**УДК 004.4’22**

**Л. М. Захарія**
Національний університет “Львівська політехніка”,
кафедра інформаційних систем та мереж

# АЛГЕБРО-АЛГОРИТМІЧНІ ПІДХОДИ ДО ОПИСУ ПРЕДМЕТНИХ ОБЛАСТЕЙ ТА СИНТЕЗУ ПРОГРАМНИХ СЕРЕДОВИЩ ДЛЯ НИХ



**Наведено основні алгебро-алгоритмічні підходи (назвемо алгеброю алгоритміки - <АА>) та інструментальні засоби її підтримки. <АА> – перспективний напрям української алгебро-кібернетичної школи, започаткованої академіком В. М. Глушковим, пов'язаний із розробленням систем алгоритмічних алгебр (САА) і синтезатора програм за їх алгебраїчними описами. Першим таким синтезатором є синтезатор Мультипроцесист, який у світлі сучасних досліджень можна вважати однією з перших систем породжувального програмування. Розглянуто алгебраїчний аппарат представлення алгоритмів та структур даних, налаштування його для представлення знань різних предметних областей, засоби алгебраїчних перетворень алгоритмічних знань та автоматизованого встановлення алгоритмічного взаємозв’язку між спорідненими предметними областями. Клони різних алгоритмічних представлень алгоритмів та структур даних задають єдиний алгебраїчний підхід у представленні алгоритмів, даних, баз даних.**

**Ключові слова: породжувальне програмування, алгебра алгоритмів та програм, схеми алгоритмів, середовища автоматизованого алгебраїчного програмування, системи алгоритмічних алгебр, клони.**

**The article is devoted to basic algebraic-algorithmic approach (call algorithmics algebra - <AA>) and tools support. <AA> - Ukrainian promising direction algebraic-cyber school launched an academician V. Glushkov associated with the development of algorithmic algebras (SAA) and software synthesizer on their algebraic description. The first such Multi-procetyst synthesizer is a synthesizer, which in light of current research can be considered as one of the first systems of generative programming. Considered apparatus algebraic representation of algorithms and data structures, setting it to represent different domains of knowledge, means of algebraic manipulations algorithmic knowledge and establish automated algorithmic relationship between related subject areas. Clones various algorithmic representations of algorithms and data structures set single algebraic approach to the representation of algorithms, data, databases.**

**Key words: generating programming algorithms and algebra programs, schemes algorithms environment of algebraic programming system of algorithmic algebras clones.**

## Вступ

Алгебраїзація програмування – одна із сучасних тенденцій, що розвиваються в теорії і практиці інформатики. В Україні ця тенденція стала актуальною ще в 1965 р, отримавши своє втілення в системах алгоритмічних алгебр (САА), які запропонував спочатку для опису мікропрограм академік В. М. Глушков та розвинув для задач автоматизованого програмування Г. О. Цейтлін. Специфічною рисою САА є формалізація процесів проектування і синтезу (складання) алгоритмів і програм. Ці об'єкти проектуються в термінах регулярних схем – алгебраїчних виразів в САА. Розвиток апарату еквівалентних перетворень САА – схем забезпечив можливість поліпшення об'єктів проектування за обраними критеріями (наприклад, пам'ять,

швидкодія та ін. залежно від типу еквівалентних співвідношень) та надав засоби для виявлення алгоритмічних взаємозв'язків між спорідненими предметними областями.

Мета роботи – розвинути теорію клонів, розроблену для формалізованого проектування алгоритмів і програм, доповнивши її клоном n-відношень для формалізованого проектування баз даних та встановити взаємозв'язок його з клоном регулярних виразів Кліні, що дозволяє перенести результати клону Кліні за функціональною повнотою на клон n-відношень.

Виклад матеріалу статті підпорядкований такій структурі. Різним взаємопов'язаним формам алгебраїчних представлень алгоритмів та структур даних присвячений Розділ 2. У Розділі 3 описано інструментальні засоби підтримки процесу проектування програм за їх алгебраїчними описами в САА.Представлено інструментарій підтримки алгебраїчного проектування алгоритмів та даних, абстрактна САА машина як універсальний механізм трансляції та виконання САА схем. У Висновку сформульовано низку важливих завдань, що визначають перспективи розвитку алгебри алгоритміки (<АА>).

**Породжувальне програмування та алгебра алгоритміки**

Одним з важливих напрямків сучасної інформатики є розвиток алгебраїчних засобів представлення алгоритмів. Більшість робіт цього напряму не передбачає автоматизованого синтезу програм за їх аналітичними алгебраїчними описами і не використовується для еквівалентних перетворень таких описів. В статті презентується як теоретичний алгебраїчний апарат опису алгоритмів, так і інструментальні засоби підтримки таких описів для різних предметних областей. Системи алгоритмічних алгебр алгоритмів та структур даних надають можливість встановлення структурних взаємозв'язків між різними алгоритмами та програмами, формують засоби перетворення та синтезу програм за їх алгебраїчними описами, визначають апарат алгоритмічного опису предметних областей та встановлюють алгоритмічну спорідненість між різними предметними областями, формалізують алгоритмічні знання в цих предметних областях. Різні САА та взаємозв'язок між ними об'єднані терміном алгебраїчна алгоритміка – АА. АА об'єднує різні алгебри і алгоритми обробки даних, які використовуються при доведенні основних теорем у відповідних алгебрах. Цей підхід характеризується цілісним комплексом алгебраїчних, мовно - лінгвістичних та графічних засобів представлення алгоритмів та структур даних, за якими відбувається синтез програм у різних програмних середовищах.

Інший напрям сучасної інформатики – породжувальне програмування ІР [3]. Мета ІР полягає у створенні програмних застосувань для різних предметних областей та їх інтеграції. Для досягнення зазначеної мети використовують такі компоненти: абстракції, біологію та екологію програмування. Поняття абстракції – це аналітичне подання знань, що належить до обраної ПО (наприклад, формули у відповідних алгебрах), біологія відповідає за взаємозв'язок між предметними областями та "генетичні" спадковості між ними, а екологія – компонента, яка надає автоматизовані засоби підтримки абстракцій та перетворень між описами предметних областей. Запропонована в цій статті алгебра алгоритміки - <АА> [2] сприяє вирішенню цих проблем – вона формалізує алгоритмічні знання про предметні області алгебраїчними засобами. Біологічну компоненту в <АА> відображено в теорії клонів, а екологічну – у системі інструментальних засобів автоматизації проектування і синтезу програмного забезпечення цих ПО. На відміну від підходів інших авторів, в алгебрі алгоритміки розглядаються три взаємопов'язані форми подання знань:

– Аналітична – у вигляді алгебраїчної формули;

– Природно-лінгвістична – текст традиційною мовою людського спілкування;

– Візуальна – за допомогою графу (блок-схеми або граф-схеми).

Приклад 2.1. Наведемо аналітичну форму подання алгоритму сортування бульбашкою на розміченому масиві

$$\text{mass: } \mathbf{H}\ a_1 a_2 \ldots a_i\ \Delta\ a_{i+1} \ldots a_n\ K, \qquad (1)$$

де $a_i$ - елементи масиву $i = (\overline{1,n})$, Н – маркер початку масиву, К – маркер кінця масиву, $\Delta$ – вказівник, що переміщується по масиву і фіксує місця перестановки елементів при виконанні умови $l > r(\Delta)$ – елемент ліворуч $l$ від вказівника $\Delta$, більший від елемента $r$ праворуч від нього. Аналітичне представлення оператора перестановки елементів по вказівнику У матиме вигляд

$$\text{АЛЬТ}=([l > r(\Delta)]\ \text{T}(l,r) * \text{R}(\Delta), \text{R}(\Delta)). \quad (2)$$

Основним складовим оператором наведеної схеми є тримісна функція типу “if-then-else”, яка в аналітичній формі має таку структуру ([u] A, B), тут u – предикат, істинне значення якого викличе виконання оператора A, а хибне – оператора B. Операція * позначає послідовне виконання операторів $\text{T}(l,r)$, $\text{R}(\Delta)$, де $\text{T}(l,r)$ – оператор перестановки елементів $l$ та $r$; $\text{R}(\Delta)$ – оператор зсуву $\Delta$ по масиву на 1 позицію праворуч.

Загалом АЛЬТ – складений оператор, що являє собою конструкцію фрагмента знань, входить до складу відомого алгоритму бульбашкового сортування. Зазначимо, що бульбашковий алгоритм може бути за допомогою перетворень трансформований у достатньо ефективний паралельний адаптивний алгоритм сортування альтернативними вставками [1].

Наведемо адекватне текстове представлення наведеного фрагмента алгоритму сортування:

АЛЬТ ==== ЯКЩО ' $l > r$ за вказівником $\Delta$ в (mass)'

ТО "Переставити $l$ , r за вказівником $\Delta$ в (mass)" *

" Зсунути вказівник $\Delta$ по масиву (mass) на (1) елемент вправо "

ІНАКШЕ "Зсунути вказівник $\Delta$ по масиву (mass) на (1) елемент вправо "

КІНЕЦЬ ЯКЩО

Візуальна форма цього ж алгоритму може бути представлена відповідним фрагментом блок-схеми (рис. 2.1), при цьому інструментальні засоби алгебри алгоритміки підтримують графічне проектування алгоритмів і синтез по них програм у вибраних користувачем програмних середовищах.

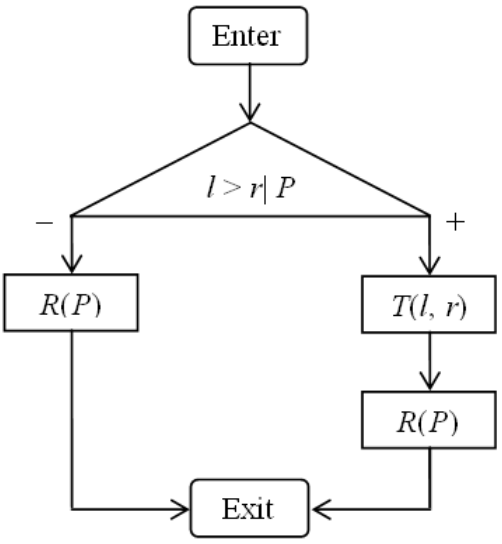


*Рис. 1. Граф-схема фрагмента сортування бульбашкою*

Отже, ця стаття презентує алгебру алгоритміки, в якій виділимо такі компоненти:

– теорія клонів, яка є формальним алгебраїчним апаратом, що забезпечує представлення алгоритмів у вигляді алгебраїчних виразів, надає апарат еквівалентних перетворень цих алгоритмів на основі системи еквівалентних перетворень;

– біологічна компонента, відповідальна за формування типових операцій предметної області у вигляді базових операторів та предикатів, характерних для алгоритмів цієї предметної області та стратегій обробки даних у цих предметних областях. Назвемо стратегією алгоритм, описаний засобами алгебр алгоритмів, який фіксує послідовність алгоритмічних операцій у стандартних модулях цієї предметної області. Алгоритмічні стратегії обробки переносяться на споріднені предметні області. Система еквівалентних перетворень у цій компоненті розширюється внаслідок представлення еквівалентних співвідношень, які визначаються вже властивостями самої предметної області, при цьому ці еквівалентні співвідношення подають у вигляді формул САА алгебр;

– екологічна компонента – інструментальні засоби алгебри алгоритміки, що підтримують процес проектування алгоритмів у вигляді виразів алгебри алгоритмів, автоматизований синтез

програм в потрібному користувачеві середовищі за стратегіями та реалізаціями базових операторів, предикатів цього алгоритму.

Коротко опишемо кожну компоненту алгебри алгоритміки.

**Поняття клону і формалізоване проектування алгоритмів**

Поняття клону належить до фундаментальних понять універсальної та загальної алгебри [4,5]. В алгебрі алгоритміки запропоновано підхід до використання клонів як основного інструменту опису різних засобів проектування алгоритмічних знань в предметних областях з урахуванням специфіки алгоритмічних операцій і структур даних у них.

Під клоном будемо розуміти одно- або багатоосновну алгебру:

$$C :: = <F / q; SUPER>, \quad (3)$$

де F / q – сукупність основ q = 1, 2, ..., n. Кожна основа складається з множини функцій певного типу (логічного, операторного, структур пам'яті і даних та ін.); SUPER – сигнатура клону, що представляє операцію суперпозиції функцій, що належать основам. В одну – головну функцію – замість змінних певних типів підставляються інші функції відповідного типу. Кожна основа клону складається з множини функцій одного і того самого типу.

Клон Поста (КП) – одноосновний клон; його основа – множина всіх булевих функцій (БФ). КП орієнтований на побудову сімейства алгебр БФ. Необхідність у побудові такого сімейства пов'язана з конструюванням комбінаційних схем (комп'ютерна апаратура), алгоритмів і програм в мовах програмування (МП).

Перераховані далі клони формалізують проектування алгоритмічної структури програмних продуктів.

Представницькою назвемо елементарну півгрупову алгебру А (з операцією *), за якою будується клон.

Наведемо досліджені пари: представницька алгебра і відповідний їй алгоритмічний клон [2]. Підкреслимо, що кожній парі відповідає певний метод і технологія програмування.

*Структурне програмування*: алгебра Дейкстри (АТ) – клон Дейкстри (КД);
САА Глушкова – клон Глушкова (КГ);
*неструктурне програмування*: алгебра Янова (АЯ) – клон Янова (КЯ);
*візуальне програмування:* алгебра граф-схем Калужніна (АК) – клон Калужніна (КК).

Прокоментуємо наведені пари. Кожна представницька алгебра є двоосновною: характеризується логічною і операторною компонентами. Логічна основа складається з булевих операцій. У клоні Глушкова логічна компонента доповнена операцією прогнозування, яка перевіряє логічну умову на очікуваному стані пам'яті після виконання використаного в операції прогнозування оператора. Введення операції прогнозування в клон Глушкова суттєво збільшило потужність засобів проектування.

Клони Дейкстри, Калужніна і Янова як логічні компоненти використовують клон Поста. Проблему функціональної повноти для перерахованих клонів вирішено в [2] з урахуванням наявності операції прогнозування в клоні Глушкова, що і пояснює алгоритмічну проектувальну потужність САА та відповідного клону Глушкова порівняно з іншими клонами.

Розглянемо алгебру Кліні АК = <RS; Сигни>, де RS – основа – множина всіх регулярних подій (РП) над алфавітом А; Сигни – сигнатура, що складається з операцій: множення x1 * x2; об'єднання x1 x2; ітерації.

Відповідний цій алгебрі одноосновний клон Кліні KL / A визначається як

KL / A :: = < АК; СУПЕР>,

де АК – основа, яку становить сукупність операцій алгебри Кліні; СУПЕР – сигнатура, що містить тільки суперпозицію функцій з можливим ототожненням і перейменуванням їх змінних.

Зазначимо, що поряд з одноосновними клонами побудовані їхні багатоосновні узагальнення, системи твірних яких містять контексно-вільні операції. Як наслідок клони формальних мов адекватні за своїми властивостями k-значним логікам (k > 2). Клон Кліні і дослідження проблеми його функціональної повноти важливий не тільки в контексті теорії формальних мов, але і в його зв'язку з клоном n-відношень.

Для клону n-відношень визначимо відповідну їй триосновну представницьку алгебру. Двома основами такої алгебри є відомі логічна і операторна компоненти систем алгоритмічних алгебр Глушкова. Як третю компоненту виберемо алгебру n-відношень. Визначимо основне поняття такої алгебри – поняття n-арного відношення.

Під операцією $M_1 \times M_2 \times ... \times M_n$ декартового добутку множин $M_1, M_2, ... M_n$ розумітимемо сукупність всіх упорядкованих n-ок ($a_1 a_2 ... a_n$), $a_1$ з $M_1$, $a_2$ з $M_2$, ..., $a_n$ з $M_n$. Зокрема, при n = 2 маємо операцію декартового добутку двох множин M x M' як сукупність всіх упорядкованих пар $(a, a')$, де $a$ з $M$, $a'$ з $M'$.

N-арне відношення визначається як довільна підмножина декартового добутку $M_1 \times M_2 \times ... \times M_n$. Отже, бінарне відношення – будь-яка сукупність упорядкованих пар виду $(a, a')$.

Визначення клону функціональних n-відношень пов'язане з операцією суперпозиції функцій, асоційованих із зазначеними відношеннями.

Наведемо більш строге визначення суперпозиції функціональних бінарних відношень. Нехай F і G – зазначені відношення, з якими пов'язані одномісні функції f і g, відповідно. Суперпозиція (або композиція) відношень F x G породжує нове відношення H, з яким пов'язана одномісна функція h(x) = f(g(x)),, що є зазначеною суперпозицією функцій f і g. Інакше кажучи, пара (a,s) ∈ H, якщо знайдеться допоміжний елемент b такий, що пара (a,b) ∈ F, а пара (b, s) ∈G. Отже, b – значення функції g на елементі s, тоді як a – значення функції f на результаті застосування функції g при значенні її аргумента s. Наведені визначення суперпозиції бінарних відношень і одномісних функцій можуть бути узагальнені на випадок n- відношень і (n-1) -арних функцій. Узагальнена згортка де Моргана (прямий кут) – суперпозиція n-арних відношень і (n-1)- арних функцій визначається так:

Нехай F, $G_1, G_2, ..., G_{n-1}$, – n-арні функціональні відношення, з кожним з них пов'язана (n-1) місна функція f, $g_1, g_2, ..., g_{n-1}$. До суперпозиції H = F ($G_1, G_2, ..., G_{n-1}$) належить n-ка ($a_1 a_2 ... a_n$), якщо знайдуться допоміжні елементи $b_1, b_2, ..., b_{n-1}$, такі що:

| | | | | | | |
|---|---|---|---|---|---|---|
| $(a_1$ | $a_2$ | ⋯ | $a_{n-1}$ | $b_1)$ | ∈ | $G_1$ |
| $(a_1$ | $a_2$ | ⋯ | $a_{n-1}$ | $b_2)$ | ∈ | $G_2$ |
| … | | . | … | … | | … |
| $(a_1$ | $a_2$ | ⋯ | $a_{n-1}$ | $b_{n-1})$ | ∈ | $G_{n-1}$ |
| $(b_1$ | $b_2$ | ⋯ | $b_{n-1}$ | $a_n)$ | ∈ | F |

Визначимо $i$-суперпозицію (хрест) – узагальнення згортки де Моргана.

Нехай $G_1, G_2, ..., G_{i-1}, G_i, G_{i+1}, ... G_n$ – функціональні відношення, з кожним з них пов'язана $n-1$-місна функція $g_i(x_{i1}, x_{i2}, ..., x_{in-1})$, де i = 1,2, ..., n.

Як узагальнення композиції бінарних відношень введемо поняття $S_i$-суперпозиції, яка породжує нове n-відношення W = $S_i$ ($G_1, G_2, ..., G_n$) таке, що n-ка ($a_1 a_2 ... a_n$) ∈ W, якщо існують $n-1$ допоміжних елементів: $b_1, b_2, ..., b_n$ для яких виконуються такі приналежності:

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| $(a_1$ | $a_2$ | $\cdots$ | $a_{i-1}$ | $b_1$ | $a_{i+1}$ | $\cdots$ | $a_n)$ | $\in$ | $G_1$ |
| $(a_1$ | $a_2$ | $\cdots$ | $a_{i-1}$ | $b_2$ | $a_{i+1}$ | $\cdots$ | $a_n)$ | $\in$ | $G_2$ |
| $\cdots$ | | $\cdots$ | $\cdots$ | $\cdots$ | $\cdots$ | | | $\in$ | $\cdots$ |
| $(a_1$ | $a_2$ | $\cdots$ | $a_{i-1}$ | $b_{i-1}$ | $a_{i+1}$ | $\cdots$ | $a_n)$ | $\in$ | $G_{i-1}$ |
| $b_1$ | $b_2$ | $\cdots$ | $b_{i-1}$ | ai | $b_{i+1}$ | $\ldots$ | $b_n)$ | $\in$ | $G_i$ |
| $(a_1$ | $a_2$ | $\cdots$ | $a_{i-1}$ | $b_{i+1}$ | $a_{i+1}$ | $\cdots$ | $a_n)$ | $\in$ | $G_{i+1}$ |
| $\cdots$ | $\cdots$ | $\cdots$ | $\cdots$ | | $\ldots.$ | | | $\in$ | $\cdots$ |
| $(a_1$ | $a_2$ | $\cdots$ | $a_{i-1}$ | $b_n$ | $a_{i+1}$ | $\cdots$ | $a_n)$ | $\in$ | $G_n$ |

при i = n $S_i$ – суперпозиція утворює згортку де Моргана.

Основа відмінності клонів – той чи інший тип суперпозиції. Побудова різних клонів n-відношень пов'язана з різними трактуваннями операції суперпозиції (і-суперпозиція, згортка де Моргана та її узагальнення).

Подібно до функціональних побудов, прийнятих у логічних клонах, поряд з тією чи іншою суперпозицією функцій зазвичай використовують запропоновані Мальцевим унарні операції перейменування, ототожнення і циклічної перестановки змінних, а також введення і виключення фіктивних змінних аналізованої функції. Аналогічні операції має сенс ввести і в клони n-відношень:

– перейменування, ототожнення і циклічну перестановку стовпців;

– введення і виключення фіктивних стовпців розглянутого відношень.

У [6] розглянуто й інші операції над n-відношеннями: проекція, фільтрація, з'єднання, ділення, теоретико-множинні перетин, об'єднання, різниця, декартовий добуток, які поширені при побудові реляційних баз даних. Залежно від того чи іншого виду представницької алгебри n-відношень, отримуємо різні клони n-відношень. Такі клони формалізують процес побудови як структур даних, так і алгоритмів їх обробки, алгоритми описуються операторною складовою клону. При цьому логічна компонента клону є сполучною ланкою в клоні, оскільки вона використовується як для побудови баз даних, так і для опису логічної структури алгоритму. Назвемо клоном Кодда клон, в якому як основу представницької алгебри використовують реляційну алгебру.

При введенні в клон Кодда операцій альтернативи, прогнозування ми отримуємо потужніші можливості для побудови структур даних і, відповідно, потужніший клон n-відношень. Цікавим видається введення як в логічну компоненту представницької алгебри, так і в алгебру n-відношень кванторів існування і загальності. Це не тільки розширює можливості побудов баз даних і обробних їх додатків, але й розширює можливості дослідження еквівалентних перетворень з метою оптимізації таких побудов.

Дослідження проблеми функціональної повноти клону n-відношень дозволило встановити факт ізоморфізму зазначеного клону та клону регулярних подій Кліні. Отже, це дало змогу використовувати таку техніку побудови решітки підалгебр для клону n-відношень.

*Теорема.* Клон n-відносин ізоморфний клону Кліні.

Теорія клонів n-відношень слугує математичним фундаментом в прикладному програмуванні з використанням реляційних баз даних і знань.

Побудова клону n-відношень і введення в сигнатуру операцій представницької алгебри кванторів існування і загальності дозволяє вирішити проблему побудови нормальних канонічних форм n-відношень. Перетворення виразів n-відношень до канонічної форми вирішує проблему еквівалентності в такому клоні і дає змогу стандартизувати вид запитів у базах даних, оптимізувати і перетворювати їх.

У клоні n-відношень актуальним стає питання взаємозв'язку логічної, алгоритмічної і табличної компонент. Для цього пропонується використовувати розмітку. По табличній компоненті клону n-відношень пересуваються вказівники, кожен з яких пов'язаний з циклічно застосовуваним

за місцем вказівника оператором обробки рядків або стовпців таблиці, тоді як сама алгоритмічна схема обробки таблиць бази даних задається САА-схемою вищого рівня.

Техніка вказівників дозволяє пов'язувати виконання оператора над даними, які відмічені вказівниками. Вона була дуже ефективна для розроблення експертної системи за алгоритмами сортування та пошуку. Для цих обох областей вдалося використати ті самі алгоритмічні стратегії, – заміні підлягали тільки базові оператори роботи з масивами при пошуку чи сортуванні. Такий підхід дозволив перенести на пошук як предметну область стратегії паралельного та послідовного сортування.

## Інструментальніе засоби підтримки теорії клонів – екологічна компонента алгебри алгоритміки

Представлений у цій статті математичний апарат дає змогу досліджувати алгоритмічну структуру знань предметних областей і тим самим визначати важливі для цієї предметної області алгоритмічні залежності, породжувати нові алгоритмічні знання, встановлювати взаємозв'язки між близькими предметними областями. Водночас він покладений в основу інструментальних засобів створення програмних систем. Апарат алгебри алгоритміки пропонує високорівневі засоби проектування алгоритмів і структур даних фахівцям предметних областей, які не володіють досконало технікою об'єктно-орієнтованого програмування. Ці засоби складають основу інтегрованого інструментарію проектування та синтезу алгоритмів і програм для занурення в об'єктно-орієнтовані середовища з підключенням розвинених в них інструментальних засобів, першим прототипом яких був синтезатор програм з САА-схем – мультипроцесист. Подальший розвиток алгебри алгоритміки в плані побудови клонів n-відношень і відповідних алгебр, орієнтованих на формалізацію і проектування структур даних для різних предметних областей служить потужним стимулом для подальшого розвитку досліджень в теоретичному, системному і прикладному програмуванні.

У [9] розроблено Інтегрований інструментарій проектування і синтезу алгоритмів і програм (ІПС). Цей інструментарій орієнтований на генерацію програм і може бути віднесений до прототипу ментального програмування. В основу цієї системи було покладено метод багаторівневого структурного проектування програм (БСПП) за їх описами в мові САА / 1. На відміну від системи Мультипроцесист, орієнтованої на синтаксичний аналіз САА-схеми, в ІПС схеми проектуються на основі методу діалогового конструювання синтаксично правильних програм (ДСП-методу), що забезпечує синтаксичну правильність схеми. Особливістю ІПС є також інтеграція всіх трьох форм представлення алгоритмів: аналітичного, природньо-лінгвістичного і графового при їх проектуванні.

До складу Інтегрованого інструментарію входять такі компоненти (рис. 2):

– ДСП-конструктор, призначений для діалогового проектування синтаксично правильних схем послідовних і паралельних алгоритмів і генерації програм, зокрема програм для об'єктно-орієнтованих середовищ (зокрема, C ++ і Java);

– ДСП -конструктор граф-схем алгоритмів;

– ДСП конструктор табличних структур даних (n-відношень);

– редактори САА-схем, граф-схем, таблиць;

– трансформатор, призначений для інтерактивної трансформації схем алгоритмів і програм з метою їх поліпшення за різними критеріями (використовувана пам'ять, час виконання та ін.);

– генератор САА-схем за схемами вищого рівня – гіперсхемами, що являють собою узагальнення граматик структурного проектування;

– середовища конструювання алгеброалгоритмічних описів, що містять

- опис конструкцій САА, операцій алгебри n-відношень в термінах цільової мови програмування, в середовищі якої проектуються програмні продукти
- базисні поняття та їх програмні реалізації;
- метаправила конструювання алгоритмів: згортка, розгортка, трансформація, переорієнтація;

- схеми алгоритмів, стратегії обробки (схеми, в яких базисні поняття замінені абстрактними змінними, стратегії обробки фіксують алгоритмічну структуру обробки) і гіперсхеми.

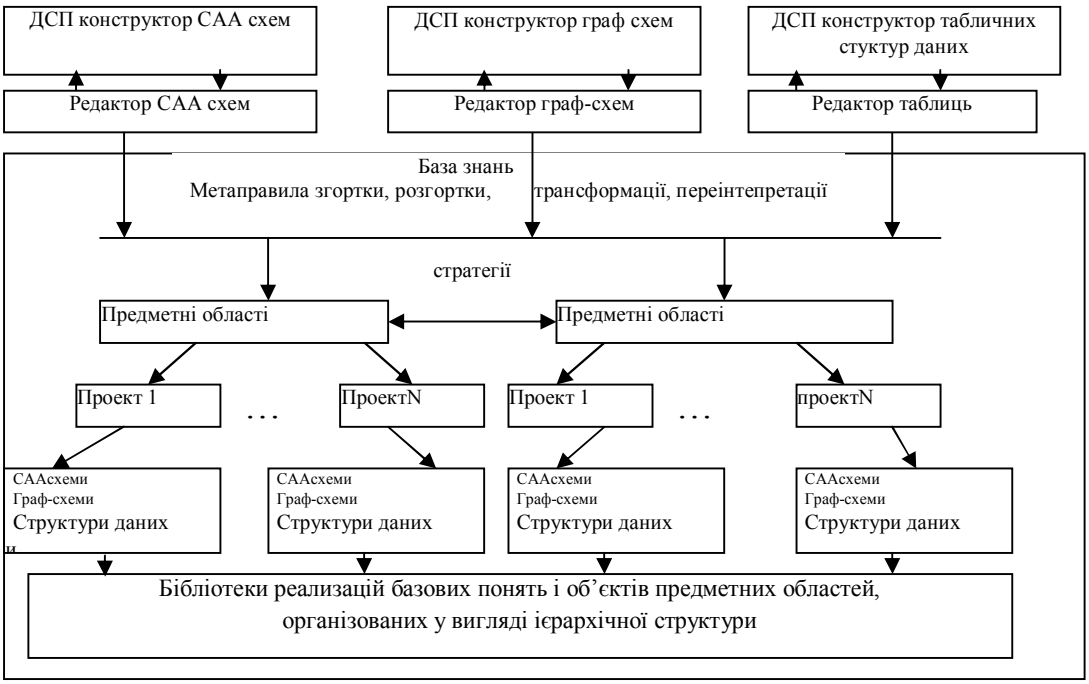


*Рис. 2. Архітектура інструментарію ІПС*

Основна ідея ДСП-конструктора полягає в порівневому конструюванні схем згори-донизу деталізацією мовних конструкцій САА. Процес проектування представлений у вигляді дерева конструювання алгоритму. На кожному кроці конструювання система в діалозі з користувачем надає на вибір лише ті конструкції, підстановка яких у формований текст не порушує синтаксичної правильності схеми. За отриманим в результаті конструювання деревом алгоритму можна синтезувати програму на основі програмних реалізацій з бібліотеки реалізацій.

ДСП – конструктор табличних структур даних – дозволяє візуальними засобами формувати таблиці баз даних і на їх основі конструювати нові як результат виконань операцій клону n-відношень над ними. ДСП конструктор граф-схем будує граф-схему алгоритму з базових конструкцій операцій клону Калужніна за тими самими принципами, що ДСП-конструктор САА_схем. Алгебраїчні засоби проектування алгоритмів і структур даних відповідають прийнятим в об'єктно-орієнтованих середовищах методам конструювання об'єктів у термінах шаблонів. При цьому порівневе присвоєння логічним і операторних змінним їх інтерпретацій зводиться до заповнення полів шаблонів у межах об'єктних середовищ. Тут слід зазначити можливість використання алгебраїчних перетворень схем на основі властивостей алгебраїчних операцій, що входять до формули, яка представляє проектований об'єкт. Поєднання поряд з алгебраїчною формою подання об'єктів, їх адекватних текстових і графових описів сприяє аналізу взаємодоповнюючих аспектів проектованих об'єктів і відповідних засобів автоматизованого синтезу. Так, граф-схемна форма проектування адекватна засобам візуального представлення знань в UML з використанням системи автоматизації об'єктно-орієнтованого програмування Rational Rose.

Для підтримки в базі знань проектів з різних предметних областей необхідно мати можливість встановлювати між усіма її компонентами ієрархічні зв'язки різної природи – зв'язки підпорядкованості, взаємозамінності, успадкування тощо Для підтримки такої ієрархічної структури в інструментарії використовується граматичний механізм синтаксичного аналізу LL (к) граматик - так звана САА-машина. САА машина не тільки відповідає за роботу ієрархічної бази

знань, але і є механізмом синтаксичного аналізу САА схем, створених без використання ДСП конструктора, механізмом синтезу програм по їх САА-схемах і інтепретації базових понять, тобто така САА-машина являє собою універсальний механізм роботи з ієрархічними структурами.

САА-машина реалізована як абстрактний автомат, який виконує певний набір команд. LL/к граматика, що описує синтаксис мови, на приналежність до якої аналізується вхідний символьний рядок, являє собою програму роботи автомата. Через близькість форм представлення формул САА-схем і LL/к граматик цей автомат здатний здійснювати інтерпретацію САА-схем. Тому опис команд автомата розглянемо з двох аспектів: реалізація алгоритму синтаксичного аналізу по LL / 1 граматиці та інтерпретації САА-схем. Нижче наведено приклади фрагментів програм автомата LL (к) граматик і внутрішнього подання формули САА-схеми, пропонуємо порівняти з відповідною формулою САА-схеми

| програма автомату інтерпретації LL/к грамматики | программа автомату інтерпретації САА-схем | відповідна формула САА-схеми |
|---|---|---|
| B1=L'схема' B2 A1 X ;<br>K'об'єкт' B1 A2 X;<br>E X ;<br>B2=&L'кінець'K'='XL A2 X;<br>E X ; | B1=!V1 V2 XL A1 X ;<br>E B2 X ;<br>B2=C V3 XL A2 X ; | B1= ЯКЩО V1+V2<br>ТО A1<br>ІНАКШЕ B2 ;<br>B2= ПОКИ НЕ V3<br>ЦИКЛ A2 |

САА – схема перетворюється в списокову внутрішню форму, і стає вхідним символьним рядком, прохід по якому викликає інтерпретаційне виконання такої САА-схеми САА-машиною.

Механізм САА-машини в новій версії інструментарію вдосконалений і використовується для обходу будь-яких ієрархічних структур.

**Висновки та перспективи подальших наукових розвідок**

Автор отримав такі результати: побудовано клони n-відношень для створення алгебраїчного апарату, скерованого на спільне проектування структур даних різної природи і алгоритмів їх обробки; накреслено підхід до поширення результатів з теорії півгрупових клонів (клони формальних мов і граматик, автоматні та алгоритмічні клони) на наведений у статті клон n-відношень; на основі отриманих результатів розширено інтегрований інструментарій для формалізованого проектування програмних застосувань із базами даних у межах алгебри алгоритміки з метою забезпечення різноаспектного проектування та синтезу програм за відомою формулою Вірта: алгоритми + структури даних = програми.

*1. Ющенко Е. Л. Многоуровневое структурное проектирование программ: Теоретические основы, инструментарий / Е. Л. Ющенко, Г. Е. Цейтлин, В. П. Грицай, Т. К. Терзян. – М.: Финансы и статистика, 1989. – 208 с. 2. Цейтлин Г. Е. Введение в алгоритмику / Г. Е. Цейтлин. – К.: Сфера, 1998. – 310 с. 3. Czarnecki K. Generative Programming: Methods, Tools and Applications./ Czarnecki, K., Eisenecker, U.- 1st edn. Addison-Wesley Professional (2000) 864 p. 4. Кон П. Универсальная алгебра / Кон П. – М.: Мир, 1968. – 348 с. 5. Курош А. Г. Лекции по общей алгебре / А. Г. Курош. – М.: Физматгиз. – 1962. – 396с. 6. Codd E. F. A Relational Model of Data for Large Shared Data Banks / Codd E. F. Comm. ACM. – 1970. – Vol. 13. – №6. – P. 377, 387. 7. Захарія Л. М. Мультипроцесист-автоматизована система багаторівневого проектування програм. 0.3 др. л. / В. П. Грицай, Л. М. Захарія., Тези доповідей 9-ї Всесоюзної наради з питань управління. – Єреван, 1983. 8. Захарія Л. М. Інструментарій синтезу програм для міні- і мікро ЕОМ / В. П. Захарія, Л. М. Захарія. – 1.5 др. // Кібернетика. – № 6. – 1989. 9. Яценко Е. А. Инструментальные средства конструирования синтаксически правильных параллельных алгоритмов и программ / Е. А. Яценко, А. С. Мохница // Пробл. программирования. Спец. выпуск по материалам 4-й Междунар. науч.-практ. конф. по программированию УкрПРОГ'2004. – 2004. – № 2–3. – С. 444–450.*