УДК 681.3.06

СИСТЕМИ СПЕЦИФІКАЦІЙ ОБ'ЄКТНО-ОРІЄНТОВАНИХ ПРОГРАМ НАД НОМІНАТИВНИМИ ДАНИМИ

© Л. Л. Омельчук

Київський національний університет імені Тараса Шевченка
факультет кібернетики
кафедра теорії та технології програмування
пр-т Академіка Глушкова, 4д, м. Київ, 03680, Україна
e-mail: *l.omelchuk@gmail.com*

The specification systems of the object-oriented programs with nominative data.

Omelchuk L. L.

Abstract. Today the problem of fast and economical design of reliable software is up to date. For a quick and economical design of reliable software is possible to apply formal methods of software specifications. Formal methods allow to prove some properties of programs using mathematical methods.

Specifications program should include a description of program goals, functional program structure, input application program output. Problem improving the adequacy of representation of data structures, functions, and compositions used in programming is important.

In this article the notion of abstract computably has been defined. Complete classes of computable functions of various abstraction levels have been described. Formal semantic-syntactic models of specification and programming languages have been defined and investigated.

Imperative and declarative models of nondeterministic programs based on composition-nominative approach are constructed and investigated. Semantics of such programs is presented by partial multi-valued functions. The complete class of naturally computable functions of described type is defined and its algebraic representation is built. A special computability considered in this paper is nominative computability. Nominative computability allows to set adequately a complete class of computable functions over nominative data. At the same time nominative computability is invariant relative to a set of basic elements. Moreover, it is oriented to functions and compositions, which are close to function and compositions of programming languages.

You can increase the adequacy of the default data structures, functions, and compositions used in programming languages if you use nominative data.

Axiomatic theory of nominative data develops a theory of admissible sets. This theory has a number of advantages with respect to the adequacy of programming. It is powerful enough to produce computable function over various data structures. It is not as restrictive as different versions of constructive logic, but not too strong and does not allow, for example, the use of

axioms constructing the set of all subsets (compared to the set theory ZF). This theory uses the basic data (praelementy) corresponding methods of constructing data programming.

An axiomatic theory of nominative data, which is capable to specify all computable functions, is constructed. The axiomatic system of program specification over nominative data is constructed.

Prototype axiomatic system software specifications of nominative data (OBJ-NDSL) is constructed. It is based on composition nominative method, axiomatic system software specifications of nominative data and it is based on the sequent calculi for logics. This prototype is based also on language Object-Z.

OBJ-NDSL system allows to prove some properties of programs. shown that for an axiomatic system specification software method can effectively use composition nominative approach. Composition nominative approach is sufficient to adequately meet the needs of programming.

Вступ

Постійне розширення сфери застосування обчислювальної техніки та необхідність побудови все більш складних програмних систем загострює проблему швидкого та економічного конструювання надійного програмного забезпечення. Ускладнення програмних комплексів, а також збільшення залежності людей від правильного функціонування систем, викликає зростання вимог до їх надійності. В багатьох випадках «ручні» методи розробки програмного забезпечення стають незадовільними.

Одним з кроків до вирішення проблеми швидкого та економного конструювання надійного програмного забезпечення в умовах постійно зростаючої складності конструюємих програмних систем є застосування формальних методів специфікацій програм, що дозволяє доведення певних властивостей програм, зокрема властивості правильності, за допомогою математичних методів.

Специфікація програми повинна включати в себе опис цілей програми, її функціональну структуру, вхідні та вихідні дані програми. При побудові мов специфікацій програм, важливою є задача підвищення рівня адекватності подання структур даних, функцій та композицій, що використовуються в програмуванні.

В [1, 2] показано, що з точки зору композиційно-номінативного підходу (КНП) [1, 2, 3, 4], використовуючи номінативні дані [1, 2, 3, 4] можна підвищити адекватність задання структур даних, функцій та композицій, що використовуються в мовах програмування (процедурне програмування), та будувати системи специфікацій програм на єдиній концептуальній основі. В композиційному програмуванні досліджуються системи на різних рівнях абстракції [3, 4], які виникають на шляху експлікації програмування — абстрактному, булевському та номінативному (атрибутному) рівнях.

Системи останнього рівня, композиційно-номінативні системи, є досить багатими для адекватного задання моделей структур даних, програм та засобів їх конструювання.

1. Аксіоматична теорія номінативних даних

Аксіоматична теорія номінативних даних [1, 2, 3] розвивається в дусі теорії допустимих множин (С. Кріпке, Р. Платек, Дж. Барвайз, Ю.Л. Єршов). Ця теорія має низку переваг, у відношенні адекватності до програмування: з одного боку, вона досить потужна, щоб породжувати обчислювані функції над різними структурами даних, з іншого ж боку, не настільки обмежувальна, як різні варіанти конструктивних логік, але і не надмірно потужна та не допускає, наприклад, застосування аксіоми побудови множини всіх підмножин (в порівнянні з теорією множин ZF). Крім того, ця теорія використовує базові дані (праелементи), що відповідає методам побудови даних у програмуванні. Теорія номінативних даних будується як аксіоматична теорія 1-го порядку з рівністю та тернарною зв'язкою (предикатом) номінативної належності, що записуються в інфіксній формі $x \mapsto y{\in_n}a$ (або $(x, y){\in_n}a$).

Класом Δ_0-формул називається найменший клас Y, що містить елементарні формули і замкнений відносно наступних операцій: 1) якщо $\varphi \in Y$, то і $\bar{\varphi} \in Y$, 2) якщо $\varphi, \psi \in Y$, то і $\varphi \vee \psi \in Y$ і $\varphi \wedge \psi \in Y$, 3) якщо $\varphi \in Y$, то і $\forall x \mapsto y{\in_n}a\varphi, \exists x \mapsto y{\in_n}a\varphi \in Y$ для всіх змінних x, y, a.

Клас Σ-формул є найменший клас Z, що містить Δ_0-формули і замкнений відносно умов 2) і 3) визначення класу Δ_0-формул і наступної умови екзистенціальної квантифікації: якщо $\varphi \in Z$, то $\exists u\varphi \in Z$.

Спеціальні аксіоми аксіоматичної системи специфікацій програм над НД діляться на три групи: перша група описує властивості рівності; друга — описує властивості множини імен та даних; третя група аксіом описує властивості НД:

1. екстенсіональність: $\forall x \forall y(x \mapsto y{\in_n}a \leftrightarrow x \mapsto y{\in_n}b) \to a = b$;
2. фундованість (індукція за належністю):
   $(\forall a(\forall x \mapsto y{\in_n}a\varphi(x) \wedge \varphi(y) \to \varphi(a))) \to \forall a\varphi(a)$;
3. індукція за включенням: $(\forall a(\forall b \subset a\varphi(b) \to \varphi(a))) \to \forall a\varphi(a)$;
4. Δ_0-виділення: $\exists b \forall x \forall y(x \mapsto y{\in_n}b \leftrightarrow x \mapsto y{\in_n}a \wedge \varphi_0(a))$;
5. іменування: $\exists c x \mapsto y{\in_n}c$;
6. об'єднання: $\exists c(a \subseteq c \wedge b \subseteq c)$;
7. нетривіальність: $\exists a \exists x \exists y(x \mapsto y{\in_n}a)$.

2. Номінативна обчислюваність

В [3] введено до розгляду та досліджено спеціальний вид обчислюваності — номінативну обчислюваність. Номінативно обчислюваними називаються функції над номінативними даними, отримані замиканням функцій $\{\Rightarrow 0, \Rightarrow 1, \bar{[]}_D, \searrow_D, \bigcup_D, \setminus_D, (=_W)_D, as_D, cn_D, \in W_D\}$ відносно множини композицій $\{\circ_D, \diamond_D, *_D, \Theta_D\}$. Показано [1], що довільна частково-рекурсивна функція може бути представлена номінативно обчислюваними функціями над множиною натуральних чисел при їх моделюванні у класі номінативних даних. Крім того, показано [1, 2], що кожна номінативно обчислювана [1, 2] функція представима деяким бінарним Σ-предикатом $P(x, y)$, тобто $f(x) = y$ тоді і тільки тоді, коли $P(x, y)$. Для цього будуються представлення всіх функцій, зазначених у визначенні номінативної обчислюваності, а також всіх функцій, отриманих застосуванням композицій.

3. Системи специфікацій програм для ООП над номінативними даними

На сьогоднішній день абсолютним лідером в прикладному програмуванні є об'єктно-орієнтоване програмування (ООП) [5]. Таким чином при побудові сучасних мов специфікацій програм необхідно враховувати специфіку об'єктно-орієнтованих мов програмування, а саме те, що методологія ООП базується на представленні програми у вигляді сукупності об'єктів, кожен з яких є екземпляром певного класу, а класи утворюють ієрархію наслідування [5]. Покажемо, що КНП може бути застосований для побудови систем специфікацій програм для ООП [5]. Для цього спочатку формалізуємо поняття об'єкта, використовуючи КНП. Базові типи даних мов програмування були задані в [1, 2], крім того в [1, 2] було задано функції над номінативними даними. Об'єкт можна представити номінативними даними наступного вигляду:

$<$ об'єкт $>::= []|[<$ ім'я $-$ об'єкту $>\mapsto<$ опис $-$ об'єкту $>]$
$<$ опис $-$ об'єкту $>::= [class \mapsto [name \mapsto <$ ім'я $-$ класу $>, base \mapsto []|$
$<$ об'єкт $>, interface - list \mapsto []| <$ інтерфейси $>, members \mapsto []|$
$[attributes \mapsto []|<$ атрибут $>, \ldots, <$ атрибут $>, methods \mapsto []|$
$<$ метод $>, \ldots, <$ метод $>, properties \mapsto []|<$ властивість $>, \ldots, <$ властивість $>]]$
$<$ інтерфейси $>::= <$ інтерфейс $>, \ldots, <$ інтерфейс $>$
$<$ інтерфейс $>::= [<$ ім'я $-$ інтерфейсу $>\mapsto<$ опис $-$ інтерфейсу $>]$
$<$ опис $-$ інтерфейсу $>::= [interface \mapsto [name \mapsto <$ ім'я $-$ інтерфейсу $>,$
$interface - list \mapsto []| <$ інтерфейси $>, members \mapsto []|[methods \mapsto []$
$|<$ метод $>, \ldots, <$ метод $>, properties \mapsto []|<$ властивість $>, \ldots, <$ властивість $>]]$
$<$ атрибут $>::= [visibility \mapsto <$ видимість $>, name \mapsto <$ номінативне$_{\text{д}}$ане $>]$

$<$ метод $>::= [visibility \mapsto <$ видимість $>,$
$modif \mapsto <$ модифікатор $>, name \mapsto <$ номінативне $-$ дане $>]$
$<$ властивість $>= [visibility \mapsto <$ видимість $>,$
$get \mapsto [visibility \mapsto <$ видимість $>, prop \mapsto <$ номінативна $-$ функція $>],$
$set \mapsto [visibility \mapsto <$ видимість $>, prop \mapsto <$ номінативна $-$ функція $>]]$
$<$ видимість $> \in \{0, 1, 2, 3\}$
//позначають модифікатори public, protected, private, internal
$<$ модифікатор $> \in \{0, 1\}$ //позначають модифікатори virtual, override

Таке дане складається з композиції типів даних визначених в [1, 3, 4], таким чином, очевидно, що вище представлені об'єкти можна задати в класі номінативних даних.

Запропоноване представлення об'єктів підтримує такі основні властивості ООП, як наслідування, інкапсуляцію та може підтримувати поліморфізм.

Зважаючи на можливість представлення об'єкту за допомогою номінативних даних можна розширити запропонований в [1] прототип аксіоматичної системи специфікацій програм над номінативними даними (NDSL), що побудований на основі композиційно-номінативного методу уточнення поняття програми [1, 2, 3, 4], аксіоматичної системи специфікацій програм над номінативними даними [1, 3], секвенційного числення над номінативними даними [1, 6], та бере за основу синтаксичну нотацію мови специфікацій Z [7]. Для такого розширення (OBJ-NDSL) візьмемо за основу запропоноване представлення об'єктів, синтаксичну нотацію мови Object-Z [8] та розширимо мову NDSL поняттям класу.

OBJ-NDSL специфікація складається з формального математичного тексту та інтуїтивного неофіційного пояснення (у вигляді коментарів). Формальний текст складається з послідовності параграфів, що представляють схеми-класи, схеми, глобальні змінні, базові типи специфікацій. Кожен параграф базується на попередніх та може визначати один чи більше імен схем-класів, схем, основних типів, глобальних змінних та глобальних констант. Він може використовувати імена, визначені в інших параграфах.

Існує кілька видів параграфів. Основні визначення типу, схема стану (обов'язково присутня та єдина), схема ініціалізації (обов'язково присутня та єдина), визначення схеми, операції, предикати та інше.

Визначення базових типів представляє один, чи кілька основних типів. Імена, що використовуються не повинні мати попередньої глобальної декларації. Область їх дії простягається від визначення до кінця специфікації, їх імена стають частиною глобального словника основних типів.

Визначення схеми включає вид схеми (схема-клас, схема стану, схема ініціалізації, операція), ім'я, декларативну та аксіоматичну частини. При цьому, декларативна частина складається з набору декларацій змінних з типами, що є глобальними типами, або побудовані за допомогою конструкторів типів (декартовий добуток, множина, список, номінативна множина, клас). Аксіоматична частина складається з набору Δ_0-предикатів [1, 3].

Визначення схеми-класу включає ім'я, частину наслідування, декларативну та предикатну частини, схему ініціалізації об'єкта (конструктор), схема-деструктор, схеми-методи. Частина наслідування може містити один параграф з об'єктом відповідного базового класу, та (або) декілька реалізацій інтерфейсів. Декларативна частина може містити public, protected, private та internal параграфи, що містять відповідні набори декларацій атрибутів з типами, що є глобальними типами, або побудовані за допомогою конструкторів типів (декартовий добуток, множина, список, номінативна множина, клас). Предикатна (аксіоматична) частина складається з набору Δ_0-предикатів [1, 3]. Схема ініціалізації та схема деструктор є відповідно конструкторами та деструкторами об'єкта, схеми-методи та схеми-властивості — є звичайними схемами виду операція.

Список предикатів може з'явитися і як окремий параграф. У цьому випадку, він визначає властивості специфікації виконання яких потрібно перевірити. При цьому використовуються глобальні змінні.

Висновки

Базуючись на композиційно-номінативному методі уточнення поняття програми [1, 3], аксіоматичній системі специфікацій програм над номінативними даними [1, 2, 3], секвенційному численні композиційно-номінативних логік [1, 5] та мові Object-Z [8] побудовано прототип аксіоматичної системи специфікацій програм над номінативними даними (OBJ-NDSL). Система OBJ-NDSL дозволяє доводити певні властивості програм. Тим самим показано, що КНП може ефективно використовуватися для побудови аксіоматичної системи специфікацій програм (в тому числі і об'єктно-орієнтованих) над номінативними даними, що достатньо адекватно відповідає проблемам програмування.

Список літератури

1. *Омельчук Л. Л.* Аксіоматичні системи специфікацій програм над номінативними даними: Дисертація к.ф.-м.н.: 01.05.01. — К., 2007. — 142 с.

Omelchuk, L. L. 2007. Axiomatic Systems of Specifications of Programs over Nominative Data. *Ph. D. Thesis: 01.05.01. (Ukrainian)*, Kyiv, 142 p.

2. *Омельчук Л. Л.* Прототип реалізації аксіоматичної системиспецифікацій програм над метаномінативними даними. Theoretical and applied aspects of program systems development (TAAPSD'2007). — Abstracts. — Berdyansk, Ukraine, 2007. P. 107–113.

   Omelchuk, L. L. 2007. A prototype of axiomatic system of specifications of programs over metanominative data. *Theoretical and applied aspects of program systems development (Ukrainian)*, Abstracts. Berdyansk, Ukraine, P. 107–113.

3. *Никитченко Н. С.* Композиционно–номинативный подход к уточнению понятия программы // Проблемы программирования. — 1999. — №. 1. — С. 16–31.

   Nikitchenko, N. S. 1999. Composite-nominative approach to updating the concept of the program *Problems in programming (Ukrainian)*, 1, pp. 16–31.

4. *Редько В. Н.* Основания композиционного программирования // Программирование. — 1979. — № 3. — С. 3–13.

   Redko, V. N. 1979. The grounds of the composite programming. *Programming (Russia)*, Kyiv, 3, pp. 3–13.

5. *Grady Booch.* Object-Oriented Analysis and Design with Applications (3rd Edition)*Addison Wesley Longman Publishing Co., Inc. Redwood City, CA*, USA 2004 ISBN:020189551X. p. 534.

6. *Нікітченко М. С., Шкільняк С. С.* Математична логіка. Додаткові розділи: Навчальний посібник. — Київ.: Видавничо-поліграфічний центр "Київський університет", 2004. — 77 с.

   Nikitchenko, M. S. and Shkilnyak, S. S. 2004. Logic. More sections: Tutorial*Publishing center "Kyiv University" (Ukrainian)*, Kyiv, 77 p.

7. *J. M./;Spivey* 1998. The Z Notation: A Reference Manual.–Oriel College, Oxford, OX1 4EW, England.

8. *Graeme Smith* 2000. The Object-Z Specification Language. — Advances in Formal Methods Series, Kluwer Academic Publishers.

Статья поступила в редакцию 24.09.2013