УДК 519.683.8

Т.Л. Захарченко, І.В. Редько
Національний технічний університет України "КПІ", Київ, Україна

ПРОБЛЕМА ПОВНОТИ В КЛАСІ ФУНКЦІЙ НАД ЗАПИСАМИ, ЯКІ ЗБЕРІГАЮТЬ ДЕНОТАТИ

Background. Modern range of IT problems forces researchers to not only consider solutions of programmers' problems, but processes of their solution. That is why researches of generally valid organization structures of the processes become the most important. Distinct place in the researches take problems related with building of algebraic characteristics of pragmatically-conditioned function classes, including completeness problems solutions in corresponding algebras.
Objective. Solution of completeness problem for class of computable manipulative functions on records. Term "manipulative" is specified as property of function to preserve denotations (atomic elements, which form records).
Methods. Primitive program algebra (PPA) has been chosen as instrument for the research of this class of computable functions. Solution of PPA completeness problem for manipulative functions on n-tuples formed solution basis of completeness problem and basis of generative functions system creation. Buildings, made in the paper, are based on algebraic methods of program analysis and methods of compositional programming.
Results. Solution of completeness problem for class of computable manipulative functions on records is described.
Conclusions. Results yielded may be utilized for further theoretical and applied researches of record manipulation methods, uncovering algebraic characteristics of manipulation functions classes on perspective carriers, for instance, lists, stacks, queues etc., programming languages semantics formalization, which use records as data type.
Keywords: completeness problem; denotations; primitive program algebras.

Вступ

З кожним роком рівень складності задач, які потребують розв'язання у сфері розробки програмного забезпечення, зростає прогресуючими темпами. При цьому використання отриманих розв'язків набуло в останні роки масового характеру. Це не могло не вплинути на посилення вимог як до якості цих розв'язків, так і до підвищення ефективності процесів їх продукування. Стає все більш очевидним те, що забезпечити необхідний рівень якості продукції та ефективності її виробництва без проведення серйозних процесів розробки програм буде надзвичайно складно, а в найближчій перспективі і неможливо. Саме такі дослідження можуть та повинні стати фундаментом якісного прориву у сфері, адекватній сучасним викликам технологізації процесів розробки програм. На жаль, поки доводиться констатувати, що у цьому напрямі мало що зроблено.

Звісно, вже перші ознаки депресивних явищ у програмуванні спонукали до пошуку причин їх виникнення. Було отримано велику кількість результатів різної глибини та значущості, висвітлено багато більш або менш вагомих причин, які призвели до "кризи жанру". Однак, як і варто було очікувати, всі найбільш принципові з них виявились не причинами, а всього лиш наслідками більш глибокої першопричини – спрощеного розуміння програмістської діяльності, основу якого становить абсолютизація інтуїтивного початку в програмуванні. Проявляється це в тому, що засоби розробки програмного забезпечення жорстко орієнтовані на програми як результати програмування і жодним чином не підтримують розгляд процесів їх створення на належному рівні строгості. Строгість тут підміняється суб'єктивною інтуїцією програміста.

В [1] обґрунтовувалась виключна значимість генетичних структур як структур процесів побудови програм для залучення процесів програмування в реальний розгляд. У композиційній парадигмі програмування такі структури уточнюються поняттям композиції [2–5]. Залучення їх до розгляду дає змогу строго поставити та вирішити низку важливих прагматико-семантичних проблем програмування. Дослідження проблеми отримання алгебричної характеристики прагматично обґрунтованого класу обчислюваних маніпуляційних функцій над записами є однією з багатьох задач, які стоять перед дослідниками у цій предметній галузі.

Сенс властивості маніпуляційності полягає в тому, що функція, яка ним володіє, може як завгодно глибоко перетворювати структуру вхідного даного та лиш в дуже обмеженій мірі змінювати складові цієї структури [6]. В праці [7] маніпуляційність функції уточнюється як властивість її зберігати денотати. Такої точки зору на маніпуляційність будемо дотримуватись і в

нашій роботі. Репрезентативним прикладом такого типу функцій виступають, наприклад, широко використовувані запити на отримання агрегованої інформації з бази даних.

Запис тут розуміється традиційно як скінченне функціональне бінарне відношення на множинах імен та значень або, слідуючи термінології композиційної парадигми програмування, – як іменна множина [8] (детальне означення поняття запису дано в наступному розділі). Значимість структури даних такого типу полягає в тому, що, як показано, наприклад, в [1], іменна множина із введеними на ній традиційними та спеціальними операціями є потужним універсальним інструментом формальної специфікації типів даних як завгодно складної природи. Наприклад, за допомогою таких записів легко представити дані типу таблиць, реляцій, списків, кортежів тощо.

У роботі [8] була розглянута загальна постановка проблеми повноти примітивної програмної алгебри (ППА) [9, 10] у класах багатомісних обчислюваних функцій та предикатів над зліченними носіями та був запропонований єдиний метод її вирішення. З його допомогою було досліджено клас багатомісних функцій та предикатів над репрезентативним для програмування носієм – записами. Однак, як це часто буває, поряд з вирішенням загальної проблеми не меншу значимість має можливість спеціалізації цього загального рішення на прагматично значимі підкласи досліджуваного класу функцій. Це й визначає актуальність нашого дослідження.

Всі невизначені у статті загальноматематичні поняття та позначення трактуються в сенсі [12, 13], а поняття теорії алгоритмів та теорії нумерацій – у сенсі [14].

Відзначимо, що ця робота є невід'ємною складовою частиною загального дослідження примітивних програмних алгебр, яке проводиться в рамках композиційної парадигми програмування та вчення про програми і побудову програм – програмології [11, 14–16], що тут розвивається.

Постановка задачі

Об'єкт дослідження – згаданий вище клас обчислюваних маніпуляційних функцій над записами.

Предмет дослідження – алгебрична характеристика зазначеного класу функцій.

Мета дослідження – відштовхуючись від результатів, отриманих у [9, 11], дослідити проблему повноти ППА для класу обчислюваних багатомісних функцій та предикатів над записами, що зберігають денотати.

Загальні положення, визначення і позначення

Тут будуть змістовно розглянуті та строго визначені поняття вектора і запису, а також введені окремі, важливі для подальших побудов, функції над ними. Далі, в рамках ППА, буде дана алгебрична характеристики класу частково-рекурсивних маніпуляційних функцій та частково-рекурсивних предикатів. Побудови будуть спиратись на отриману в [11] алгебричну характеристику класу обчислюваних маніпуляційних функцій та обчислюваних предикатів над векторами, а також на результати роботи [8].

Вектори, записи та функції над ними. Не обмежуючи загальність побудов, що слідує, наприклад, з результату по нумераційній обчислюваності [9] та результатів дослідження ППА арифметичних функцій [16], домовимось далі розуміти під векторами вектори натуральних чисел, а під записами – записи натуральних чисел, які позначатимуться відповідно N^* та $Z^{(N,N)}$ [8, 11]. Тоді $N^* = \bigcup_{i=0}^{\infty} N^i$, де $N^0 = \varnothing$ – так званий нульовий вектор, $N^1 = N$, а $N = \underbrace{N \times ... \times N}_{i}$, $i \in 2,3...$ Для позначення векторів будемо використовувати прописні літери $P, Q, R, ...$, а елементи векторів будемо відповідно позначати рядковими літерами $p, q, r...$ При цьому допускається використання індексів у позначеннях. Для позначення конкретного вектора будемо використовувати, залежно від ситуації, записи вигляду $\langle p_1, p_2, ..., p_n \rangle$ та $\overline{p_1 p_2 ... p_n}$, розуміючи їх еквівалентними, інакше кажучи $\langle p_1, p_2, ..., p_n \rangle \equiv \overline{p_1 p_2 ... p_n}$. Пустий вектор (вектор, що не містить елементів) будемо позначати Λ. Як оцінку векторного універсуму зафіксуємо функцію $\beta : N^* \to 2^N_{\text{fin}}$, де 2^N_{fin} – множина усіх скінченних підмножин множини N і таку, що: $\beta(\Lambda) = \Lambda$, $\beta(\langle p_1, p_2, ..., p_n \rangle) = \{p_1, p_2, ..., p_n\}$.

Відповідно, під *записом* над множинами імен та значень з N (далі просто записом) будемо розуміти скінченне функціональне бі-

нарне відношення на множині N. Для позначення записів домовимось користуватися прописними літерами $I, J, K, \ldots$. Відповідно, під записом над множинами імен та значень з N (далі просто записом) будемо розуміти скінченне функціональне бінарне відношення на множині. Рядковими літерами $u, v, w, \ldots$ будемо позначати імена елементів записів, літерами $a, b, c, d, \ldots$ – їх значення, а літерами $\lambda, \mu, \eta, \ldots$ – власне елементи записів – пари, які складаються з імені та значення. В усіх випадках при необхідності можна використовувати індекси. Оцінка β на записах задається аналогічно випадку з векторами: $\beta : Z^{(N,N)} \to \to 2^N_{\text{fin}}$, де 2^N_{fin} має такий же сенс, як згадувалось раніше, а $\beta(\{\varnothing\}) = \varnothing$, $\beta(\{(\nu_1, d_1), \ldots, (\nu_n, d_n)\}) = \{d_1, \ldots, d_n\}$, $\nu_i, d_j \in N$; $i, j \in \{1, \ldots, n\}$.

Відзначимо, що будь-який вектор можна промоделювати за допомогою деякого спеціального запису. Можливе і обернене представлення запису деяким спеціального вигляду вектором. Такі представлення будуть досить важливими у наступних побудовах. Тому їх варто розглянути окремо.

Нехай $\langle p_1, p_2, \ldots, p_n \rangle$ – довільний вектор. Тоді запис, що його представляє (моделює), може виглядати, наприклад, так: $\{(1, p_1), \ldots, (n, p_n)\}$. Інакше кажучи, вектор легко моделюється записом зі "стандартними" іменами від 1 до n. Ім'я тут не просто іменує, але і фіксує "стандартну" позицію кожного елемента вектора. Саме у зв'язку з цим таке іменування ми будемо називати *стандартним*, а імена, відповідно, "стандартними", інакше кажучи такими, що відповідають стандарту вектора.

Своєю чергою вектор, що моделює довільний запис $\{(\nu_1, d_1), \ldots, (\nu_n, d_n)\}$, буде виглядати як $\langle 0^{\nu_1+1} 1^{d_1+1} \ldots 0^{\nu_n+1} 1^{d_n+1} \rangle$, де p^k позначає k-мірний вектор $\underbrace{p \ldots p}_{k}$. Наприклад, запис $\{(4,3),(1,2)\}$ буде представлений вектором $\overline{\underbrace{00000}_{4+1}\underbrace{1111}_{3+1}\underbrace{00}_{1+1}\underbrace{111}_{2+1}}$, а $\{(1,2),(2,3)\}$ – вектором $\overline{001110001111}$.

Тепер введемо корисні функції на векторах та записах. Спочатку звернемось до векторів. Нехай $\sigma_{N^*} = \{\bar{p}\}_{p=1,2,\ldots} \cup \{\mathrm{H}, \mathrm{E}, \Pi, \circ, =_{N^*}, I^n_m\}^{n=1,2,\ldots}_{m=1,\ldots,n}$ – множина чр-функцій та чр-предикатів на векторах таких, що $\mathrm{H}(\bar{n}) = \underbrace{\overline{0 \ldots 0}}_{n+1} \equiv 0^{n+1}$, $H(\Lambda) = \Lambda$, $n \in N$, – функція відміток; функція Е(П) – вибір (видалення) першої компоненти вектор, $\circ$ – операція конкатенації двох векторів, а $=_{N^*}$ – предикат рівності двох векторів [11].

Звернемося до записів. Відзначимо, що оскільки множина імен записів є частково впорядкованою, то вважатимемо, що елементи будь-якого непустого запису впорядковані за іменами. Розглянемо таку сукупність $Z^{(N,N)}$-чр-функцій та $Z^{(N,N)}$-чр-предикатів на записах:

$$\sigma_{Z^{(N,N)},\beta} = \{=_Z, Od, Od^{-1}, Zn, \nabla, \div^Z, Mn, Rn,$$

$$Nm^+, \{(1,0)\}, I^n_m\}^{n=1,2,\ldots}_{m=1,\ldots,n}.$$

Функція стандартизації імен Rn перетворює будь-який запис на запис зі стандартними іменами відповідно до початкової впорядкованості елементів вихідного запису:

$$Rn(\{(\nu_1, d_1), \ldots, (\nu_n, d_n)\}) =$$

$$= \{(1, d_1), (2, d_2), \ldots, (n, d_n)\}\big|_{\nu_1 < \nu_2 < \ldots < \nu_n},$$

$$Rn(\{(\nu, d)\}) = \{(1, d)\}, \; Rn(\varnothing) = \varnothing.$$

Функція кодування денотату Od ставить у відповідність будь-якому непорожньому запису новий запис, який є "одиничним кодом" елемента запису з найменшим іменем. Порожній запис перетворюється сам на себе, іншими словами, $Od(\{(\nu_1, d_1), \ldots, (\nu_n, d_n)\})\big|_{\nu_1 < \nu_2 < \ldots < \nu_n} = \{(1,1), (2,1), \ldots, (d_1 + 1, 1)\}$, $Od(\varnothing) = \varnothing$. Пояснимо сказане на прикладах:

$Od(\{(2,4)\}) = \{(1,1),(2,1),(3,1),(4,1),(5,1)\}$ – модель вектора-коду $\langle 1,1,1,1,1 \rangle$;

$Od(\{(3,2),(4,6),(5,1)\}) = \{(1,1),(2,1),(3,1)\}$ – модель вектора-коду $\langle 1,1,1 \rangle$; $Od(\varnothing) = \varnothing$.

Функція декодування денотату Od^{-1}. Змістовно ця функція є оберненою до введеної вище функції Od. Формально:

$$Od^{-1}(L, \underbrace{\{(\nu_1, 1), (\nu_2, 1), \ldots, (\nu_k, 1)}_{\text{те, що декодується}}, (\nu_{k+1}, d_1), \ldots, (\nu_n, d_n)\}) =$$

$$= \{\underbrace{(\nu_1, k-1)}_{\substack{\text{результат} \\ \text{декодування}}}, (\nu_{k+1}, d_1), \ldots, (\nu_n, d_n)\},$$

де L – деякий запис, який містить денотат $k-1$, $\nu_1 < \nu_2 < ... < \nu_k < ... < \nu_n$, $d_1 \in N \setminus \{1\}$. Наприклад,

$$Od^{-1}(\{(1,2),(2,3)\}, \{\underbrace{(1,1),(2,1),(3,1)}_{\text{те, що декодується}},(4,0),(5,0),$$

$$(6,1)\}) = \{(1,2),(4,0),(5,0),(6,1)\};$$

$$Od^{-1}(\{(3,5)\}, \{\underbrace{(1,1),(2,1),(3,1),(4,1),(5,1),(6,1)}_{\text{те, що декодується}}\}) =$$

$$= \{(1,5)\}.$$

Функція кодування імені Zn. Змістовно ця функція аналогічна введеній вище функції Od, але кодується вже ім'я, а не денотат. Строго кажучи: $Zn(\{(\nu_1,d_1),\ldots,(\nu_n,d_n)\})|_{\nu_1<\nu_2<\ldots<\nu_n} = \{(1,0),$ $(2,0),\ldots,(\nu_1+1,0)\}$, $Zn(\varnothing)=\varnothing$. Наприклад: $Zn(\{(3,2),(4,6),(5,1)\}) = \{(1,0),(2,0),(3,0),(4,0)\}$.

Функція видалення по зразку $\div^Z$: $\div^Z(I,J) = pr_{pr_1(I)\setminus pr_1(J)}(I)$. Інакше кажучи, ця функція видаляє з першого аргументу всі елементи, імена яких містяться в другому аргументі – зразку. Для частинного випадку $pr_1(I) \cap pr_1(J) = \varnothing$, $\div^Z(I,J) = I$. Тут pr_i – функція проекції по i-й компоненті ($1 \le i \le m$) m-арного відношення [11], а $pr_{\{v_{i_1},\ldots,v_{i_k}\}}$ – допоміжна параметрична функція проекції запису $pr_{\{v_{i_1},\ldots,v_{i_k}\}}$, що будь-якому запису, наприклад, $I = \{(v_1,d_1),\ldots,(v_n,d_n)\}$ ставить у відповідність новий запис $pr_{\{v_{i_1},\ldots,v_{i_k}\}}(I) \equiv \{(v_{j_1},d_{j_1}),\ldots,(v_{j_p},d_{j_p})\}$, де $\{v_{j_1},\ldots,v_{j_p}\} \equiv$ $\equiv \{v_{i_1},\ldots,v_{i_k}\} \cap \{v_1,\ldots,v_n\}$ та $pr_{\{v_{i_1},\ldots,v_{i_k}\}}(I) \subseteq I$. Продемонструємо роботу $\div^Z$ на прикладах:

$$\div^Z(\{(1,3),(2,10),(5,7)\},\{(1,1),(2,5),(3,7)\}) = \{(5,7)\};$$

$$\div^Z(\{(5,7)\},\{(6,5),(3,7)\}) = \{(5,7)\};$$

$$\div^Z(\{(6,5),(3,7)\},\varnothing) = \{(6,5),(3,7)\};$$

$$\div^Z(I,I) = \varnothing.$$

Вибір елемента з мінімальним іменем Mn: $Mn(\{(\nu_1,d_1),\ldots,(\nu_n,d_n)\})|_{\nu_1<\nu_2<\ldots<\nu_n} = \{(\nu_1,d_1)\}$. У випадку з пустим записом, функція повертає пустий запис $\varnothing$: $Mn(\varnothing)=\varnothing$.

Функція слідування імен Nm^+. Збільшує імена елементів непустого запису на одиницю: $Nm^+(\{(\nu_1,d_1),\ldots,(\nu_n,d_n)\}) = \{(\nu_1+1,d_1),(\nu_2+1,d_2),$ $\ldots,(\nu_n+1,d_n)\}$. Для пустого запису: $Nm^+(\varnothing)=\varnothing$.

Функція накладання ∇. Для будь-яких $I,J \in Z^{(N,N)}$ $\nabla(I,J) = J \cup pr_{pr_1(I)\setminus pr_1(J)}(I)$. У часткових випадках: $\nabla(I,\varnothing)=I$, $\nabla(\varnothing,J)=\varnothing$, а у випадку, коли $pr_1(I)\cap pr_1(J)=\varnothing$, $\nabla(I,J) =$ $= \nabla(J,I) = J \cup I$. Наприклад, $\nabla(\{(1,3),(5,7)\},\{(1,1),$ $(2,5),(3,7)\}) = \{(1,1),(2,5),(3,7),(5,7)\}$.

Функція ініціалізації $\{(1,0)\}$ ставить у відповідність будь-якому аргументу запис $\{(1,0)\}$.

Предикат рівності $=_Z$ вводиться як звуження на множину $Z^{(N,N)}$ стандартного предиката рівності множин.

Повнота ППА у класі чр-функцій, що зберігають денотати, і чр-предикатів над записами

Повнота ППА у класі чр-функцій, що зберігають денотати, і чр-предикатів над множиною записів буде доведена з використанням результату про повноту ППА у класі чр-функцій, що зберігають денотати, і чр-предикатів над множиною векторів [11].

ППА чр-функцій, що зберігають денотати, і чр-предикатів над записами та векторами позначимо відповідно $A^{\text{чр}}_{Z^{(V,W)},\beta}$ та $A^{\text{чр}}_{N^*,\beta}$. Також згадаємо, що згідно з [11] $\sigma_{N^*,\beta} = \{\overline{p}\}_{p=1,2,\ldots} \cup \{\text{H},$ $\text{E},\Pi,\circ,=_{N^*},I^n_m\}^{n=1,2,\ldots}_{m=1,\ldots,n}$ – система утворюючих ППА $A^{\text{чр}}_{N^*,\beta}$. Нехай $\varphi: Z^{(N,N)} \to N^*$ та $\Phi: N^* \to Z^{(N,N)}$ такі ін'єкції:

$$\varphi(\{\varnothing\}) = (\varnothing),$$

$$\varphi(\{(\nu_1,d_1),\ldots,(\nu_n,d_n)\}) =$$

$$= \begin{cases} (d_1 d_2 ... d_n), \text{якщо } \nu_i = i, i = \overline{1,n}, \\ (0^{\nu_1+1}1^{d_1+1}.....0^{\nu_n+1}1^{d_n+1}) \text{ інкаше}, \end{cases}$$

$$\Phi((\varnothing)) = \{\varnothing\}, \Phi((p_1,\ldots,p_n)) = \{(1,p_1),\ldots,(n,p_n)\}.$$

Змістовно кажучи, ці функції використовують очевидне співвідношення вектора, наприклад $(d_1 d_2 ... d_n)$, та відповідного запису зі стандартними іменами, у цьому випадку запису $\{(1,d_1),(2,d_2),\ldots,(n,d_n)\}$, і навпаки. Таким чином, якщо аргументом φ є запис зі стандарт-

ними іменами, то він відображається у вектор, який моделює. Інакше будується вектор, що його кодує за вказаним вище правилом. Функція Ф будує модель вектора $< d_1, d_2, ..., d_n >$ у множині $Z^{(N,N)} - \{(1, d_1), (2, d_2), ...(n, d_n)\}$. Нижче ці функції грають роль відображень, одне з яких кодує, а інше – розкодовує.

***Означення* 1**. N^*-чр-функцію $f(P_1, P_2, ...P_n)$, $n \in N$, називають N^*-образом $Z^{(N,N)}$-чр-функції $F(I_1, I_2, ...I_n)$, якщо $f(\varphi(I_1), \varphi(I_2), ..., \varphi(I_n)) \cong$ $\cong \varphi(F(I_1, I_2, ..., I_n))$ для всіх $I \in Z^{(N,N)}$. Аналогічно для предикатів.

***Означення* 2.** $Z^{(N,N)}$-чр-функцію $F(I_1, I_2, ..., I_n)$, $n \in N$, назвемо $Z^{(N,N)}$-моделлю N^*-чр-функції $f(P_1, P_2, ..., P_n)$, якщо $F(\Phi(P_1), \Phi(P_2), ..., \Phi(P_n)) \cong \Phi(f(P_1, P_2, ..., P_n))$ для будь-яких $P_1, P_2, ..., P_n \in N^*$. $Z^{(N,N)}$-модель N^*-чр-предиката вводиться аналогічним чином.

Виходячи із визначень N^*-образу $Z^{(N,N)}$-чр-функції, $Z^{(N,N)}$-моделі N^*-чр-функції, повноти сукупності $\sigma_{N^*,\beta}$, можна встановити, що справедливі такі леми.

Лема 1. Для будь-якої $Z^{(N,N)}$-чр-функції існує її N^*-образ, який є N^*-чр-функцією. Те ж саме для предикатів.

Лема 2. Для будь-яких векторних чр-функцій та чр-предикатів існують їх $Z^{(N,N)}$-моделі, які належать до замикання $[\sigma_{Z^{(N,N)},\beta}]_\Omega$, де Ω – множина композицій ППА.

Доведення проводяться індукцією по довжині відповідних константних термів N^* і $Z^{(N,N)}$ ППА.

Позначимо $\Psi : Z^{(N,N)} \to \Phi(\varphi(Z^{(N,N)}))$ таку бієкцію:

$$\Psi(\{(\nu_1, d_1), (\nu_2, d_2)...(\nu_n, d_n)\}) =$$

$$= \begin{cases} \{(\nu_1, d_1), (\nu_2, d_2)...(\nu_n, d_n)\}, & \text{якщо } \nu_i = i, i = \overline{1,n}, \\ \{\underbrace{(1,0),...,(k,0)}_{\nu_1+1}, \underbrace{(k+1,1),...,(l,1)}_{d_1+1}, \underbrace{(l+1,0),...,(r,0)}_{\nu_2+1}, \\ \underbrace{(r+1,1),...,(s,1)}_{d_2+1}, (s+1,0),..., \\ \underbrace{(m,0),...,(p,0)}_{\nu_n+1}, \underbrace{(p+1,1),...,(o,1)}_{d_n+1}\} \text{ інакше,} \end{cases}$$

де $(k < l < r < s < m < p < o) \in N$. Відзначимо, що якщо аргументом є модель вектора – запис зі стандартними іменами, то значення функції дорівнює самому значенню аргументу. Таким чином функція Ψ зберігає денотати та має характеристику $\{0,1\}$.

Нехай χ – деяке розширення $Z^{(N,N)}$-чр-функції Ψ^{-1}. Додатковою складністю на шляху створення такої функції є збереження денотатів. Адже така функція може мати безкінечну характеристику – аргументом функції буде запис з денотатами із множини $\{0,1\}$, а результатом функції може бути запис із будь-якими денотатами. Для вирішення цієї ситуації вводиться додатковий аргумент, який несе у собі необхідні денотати. Отже, функція χ може бути визначеною таким чином:

$$K = \chi(I, \{(\nu_1, d_1), ..., (\nu_n, d_n)\}) =$$

$$= \begin{cases} \Psi^{-1}, \text{ якщо } \beta(K) \in \beta(I), \\ \{(\nu_1, d_1), ..., (\nu_n, d_n)\} \text{ інакше.} \end{cases}$$

Лема 3. Нехай $F(I_1, I_2, ..., I_n)$ – $Z^{(N,N)}$-чр-функція, яка зберігає денотати, і $\{d_1, d_2, ..., d_k\}$, $k \geq 0$, – її характеристика. Нехай $G(J_1, J_2, ..., J_n)$ – $Z^{(N,N)}$-модель векторного образу функції $F(I_1, I_2, ..., I_n)$. Тоді

$$F(K_1, K_2, ..., K_n) \cong$$

$$\cong \chi(L, G(\Psi(K_1), \Psi(K_2), ..., \Psi(K_n)))$$

для всіх $K_1, K_2, ..., K_n \in Z^{(N,N)}$. Аналогічно для предикатів.

Отже, для отримання результату по повноті ППА $A^{\text{чр}}_{Z^{(N,N)},\beta}$ залишається показати, що справедлива така лема.

Лема 4. $\Psi, \chi, \Pi_Z, \mathrm{E}_Z, \mathrm{H}_Z, \circ_Z \in [\sigma_{Z^{(N,N)},\beta}]_\Omega$, де $\Pi_Z, \mathrm{E}_Z, \mathrm{H}_Z \circ_Z$ – $Z^{(N,N)}$-образи N^*-чр-функцій $\Pi, \mathrm{E}, \mathrm{H}, \circ$.

Визначимо образи, які необхідно промоделювати. Функції Н та Е вже мають свої образи в $\sigma_{Z^{(N,N)},\beta}$, а саме Zn та Mn. Образ Π_Z можна задати як результат віднімання двох множин: $\Pi_Z(\lambda) = \lambda \setminus \mathrm{E}_Z(\lambda)$. Образ функції конкатенації можна визначити так:

$$\circ_Z(\{(\nu_1,d_1),...,(\nu_n,d_n)\},\{(u_1,c_1),...,$$
$$(u_m,c_m)\})|_{\nu_1<\nu_2<...<\nu_n\, u_1<u_2<...<u_m} = \{(\nu_1,d_1),...,(\nu_n,d_n)\}\cup$$
$$\cup\{(u_1+\nu_n,c_1),...,(u_m+\nu_n,c_m)\}.$$

Для доведення леми 4 необхідно промоделювати за допомогою функцій $\sigma_{Z^{(N,N)},\beta}$ $Z^{(N,N)}$-образи N^*-функцій з $\sigma_{N^*,\beta}$. Проведемо таке моделювання для кожної з указаних функцій.

Функція Π_Z: $\Pi_Z = S^2(Rn, S^3(\div, Mn, I_1^1))$.

Конкатенація $\circ_Z$. Для початку введемо декілька допоміжних функцій: отримання пустої множини $\varnothing_Z = S^3(\div, I_1^1, I_1^1)$, завжди хибний предикат $False = S^3(=_Z, \varnothing_Z, \{(1,0)\})$, завжди істинний предикат $True = S^3(=_z, \{(1,0)\}, \{(1,0)\})$, предикат нерівності $Neq = \Diamond(=_Z, False, True)$. Тоді сама функція конкатенації буде визначена як

$$\circ_Z = S^3(\nabla, I_1^2, S(*^3(S^3(Neq, \varnothing_Z, S^3(\div, I_2^2, I_1^2)),$$
$$Nm^+, I_2^2), I_2^2, I_2^2)).$$

Функцію Ψ можна представити композицією з предиката і двох функцій: $\Psi = \Diamond(StNames, I_1^1, notStNames)$, де *notStNames* – предикат, що перевіряє, чи аргумент є моделлю вектора, *notStNames* – функція, що будує модель вектора, який кодує запис-аргумент. У випадку, коли аргумент функції Ψ – модель вектора, він і є результатом функції, інакше результатом функції є значення *notStNames* на її аргументі.

Предикат *StNames* визначається як

$$StNames = S^5(*^5(S^3(=_Z, \varnothing_Z, S^2(\Pi_Z, I_2^4)),$$
$$\Diamond(S^3(=_Z, I_1^4, False)), False,$$
$$\Diamond(S^3(=_Z, S^2(E_Z, I_2^4), I_3^4), True,$$
$$S^3(Neq, S^2(\mathrm{E}_Z, I_2^4), I_4^4)),$$
$$S^2(\Pi_Z, I_2^4), Nm^+, Nm^+), True,$$
$$I_1^1, \{(1,0)\}, S^2(Od, \{(1,0)\})).$$

Для побудови функції *notStnames* знадобиться функція *convert*. Очевидно, що вона може бути представлена таким термом: $convert = S^3(\circ_Z, S^2(Zn, I_2^2), S^2(Od, I_2^2))$. Тоді:

$$notStnames = S^3(*^3(S^3(Neq, \varnothing_Z, I_2^2),$$
$$S^3(\circ_Z, I_1^2, convert), S^2(\Pi_Z, I_2^2)), \varnothing_Z, I_1^2).$$

Функція χ є композицією двох функцій та предиката: $\chi = \Diamond(isModel, restore, I_1^1)$. Предикат *isModel* перевіряє, чи аргумент є моделлю коду запису, якщо його значення істинне; по цій моделі будується вихідний запис, у випадку хиби повертається сам аргумент.

Предикат *isModel* істинний при одночасному виконанні таких умов: денотат першого елемента запису дорівнює нулю (*FirstZero*), денотат останнього елемента запису дорівнює одиниці (*LastOne*), запис має стандартні імена (*StNames*), у записі всі денотати мають значення 1 або 0 (*Denotates*10), імена елементів запису, закодовані вектором-кодом, який моделюється записом-аргументом, впорядковані за зростанням (*AscNames*). Неважко переконатись, що ці умови можуть бути представлені у вигляді таких $Z^{(N,N)}$-чр-предикатів:

$$FirstZero = S^3(=_Z, \{(1,0)\}, E_Z),$$
$$LastOne = S^3(*^3(S^3(=_Z, \varnothing_Z, S^2(\Pi_Z, I_2^2)),$$
$$S^3(=_Z, S^2(Od, \{(1,0)\}), S^2(Rn, I_2^2)), S^2(\Pi_Z, I_2^2)),$$
$$False, I_1^1).$$

Визначимо предикат *Denotates*10. Для початку задаймо два допоміжних предикати, які визначають приналежність денотата, що розглядається, до множини $\{0,1\}$. Це а) предикат рівності денотата одиниці $Eq1 = S^3(=_Z, S^2(Od, \{(1,0)\}), S^2(\mathrm{E}_Z, S^2(Rn, I_2^2)))$ та б) предикат рівності його нулю $Eq0 = S^3(=_Z, \{(1,0)\}, S^2(\mathrm{E}_Z, S^2(Rn, I_2^2)))$. Тоді власне предикат *Denotates*10 визначатиметься так:

$$Denotates10 = S^3(*^3(S^3(Neq, \varnothing_Z, S^2(\Pi_Z, I_2^2)),$$
$$\Diamond(S^3(=_Z, I_1^2, False), False,$$
$$S^3(=_Z, I_1^2, \Diamond(Eq1, True, Eq0))), S^2(\Pi_Z, I_2^2)), True, I_1^1).$$

Також визначимо предикат $AscNames$:

$$AscNames = \Diamond(S^3(=_Z, \varnothing_Z, I_1^1), False,$$
$$S^3(*^3(\Diamond(S^3(Neq, S^2(NextPair, I_2^2), \varnothing_Z),$$
$$S^3(GtName, S^2(NextPair, I_2^2), I_2^2), False),$$
$$S^3(GtName, S^2(NextPair, I_2^2), I_2^2),$$
$$S^2(NextPair, I_2^2)), True, I_1^1)).$$

Тут функція $NextPair$ видаляє "ділянку" моделі коду запису, яка є представленням елемента закодованого запису та розміщена "на початку" моделі коду, вона є композицією двох функцій: $NextPair = S^2(Rn, S^2(delO, delZ))$, де $delZ$ видаляє "код" імені елемента, що видаляється, $delO$ – "код" його денотату, при цьому виконується стандартизація імен у частині коду, яка залишилась, що важливо для наступних побудов. Формально вказані функції мають такий вигляд:

$$delO = *^2(S^3(=_Z, S^2(Od, \{(1,0)\}),$$
$$S^2(Rn, S^2(\mathrm{E}_Z, I_1^1))), S^2(\Pi_Z, I_1^1)),$$

$$delZ = *^2(S^3(=_Z, \{(1,0)\}, S^2(Rn, S^2(\mathrm{E}_Z, I_1^1))),$$
$$S^2(\Pi_Z, I_1^1)).$$

У визначенні $AscNames$ також існує раніше не заданий предикат $GtNames$, що дає змогу порівняти два найменших закодованих імені у записах-аргументах, які є моделями векторів, що кодують деякий вихідний запис. Цей предикат можна визначити так:

$$GtNames = S^4(*^4(\Diamond(S^3(=_Z,$$
$$S^2(Rn, S^2(\mathrm{E}, I_3^3)), \{(1,0)\}), S^3(=_Z, S^2(Rn, S^2(\mathrm{E}, I_2^3)),$$
$$\{(1,0)\}), False), S^3(Neq, S^2(Rn, S^2(\mathrm{E}, S^2(\Pi, I_3^3))),$$
$$\{(1,0)\}), S^2(\Pi, I_2^3), S^2(\Pi, I_3^3)), True, I_1^2, I_2^2).$$

Тепер сам предикат $isModel$ представляється, наприклад, такою композицією чотирьох загаданих вище предикатів:

$$isModel = \Diamond(\Diamond(\Diamond(\Diamond(LastOne, FirstZero, False),$$
$$Denotates10, False), StNames, False),$$
$$AscNames, False).$$

Визначимо функцію $restore$, яка по моделі коду запису елемент за елементом відновлює вихідний запис. Конкретно, $restore = S^3(*^3(S^3(=_Z,$ $\varnothing_Z, I_2^2), S^3(\nabla, I_1^2, ConvertPair), NextPair), \varnothing_Z, I_1^1)$, де $ConvertPair$ – функція, що відновлює по моделі коду запису перший елемент цього запису. Розглянемо цю функцію більш детально. Вона визначається через дві допоміжні функції. Одна, $convName$, перетворює послідовність нулів на ім'я елемента запису, інша, $convValue$ – послідовність одиниць на денотат того ж елемента. Таким чином $ConvertPair = S^3(convName,$ $S^2(convValue, I_2^2), I_2^2)$, а $convName = *^3(S^3(=_Z,$ $\{(1,0)\}, S^2(\mathrm{E}_Z, S^2(Rn, I_2^2))), S^2(Nm^+, I_1^2), S^2(\Pi_Z, I_2^2))$ і $convValue = S^2(\mathrm{E}_Z, S^2(Rn, S^3(Od^{-1}, L, S^2(delZ, I_1^1))))$.

Таким чином, лема 4 повністю доведена. Звідси безпосередньо слідує справедливість основного результату цього розділу.

Теорема 1. $\sigma_{Z^{(N,N)},\beta}$ – система утворюючих алгебри $A^{\text{чр}}_{Z^{(N,N)},\beta}$.

Висновки

У праці [8] викладено загальну методику віднаходження породжувальних сукупностей у примітивних програмних алгебрах, разом із цим необхідно усвідомлювати, що не менш важливим є і вирішення проблем повноти в класах функцій на різноманітних актуальних носіях. У цій статті запропоновано вирішення проблеми повноти для класу функцій, що зберігають денотати на записах. Записи – носій, що має широке застосування в інформаційних технологіях, зокрема як спосіб представлення інформації в базах даних, а запити на мові SQL будуються з використанням функцій, що зберігають денотати на записах. У статті представлений набір представників класу функцій, що зберігають денотати, і предикатів над записами $\sigma_{Z^{(N,N)},\beta} = \{=_Z, Od, Od^{-1}, Zn, \nabla, \div^Z, Mn, Rn, Nm^+,$ $\{(1,0)\}, I_m^n\}_{m=1,..,n}^{n=1,2,...}$, а також доведена повнота ППА на ньому. Доведення виконано з використанням отриманого раніше [11] набору функцій, що зберігають денотати, і предикатів над векторами. Побудова двох ін'єкцій у множину векторів та у множину записів дала можливість

отримати доведення повноти $\sigma_{Z^{(N,N)},\beta}$ з доведення повноти $\sigma_{N^*,\beta}$.

Отримані результати є фундаментом для розвитку напряму адаптивних середовищ програмування. Наступні кроки у цьому напрямі будуть пов'язані з дослідженням композицій як поняття та розробкою пов'язаних із ними редукційних методів дослідження функцій як середовищ прагматично обумовленої декомпозиції програмістських задач.

Список літератури

1. *Басараб И.А., Никитченко Н.С., Редько В.Н.* Композиционные базы данных. – К.: Либідь, 1992. – 192 с.
2. *Редько В.Н.* Основания композиционного программирования // Программирование. – 1979. – № 3. – С. 3–13.
3. *Редько В.Н., Редько И.В.* Экзистенциальные основания композиционной парадигмы // Кибернетика и системный анализ. – 2008. – № 2. – С. 3–12.
4. *Редько В.Н., Редько И.В.* Дескриптологическая среда информационных технологий // Проблемы программирования. – 2004. – № 2-3. – С. 65–73.
5. *Редько В.Н., Редько И.В., Гришко Н.В.* Дескриптивные системы: концептуальный базис // Проблемы программирования. – 2006. – № 2-3. – С. 75–80.
6. *Буй Д.Б., Редько И.В.* Примитивные программные алгебры функций, сохраняющих денотаты // Докл. АН УССР. – 1988. – № 9. – С. 66–68.
7. *Басараб И.А., Редько В.Н.* Базы данных с логико-функциональной точки зрения // Программирование. – 1984. – № 2. – С. 53–67.
8. *Примітивна* програмна алгебра обчислювальних функцій над записами / Т.Л. Захарченко, Д.І. Редько, І.В. Редько, П.О. Яганов // Наукові вісті НТУУ "КПІ". – 2015. – № 2. – С. 29–40.
9. *Мальцев А.И.* Конструктивные алгебры. 1 // Успехи мат. наук. – 1961. – **16**, № 3. – С. 3–60.
10. *Губский Б.В., Крапива Е.В., Редько И.В.* Вычислимые функции на реляциях и таблицах в счетном и конечном алфавитах // Докл. АН Украины. – 1988. – № 10. – С. 74–76.
11. *Буй Д.Б., Редько И.В* Проблемы полноты в классах вычислимых функций, сохраняющих денотаты // Программирование. – 1991. – № 4. – С. 56–68.
12. *Мальцев А.И.* Алгоритмы и рекурсивные функции. – М.: Наука, 1965. – 391 с.
13. *Мальцев А.И.* Алгоритмические системы. – М.: Наука, 1970. – 392 с.
14. *Редько І.В., Снігур Н.М.* Алгебраїчна характеристика класу графових перетворювачів, які зберігають денотати // Реєстрація, зберігання і обробка даних. – 2010. – **12**, № 4.– С. 54–61.
15. *Горєлов А.В., Редько І.В., Яганов П.О.* Композиційні засади програмістської діяльності // Тези XIII Міжнар. конф. "Приладобудування: стан і перспективи", 23–24 квітня 2014 р., Київ. – К., 2014. – С. 110.
16. *Буй Д.Б.* Примитивные программные алгебры: Дис. ... канд. ф.-м. наук: 01.01.09. – К., 1984. – 150 с.

References

1. I.A. Basarab *et al.*, *Compositional Databases*. Kyiv, Ukraine: Lybid, 1992, 192 p. (in Russian).
2. V.N. Redko, "Basics of compositional programming", *Programmirovanie*, no. 3, pp. 3–13, 1979 (in Russian).
3. V.N. Redko and I.V. Redko, "Existential basis of compositional paradigm", *Kibernetika i Systemnyi Analiz*, no. 2, pp. 3–12, 2008 (in Russian).
4. V.N. Redkon and I.V. Redko, "Descriptological environment of Information Technologies", *Problemy Programmirovania*, no. 2-3, pp. 65–73, 2004 (in Russian).
5. V.N. Redko *et al.*, "Descriptive systems: conceptual basis", *Problemy Programmirovania*, no. 2-3, pp. 75–80, 2006 (in Russian).
6. D.B. Bui and I.V. Redko, "Primitive program algebras of functions, which preserve denotations", *Doklady AN Ukrainy*, no. 9, pp. 66–68, 1988 (in Russian).
7. I.A. Basrab and V.N. Redko, "Data bases from logic-functional point of view", *Programmirovanie*, no. 2, pp. 53–67, 1984 (in Russian).
8. T.L. Zakharchenko *et al.*, "Primitive programming algebra: general approach to a problem of functional fullness", *Nukovi Visti NTUU KPI*, no. 2, pp. 29–40, 2015 (in Ukrainian).
9. A.I. Maltsev, "Constructive algebras. 1", *Uspehi Mat. Nauk*, no. 3, vol. 16, pp. 3–60, 1961 (in Russian).
10. B.V. Gubsky *et al.*, "Computable functions on relations and tables in countable and finite alphabets", *Doklady AN Ukrainy*, no. 10, pp. 74–76, 1988 (in Russian).

11. D.B. Bui and I.V. Redko, "Completeness problems in classes of computable functions preserving denotations", *Programmirovanie*, no. 4, pp. 56–67, 1991 (in Russian).
12. A.I. Maltsev, *Algorithms and Recursive Functions*. Moscow, USSR: Nauka, 1965, 391 p. (in Russian).
13. A.I. Maltsev, *Algorythmic Systems*. Moscow, USSR: Nauka, 1970, 392 p. (in Russian).
14. I.V. Redko and N.M. Snigur, "Algebraic characteristics of graph transformers class, which preserve denotations", *Registratsia, Khranenie i Obrabotka Dannykh*, vol. 12, no. 4, pp. 54–61, 2010 (in Russian).
15. A.V. Gorelov *et al.*, "Compositional basics of programmer's activity", in *Proc. 8th Int. Conf. "Instrumentation: State and Perspectives"*, Kyiv, Ukraine, 2014, p. 110 (in Ukrainian).
16. D.B. Bui, "Primitive programming algebras", Ph.D. dissertation, T. Shevchenko Kyiv National University, Kyiv, Ukraine, 1984 (in Russian).

Т.Л. Захарченко, І.В. Редько

ПРОБЛЕМА ПОВНОТИ В КЛАСІ ФУНКЦІЙ НАД ЗАПИСАМИ, ЯКІ ЗБЕРІГАЮТЬ ДЕНОТАТИ

Проблематика. Сучасна інформатико-технологічна проблематика така, що необхідний безпосередній розгляд навіть не стільки результатів розв'язків програмістських задач, скільки процесів їх розв'язання. Тому дослідження загальнозначимих структур організації цих процесів стають сьогодні надзвичайно актуальними. Особливе місце у цих дослідження посідає проблематика, пов'язана з побудовою алгебричних характеристик прагматично обумовлених класів функцій, у т.ч. з рішенням проблем повноти у відповідних алгебрах.

Мета дослідження. Вирішення проблеми повноти в класі обчислюваних маніпуляційних функцій над записами, які зберігають денотати. Тут маніпуляційність уточнюється за допомогою властивості функції зберігати денотати (атомарні елементи, що формують записи).

Методика реалізації. Як інструмент дослідження цього класу обчислюваних функцій у роботі вибрана примітивна програмна алгебра (ППА). В основу вирішення проблеми повноти цієї ППА та побудови системи її породжувальних покладено результат щодо повноти ППА над класом маніпуляційних функцій над кортежами. Проведені в роботі побудови базуються на алгебричних методах дослідження програм і методах композиційного програмування.

Результати дослідження. Описано рішення проблеми повноти в класі обчислюваних маніпуляційних функцій над актуальним носієм – записами.

Висновки. Отримані результати можуть бути застосовані для подальших теоретичних та прикладних досліджень методів маніпулювання записами, побудови алгебричних характеристик класів маніпуляційних функцій над перспективними носіями, наприклад списками, стеками, чергами тощо, для формалізації семантики мов програмування, які використовують записи як тип даних.

Ключові слова: проблема повноти; денотати; примітивні програмні алгебри.

Т.Л. Захарченко, И.В. Редько

ПРОБЛЕМА ПОЛНОТЫ В КЛАССЕ СОХРАНЯЮЩИХ ДЕНОТАТЫ ФУНКЦИЙ НАД ЗАПИСЯМИ

Проблематика. Современная информатико-технологическая проблематика такова, что необходимо непосредственное рассмотрение даже не столько результатов решений программистских задач, сколько процессов их решения. Поэтому исследования общезначимых структур организации этих процессов становятся очень актуальными. Особенное место в этих исследованиях занимает проблематика, связанная с построением алгебраических характеристик прагматически обусловленных классов функций, в т.ч. с решением проблем полноты в соответствующих алгебрах.

Цель исследования. Решение проблемы полноты в классе вычислимых манипуляционных сохраняющих денотаты функций над записями. Манипуляционность уточняется посредством свойства функции сохранять денотаты (атомарные элементы, формирующие записи).

Методика реализации. В качестве инструмента исследования данного класса вычислимых функций в работе выбрана примитивная программная алгебра (ППА). В основу решения проблемы полноты данной ППА и построения системы ее порождающих положен результат по полноте ППА над классом манипуляционных функций над кортежами. Проведенные в работе построения базируются на алгебраических методах исследования программ и методах композиционного программирования.

Результаты исследования. Описано решение проблемы полноты в классе вычислимых манипуляционных функций над актуальными носителем – записями.

Выводы. Полученные результаты могут быть применены для дальнейших теоретических и прикладных исследований методов манипулирования записями, построения алгебраических характеристик классов манипуляционных функций над перспективными носителями, например списками, стеками, очередями и т.п., для формализации семантики языков программирования, использующих записи как тип данных.

Ключевые слова: проблема полноты; денотаты; примитивные программные алгебры.

Рекомендована Радою
факультету електроніки
НТУУ "КПІ"

Надійшла до редакції
26 травня 2015 року