УДК 618.3

В.Ю. Вінник, асист.
Житомирський інженерно-технологічний інститут

ЕКСПЛІКАТИВНІ МОДЕЛІ НОРМАЛЬНИХ АЛГОРИТМІВ

(Представлено д.ф.-м.н., акад. В.Н, Редьком)

З позицій експлікативного програмування аналізуються нормальні алгоритми А.А. Маркова (НА), які є теоретичним фундаментом багатьох практично значущих відгалужень символьної обробки. Виявлено базові поняття, необхідні та достатні для адекватного відображення сутності НА. Описано композиційну семантику формул підстановки та механізми їх взаємодії у складі НА.

Символьна обробка (СО) є однією з найважливіших з практичної точки зору галузей програмування, і спроби її теоретичного моделювання мають велике прикладне значення. Задача даної роботи – застосувавши методологію експлікативного програмування (ЕП) [1, 2] до нормальних алгоритмів А.А. Маркова (НА) [3], побудувати їх експліцитну модель. Важливість цієї задачі слідує з виняткової ролі НА як теоретичного фундаменту СО. Зокрема, побудована в цій роботі модель може слугувати базою для розробки більш складних моделей СО, збагачених додатковою специфікою, наприклад, моделей мови Рефал.

Перш ніж перейти до безпосереднього розгляду предмета статті, нагадаємо деякі ключові положення ЕП, які будуть використовуватися у подальшому. За більш детальною інформацією відсилаємо до основоположних робіт [1, 2].

Під *експлікацією* розуміють адекватне роз'яснення сутності предмета, яке починається з виявлення його найзагальніших сутнісних властивостей та здійснюється шляхом їх послідовного розгортання та уточнення. Концептуальне середовище для експлікації програмування в цілому і окремих прикладних програмувань становить система принципів, в якій центральне місце посідає принцип *програмологічності*. Згідно з цим принципом, сутнісним ядром програмування, яке повинно підлягати експлікації, є (прагматико-обумовлена – принцип *прагматичності*) *логіка* програмування – сукупність понять (як загальнопрограмологічних, так і специфічних для даного прикладного програмування) і засобів оперування поняттями, що характерні для процесів побудови програм та інших програмних об'єктів. Для того, щоб бути програмологічним засобом, програмобудівний засіб повинен відповідати неформальному критерію – узгоджуватися з інтуїтивною, природною логікою, внутрішньо притаманною прикладній галузі.

Програмний об'єкт, що розглядається крізь призму прагматико-обумовленої логіки програмування, називається *програмологічним*. Властивості програмологічних об'єктів у ЕП поділяються на *абстрактні* та *інкапсульовані*. Перші істотним чином проявляються на рівні програмної логіки і тому можуть і повинні братися до уваги при прийнятті проектних рішень в процесі конструювання об'єктів. Другі, навпаки, не проявляються на програмологічному рівні, несуттєві з прагматико-інтуїтивної точки зору і тому не повинні враховуватися у теоретичних моделях, що претендують на адекватність, щоб не "засмічувати" їх. Таким чином, програмологічний об'єкт характеризується не стільки своїми іманентними властивостями, скільки зануреністю у те чи інше середовище програмологічних відношень, що є одним з проявів *інтенсиональності* програмування.

Принцип *композиційності* уточнює поняття програмологічної операції до поняття *композиції* – операції, яка з відносно простих програмних об'єктів як частин утворює більш складне ціле. Поняття композиції є центральним у композиційному програмуванні (КП), яке за змістом включається в ЕП як його рання форма. Нарешті, принцип *комутаційності* конкретизує поняття композиції до поняття комутації. Іншими словами, композиція як каркас, що з'єднує між собою об'єкти-компоненти, виконує роль комутатора потоків даних та управління.

Експлікація прикладних галузей базується на ретельному неформальному, якісному аналізі сутнісновизначальних характеристик предмета. Задача аналізу – виявити такі характеристики предмета, без яких цей предмет не може адекватно мислитися, та яких у сукупності для цього достатньо. Основним специфічним принципом, що відображає принципово важливу особливість СО, є принцип *декомпозиційної драйверованості*, згідно з яким "рушієм" обчислювального процесу у СО є процес *декомпозиції* (розбору) слова, що обробляється [4, с. 244]. Важливий наслідок цього принципу полягає в тому, що між композиціями, що визначають структуру слів як об'єктів даних, та композиціями, що визначають структуру вербальних функцій, існує певний дуалізм, а саме: кожній композиції K над словами має відповідати композиція над функціями, що здійснює управління обчислювальним процесом, скероване розбором K-структури слова.

Вважаємо за потрібне зупинитися на відмінностях і водночас зв'язках експлікативної теорії НА як складової частини експлікативної теорії СО з теорією НА, побудованою А.А. Марковим та його шко-

лою [3]. Цільові установки класичної теорії алгоритмів істотно відрізняються від цілей ЕП. Центральною у теорії алгоритмів є проблема математичного уточнення інтуїтивного поняття *конструктивного процесу*, тоді як центральною проблемою ЕП є уточнення інтуїтивних уявлень про *програмування* як про керований прагматико-обумовленою логікою процес створення програм [2]. Орієнтація на конструктивність зумовлює *висхідну* логіку побудови теорії НА, коли за відправні беруться елементарні об'єкти і також елементарні дії над об'єктами, конструктивність яких самоочевидна, а усі складніші об'єкти та процеси явно, або *конотативно*, моделюються через базові. Звідси слідує порівняно високий рівень конкретизації теорії, оскільки усі поняття вищих рівнів успадковують певну специфіку від нижчих рівнів ієрархії, через які вони означені. Наприклад, навіть схема НА – поняття вищого рівня у даній теорії – вводиться в [3] як слово спеціального вигляду.

Для ЕП характерне, навпаки, низхідне розгортання від якомога абстрактних (але разом з тим достатньо конкретних, щоб бути адекватними) базових понять та принципів шляхом збагачення додатковою конкретикою, причому головним критерієм є не конструктивність моделі, а її адекватність. Це дозволяє запобігти потраплянню до еталонної моделі зайвої специфіки – такої, яка не має безпосереднього відношення до сутності предмета та не обумовлена прагматичними потребами. Як буде видно з подальшого, деякі з понять та сутностей, що були необхідні для побудови марківської теорії НА (в першу чергу – для обґрунтування конструктивності), не будуть необхідними для експлікації НА. Отже, в цьому сенсі еталонна модель НА виявляється абстрактнішою та простішою від теорії НА у її класичному вигляді.

З іншого боку, теорія алгоритмів як розділ "чистої" математики не враховує (і відповідно до своїх цілей не повинна враховувати!) специфіку процесів конструювання алгоритмів – процесів програмування, і тому абстрагується від центральних з точки зору ЕП питань програмологічності та інтенсиональних аспектів. В цьому сенсі еталонна модель НА виступає як конкретніша за марковську теорію, бо враховує вказану специфіку.

Таким чином, добре відома класична теорія НА як уточнення поняття обчислюваності над словами та експлікативна модель НА як уточнення поняття програмування символьної обробки мають хоча й однаковий *об'єкт*, але різний *предмет* та різні області застосування і тому не входять у суперечність, органічно доповнюючи одна одну.

Зазначимо, що наявність ретельно розробленої теорії НА [3], де формально-математичному уточненню передує вичерпний та надзвичайно глибокий неформальний аналіз, істотно спрощує нашу задачу. Замість того, щоб розробляти відповідну систему базових понять "з чистого листа" та досліджувати зв'язки між ними, ми можемо значною мірою скористатися результатами такого аналізу, наведеними в [3], та здійснити їх переорієнтацію на експлікативну парадигму.

З огляду на зазначену вище відмінність між висхідною логікою побудови марковської теорії НА та низхідною логікою експлікації, сподіваємося, зрозуміло, що за відправну точку експлікації слід взяти поняття найвищого рівня – поняття НА, яке у теорії Маркова з'являється лише після цілої низки проміжних понять та теорем. Далі ми відтворимо, в загальних рисах, марковський хід побудови теорії у зворотному порядку. А саме, для кожного поняття високого рівня абстракції будемо з'ясовувати, через які більш прості поняття воно експлікується за Марковим та з якими поняттями воно має сутнісні зв'язки. Експлікацію буде завершено тоді, коли вихідне поняття НА буде зведено до понять, достатньо прозорих та очевидних з інтуїтивної точки зору. Підкреслимо, що з точки зору конструктивності ці поняття все ще залишаються досить складними і такими, що потребують подальшого розгортання, але в контексті ЕП доцільно зупинитися на них з огляду на прагматику. Зазначимо також, що завдяки класичним дослідженням з теорії НА та конструктивної математики ми звільняємося від необхідності кожного разу обґрунтовувати конструктивність того чи іншого поняття.

Позначимо універсум слів через W. З найабстрактнішої точки зору НА постає як перетворювач над словами – функція A типу $W \to W$. Єдиними програмологічними ознаками такої моделі НА є властивості застосовуватися до слова та давати слово на виході, що описується тривіальною формулою:

$$\mathbf{Apply}(A, P) = Q,$$

де через $\mathbf{Apply}$ позначено композицію аплікації (застосування функції до аргументу).

Зазначимо, що поняття *слова*, яке було щойно використано, не є абсолютно елементарним і саме потребує експлікації. Але для того, щоб зробити це обґрунтованим чином, потрібно спочатку уточнити специфіку процедур обробки слів у даній галузі – лише знаючи, які операції над словами суттєві з прагматичної точки зору, можна обрати адекватний тип абстракції для розгляду слів, тобто такий, що максимально прозоро висвітлює саме ці суттєві операції над словами і в той же час не перевантажений зайвими характеристиками. Отже, експлікацію слів відкладемо на потім.

Ясно, що модель НА як "чорного ящика" надто абстрактна, бідна за змістом та не дає можливості для більш детального розгляду. Разом з тим, саме через свою абстрактність вона багата потенційними можливостями найрізноманітніших конкретизацій, з яких нам слід обрати найбільш адекватну. Щоб семан-

тику НА описати на рівні, відмінному від тривіального, немає іншого виходу, як розкрити "чорний ящик", виявивши його внутрішню будову: складові елементи та структурні відношення між ними. Відомо, що складовими елементами НА є *формули підстановки* [3]. Таким чином, задача експлікації НА зводиться до двох задач: експлікації формул підстановки та експлікації способу поєднання цих формул у складі НА, що, зрозуміло, є композицією. Цим обумовлено розгляд формул підстановки на двох рівнях абстракції.

На верхньому рівні формулу підстановки будемо розглядати як самостійний об'єкт, безвідносно аспектів взаємодії з іншими формулами у складі НА. У цьому сенсі формула підстановки експлікується як перетворювач над словами – деяка функція типу $W \to W$, яку будемо називати *функцією підстановки*.

Функціональний аспект, яким би він не був важливим, не вичерпує змісту поняття формули підстановки. Справді, як видно зі стандартних викладів теорії НА, формула не лише породжує результат підстановки, але ще й визначає, що з ним робити далі, передаючи управління в залежності від успіху чи неуспіху підстановки або на початок НА, або на вихід НА, або на наступну формулу. Іншими словами, має місце нетривіальна *комутація* потоків даних та управління. Саме ця властивість є, очевидно, ключовою для поділу формул підстановки на *звичайні* та *завершальні*, який на попередньому рівні абстракції був інкапсульований. Враховуючи сказане, на другому, конкретнішому, рівні абстракції ми конкретизуємо функцію підстановки властивістю керувати викликом функції-наступника, щоб потім на цій основі описати семантику НА. Але поки що зосередимося на першому, абстрактнішому рівні розгляду формул підстановки.

Розглядаючи функції підстановки як перетворювачі над словами, ми знов не можемо обмежитися тривіальним розглядом їх як абстрактних функцій типу "чорного ящика" і, в свою чергу, маємо розкрити їх будову та механізми застосування до аргументу. Почнемо з неформальної характеристики. Функція підстановки є не елементарним об'єктом, а будується з інших об'єктів за допомогою певної композиції. Конкретніше, функція підстановки утворюється з пари слів S, T спеціальною композицією, яку позначимо знаком $\triangleright$ (позначення $\to$ та $\to$□ зарезервуємо для більш конкретного рівня розгляду підстановок, де буде враховано їх роль в управлінні обчислювальним процесом у складі НА). Іншими словами, терм вигляду $\triangleright(S,T)$, або, якщо використати більш зручне інфіксне позначення, $S \triangleright T$ є експліцитним зображенням функції підстановки.

Слово S називають *лівою частиною*, а слово T – *правою частиною* підстановки. Звичайно, властивість бути лівою (правою) частиною не належить слову як такому і не виділяє власну підмножину у множині слів, а відображає *роль*, у якій дане слово виступає у складі композиційного об'єкта, тобто є не внутрішньою, а зовнішньою властивістю.

Принципово важливою особливістю функції підстановки є те, що процедура її застосування до слова-аргументу P чітко розпадається на два суттєво різнотипні кроки. На першому з них здійснюється розпізнавання допустимості даної функції для даного аргументу шляхом аналізу структури останнього з виділенням у ньому деяких компонентів, тобто *співставлення зі зразком*, заданим словом S. Другий крок виконується лише у разі, коли підстановка допустима, та полягає у синтезі результату з використанням цих компонентів і слова T. Таким чином, ліва частина здійснює *аналіз аргументу*, а права – *синтез результату*. Між іншим, поділ процедури застосування символьного перетворювача на кроки двох вказаних типів віднесено до загальної сутнісної ознаки символьної обробки [4, с. 244].

Спроби подальшої деталізації внутрішніх механізмів функції підстановки наштовхуються на потребу означити деякі операції над словами, а значить конкретизувати структури слів і, зрештою, дати певну експлікацію поняття слова. Згідно з канонічним викладом теорії НА [3], ключовими для нас мають бути поняття *конкатенації* слів, *підслова*, *входження*, *лівого* та *правого крила* входження. Крім того, нас цікавить не будь-яке, а *перше* зліва входження, для чого потрібно мати над словами відношення часткового *порядку*. Для адекватності моделі потрібно визначити такий тип абстракції у розгляді слів, який був би достатньо конкретним, щоб відобразити всі суттєві аспекти предмета, але водночас і досить абстрактним, щоб приховати несуттєві подробиці. Літерний рівень абстракції, взятий за основу в [3], ми вважаємо надто конкретним та перевантаженим подробицями. Справді, специфіка літерної зіставленості безпосередньо не використовується для опису механізмів НА. На нашу думку, більш адекватним є тип абстракції, де структури слів розглядаються з точки зору однієї лише операції – конкатенації.

Для слів, за базовим припущенням, має сенс абстракція ототожнення, тобто слова можливо порівнювати, встановлюючи їх однаковість чи відмінність. Тотожність слів виступає в першу чергу як *композиція* – фундаментальна характеристика способу існування слів. Це поняття вважається первинним, таким, що не зводиться до жодних інших понять. Крім того, рівність може виступати як *операція порівняння* – функція типу предикату, що може використовуватися у складі більш складної функції, наприклад, як умова циклу. В обох ролях рівність слів позначається знаком "$\doteqdot$", що не повинно призводити до непоро-

зумінь, оскільки, по-перше, між цими ролями існує глибока єдність, а по-друге, з контексту завжди ясно, яка з них мається на увазі. Слова всюди далі розглядаються з точністю до рівності $\doteq$.

Фундаментальною композицією, що характеризує внутрішню структуру слів, є *конкатенація*, яка позначається знаком "$\square$". Цим же знаком позначається і *функція* конкатенації. Поняття конкатенації у сенсі композиції є первинним, не означується через інші поняття, але експлікується шляхом опису властивостей. Найважливішою з них є асоціативність:

$$(P \square Q) \square R \doteq P \square (Q \square R). \qquad (1)$$

Крім того, постулюється існування в універсумі слів виділеного об'єкта – порожнього слова Λ з характеристичною властивістю

$$\Lambda \square P \doteq P \square \Lambda \doteq P. \qquad (2)$$

Нарешті, вводиться негативний принцип, згідно з яким у алгебрі слів немає інших тотожностей та визначальних співвідношень, крім виписаних вище та їхніх наслідків. Тим самим, універсум слів уточнюється як вільна напівгрупа з множенням $\square$, одиницею Λ та без виділеної сукупності твірних елементів.

Нехай слова P, Q та R такі, що $R \doteq P \square Q$. Тоді слово P називається (*власним*, якщо при цьому $Q \doteq \Lambda$) *початком*, а слово Q – (*власним*, якщо при цьому $P \doteq \Lambda$) *кінцем* слова R. Слово Q називають також *кінцевим доповненням* слова P у слові R. Трійка (Q_1, P, Q_2), така, що $Q_1 \square P \square Q_2 = R$, називається *входженням* слова P у слово R, а слова Q_1 та Q_2 – відповідно *лівим* та *правим крилом* даного входження.

Якщо P – початок (власний початок) слова R, то за означенням $P \circ R$ (відповідно, $P \prec R$). Очевидно, дане відношення є частковим порядком. Входження (Q_1, P, Q_2), у якого ліве крило Q_1 найменше (у сенсі порядку $\prec$) серед усіх входжень слова P у слово R, називається *першим зліва*.

Тепер, на основі введених понять, процедура застосування функції підстановки $S \triangleright T$ до слова P як аргументу уточнюється так. Нехай (P_1, S, P_2) – перше зліва входження слова S у слово P. Тоді

$$\mathbf{Apply}(S \triangleright T,\ P) = P_1 \square T \square P_2.$$

Якщо такого входження не існує, значення функції підстановки на даному аргументі вважаємо невизначеним.

Як видно звідси, щоб завершити експлікацію функцій підстановки, потрібно експлікувати процедуру пошуку першого зліва входження. Для цього потрібно залучити засоби аналізу (розбору, декомпозиції) конкатенаційних структур, які, згідно з принципом декомпозиційної драйверованості, становлять композиційний фундамент структур вербальних функцій у даній галузі СО. Основою сімейства таких композицій є деконкатенаційний цикл $\mathbf{W}\overrightarrow{\mathrm{WW}}$.

Нехай U – множина імен, $u, v \in U$, функція f – іменна, U-арна, $U \setminus \{u, v\}$-арнозначна, d – U-іменна множина, в якій значенням імені v є деяке слово R. Тоді функція $g = \mathbf{W}\overrightarrow{\mathrm{WW}}^{\langle u,v \rangle}(f)$ – U-арна, U-арнозначна, процедура застосування якої до аргументу d є ітеративний процес, що описується співвідношеннями:

$$d_0 = d,$$
$$d'_{i+1} = d_i \nabla (u \Leftarrow P_i \square v \Leftarrow Q_i),$$
$$d_{i+1} = d'_{i+1} \nabla f(d'_{i+1}),$$
$$g(d) = d_n,$$

де через ∇ позначено заміщення іменних множин; через $\square$ – композицію паралельного виконання (більш детальну інформацію можна знайти у згаданих вище роботах з ЕП); n – найменше, для якого $Q_n = \Lambda$, а кожна пара (P_i, Q_i) є розбиття вихідного слова R на початок та кінець, $R = P_i \square Q_i$, причому послідовність P_i монотонно зростає: $P_i \prec P_{i+1}$. Іншими словами, дана композиція як спеціалізований різновид циклу здійснює перебір усіх розбиттів вихідного слова на початок P_i (що поіменовується ім'ям u) та кінець Q_i (що поіменовується ім'ям v) у порядку зростання початків, для кожного такого розбиття застосовуючи тіло циклу, до тих пір, поки не буде перебрано усі можливі пари.

Тепер, користуючись щойно описаною композицією, яка є загальнозначущим засобом конструювання функцій над конкатенаційно структурованими словами, означимо специфічну композицію, яка знадобиться для нашої конкретної задачі, а саме: параметризовану композицію *диз'юнкції за першим вхо-*

дженням підслова, яку позначимо $\mathbf{if}\overrightarrow{entry}$. Якщо u та v – деякі імена, S – довільне, але фіксоване слово, то композиція $\mathbf{if}\overrightarrow{entry}_S^{\langle u,v\rangle}$ ставить у відповідність $\{u, w\}$-арній вербальнозначній функції h_1 та вербальній вербальнозначній функції h_2 вербальну вербальнозначну функцію $g = \mathbf{if}\overrightarrow{entry}_S^{\langle u,w\rangle}(h_1, h_2)$, таку, що для довільного слова P результат застосування функції g визначається наступним чином. Якщо існує входження слова S у слово P і (P_1, S, P_2) – перше зліва таке входження, то $g(P) = h_1(u \Leftarrow P_1 \,\square\, u \Leftarrow P_2)$. Якщо такого входження не існує, то $g(P) = h_2(P)$. Неважко переконатися, що саме цю композицію задає блок-схема, показана на рис. *1*.

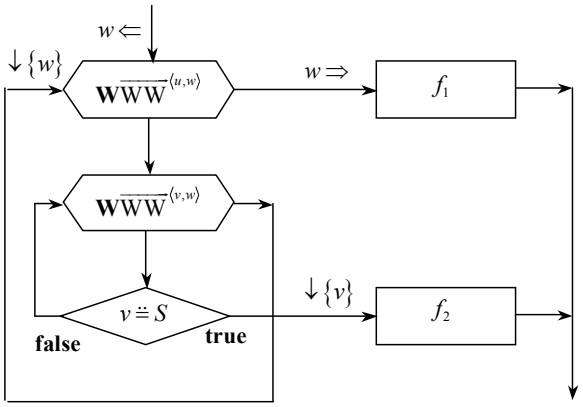

Рис. 1. Композиція диз'юнкції за першим входженням

Слід пояснити, що помітки на ребрах цього графу означають допоміжні інтерфейсні композиції, що відповідають за узгодження типів об'єктів. Помітка $w \Leftarrow$ на вхідній стрілці означає, що подане на вхід конструкції слово P поіменовується ім'ям w і перетворюється на одноелементну іменну множину $\{(w, P)\}$, помітка $w \Rightarrow$ означає, навпаки, розіменування імені w, а помітки $\downarrow\{w\}$ та $\downarrow\{v\}$ – видалення з об'єкта типу іменної множини відповідного імені. За більш детальною інформацією про семантичне конструювання програм у КП та ЕП знов відсилаємо до першоджерел.

Тепер функція підстановки еталонується наступним композиційним термом:

$$(S \triangleright T) = \mathbf{if}\overrightarrow{entry}_S^{\langle u,v\rangle}(u \Rightarrow \bullet T \bullet v \Rightarrow, \perp),$$

де через ^ позначено всюди невизначену функцію. Таким чином, функціональний аспект формул підстановки остаточно експліковано.

Тепер потрібно уточнити, як формули підстановки здійснюють управління обчислювальним процесом. Сказаного вище, сподіваємося, досить, щоб обґрунтувати адекватність наступної експлікації даного поняття:

$$(S \,\%_{f_2}^{f_1}\, T) = \mathbf{if}\overrightarrow{entry}_S^{\langle u,v\rangle}((u \Rightarrow \square T \,\square v \Rightarrow) \circ f_1, f_2),$$

де f_1 та f_2 – вербальні функції, перша з яких викликається у випадку успішної, а друга – у випадку неуспішної підстановки, а через % позначено композицію, яку назвемо *диз'юнкцією за підстановкою*.

Залишається показати, як з формул підстановки компонуються НА. Розглянемо схему НА, що складається з n формул вигляду $(S_i\ \xi_i\ T_i)$, де S_i та T_i – слова, що відіграють ролі лівої та правої частин, а через ξ_i позначено формулоутворюючу композицію: $\rightarrow$ для звичайної та $\rightarrow\square$ для завершальної формул.

Семантику формули $(S_i\ \xi_i\ T_i)$ як перетворювача над словами, у якій враховано дію на слово-аргумент з боку не лише самої підстановки, але й з боку тих функцій, виклик яких керується даною формулою, позначимо через f_i, $i = 1, \ldots, n$. Тоді функція f_1 разом з семантикою першої формули схеми відображає і семантику усього алгоритму. В допоміжних цілях введемо також функцію f_{n+1}. Під позначенням φ_i, f_i, $i = 1, \ldots, n$, домовимося розуміти функцію f_1, якщо $\xi_i = \rightarrow$, та функцію $\mathbf{e}$, якщо $\xi_i = \rightarrow\square$. Тоді семантику кожної формули підстановки $(S_i\ \xi_i\ T_i)$ та усього алгоритму отримуємо як розв'язок (у сенсі найменшої нерухомої точки [5]) наступної системи рівнянь:

$$f_i = S_i \%_{f_{i+1}}^{\varphi_i} T_i \quad (i = 1, \ldots, n),$$

$$f_{n+1} = \mathbf{e}.$$

Зазначимо, що дана система за формою зручна для застосування своєрідного аналогу методу Гауса [5]. Врахувавши, що семантика кожної i-ї формули підстановки істотно використовує семантику *усіх наступних* формул, а також першої, можна запропонувати дещо модифіковану форму попереднього означення.

Позначимо подвійними дужками $^{\mu}\llbracket t \rrbracket$ композицію, яка перетворює композиційний терм t на функцію з ім'ям μ, причому дане ім'я локальне, тобто має сенс лише у межах цих дужок. Всяке входження імені μ у терм t розглядається як композиція рекурсії.

Розглянемо сукупність формул підстановки, що разом утворюють НА, як *список* з виділеним *зліва* початком та хвостом. А саме, за основу візьмемо порожній список, який позначимо через ε, та композицію *лівого приписування*, яка формулі F та списку L ставить у відповідність список $F \mid L$.

Тоді семантику списків формул підстановки дає наступне індуктивне означення:

$$\varepsilon = \mathbf{e}$$

$$(S \to T) \mid L = \left(S \,\%_L^{\mathbf{M}}\, T\right),$$

$$(S \to \llbracket T \rrbracket) \mid L = \left(S \,\%_L^{\mathbf{e}}\, T\right).$$

Остання функція є параметризованою: у ролі параметра виступає (вільне) ім'я μ. Нарешті, щоб з семантики списку формул L отримати семантику f відповідного алгоритму, достатньо зв'язати вільне ім'я μ композицією $^{\mu}\llbracket \ldots \rrbracket$, зробивши функцію f денотатом даного імені:

$$f = {}^{\mu}\llbracket L \rrbracket.$$

Отже, здійснена експлікація нормальних алгоритмів дозволила краще відобразити їх програмологічні аспекти, ніж традиційний підхід, орієнтований на проблеми "чистої" математики. Крім того, дана експлікація важлива не лише сама по собі, але і як база для експлікації більш конкретних відгалужень СО, наприклад, мови Рефал.

ЛІТЕРАТУРА:

1. *Редько В.Н.* Основания программологии // Кибернетика и системный анализ. – 2000. – № 1. – С. 35–57.
2. *Редько В. Н.* Эталонное программирование: ретроспективы и перспективы // Проблемы программирования. – 2000. – № 1–2. Спец. выпуск: Материалы 2-й Межд. науч.-практ. конф. по программированию УкрПрог 2000. – С. 13–28.
3. *Марков А.А.*, *Нагорный Н.М.* Теория алгорифмов. – М.: Наука, 1984. – 432 с.
4. *Кауфман В.Ш.* Языки программирования. Концепции и принципы. – М.: Радио и связь, 1993. – 430 с.
5. *Буй Д.Б.* Системи рівнянь в індуктивних множинах: метод Гауса, інваріантні перетворення, взаємозв'язок між рекурсією та суперпозицією, похідність багатомісної рекурсії // Спец. выпуск. Материалы 2-й Межд. науч.-практ. конф. по программированию УкрПрог 2000. – С. 63–74.

ВІННИК Вадим Юрійович – асистент кафедри програмного забезпечення обчислювальної техніки Житомирського інженерно-технологічного інституту.

Наукові інтереси:

– фундаментальні проблеми теоретичної програмології, експлікативне програмування;

– формальні методи у програмуванні, семантика мов програмування;

– прикладна конструктивна математика та логіка;

– методи програмного моделювання математичних об'єктів, галузей та систем.

Тел. (0412) 208-542

E-mail: vvinnik@ziet.zhitomir.ua; vvin@ratibor.zt.ukrtel.net

Подано 26.03.2002

Вінник В.Ю. Експлікативні моделі нормальних алгоритмів
Винник В.Ю. Экспликативные модели нормальных алгоритмов
Vinnik V.Yu. Explicative models of normal algorihtms

УДК 618.3

Экспликативные модели нормальных алгоритмов / В.Ю. Винник

С позиций экспликативного программирования анализируются нормальные алгоритмы А.А.Маркова (НА), являющиеся теоретическим фундаментом многих практически значимых областей символьной обработки. Выявлены базовые понятия, необходимые и достаточные для адекватного отображения сущности НА. Описана композиционная семантика формул подстановки и механизмы их взаимодействия в составе НА.

УДК 618.3

Explicative models of normal algorihtms / V.Yu. Vinnik

From the perspective of explicative programming, Markovian normal algorithms (NA), that constitute a theoretical foundation for many practically valuable branches of symbol processing, are treated. Fundamental concepts that are necessary and sufficient for an adequate representation of the essence of NA. Compositional semantics of substitution formulae and mechanisms of interaction between them within NA are described.