УДК 004.415.28

М.С. Львов

Херсонський державний університет, Херсон

МЕТОД МОРФІЗМІВ РЕАЛІЗАЦІЇ АЛГЕБРАЇЧНИХ ОБЧИСЛЕНЬ В МАТЕМАТИЧНИХ СИСТЕМАХ НАВЧАЛЬНОГО ПРИЗНАЧЕННЯ

Реалізація 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 (БАС). У даній роботі ми аналізуємо один з основних методів побудови специфікацій багатосортних алгебраїчних систем, оснований на понятті морфізмів (тобто ізоморфізмів та гомоморфізмів) БАС. Цей метод ми застосовуємо до побудови ієрархії БАС математичних систем навчального призначення. Численні приклади ілюструють основні теоретичні положення. Практика використання методу морфізмів при розробленні математичних систем навчального призначення показала його ефективність.

Ключові слова: *математичні системи, символьні перетворення, багатосортні алгебраїчні системи, ізоморфізми та гомоморфізми, інтерпретатори алгебраїчних операцій.*

Вступ

Розроблення алгоритмів виконання алгебраїчних обчислень є однією з основних задач, що виникають при реалізації математичних систем, основаних на символьних перетвореннях [8, 9]. Математичною моделлю цієї задачі є багатосортні алгебраїчні системи (надалі БАС). Практика розроблення навіть достатньо простих математичних систем навчального призначення [1 – 3] показала, що реалізація алгебраїчних обчислень потребує ретельного попереднього проектування БАС шляхом розроблення ієрархій сортів БАС та специфікацій інтерпретаторів багатосортних алгебраїчних операцій [8].

Для реалізації обчислень, основаних на символьних перетвореннях, ми використовуємо систему алгебраїчного програмування APS [4 – 6], адаптовану В. Песчаненко [10] для наших цілей. APS використовує технології алгебраїчного програмування, основані на використанні систем правил переписувань та стратегій переписувань на верхньому рівні системи (мовою APLAN) та можливості ефективної реалізації алгоритмів алгебраїчних обчислень на нижньому рівні (мовою C++). Метод морфізмів, який ми розглядаємо, по суті є методом перетворення типів даних. Застосування методу морфізмів у поєднанні з іншими методами програмування БАС [11 – 13] дозволяє з єдиних теоретичних позицій підходити до розв'язання задач проектування БАС. Побудова БАС як структури ізоморфних представлень алгебр дозволяє розробити таке ядро алгебраїчних обчислень математичної системи, яке дуже просто розширювати на нові предметні області.

Практика використання цього підходу при розробленні математичних систем навчального призначення довело його ефективність і навіть універсальність.

1. Багатосортні алгебри як математична модель алгебраїчних обчислень

1.1. Сигнатури багатосортної алгебри.

Означення 1.1 Нехай $\mathbf{U} = \{u_1,\ldots,u_k\}$ – 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_{Real} – множина дійсних чисел.

Означення 1.3 Багатосортною операцією f над сімейством $\mathbf{S}$ 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

© М.С. Львов

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

Mult : *Real* × *VectorSpace* → *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 → VectorSpace.

Означення 1.4. Визначимо сорт *Bool* з областю значень $S_{Bool} = \{True, False\}$. Багатосортним предикатом P називається відображення $P : S_{u_1} \times ... \times S_{u_m} \to S_{Bool}$, 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. Отже, нехай $\mathbf{A} =< \mathbf{S}, \mathbf{U}, \Sigma >$ багатосортна алгебра. Нехай $u, v \in \mathbf{U}$ символи сортів цієї алгебри. Будемо казати, що сорт *v* залежить від сорту u, якщо одна з операцій сигнатури Σ має тип $u_1 \times ... \times u \times ... \times u_m \to v$. Через $\mathbf{U}_v$ позначимо підмножину сортів, від яких залежать сорт *v*. Підмножину елементів Σ, що мають тип $u_1 \times ... \times u \times ... \times u_m \to v$ 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. Завдання полягає у тому, щоб побудувати програмну систему, яка підтримує спрощення цілих алгебраїчних та тригонометричних виразів. Таким чином, ядро цієї системи має підтримувати обчислення у кільці поліномів та кільці тригонометричних виразів багатьох змінних над полем раціональних чисел. Отже специфікаціям підлягають такі алгебри – обмеження на указані сорти:

1. *MultiPolynom* – кільце поліномів багатьох змінних.

Сигнатура операцій:

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

2. MultiTrig – кільце тригонометричних поліномів багатьох змінних.

Сигнатура операцій:

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

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

Сигнатура операцій:

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

4. 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*

5. RatBool – доповнення Bool предикатами рівності та порядку на раціональних числах.

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.

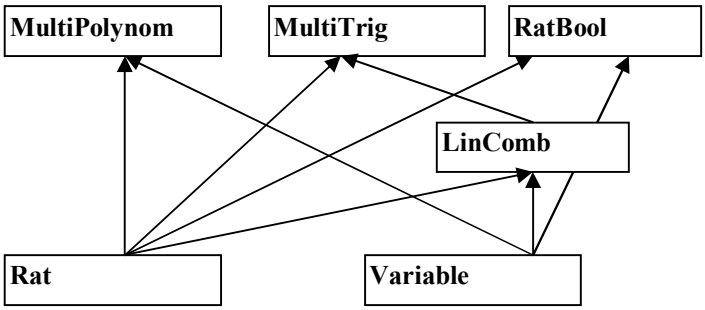

Рис. 1. Діаграма відношення залежності алгебр прикладу 1.

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

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 $\mathbf{A}_u$.

В математиці, зокрема, в алгебрі, аксіоматичний метод широко застосовується для визначення властивостей різноманітних предметних областей. За допомогою аксіом визначаються так звані абстрактні властивості – властивості, незалежні від природи множини, на якій діють алгебраїчні операції. Ми будемо користуватися алгебраїчною термінологією та відповідними системами аксіом з підручника [*Ван дер Варден. Алгебра*].

Для визначення конкретної алгебри разом з аксіомами часто використовують так звані постулати мінімальності. Наприклад, поле раціональних чисел можна визначити як мінімальне поле, яке містить множину натуральних чисел.

Конструктивний опис алгебри $\mathbf{A}_u$ – це означення конструктору сорту $\mathbf{S}_u$ (тобто визначення термів, множина яких утворює сорт $\mathbf{S}_u$) 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

Елементи цього поля – раціональні числа, які представляють у вигляді звичайних дробів. Конструктор сорту визначає стандартну форму представлення елементу цього сорту. Частіше за все це *канонічна форма*. Таким чином,

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

Роль конструктора сорту грає символ - горизонтальна риска. Цей же символ математики використовують для позначення операції ділення, зокрема, у *Rat.* Це явище у математиці є розповсюдженим. Таку ж роль грає, наприклад, символ кореня, символи додавання та множення ($a + b * \sqrt{2}$). І ці приклади можна продовжувати. Виявилося, однак, що для задачі специфікацій алгебраїчних обчислень це не зовсім зручно. Конструктори сортів, з нашої точки зору, мають бути позначеними окремо. Тому ми вводимо окремо поняття сигнатури операцій Σ та сигнатури конструкторів T. Зокрема, для конструктора сорту *Rat* ми використовуємо *подвійну нахильну риску*:

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

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 звичайно у вигляді суми мономів, упорядкованих у порядку спадання степенів.

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

У специфікаціях сорту *Polynom* це означення має бути рекурсивним, оскільки треба видалити з формули символ "три точки" та визначити окремо випадок, коли поліном містить лише один моном. Як ми побачимо, при такому підході окремо треба визначити ще поняття степеня полінома.

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

Очевидно, що традиційні математичні формули такого типу не пристосовані для специфікацій. Отже, для визначення носіїв сортів ми будемо користуватися спеціальною мовою специфікацій, яка допускає нерекурсивні та рекурсивні синтаксичні визначення елементів сортів, визначення функцій доступу (зокрема, для формулювання контекстних умов) та контекстні умови. Наведемо визначення сортів наших прикладів цією мовою:

```
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: $\mathbf{A}_v =< \mathbf{S}_v, \mathbf{U}_v, T_v, \Sigma_v, Ax_v, I_v >$.

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 ізоморфізмів та гомоморфізмів) та діаграми спадкування. Ці діаграми є графічними моделями специфікацій конструкторів алгебр та інтерпретаторів відповідних алгебраїчних операцій. Зараз ми розглянемо метод морфізмів. Метод розширень описано у [11, 12]. Метод спадкування описано у [13].

2. Метод морфізмів специфікацій алгебраїчних обчислень

Означення 2.1. Нехай $\mathbf{A}_u$ та $\mathbf{A}_v$ – багатосортні алгебри. Багатосортна алгебра $\mathbf{A}_v$ називається *гомоморфною* багатосортній алгебрі $\mathbf{A}_u$, якщо існує таке відображення $H : S_u \to S_v$, що для будь-якої операції f_1 типу $\varphi : (u_1, ..., u, ..., u, ..., u_m) \to u_{m+1}$ існує операція f_2 типу $\phi : (u_1, ..., v, ..., v, ..., u_m) \to u_{m+1}$ та виконується рівність

$$H(f_1(a_1, ..., a, ..., b, ..., a_m)) = f_2(a_1, ..., H(a), ..., H(b), ..., a_m),$$

де $(a_1, ..., a, H(a), b, H(b), ..., a_m) \in S_{u_1} \times ... \times S_u \times \times S_v \times S_u \times S_v \times ... \times S_{u_m}$.

Багатосортні алгебри $\mathbf{A}_u$ та $\mathbf{A}_v$ називаються ізоморфними, якщо існує взаємно однозначний гомоморфізм: $M : S_u \leftrightarrow S_v$.

Гомоморфізми та ізоморфізми багатосортних алгебр – зручний засіб для опису алгебраїчних обчислень в багатьох ситуаціях. З точки зору програмування – це функції перетворення типу.

Двоспрямована пунктирна стрілка на рис. 2 означає той факт, що між алгебрами A_{u_1} та A_{u_2} визначено ізоморфізм, а односпрямована пунктирна стрілка означає, що визначено гомоморфізм з A_{u_2} у A_{u_3}.

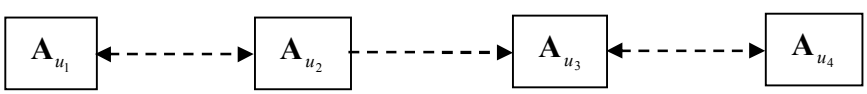

Рис. 2. Фрагмент діаграми морфізмів

Приклад 4. Кільце *Polynom* поліномів однієї змінної над полем *Rat*. Елементи цього поля визначені конструкторами, наведеними у прикладі 2. Конструктор сорту *Monom* визначає моном виду $a * x^k$. У такому вигляді, користуючись математичним редактором, здійснюють введення поліномів-даних і отримують поліноми-результати обчислень. Відомо, однак, що більш ефективним представленням поліномів однієї змінної у пам'яті є так зване розріджене представлення, у якому мономи є парами (коефіцієнт, степінь), а поліном – упорядкованою послідовністю мономів. Таким чином, ми маємо справу з двома ізоморфними представленнями по суті однієї алгебри і проблема полягає у побудові специфікацій, які адекватно відображають метод обчислень: треба спочатку здійснити перетворення введених даних у внутрішнє (ефективне) представлення, виконати обчислення, а потім здійснити обернене перетворення у зовнішнє представлення. Специфікації функцій перетворень H та H^{-1} задаються рівностями:

$$H(a * x^k) = ((a,k),x); \quad H^{-1}((a,k),x) = a * x^k;$$

$$H((m,x)++P) = ((m,\ H(P)),x),$$

де $M = (m,x), m = (a,k)$

Зауваження. У цих рівностях замість явного виду M можна користуватися функціями доступу. Наприклад, *m* та *x* можна замінити на функції доступу, які мають бути визначені у специфікаціях сорту *Monom:*

$$Cont(M) = (a, k),\ Arg(M) = x.$$

Тоді отримаємо

$$H(M) = (Cont(M), Arg(M));$$

$$H(M++P) = ((Cont(M),\ H(P)), Arg(M)).$$

Іншим прикладом використання ізоморфізмів є ізоморфізми, за допомогою яких визначаються три представлення раціональних чисел, якими оперує шкільний курс алгебри: звичайні дроби, змішані дроби та десяткові періодичні дроби.

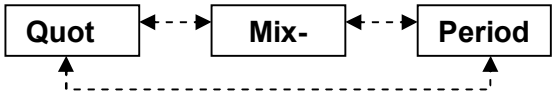

Рис. 3. Ізоморфні представлення поля раціональних чисел

Як уже було відзначено, метод морфізмів є одним з основних методів специфікацій БАС. У прикладі 4 також відзначено одна з важливих ситуацій, в якій для специфікацій багатосортної алгебри застосовуються ізоморфізми – специфікації конвертацій представлень елементів алгебр. Насправді необхідність у перетвореннях даних, які по суті є ізоморфізмами алгебр, виникає на етапі введення даних, існує протягом виконання обчислень та завершується етапом відображення даних.

1. На етапі введення даних або з клавіатури, або з інших джерел математичні формули мають бути представлені у стандартних формах (MathML, OpenMath тощо). Система алгебраїчного програмування (наприклад, APS) пропонує користувачу свій власний формат. Таким чином, має бути реалізованим перетворення MathML < == > APS, яке по суті є ізоморфним перетворенням математичних виразів. Вимога сумісності системи APS з іншими математичними пакетами (Maple, Mathematica) означає необхідність програмування конверторів MathML < == > OpenMath.

2. Як уже було відзначено, у зв'язку з вимогами ефективності алгоритмів алгебраїчних обчислень треба реалізувати перетворення представлень елементів багатосортної алгебри. Перелічимо основні канонічні форми лише поліномів однієї змінної:

- у вигляді суми мономів, упорядкованих за спаданням степенів – для введення - виведення;
- у рекурсивному вигляді (схема Горнера) – для ефективного алгоритма обчислення значення многочлену
- у вигляді списку "коефіцієнт-степінь" – для ефективного використання пам'яті та алгоритмів лінійних операцій додавання та множення на число;
- у вигляді вектора коефіцієнтів – для ефективних алгоритмів множення;
- у вигляді вектора коефіцієнтів «молодша половина – старша половина» - для алгоритму множення Карацуби;
- у вигляді вектора коефіцієнтів «мономи парних степенів – мономи непарних степенів" – для алгоритму швидкого перетворення Фур'є.

Зауважимо, що поряд з ізоморфними відображеннями алгебр, деякі важливі алгоритми потребують також гомоморфних відображень. Класичним прикладом є алгоритм Берлекемпа факторизації поліномів, першим етапом якого є гомоморфне відображення поліному з цілими коефіцієнтами у поліном з коефіцієнтами зі скінченого поля GF(p). Більш детально метод гомоморфізмів розглянуто в [15].

Оскільки ізоморфізми визначаються як відображення носіїв подібних алгебр, їх специфікація здійснюється за діаграмою (рис. 4).

Продовжимо розгляд прикладів.

Приклад 5. Обчислення у полі *Rad* квадратних радикалів.

Нашою метою є розв'язання задачі спрощення раціональних числових виразів, які містять квадратні корені з раціональних чисел. Наведемо приклад такого виразу:

$$A = \frac{(2\sqrt{20} + 3\sqrt{12})^2 + \sqrt{4/15}}{(5+\sqrt{6})\cdot(2-3\sqrt{5/12})}.$$

Результатом такого спрощення є цілий числовий вираз, який можна представити у вигляді лінійної комбінації коренів $\sqrt{2}, \sqrt{3}, \sqrt{5}$ та їх добутків з раціональними коефіцієнтами

$$A = r_0 + r_1\sqrt{2} + r_2\sqrt{3} + r_3\sqrt{5} + r_4\sqrt{6} + r_5\sqrt{10} + r_6\sqrt{15} + r_7\sqrt{30}$$

Отже, у цій задачі ми маємо справу з полем *Rad1*, елементи якого мають вид $B = r_0 + \sum_i r_i\sqrt{q_i}$, де r_i – раціональні числа, q_i - натуральні числа, вільні від квадратів, причому $q_1 < q_2 < ... < q_i < ...$.

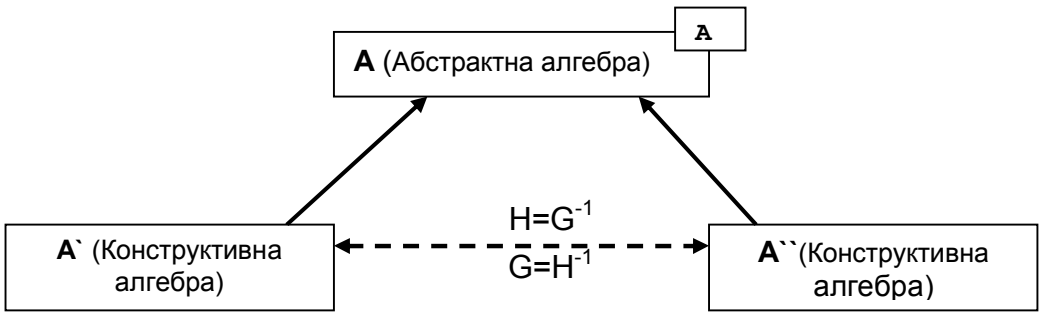

Рис. 4. Діаграма – принцип специфікацій ізоморфізмів алгебр

Діаграма специфікацій алгебр для нашого прикладу має вигляд:

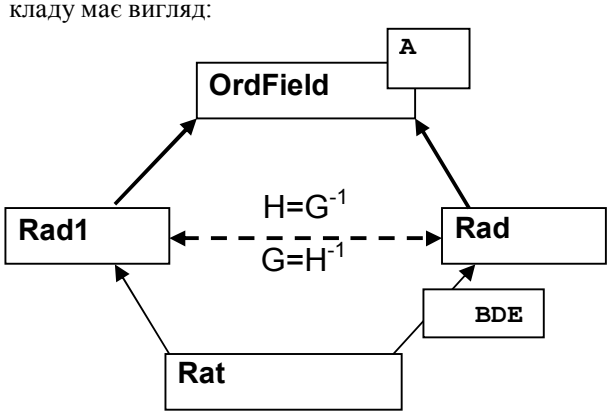

Рис. 5. Діаграма специфікацій алгебр прикладу 5

Оскільки *Rad1* є упорядкованим полем, ми маємо визначити абстрактне упорядковане поле *OrdField*, яке спадковує *Field, LinOrd* та містить аксіоми, які пов'язують операції поля та відношення порядку:

```
Sort OrdField::Field, LinOrd;
Axioms
a < b -> a+c < b+c,
(c > 0)&(a < b) -> a*c < b*c,
a<>0 -> a*a > 0;
…
```

Поле *Rad1* є розширенням *Rat*. Поле *Rad* є бінарним динамічним розширенням *Rat*. Один з можливих ефективних підходів до реалізації алгоритмів спрощення виразів у полі *Rad1* полягає у реалізації ізоморфізму $H : \mathrm{Rad1} \to \mathrm{Rad}$, обчисленні значення виразу A у полі *Rad,* і реалізації зворотного гомоморфізму $G : \mathrm{Rad} \to \mathrm{Rad1}$.

Оскільки ізоморфізм H залишає нерухомими раціональні числа, достатньо реалізувати його для коренів квадратних з раціональних чисел: $H\left(\sqrt{\frac{a}{b}}\right) = \frac{1}{b}H(\sqrt{ab}) = \frac{p}{b}\sqrt{q}$, де числа p, q є результатом застосування алгоритму звільнення від квадратів до підкорінного виразу ab.

Позначимо через $\mathrm{rad}(a, b, p)$ елемент поля Rad, а через SqrFree –функцію звільнення від квадратів натурального числа. *SqrFree: Nat –> Rad*. Функція SqrFree задовольняє умові

$\mathrm{SqrFree}(x) = \mathrm{rad}(0, y, p) \to x = y^2 p$, p не містить множників – повних квадратів. Таким чином,

$$H\left(\sqrt{\frac{a}{b}}\right) = \frac{1}{b} * \mathrm{SqrFree}(ab). \qquad (4)$$

У нашому прикладі

$H(\sqrt{20}) = \mathrm{rad}(0, 2, 5); H(\sqrt{12}) = \mathrm{rad}(0, 2, 3);$.

$$H\left(\sqrt{\frac{4}{15}}\right) = \mathrm{rad}\left(0, \frac{2}{15}, 15\right); H(\sqrt{6}) = \mathrm{rad}(0, 1, 6);$$

$$H\left(\sqrt{\frac{5}{12}}\right) = \mathrm{rad}\left(0, \frac{2}{12}, 15\right).$$

2-ий етап обчислень – обчислення у полі *Rad*. На третьому етапі треба застосувати зворотній ізоморфізм $G = H^{-1}$. Розглянемо для цього структуру поля *Rad*. *Rad* є бінарним динамічним розширенням *Rat*. Тому в традиційних математичних позначеннях $A = a + b\sqrt{p}$, де a, b взагалі кажучи, мають той же вид. Для упорядкування радикалів вводимо функцію доступу *Index:* $\mathrm{Index}(\mathrm{rad}(a, b, p) = p$ з контекстними умовами

$$\mathrm{Index}(A) > \max(\mathrm{Index}(a), \mathrm{Index}(b)),$$
$$\mathrm{GCD}(\mathrm{Index}(A), \mathrm{Index}(a)) = 1,$$
$$\mathrm{GCD}(\mathrm{Index}(A), \mathrm{Index}(b)) = 1.$$

Це означає, що

$$a = a_1 + b_1\sqrt{p_1}\ ;\ b = a_2 + b_2\sqrt{p_2}\ ;$$

$$p_1 < p;\ p_2 < p;\ \gcd(p, p_1) = 1;\ \gcd(p, p_2) = 1.. \qquad (5)$$

Отже, ізоморфізм *G* задається співвідношенням $G(\mathrm{rad}(a, b, p)) = a + b\sqrt{p}$, розкриттям скобок $b\sqrt{p} = a_2\sqrt{p} + b_2\sqrt{p_2}\sqrt{p}$, множенням радикалів $\sqrt{p_2}\sqrt{p} = \sqrt{p_2 p}$ та спрощеннями, які визначені для нуля та одиниці поля..

Отже, специфікації ізоморфного відображення H, якому (за традицією) надано ім'я *Rad1ToRad* мають вид:

```
Sort Rad1::OrdField;
Parameter Rat;
Signature
  RadToRad1(1): Rad -> Rad1;
```

```
    Sqrt(1): Rad -> Rad1;
    Operations
    RadToRad1:      rad(a,b,p)    =    a    +
b*Sqrt(p);
    Sqrt:
    (a<0)   ->   Sqrt(a))=   Exeption('Square
Root from negative number'),
    Sqrt(0) = 0,
    Sqrt(p)*Sqrt(q) = Sqrt(p*q);

    Add:      a+b                              =
RadToRad1(Rad1ToRad(a)+Rad1ToRad(b)),
    Mult:     a*b                              =
RadToRad1(Rad1ToRad(a)*Rad1ToRad(b)),
    Div:      a/b                              =
RadToRad1(Rad1ToRad(a)/Rad1ToRad(b)),
    …

    Sort Rad::OrgField;
    Parameter Rat;
    Signature
       Rad1ToRad(1): Rad1 -> Rad;
    SqrFree: Nat -> Rad;
    Operation
    Rad1ToRad:
    (a//b>0)    ->    Rad1ToRad(Sqrt(a//b))=
rad(0,1//b*SqrFree(a*b)),
       //частковий випадок натуральне число під знаком
радикалу
    (a>0)                                    ->
Rad1ToRad(Sqrt(a)=rad(0,SqrFree(a)),
    …
```

Наведемо специфікації операцій додавання, множення та ділення у полі *Rad.*

```
    Add:
    rad(a,b,p)+rad(c,d,p)=rad(a+c,b+d,p),
    (p>q)&(gcd(p,q)=1)->
rad(a,b,p)+rad(c,d,q)=rad(a+rad(c,d,q),b,
p),
    (p<q)&(gcd(p,q)=1)->
rad(a,b,p)+rad(c,d,q)=rad(rad(a,b,p)+c,d,
q),
    gcd(p,q)<> 1 ->
    rad(a,b,p)+rad(c,d,q)=    a+rad(0,b,(p
div gcd(p,q))*rad(0,1,gcd(p,q))+
       (c+rad(0,d,(q                    div
gcd(p,q))*rad(0,1,gcd(p,q));
    Mult:
    rad(a,b,p)*rad(c,d,p)=rad(a*c+b*d*p,a*
d+d*c,p),
    (p>q)&(gcd(p,q)=1)->

rad(a,b,p)*rad(c,d,q)=rad(a*rad(c,d,q),b*
rad(c,d,q),p),
    (p<q)&(gcd(p,q)=1)->

rad(a,b,p)*rad(c,d,q)=rad(rad(a,b,p)*c,ra
d(a,b,p)*d,q),
    gcd(p,q)<> 1 ->
      rad(a,b,p)*rad(c,d,q)=(a+rad(0,b,(p
div gcd(p,q))*rad(0,1,gcd(p,q))*
       (c+rad(0,d,(q                    div
gcd(p,q))*rad(0,1,gcd(p,q));
    Div:
    rad(a,b,p)/rad(c,d,p)=
    rad(a,b,p)*rad(c/(c^2-d^2*q), -d/(c^2-
d^2*q),q);
```

Надамо необхідні пояснення до правил операції додавання та множення. Контекстні умови конструктору *rad* (5) потребують взаємної простоти радикалів, які визначають індекси в бінарному алгебраїчному розширенні. Тому при додаванні або множенні чисел $a+b\sqrt{p}$, $c+d\sqrt{q}$, перш, ніж перевіряти нерівності $p<q, q<p$ слід забезпечити взаємну простору індексів: $\gcd(p,q)=1$. Позначимо

$$d=\gcd(p,q),\ p_1=p\,\mathrm{div}\,d,\ q_1=q\,\mathrm{div}\,d.$$

Тоді

$$a+b\sqrt{p}=a+b\sqrt{p_1}*\sqrt{d},\ c+d\sqrt{q}=c+d\sqrt{q_1}*\sqrt{d}\ .(6)$$

В цій формулі радикали задовольняють умові взаємної простоти. Тому, якщо $\gcd(p,q)\neq 1$, ми виконуємо попередньо перетворення (6), а потім – операцію додавання або множення.

Зауважимо, що поле *Rad* є динамічним розширенням *Rat*, тому специфікації операцій мають бути доповнені правилами, один з аргументів яких є раціональним числом - елементом *Rat*. Для того, щоб уникнути цих доповнень, можна представити раціональне число r у вигляді

$$r=\mathrm{Rad}(r,0,1)\ (r=r+0*\sqrt{1}).$$

Тоді, як легко перевірити, при виконанні операції у випадку, коли один з її аргументів – раціональне число, виконається одне з перших трьох правил, оскільки gcd(p,1)=1 і)&(gcd(1,q)=1.

Нарешті, специфікації *Rad* мають визначати відношення строгого порядку

$$\mathrm{rad}(a,b,p)<\mathrm{rad}(c,d,q),$$

яке є продовженням цього відношення на *Rat.*

```
(rad(a,b,p) < rad(c,d,p)) = (rad(a – c,b – d,p) < 0),
(p < q)–> (rad(a,b,p) < rad(c,d,q)) =
= rad(a – c + rad(0,b,p),–d,q) < 0,
(p > q)–> (rad(a,b,p) < rad(c,d,q)) =
= rad(a – c – rad(0,d,q), b,q) < 0,
rad(a,b,p) < 0 = (b < 0) & ((a <= 0)|
(a > 0) & (b ^ 2*p > a ^ 2))|
(b > 0) & (a < 0) & (b ^ 2*p < a ^ 2)).
```

Ці обчислення по суті зводять нерівності виду $A<B$ до нерівностей $A-B<0$ та відповідають структурі *Rad* як бінарного динамічного розширення *Rat*. Тому їх можна реалізувати за допомогою функції

```
Neg(1):Rad -> Bool, Neg(a) = a < 0 ,
(rad(a,b,p)< rad(c,d,p)) = Neg(rad(a – c,b – d,p)),
(p<q)–> (rad(a,b,p)<rad(c,d,q)) =
= Neg(rad((a – c + rad(0,b,p),–d,q)),
(p>q)–> (rad(a,b,p)<rad(c,d,q)) =
= Neg(rad((a – c – rad(0,d,q), b,q)),
Neg(a,b,p) = Neg(b) & (Neg(a)|(a = 0)
 |Neg(–a)) & Neg(a^2 – b^2*p))
 |Neg(–b) & Neg(a) & Neg(b^2*p – a^2)).
```

Висновки: парадигма алгебраїчних обчислень

Практика реалізації обчислень у багатосортній алгебрі програмної системи ТерМ, інших математичних системах навчального призначення показала, що підхід, застосований нами у прикладі для специфікацій *Rad,* є по суті універсальним. Цей підхід ми розглянули детально, починаючи з означень багатосортної алгебри і закінчуючи застосуванням методів розширень, спадкування [11 – 13] та морфізмів. Численні приклади всебічно демонструють його. Заключний приклад 5, ілюстрований діаграмою (рис 5), показує роль і місце методів розширень, спадкувань та морфізмів як основних методів алгебраїчних обчислень. Таким чином, цей підхід можна вважати *парадигмою алгебраїчних обчислень*.

Список літератури

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

2. Lvov M. AIST: Applied Computer Algebra System / M. Lvov // Proc. of ICCTE'93. – Kiev, 1993. – P. 25-26.

3. Львов М.С. Терм VII – шкільна система комп'ютерної алгебри / Львов М.С. // Комп'ютер у школі та сім'ї. – 2004. – № 7.– С. 27-30.

4. Algebraic programming system APS (user manual) Glushkov Institute of Cybernetics, National Acad. of Sciences of Ukraine / A. Letichevsky, J. Kapitonova, V. Volkov, A. Chugajenko, V. Chomenko. – K iev, 1998. – 320 p.

5. Капитонова Ю.В. Дедуктивные средства системы алгебраического программирования / Ю.В. Капитонова, А.А. Летичевский, В.А. Волков // Кибернетика и системный анализ. – 2000. – 1. – С. 17-35.

6. Tools for solving problems in the scope of algebraic programming / J. Kapitonova, A. Letichevsky, M. Lvov, V. Volkov // Lectures Notes in Computer Sciences. – 1995. – № 958. – P. 31-46.

7. Львов М.С. Основные принципы построения педагогических программных средств поддержки практических занятий / М.С. Львов // Управляющие системы и машины. – 2006. – N 6. – С. 70-75.

8. Песчаненко В.С. Розширення стандартних модулів системи алгебраїчного програмування APS для використання у системах навчального призначення / В.С. Песчаненко // Науковий часопис НПУ імені М.П. Драгоманова Серія №2. Комп'ютерно-орієнтовані системи навчання: зб. наук. пр.; редкол. – К.: НПУ ім. М.П. Драгоманова, 2005. – № 3 (10). – С. 206-215.

9. Песчаненко В.С. Об одном подходе к проектированию алгебраических типов данных / В.С. Песчаненко // 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 / В.С. Песчаненко // Управляющие системы и машины. – 2006. – № 4. – С. 86-94.

11. Львов М.С. Синтез інтерпретаторів алгебраїчних операцій в розширеннях багатосортних алгебр (підготовлено до друку).

12. Львов М.С. Верифікація інтерпретаторів алгебраїчних операцій в розширеннях багатосортних алгебр (підготовлено до друку).

13. Львов М.С. Метод спадкування при реалізації алгебраїчних обчислень в математичних системах навчального призначення (підготовлено до друку).

14. Ван дер Варден Б.Л. Алгебра / Б.Л. Ван дер Варден. – изд. 2-ое.`– М.: ГРФМЛ, 1979. – 624 с.

15. Лауэр М. Вычисления в гомоморфных образах. В кн. «Компьютерная алгебра. Символьные и алгебраические вычисления» / М. Лауэр; ред. Б. Бухбергер, Дж. Коллинз, Р. Лоос. – М.: Мир, 1986. – 392 с. – С. 178-211.

Надійшла до редколегії 16.09.2009

Рецензент: д-р пед. наук, канд. фіз.-мат. наук, проф. О.В. Співаковський, Херсонський державний університет, Херсон.

МЕТОД МОРФИЗМОВ РЕАЛИЗАЦИИ АЛГЕБРАИЧЕСКИХ ВЫЧИСЛЕНИЙ В МАТЕМАТИЧЕСКИХ СИСТЕМАХ УЧЕБНОГО НАЗНАЧЕНИЯ

М.С. Львов

Реализация 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). В данной работе мы анализируем один из основных методов построения спецификаций МАС, основанный на понятии морфизмов (т.е. изоморфизмом и гомоморфизмов). Этот метод мы применяем к построению иерархии МАС математических систем учебного назначения. Многочисленные примеры иллюстрируют основные теоретические положения. Практика использования этого подхода при разработке математических систем учебного назначения показала его эффективность.

Ключевые слова: *Математическая система, символьные преобразования, многосортная алгебраическая система, изоморфизм, гомоморфизм, интерпретаторы алгебраических операций.*

METHOD OF MORPHISMS AT REALIZATION OF ALGEBRAIC COMPUTATIONS IN MATHEMATICAL SYSTEMS OF EDUCATIONAL SETTING

M.S. Lvov

Development of algorithms of algebraic computations implementation is one of basic tasks, which arise up during realization of mathematical systems, based on symbolic transformations. The mathematical model of this task is multi-sorted algebraic systems (MAS). In this work we analyze one of the methods of specifications construction of algebraic systems, based on the concept of morphism (i.e. isomorphism and homomorphism) of MAS. We apply this method to the construction of hierarchy MAS of the mathematical systems of educational destination. Computation examples illustrate substantive theoretical positions. The practice of the use of method of morphism at development of the mathematical systems of educational destination shows its efficiency.

Keywords: *mathematical systems, symbolic transformations, multy-sorted algebraic systems, isomorphism and homomorphism, interpreters of algebraic operations.*