньому прикладі, використовуємо умову вкладення, покладаючи `A1=B1` і позначивши `A1` через `A`. Система правил інтерпретації операції диз'юнкції має вид:

```
BF(A1,B1,x)& BF(A2,B2,x)= BF(A1&A2,B1&B2,x),
x > Arg(A)→ A&BF(A2,B2,x)= BF(A&A2,A&B2,x),
x > Arg(A)→ BF(A1,B1,x)&A = BF(A1&A,B1&A,x.
```

Можна показати, що у цьому прикладі функція *Form* не використовується.

Заключення

У даній роботі ми показали, що поняття конструктивного розширення БАС є ключовим для проектування та реалізації символьних обчислень. Більш за все це стосується символьних обчислень у математичних системах навчального призначення, де використовуються класичні алгебри та алгебраїчні системи.

Насправді конструктивний підхід, поряд з аксіоматичним підходом у алгебрі є загальновідомим. Ідея конструктивного визначення елементу алгебри, що будується, через елементи базових алгебр систематично використовується в алгебраїчних дослідженнях. З іншого боку, механізм перевантаження алгебраїчних операцій є стандартним засобом програмування математичних систем.

Отже, основним теоретичним результатом роботи є ідея систематичного застосування конструкції розширення у програмуванні сигнатур БАС як перевантажених сигнатур.

Практика використання цього підходу при розробленні математичних систем навчального призначення показала його ефективність і навіть універсальність. Це – основний практичний результат роботи.

ЛІТЕРАТУРА

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. 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.
5. Капитонова Ю.В., Летичевский А.А., Волков В.А., Дедуктивные средства системы алгебраического программирования, Кибернетика и системный анализ, 1, 2000, 17-35.
6. 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.
7. Львов М.С. Основные принципы построения педагогических программных средств поддержки практических занятий. Управляющие системы и машины.- 2006.-N6. с. 70-75.
8. 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.
9. Песчаненко В.С. Об одном подходе к проектированию алгебраических типов данных // Проблеми програмування. - 2006.- №2-3.-С. 626-634.
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. Ван дер Варден Б.Л. Алгебра. Изд. 2-ое.` М. ГРФМЛ 1979г. 624 с.
12. Goguen J., Meseguer J. Ordered-Sorted Algebra I: Partial and Overloaded Operations. Errors and Inheritance. SRI International, Computer Science Lab., 1987.
13. Гоген Дж. А., Мезегер Ж. Модели и равенство в логическом программировании. Сб. Математическая логика в программировании. М.: Мир, 1991.-с.274-310.

Надійшла у першій редакції 27.03.2009, в останній – 06.04.2009.