УДК 510.57+004.424.36 DOI: 10.31673/2412-9070.2022.055255

О. І. ПРОВОТАР, доктор фіз.-мат. наук, професор;
О. П. ІЛЬКУН, аспірант,
Київський національний університет імені Тараса Шевченка

ВИКОРИСТАННЯ КАТЕГОРІЙ ЯК АБСТРАКТНИХ СТРУКТУР ДЛЯ ФОРМАЛІЗАЦІЇ МОДЕЛЕЙ ОБЧИСЛЕНЬ

Проаналізовано різноманітні формальні моделі обчислень, зокрема рекурсивні функції, нечіткі моделі, а також категорійні моделі. Ці моделі підлягають дослідженню з метою виявлення способів виконання обчислень на їх основі. У контексті кожної з цих моделей уводиться концепція обчислюваності та базових арифметичних операцій, що дає змогу поглибити розуміння їх структур та механізмів. Основний висновок дослідження полягає в тому, що розглядувані формальні моделі обчислень можуть бути модельовані в контексті різних категорій. Такий підхід відкриває шлях для побудови абстрактної теорії обчислюваності та відповідних мов програмування на категорійній основі. Цей метод має потенціал для створення універсальної мови програмування, яка б підтримувала опис задач із різних предметних галузей. Реалізувати підхід є можливим інтерпретацією цих задач у відповідних категоріях і використанням універсального категорійного апарату для їх розв'язання. Варте особливої уваги те, що така мова має бути націлена на вирішення наукових задач, що відображає її потенційну цінність у сфері наукових досліджень. Додатково це дослідження може стати основою для подальших розробок стосовно розширення можливостей формальних моделей обчислень.

Ключові слова: формальні моделі обчислень; нечітка множина; категорія; рекурсивна функція.

Вступ

Проблемам розроблення категорійних моделей обчислень та їх застосуванню в комп’ютерних науках та програмуванні присвячені праці таких вчених, як G. Rosolini, M. Barr, C. Wells, G. Kelly, R. Milner, R. Goldblatt та ін. Але питання застосування теорії категорій в комп’ютерних науках та програмуванні набуло актуальності лише кілька років тому. Передусім це пов’язано з проблемою універсалізації інструментарію для розроблення та дослідження програмних систем.

Категорія — це надзвичайно просте поняття [1; 2]. Вона формується з об’єктів і зв’язків, які йдуть між ними. Але суть категорії полягає в композиції, яка водночас є суттю функціонального програмування [3]. Ще однією особливістю є те, що об’єкти категорії вивчаються не через їхні внутрішні характеристики, а дослідженням того, як вони себе поводять щодо інших об’єктів категорії.

Отже, якщо вважати операцію композиції та принцип невтручання у внутрішню структуру об’єкта парадигмами програмування, то зазначені властивості категорій дають змогу дійти висновку про те, що моделі як програмних систем, так і їх складових можуть бути описані на категорійній мові [4]. Це дасть можливість, по-перше, досліджувати складні програмні системи сучасними засобами категорійної математики [5] і, по-друге, створити мову програмування, яка уможливить розроблення алгоритмів і програм із різних предметних галузей. Це буде відбуватися вибором категорій відповідної структури для кожної сфери аналогічно тому, як програми працюють із різними типами даних.

Зрозуміло також, що будь-які моделі обчислень можна будувати в межах теорій, де внутрішніми засобами яких визначаються поняття числа, логіка та арифметичні операції над числами. Така подібність цих теорій та можливість описати їх в категорійних термінах дає змогу звести ці теорії за допомогою природного перетворення до категорії з натурально-числовим об’єктом (НЧО) [6]. Тому питання щодо дослідження довільних властивостей об’єктів цих теорій може бути зведене до дослідження властивостей відповідних (відносно природного перетворення) об’єктів категорії з НЧО. Наприклад, відома задача пошуку мінімального елемента в множині M (категорія Set) зводиться до задачі пошуку початкового об’єкта в категорії передпорядку $(M,\subseteq)$, або пошуку «мінімального елемента» серед категорійних моделей елементів множини M у категорії з НЧО.

Основна частина

Розглянемо теорію рекурсивних функцій. Демонстрація алгоритмічної обчислюваності функцій через їх додавання до групи частково-рекурсивних функцій часто є викликом, за винятком найбільш простих функцій. Також у кожному конкретному випадку цей процес потребує створення математичної моделі функції у формі терму, що виконує обчислювані операції на базових функціях. Як зазначено в [1], до базових функцій належать найпростіші функції $o(x)=0, s(x)$ та функції-селектори $(x_1,\ldots,x_n)=x_m$, де $n \ge m \ge 1$.

Оператори суперпозиції S^{n+1}, примітивної рекурсії R, мінімізації M виступають як основні обчислювані операції.

Отже, функція вважається алгоритмічно обчислюваною, якщо існує операторний терм, який визначає її.

Постає логічне питання стосовно використання специфічних алгоритмів у доведенні засадничих висновків теорії рекурсивних функцій. Для досягнення цього потрібно точно визначити основні алгоритмічні конструкції та переформулювати тезу Чорча для більш обмежених множин алгоритмічно обчислюваних функцій.

Теза 1. Множина функцій, обчислюваних універсальними алгоритмами без використання оператора «while…do», збігається з множиною примітивно-рекурсивних функцій.

Теза 2. Множина функцій, обчислюваних універсальними алгоритмами, збігається із множиною рекурсивних функцій.

Теза 3. Множина функцій, обчислюваних за допомогою довільних алгоритмів, збігається з множиною частково-рекурсивних функцій.

Отже, з одного боку, ми оперуємо алгебраїчним термом як формальною моделлю обчислень, що представляє алгоритм, з другого — ми маємо синтаксичну структуру, яка також виступає як формальна модель обчислень, але є ближчою до мов програмування. Інакше кажучи, обмежуючись, наприклад, набором натуральних чисел та впроваджуючи базові логічні та арифметичні операції, обчислювані в межах традиційної обчислювальної парадигми, ми можемо говорити про розроблення теорії обчислюваності в контексті деякого конкретного формалізму.

Розглянемо нечіткі моделі. Як відомо, в бінарній логіці висновок про істинність певного твердження або висловлювання базується на істинності його тверджень. Такий процес виведення часто подається у вигляді схеми: складові твердження записуються над горизонтальною лінією, тоді як саме твердження, яке виводиться, розміщується під цією лінією.

У контексті нечіткої логіки [7; 8] застосовується узагальнене (або нечітке) правило виведення, відоме як *modus ponens*. Це правило описується за допомогою такої схеми виведення:

Умова	x це A'
Імплікація	Якщо x це A, то y це B
Висновок	y це B'

Тут A, $A'\subseteq X$ і B, $B'\subseteq$X відображають нечіткі множини, тоді як x, y є так званими лінгвістичними змінними, де їхніми значеннями виступають нечіткі висловлювання, використані на природній мові. Висновок, що випливає з нечіткого правила, асоціюється з певною нечіткою множиною B', яка визначається за допомогою композиції нечіткої множини A' та нечіткої імплікації $A\to B$, тобто

$$B' = A'\circ(A\to B).$$

Нечітка імплікація $A\to B$ може бути відображена через певне нечітке відношення $R\subseteq X\times Y$ з функцією належності $\mu_R(x,y)$. Саме з цієї причини функцію належності нечіткої множини B' можна подати в такому форматі:

$$\mu_{B'}(y) = \sup_{x\in X}\{\mu_{A'}(x)^T \cdot \mu_{A\to B}(x,y)\}.$$

Варто зазначити, що $\mu_{A\to B} = \mu_R(x,y)$.

У межах теорії нечітких систем існують нечіткі множини, які характеризуються на осі дійсних чисел та вирізняються нормальністю, опуклістю, а також неперервними функціями належності. Такий клас нечітких множин відомий як нечіткі числа.

Принцип розширення дає змогу виконувати основні арифметичні операції, зокрема додавання, віднімання, множення та ділення між двома нечіткими числами, позначеними як A_1, $A_2\subseteq R$. Як приклад, розглянемо випадок розрахунку суми двох нечітких чисел A_1 і A_2, що позначається як $A_1\oplus A_2 = B$. Обчислення виконуються за такою формулою:

$$\mu_B(y) = \sup_{\substack{x_1\,x_2\\ y = x_1'+x_2}} \min\{\mu_{A_1}(x_1),\mu_{A_2}(x_2)\}.$$

У цьому контексті натуральні числа можуть бути подані як нечіткі трикутні числа. Важливо зауважити, що таке подання натуральних чисел як нечітких трикутних чисел дає можливість зберегти основні властивості натуральних чисел під час виконання арифметичних операцій.

Алгоритм для обчислення довільних функцій у множині натуральних чисел можна записати в такий спосіб:

function $s(x, y)$
  begin
    if x is (a_1, x, b_1) and y is (a_2, y, b_2) then
      s′ is $(a_1+a_2, x+y, b_1+b_2)$
    $s = x+y$
  end

У цьому поданні x та y — натуральні числа, які моделюються через нечіткі трикутні числа відповідно (a_1, x, b_1) і (a_2, y, b_2) .

Розглянемо теорію категорій. Нехай $\Im$ — категорія [7], що має кінцевий об'єкт 1 і класифікатор підоб'єктів, позначений як Ω. Категорія, яка декартово замкнена і має класифікатор підоб'єктів, називається топосом.

З огляду на те, що діапазоном значень довільної класичної логічної функції є двоелементна множина $2 = \{0,1\}$, можна досліджувати таку функцію як характеристичну функцію відповідної підмножини її діапазону визначення. В іншому контексті це можна розглядати як характеристичну функцію образу певного мономорфізму. Такий підхід дає змогу обчислювати логічні функції через діаграми в будь-якому топосі.

Для прикладу, функцію кон'юнкції $\wedge$, що відображає $\Omega \times \Omega$ у Ω, можна визначити за декартовим квадратом, зображеним на рис. 1, тобто має місце <true, true>: $1 \to \Omega$.

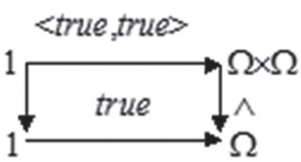

Рис. 1. Декартовий квадрат функції кон'юнкції ∧

Інші логічні операції, зокрема диз'юнкція (∨), заперечення (¬) та імплікація (⇒), також визначаються через відповідні декартові квадрати. Отже, декартові квадрати відіграють ключову роль у конструюванні логічних функцій у межах цієї структури.

Справедливою буде наведена після цього теорема.

Теорема. Поведінку операцій ∨, ¬, ⇒, ∧ може бути узагальнено поданою далі таблицею істинності.

Таблиця істинності для типових операцій

¬ true false	∧ true false	⇒ true false	∨ true false
false true	true true false	true true false	true true true
	false false false	false true true	false true false

Натурально-числовий об'єкт (НЧО) — це деякий об'єкт N разом з його визначеною парою морфізмів $1 \xrightarrow{O} N \xrightarrow{S} N$, таких, що для довільного об'єкта a і морфізмів $1 \xrightarrow{x} a \xrightarrow{f} a$ існує тільки єдиний морфізм h: $N \to a$, для якого наступна діаграма комутує (рис. 2).

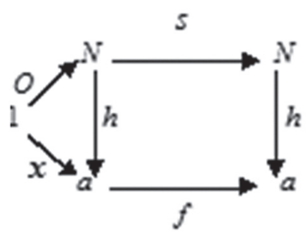

Рис. 2. Діаграма для НЧО

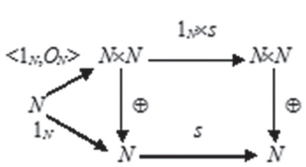

Рис. 3. Діаграма для операції додавання

За таким підходом натуральне число можна визначити як морфізм n: $1 \to N$ у топосах, що містять натурально-числовий об'єкт.

У такому топосі можна визначити різні арифметичні операції. Наприклад, операція додавання може бути подана як морфізм, в якому комутує діаграма (рис. 3).

Обчислення суми двох натуральних чисел $m + 0$ зводиться до такої композиції:

$$m \circ \langle 1_N, O_N \rangle \circ \oplus = m \circ 1_N = m.$$

Однією з переваг цього підходу до створення формальних моделей алгоритмів є те, що натурально-числовий об'єкт може бути інтерпретований досить широко, від класичних моделей у категорії Set до категорії доведень, де об'єктами є формули, а морфізми — формальні виведення одних формул з інших у межах певної системи дедуктивного виведення.

Висновки

З огляду на те, що теорії нечітких множин та частково-рекурсивних функцій можуть бути описані як категорії [10], розглянуті моделі обчислень можуть бути об'єктом аналізу в контексті цих категорій.

Це надає можливість для побудови абстрактної теорії обчислюваності та відповідних мов програмування, заснованих на категорійній основі. Розроблення мови програмування Haskell також відображає перший крок у цьому напрямку, наголошуючи на важливості та потенційних перевагах цього підходу. Можна розглянути Haskell [11] як експериментальну площину для розроблення категорійних концепцій у мовах програмування, що відкриває нові горизонти для подальших досліджень у цій сфері.

Список використаної літератури

*1. **Barr M., Wells C.** Category Theory for Computing Science // Reprints in Theory and Applications of Categories. 2012. No. 22.*

*2. **Rosolini G.** Categories and effective computation. Lecture Notes in Computer Science // Category Theory and Computer Science. Springer-Verlag, 1987. Vol. 283.*

*3. **Curien P.-L.** Categorical Combinators, Sequential Algorithms and Functional Programming // Research Notes in Theoretical Computer Science, Pitman, 1986.*

*4. **Hagino T.** A categorical programming language. University of Edinburgh, 1987. 140 p.*

*5. **Sergienko I. V., Parasyuk I. N., Provotar A. I.** On the application of categorical methods in Computer Science // KISA, 2000. № 4.*

*6. **Goldblatt R.** Topoi. The categorial analysis of logic // North-Holland Publishing Company, 1979. P. 486.*

*7. **Zadeh L.** Fuzzy sets // Inf. Control. 1965. P. 338–353.*

*8. **Ross T.** Fuzzy Logic with Engineering Applications. John Wiley & Sons, Ltd, 2004. Chichester.*

*9. **Carol L. Walker.** Categories of fuzzy sets // Soft Computing. 2004. Vol. 8, №4. P. 299–304.*

*10. **Barr M.** Fuzzy sets and topos theory // Canadian Math. Bull. 1986. Vol. 24. P. 501–508.*

*11. **Hudak P.** The Haskell School of Expression – Learning Functional Programming through Multimedia. New York: Cambridge University Press, 2000.*

O. I. Provotar, O. P. Ilkun

USING CATEGORIES AS ABSTRACT STRUCTURES TO FORMALIZE COMPUTATIONAL MODELS

The paper provides an analysis of various formal models of computation, including recursive functions, fuzzy models, and categorical models. These models are subject to research in order to identify ways of performing calculations based on them. In the context of each of these models, the concept of computability and basic arithmetic operations is introduced, which allows to deepen the understanding of their structures and mechanisms. The main conclusion of the study is that the considered formal models of calculations can be simulated in the context of different categories. This approach paves the way for the construction of an abstract theory of computability and corresponding programming languages on a categorical basis. This approach has the potential to create a universal programming language that would support the description of tasks from different subject areas. It is possible to implement the approach by interpreting these problems in the appropriate categories and using a universal category apparatus for their solution. It deserves special attention that such a language should be aimed at solving scientific problems, which reflects its potential value in the field of scientific research. In addition, this study can lay the foundation for further developments in expanding the capabilities of formal computational models. Given the presence of categories including fuzzy sets, fuzzy functions and partially recursive functions, the considered calculation models can be the object of analysis in the context of these categories. This provides an opportunity to construct an abstract theory of computability and corresponding programming languages based on a categorical basis. The development of the Haskell programming language also represents a first step in this direction, highlighting the importance and potential benefits of this approach. It is possible to consider Haskell as an experimental plane for the development of categorical concepts in programming languages, which opens new horizons for further research in this area.

Keywords: formal models of calculations; fuzzy set; category; recursive function.