УДК 681.3.06

В.Ю. Вінник

УЗАГАЛЬНЕНА КОМПОЗИЦІЙНА МОДЕЛЬ СИМВОЛЬНОЇ ОБРОБКИ

У концептуально-методологічному середовищі композиційного та експлікативного програмування розглядається задача побудови системи взаємопов'язаних адекватних математичних моделей семантики програм символьної обробки, що відображають природну логіку процесів програмування та специфіку предметної області. Запропоновано загальне поняття композиційної системи символьної обробки як абстрактну основу такої експлікації та продемонстровано його розгортання до більш конкретних моделей.

Загальна характеристика задачі

Символьна обробка (СО) є однією з найважливіших галузей практичного програмування, оскільки значну частку потреб сучасного суспільства в автоматизації становлять задачі, що не зводяться до обчислювальної математики. Цим обумовлена поява багатьох мов та систем СО, таких, як Аналітик [1], Рефал [2], Снобол [3], та ін., а також побудова ряду теоретичних моделей СО, зокрема формальної семантики мов СО [4], теорії синтаксичного розбору та її відгалужень [5].

Разом з тим розроблені моделі мають своїм предметом деякі окремі, більш чи менш специфічні аспекти СО та не охоплюють галузі в цілому [6]. З іншого боку, не є універсальними і наявні мови та системи, оскільки кожна з них орієнтована на певний клас задач СО, можливо, й вельми широкий, але обмежений. Жодна конкретна, вузька модель СО не може бути адекватною даному складному та багатоаспектному предмету. Не вирішує задачу й сукупність розрізнених специфічних моделей — через складність встановлення зв'язків між ними. Крім того, предметом наявних теорій СО є переважно алгоритми обробки символьних даних, а не процеси та засоби конструювання таких алгоритмів, що не задовольняє потребу практичного програмування — діяльності по створенню програм — в адекватному теоретичному підґрунті.

Зазначені чинники свідчать, що теоретично та практично важливою задачею є побудова загальної теоретичної моделі СО як галузі програмування. Опишемо основні вимоги до такої моделі, її характеристичні ознаки та очікувані переваги.

Перш за все абстрактна модель СО має відображати сутність предмету, а не зовнішні, похідні властивості. Абстрагування від специфічних властивостей в той же час повинно бути не беззмістовним, а плідним, креативним, щоб абстрактна модель СО допускала природне розгортання до моделей конкретного рівня. Зокрема, абстрактна модель СО має своїми частковими випадками наявні специфічні моделі. Наслідком зображення специфічних моделей СО як похідних від спільної абстрактної бази є можливість встановлювати між ними зв'язки. З практичної точки зору це дозволило б узгоджено використовувати в спільному технологічному процесі розробки програм різні моделі СО. Іншими словами, одна з особливостей узагальненої моделі СО є її класифікаційна здатність. Більш важливою є прогностична та креативна здатність, яка полягає в тому, що модель СО, хоча й бідна конкретним змістом, підтримує скероване прагматичними потребами створення специфічних моделей СО, надаючи для цього понятєвий апарат та методологічне середовище для його розгортання.

Основні відомості з експлікативного програмування

Зазначені вимоги до моделі СО збігаються з основними засадами експлікативного програмування (ЕП) — загальної методології побудови адеква-

тних моделей програмування [7]. Згадаємо деякі положення ЕП.

Адекватна модель програмування в цілому чи будь-якої достатньо складної його галузі зокрема має вигляд не "монолітної" теорії, що повністю охоплює предмет, а системи взаємопов'язаних підмоделей тих чи інших його аспектів, побудованих на спільній базі, разом з метамоделлю коректного переходу між моделями.

Далі, адекватна модель повинна відображати сутність предмету і в цьому сенсі бути його роз'ясненням, експлікацією. Сутність програмування в ЕП уточнюється як логіка процесів побудови програм, конкретніше — як процедура покрокового застосування засобів побудови складніших програм з простіших до наявних програм як "будівельного матеріалу". Такі засоби логічної побудови, що отримали назву *композицій*, покладені в основу парадигми композиційного програмування (КП) [8]. Слід зазначити, що саме в роботах з КП було сформульовано задачу побудови адекватної моделі програмування СО та намічено шляхи її вирішення [9].

Програмування настільки складне та багатоаспектне, що жодна фінітна сукупність композицій не може бути достатньою наперед [7]. Тим не менше одна з основних тез ЕП полягає в тому, що експлікацію "чистого" програмування складає доволі обмежена сукупність композицій [там же]. Цим прикладне програмування розділене на дві складові: з одного боку, загальнозначуща, сутнісно-визначальна для програмування в цілому, яку можливо описати наперед, з іншого — специфічна, що визначає сутність даного конкретного різновиду програмування та потребує окремого опису для кожного часткового випадку.

Якщо поняття даних екстенсіонально зобразити деякою множиною об'єктів $\mathfrak{D}$, то семантика програм як перетворювачів даних відображається функціями типу $\mathfrak{D}\to\mathfrak{D}$. Тоді композиції можуть екстенсіонально розглядатися як операції алгебри таких функцій.

Отже, експлікація програмування в цілому та прикладних галузей програмування зокрема спирається на математичні моделі трьох взаємопов'язаних категорій: даних, функцій та композицій, і є підстави вважати моделлю програмування *композиційну систему* — трійку $\mathfrak{S}=\langle\mathfrak{D},\mathfrak{F},\mathfrak{K}\rangle$, де $\mathfrak{D}$ — клас даних; $\mathfrak{F}$ — множина базових функцій над даними; $\mathfrak{K}$ — множина композицій. Відносно базових функцій вважається, що вони є найелементарнішими здійсненними перетвореннями даних. Клас всіх програм, що можливо виразити (побудувати) в даному програмуванні, є замикання множини базових функцій $\mathfrak{F}$ відносно множини композицій $\mathfrak{K}$.

Поліморфізм поняття програмування підтримується даним типом моделі як можливість різноманітних конкретизацій по кожній складовій, уточнення властивостей класів $\mathfrak{D}$, $\mathfrak{F}$ та $\mathfrak{K}$. При цьому опорою для введення спеціальних класів функцій та відповідних композицій є спеціальні властивості даних.

Основа експлікації символьної обробки

В середовищі ЕП поставлена задача адекватного моделювання СО формулюється як експлікація прикладної логіки програмування СО, причому формою втілення такої експлікації є композиційна система. В [10−16] описано композиційні системи, що відповідають найважливішим частковим аспектам СО, і подано погляд "знизу" на задачу експлікації СО. В даній статті зробимо узагальнений огляд зазначених композиційних систем, що відповідає підходу "зверху".

В основі експлікації специфічної сутності СО лежить поняття про слово як про характерний різновид об'єктів даних. Слід зазначити, що дане поняття доволі поліморфне та допускає різноманітні уточнення, які відображають особливості різних систем СО. Тому на верхньому рівні абстракції недоцільно обмежуватися жорсткою фіксацією певної моделі слів (наприклад, тради-

ційною моделлю в термінах вільної напівгрупи). В першому наближенні поняття слова відобразимо гранично абстрактно, екстенсіонально — як множину $\mathcal{W} \subseteq \mathfrak{D}$.

Не можна не брати до уваги існування в СО також і деяких несимвольних даних, що відіграють певну допоміжну роль. Для різних систем СО можуть бути характерні свої суто специфічні типи несимвольних даних. В той же час є принаймні один тип даних, загальнозначущий для всякого програмування та включений до ядра ЕП і КП. Експлікативний аналіз поняття програмування показує, що, незалежно від природи об'єктів даних, загальною та сутнісно важливою характеристикою є обробка одночасно сукупності об'єктів, в якій кожен член ідентифіковано іменем. Адекватну математичну модель явища іменування дають поняття іменної множини та іменної функції.

Нехай Δ — клас об'єктів-*денотатів*, $\mathcal{V}$ — клас *імен*. *Іменна множина* (ІМ) є об'єктом d вигляду $\{(v_1,x_1),\ldots,(v_n,x_n)\}$, де $x_1, \ldots, x_n \in \Delta$ — довільні; $v_1, \ldots, v_n \in \mathcal{V}$ — попарно різні. Через $\mathrm{pr}^k R$ позначимо проекцію відношення R по k-й компоненті. Якщо $\mathrm{pr}^1 d = V$, то d називають V-іменною множиною (V-ІМ). Класи ІМ, V-ІМ позначимо $\mathcal{NS}(\Delta)$, $\mathcal{NS}^V(\Delta)$. Якщо клас Δ зрозумілий з контексту, пишемо просто $\mathcal{NS}$. Умова попарної нерівності імен означає, що ім'я не може мати одночасно кілька різних значень (денотатів) — *принцип однозначності іменування*. Іншими словами, для будь-якої іменної множини d та імені v з $(v,x_1) \in d$ та $(v,x_2) \in d$ слідує $x_1 = x_2$.

Іменними функціями (ІФ) називають функції, аргументами та (або) значеннями яких є ІМ. За допомогою ІФ адекватно відображається семантика програм, що є перетворювачами над абстрактною пам'яттю. Базовими ІФ є функції іменування та розіменування. Композиції над ІФ, які називають композиціями іменного рівня, відображають типові засоби конструювання програм [7] та дозволяють моделювати будь-який стиль або парадигму програмування, зокрема функціональну та імперативну [17]. Властивості композицій іменного рівня детально досліджувалися в роботах з КП та ЕП (див. [7] та бібліографію до неї), тому не будемо на них зупинятися.

Для поточного розгляду суттєво, що оскільки основним для СО типом простих даних є слова (клас $\mathcal{W}$), то характерний тип відношень іменування уточнюється за допомогою класу $\mathcal{NS}(\mathcal{W})$. Таким чином, для експлікації поняття даних є підстави постулювати принаймні включення $\mathfrak{D} \supseteq \mathcal{W} \cup \mathcal{NS}(\mathcal{W})$. Для спрощення розгляду приймемо $\mathfrak{D} = \mathcal{W} \cup \mathcal{NS}(\mathcal{W})$.

Таким чином, модель СО наслідує загальну модель "чистого" програмування, що складається з поняття ІМ, базових функцій над ІМ і композицій іменного рівня, та доповнює її засобами, які відображають символьну специфіку. Побудова моделі конкретної мови або системи СО зводиться до опису конкретних властивостей класу $\mathcal{W}$, базових функцій над словами та специфічно символьних композицій.

Літерна обробка

Дане відгалуження СО характеризується розглядом слів виключно крізь призму властивості *бути зіставленими з літер* [13, 14]. Поняття зіставленості в свою чергу уточнюється за допомогою операцій правого приписування літер, які взято за базові операції над словами. Нехай $\mathcal{A}$ — скінченна множина деяких об'єктів, які за означенням називаємо літерами. Кожній літері ставиться у відповідність тотальна функція $\vec{\xi}$ типу $\mathcal{W} \to \mathcal{W}$, яка змістовно відповідає приписуванню літери ξ справа до заданого слова. За основу побудови класу слів береться порожнє слово Λ — виділений (індивідуалізований) об'єкт з класу $\mathcal{W}$. До базових функцій над словами вводиться

функція-константа $\bar{\Lambda}$ така, що $\bar{\Lambda}(d) \equiv_{Df} \Lambda$ для будь-якої ІМ d. Тоді клас $\mathcal{W}$ уточнюється як замикання множини $\{\Lambda\}$ відносно множини операцій $\vec{\mathcal{F}}_{\mathcal{A}} = \{\vec{\xi} \mid \xi \in \mathcal{A}\}$. Очевидно, що для довільного $P \in \mathcal{W}$ можливо за допомогою базових функцій виразити функцію-константу $\bar{P}$ таку, що $\bar{P}(d) \equiv_{Df} P$. Для скорочення замість $\vec{\xi}(P)$ писатимемо $P\xi$.

Очевидне базове припущення про літерні структури слів полягає в тому, що літерна зіставленість характеризує слово вичерпним чином. Це означає, що рівність $P = Q$ має місце тоді й тільки тоді, коли або $P = \Lambda$ та $Q = \Lambda$, або $P = P'\xi$, $Q = Q'\zeta$, причому $P' = Q'$ та $\xi = \zeta$. Звідси слідує наступний принцип літерної обробки, що полягає у здійсненності розпізнавання літерної структури слова. Для кожного слова $P \in \mathcal{W}$ можливо встановити, чи мають місце рівності $P = \Lambda$ та $P = P'\xi$ для фіксованого ξ відносно невідомого P'. Природною подальшою конкретизацією є введення композиції правої іменної літерної диз'юнкції.

Якщо функція b іменна $\tilde{V}$-арна з областю значень $\mathcal{W}$, f_0, ..., f_N — іменні функції, відповідно V_0, ..., V_N-арні, v — ім'я, то $h = \underline{\mathbf{if}}^{v}_{\mathcal{A}}(b, f_0, f_1, \ldots, f_N)$ є V-арна, $V = \tilde{V} \cup V_0 \cup \ldots \cup V_N$, іменна функція, така, що

$$h(d) = \begin{cases} f_0(d), & \text{якщо} \quad P = \Lambda, \\ f_1(d'), & \text{якщо} \quad P = Q\xi_1, \\ \ldots \\ f_N(d'), & \text{якщо} \quad P = Q\xi_N, \end{cases}$$

де $P = b(d)$, $d' = d \nabla \{(v, Q)\}$.

Порядок на множині слів означується індуктивно, тобто конструюється з елементарних операцій над словами: $P\xi \succ P$; якщо $P \succ Q$, $Q \succ R$, то $P \succ R$.

Доведено, що композиційна система літерної обробки слів забезпечує породження усієї сукупності обчислюваних функцій над словами, тобто є повною. Зокрема, в даній композиційній системі неважко виразити поняття конкатенації — асоціативної бінарної операції • над словами та дуальне до нього поняття — похідну композицію деконкатенаційного циклу по всіх зображеннях заданого слова P у вигляді $P = Q_1 \bullet Q_2$ в порядку зростання або спадання слова Q_1.

Конкатенаційна обробка

Експлікацією поняття слова, основаною виключно на операції конкатенації, є вільна напівгрупа $\mathcal{W}$ відносно операції • з одиницею Λ без виділеної множини твірних [12]. Базовою функцією над словами є конкатенація, представлена іменною функцією $\mathrm{cat}^{u,v}$ над словами, яка іменній множині $\{(u, Q_1), (v, Q_2)\}$ ставить у відповідність слово $Q_1 \bullet Q_2$.

Для слів постулюється можливість розпізнати рівність. Безпосереднім продовженням даного припущення є введення композиції $\mathbf{if}_{=}$ *диз'юнкції по порівнянню*, яка іменним функціям b_1, b_2 та f_1, f_2 ставить у відповідність іменну функцію $h = \mathbf{if}_{=}(b_1, b_2, f_1, f_2)$, таку, що

$$h(d) = \begin{cases} f_1(d), & \text{якщо} \quad P_1 = P_2, \\ f_2(d), & \text{якщо} \quad P_1 \neq P_2, \end{cases}$$

де $P_1 = b_1(d)$, $P_2 = b_2(d)$.

Порядок на множині слів означується денотативно, шляхом опису властивостей: якщо $P = Q \bullet R$, то $P \succ Q$ (тоді Q називається початком слова P). Аналіз змісту поняття обробки слів приводить до висновку, що конкатенаційна обробка може вважатися конструктивною лише тоді, коли кожне слово має скінченну множину початків. Отже, коректним є поняття найбільшого та найменшого власного по-

чатку заданого слова, і до множини базових функцій можна включити функції деконкатенації $\underline{\text{cat}}^{-1}_{u,v}$ та $\underline{\text{cat}}^{-1}_{u,v}$, які довільному слову P ставлять у відповідність ІМ $\{(u,Q_1),(v,Q_2)\}$ з найменшим (найбільшим) власним початком Q_1.

Можна довести, що в композиційній системі, побудованій із зазначених засобів, можливо виразити похідну композицію деконкатенаційного циклу. Поповнення даної композиційної системи множиною словарних функцій-констант дозволяє виразити семантику довільного нормального алгоритму, а отже й будь-яку обчислювану функцію над словами.

Подальший аналіз умови конструктивності конкатенаційної обробки дозволяє природним чином зобразити літерну обробку як частковий випадок конкатенаційної. А саме можна довести, що в класі слів існують атомарні слова — такі, які неможливо зобразити конкатенацією двох інших слів. При цьому клас слів $\mathcal{W}$ співпадає з множиною всіх слів, які будуються конкатенацією з атомарних слів. Тоді для зведення літерної обробки до конкатенаційної достатньо додатково вимагати скінченності множини атомарних слів та ототожнити їх з літерами. Обернене зведення конкатенаційної обробки до літерної, як зазначено вище, також має місце.

Обробка однозначних та неоднозначних структур слів

Описана вище модель літерної обробки базується на двох основних властивостях структур слів: по-перше, операції породження слів є унарними, по-друге, слова допускають однозначний розбір. Відомо, що потужним інструментом експлікації програмування взагалі та його прикладних галузей зокрема є заперечення у сенсі гегелівської діалектики [18]. Піддамо запереченню першу із зазначених властивостей, зберігши в той же час другу.

Базовими операціями над словами є іменні функції к типу $\mathcal{NS}(\mathcal{W})\to\mathcal{W}$, такі, що для будь-якого слова P рівняння $\text{к}(d)=P$ відносно невідомої V-арної базової функції к та V-ІМ d має в точності один розв'язок. В основі специфічної логіки програмування лежить, як і у випадку літерної обробки, композиція диз'юнкції, що грунтується на розпізнаванні структури слова.

Можна довести, що в композиційній системі обробки однозначно структурованих слів можливо виразити будь-яку обчислювану функцію над словами.

Очевидний зв'язок з літерним рівнем полягає в тому, що літерні структури є частковим випадком однозначних структур, де операції к обмежені лише унарними.

Заперечення також і другої властивості літерних структур — однозначності розбору — веде до загального поняття структур слів. На відміну від попереднього випадку рівняння $\text{к}(d)=P$ відносно невідомої V-арної базової функції к та V-ІМ d може мати більше одного, але принаймні один розв'язок.

Частковим випадком неоднозначних структур є, очевидно, конкатенаційні структури. Розгортання поняття обробки неоднозначних слів, як і у випадку конкатенаційної обробки, ґрунтується на вимозі конструктивності їх обробки. Наслідком останньої є умова скінченності множини розв'язків рівняння $\text{к}(d)=P$. Тоді функції к можна вважати адекватним уточненням поняття структури слів в загальному випадку.

Рівняння зазначеного вигляду при фіксованому к цілком природно моделює поняття співставлення слова P зі зразком, заданим структурною функцією к [15].

Теоретично та практично значущий різновид структур слів отримаємо, поєднавши конкатенаційні структури з однозначними. Іншими словами, вся неоднозначність структур слів перекладається на операцію конкатенації, тоді як атомарними словами є слова,

породжені операціями к однозначної структури слів. Велике значення даного типу символьних структур обумовлене тим, що вони використовуються в розповсюджених мовах та системах СО, наприклад в мові Рефал [16]. Відповідна композиційна система будується як об'єднання композиційних систем конкатенаційної обробки та обробки однозначно структурованих слів.

1. *Алгоритмический* язык Аналитик-74 (информационная часть) / В.М. Глушков, Т.А. Гринченко, А.А. Дородницына и др. — Киев, 1977. — 49 с. — (Препр. / Ин-т кибернетики АН УССР, № 27).
2. *Турчин В.Ф.* Язык Рефал и его использование для преобразования алгебраических выражений // Кибернетика. — 1969. — № 3. — С. 58—62.
3. *Грисуолд Р., Полонски Дж.* Язык программирования Снобол-4. — М.: Мир, 1980. — 268 с.
4. *Мансуров В.Н.* Конструктивные и развивающиеся формальные системы. — Саратов: Изд-во Сарат. ун-та, 1988. — 196 с.
5. *Ахо А., Ульман Дж.* Теория синтаксического анализа, перевода и компиляции. — М.: Мир, 1978. — Т.1. — 612 с.
6. *Мансуров Н.Н.* Реализация расширенного языка Рефал на односвязной списковой памяти: Автореф. дис. ... канд. физ.-мат. наук: 05.13.11 / МГУ им. М.В.Ломоносова, Фак. вычисл. математики и кибернетики. — М., 1992. — 16 с.
7. *Редько В.Н.* Экспликативное программирование: ретроспективы и перспективы // Труды 1-й Междунар. науч.-практ. конф. по программированию «УкрПРОГ-98». — Киев: Кибцентр НАНУ, 1998. — С. 3—24.
8. *Редько В.Н.* Композиции программ и композиционное программирование // Программирование. — 1978. — № 5. — С. 3—24.
9. *Редько В.Н.* Универсальные программные логики и их применение // Системное и теоретическое программирование: Тез. докл. IV Всесоюз. симп. — Кишинев, 1983. — С. 310—326.
10. *Вінник В.Ю.* Про експлікацію символьної обробки: загальна характеристика проблеми та методологічні аспекти // Проблемы программирования. — 2002. — № 1—2: Спец. вып. Материалы 3-й Междунар. науч.-практ. конф. по программированию «УкрПРОГ—2002». — С. 51—57.
11. *Вінник В.Ю.* Експлікативні моделі нормальних алгоритмів // Вісник Житомир. інж.-технол. ін-ту. Сер.: техн. науки. — 2002. — № 2 (21). — С. 81—87.
12. *Вінник В.Ю.* Еталонні моделі символьної обробки: конкатенаційний рівень // МКІМ-2002. Міжнар. конф. з індуктивного моделювання, Львів, 20—25 травня 2002: В 4-х т. — Львів: ДНДІ інформаційної інфраструктури, 2002. — Т. 2. — С. 47—53.
13. *Вінник В.Ю.* Композиційні моделі літерних структур слів // Вісник Київ. Ун-ту. Сер.: фіз.-мат. науки. — 2002. — № 1. — С. 199—207.
14. *Вінник В.Ю.* Структури функцій символьної обробки літерного рівня // Там же. — № 2. — 10 с.
15. *Vinnik V.Yu.* On formal models of pattern matching // Proc. of the 5th Intern. sci. conf. "Electronic Computers and Informatics 2002", Košice—Herľany (Slovakia). — 2002. — P. 57—62.
16. *Вінник В.Ю.* Композиційна семантика мови Рефал // Вісник Житомир. інж.-технол. ін-ту. Сер. техн. науки. Спец. вип. за матеріалами Міжнар. наук.-техн. конф. "Інформаційно-комп'ютерні технології 2002". — 2002. — С. 44—52.
17. *Никитченко Н.С., Редько В.Н.* Композиционное и функциональное программирование: сравнительный анализ // Программирование. — 1985. — № 2. — С. 15—28.
18. *Nikitchenko N.S.* Towards Foundations of the General Theory of Transport Domains. — UNU/IIST Report No. 88. — Macau, 1996. — 37 p.

Отримано 08.01.03

Про автора

Вінник Вадим Юрійович

асистент кафедри програмного забезпечення обчислювальної техніки

Місце роботи автора:
Житомирський державний технологічний університет, м. Житомир

Тел. (0412) 41 8542
Ел. пошта: vvin@ratibor.zt.ukrtel.net; vvinnik@ziet.zhitomir.ua