УДК 681.3.06

ПУСТОВАРОВ В.І.

СУПРОВОДЖЕННЯ КОНТРОЛЮ ВІДПОВІДНОСТІ ПРОГРАМ І МОДЕЛЕЙ ФОРМАЛЬНИМ СПЕЦИФІКАЦІЯМ ЗАДАЧ

Запропоновано механізми автоматизації доказового об'єктно-орієнтованого програмування задач за специфікаціями на базі контролю семантичної коректності взаємних перетворень кодів програм та специфікацій. Через ці механізми обгрунтовано використання аналітичного типу даних, який можна вбудувати до мов і систем програмування. Запропонований підхід дозволяє автоматизувати доказові перетворення кодів і специфікацій в системах автоматизації програмування і моделювання для задач формальної верифікації та синтезу кодів.

The proofing mechanisms of object-oriented programming are proposed for tasks with mutual transformations of program codes and formal specifications. Those mechanisms were used for justification to analytical data type which maybe built into programming languages and systems. Proposed approach allows automated proofing transformations of program codes and specifications in programming and simulation system for tasks of formal verification and code synthesis.

Однією з цілей систем автоматизації підготовки програм і моделей обладнання [3, 4, 7] є визначення підтримки і супроводження відповідності програм розв'язання задач формальним і формалізованим специфікаціям. Хоча сучасні системи програмування і моделювання (СПМ) не включають комплексів засобів достатньо повної перевірки відповідності кодів специфікаціям, комп'ютерна реалізація взаємних перетворень кодів і специфікацій [4, 6, 7] може дозволити організувати контроль і супроводження відповідності виконавчих кодів і специфікацій в процесі реалізації різноманітних програмних проектів. Створення автоматизованих засобів формальних перетворень дозволить скоротити витрати праці на створення нових і модифікацію попередніх проектів.

Перевірка коректності текстів програм і моделей, а також результатів формальних математичних перетворень, досі не автоматизовані, хоча створено ряд схем формальних специфікацій [2, 6, 7], які принципово можуть зберігатися на рівні комп'ютерних мов. Наприклад, мова Z-специфікацій [7] одержала ґрунтовну підтримку для задач управління базами даних. Механізми формальних аналітичних перетворень, стали основою математичних, а точніше, алгебраїчних методів розв'язання задач і побудови теорій типів даних [4, 6, 7] аж до взаємних перетворень математичних виразів і виконавчих кодів комп'ютера. Саме такі перетворення над моделями використовуються для формального доведення вручну тверджень відносно математичних формул і програмних кодів [4, 6, 7].

З самого початку формалізації та автоматизації задач аналітичних перетворень для полегшення впорядкування, систематизації та порівняння аналітичних виразів в логіці почали використовувати спеціальні форми [2, 4, 6], які сприяють зменшенню обсягів правил і таблиць перетворень при формальному синтезі та верифікації кодів. Найбільш загальними є графові схеми внутрішнього подання з підграфами G_{lj}, які поєднують підлеглі та рекурсивно зв'язані вузли включно з термінальними вузлами лексем G_{0j} та використовуються в різних трансляторах [1, 5]. Найбільш зручні спеціальні форми загальних конструкцій виразів і операторів комп'ютерних мов і мов специфікацій базуються на базовому порядку виконання операцій, основаному на їх передуваннях або пріоритетах. Ці форми є відображенням повного різноманіття конструкцій мов і відповідають таким вимогам:

- впорядкування складних структурованих конструкцій за послідовністю виконання елементів конструкції відповідно графового подання;
- визначення загального відношення порядку $\mathbf{S}_s(G_{lj})$ для всіх елементарних та структурованих або складених однотипних та однойменних об'єктів G_{lj} спрямованих графів (СГ) текстів комп'ютерних мов та мов специфікацій з розміщенням більш складних об'єктів після простих;
- впорядкування однорідних або подібних елементів конструкцій згідно з відношенням порядку $\mathbf{S}_s$ відносно кореневого вузла підграфа G_{lj} для конструкцій зі списками або операціями, повторюваними на одному рівні виразу;
- визначення спеціального порядку $\mathbf{S}_c(G_{lj})$ типів даних за їх структурною складністю та різноманіттям і впорядкованістю значень в доменах;
- впорядкування однорідних елементів конструкцій, що припускають довільну послідовність обробки або перестановку в списках аргументів та специфікацій даних, в порядку зменшення важливості їх змістовної обробки;
- використання єдиної системи впорядкування, нумерації і кодування однорідних та однотипних елементів конструкцій, в тому числі, операцій з можливістю перестановки аргументів (впровадження такої системи може зробити подання надлишковим, але повинно спростити порівняння за наявності спеціального механізму прискорення порівнянь);
- можливість побудови спеціально обмежених спеціальних форм для будь-яких підмножин операцій з високим пріоритетом.

Традиційна методика доказових перетворень та синтезу програм за специфікаціями [7] базується на операціях тотожних та імплікативних перетворень, які можна розглядати як узагальнення логічних операцій рівнозначності, нерівнозначності та імплікації [4], що використовується при аналітичних доведеннях. Щоб полегшити розрізнення дій перетворення в ланцюжках доведень від операцій відношення рівності виразів,

що операцію тотожності (тотожної рівності, рівності, визначення рівняння, рівнозначності) будемо позначати у формі $x \equiv y$ на відміну від $x = y$, використаного в [4], що співпадає з рівністю, а операнди називатимемо лівою частиною та правою частиною. За Хехнером [4] імплікацією називають перетворення, що визначає істинність слідування довільного математичного виразу y з виразу такого ж типу x, і записують її в формі $x \Rightarrow y$ або $y \Leftarrow x$. При цьому операнд x називають антецедентом мети (попередником), а y – консеквентом (наслідком). Далі називатимемо ці операції правосторонньою та лівосторонньою імплікаціями.

Група ключових трансформаційних дій утворює послідовні ланцюжки доказових перетворень, що зберігаються в протоколах доведень. Тому їх необхідно включити до списку основних операцій внутрішнього подання процесів перетворень, показані на Табл.1 в порівнянні з відповідними логічними операціями. Семантика цієї групи дій задається відношеннями тотожності та імплікативності між попередньою S_{j-1} та наступною S_j специфікацією або виконавчим кодом в ланцюжку перетворень.

Табл. 1. Типові операції перетворень специфікацій, моделей, програм та скриптів запитів в порівнянні з відповідними логічними операціями

Типи операцій	**Мови або процеси**	**Варіанти відповідності**
Y1. Двостороння еквівалентність або тотожність: ≡	Перетворення специфікацій і кодів	Доцільність аналізу другого аргументу після першого
Y2. Лівостороння імплікація: ⇐	Перетворення специфікацій і кодів	Необхідність аналізу першого аргументу після другого
Y3. Правостороння імплікація: ⇒	Перетворення специфікацій і кодів	Необхідність аналізу другого аргументу після першого
X1. Рівнозначність: ≡ ~^	Специфікацій; моделювання і програмування	Доцільність обчислення другого аргументу після першого
X2. Лівостороння імплікація: ⇐ :-	Специфікацій і логічного програмування	Необхідність обчислення першого аргументу після другого
? X3. Правостороння імплікація: ⇒	? Специфікацій	? Необхідність обчислення другого аргументу після першого

В протоколах ланцюжків перетворень виконуються і фіксуються результати операцій еквівалентних перетворень ‘≡’ і операцій лівосторонньої ‘⇐’ та правосторонньої ‘⇒’ імплікації, в яких стрілка вказує на менш загальну специфікацію S_j. Тобто на всій області визначення елементів та параметрів типу, або аргументів та параметрів фрагмента коду, ця специфікація S_j завжди має ті самі значення істинності, що відповідають обмеженням адекватності виконаних перетворень, але за межами цієї області S_j може не відповідати цим обмеженням.

В процесі доведення тотожність та імплікації реалізуються як композиції відповідних операцій для даних за типами з відокремленими елемен-

тами функціональних перетворень. При цьому ланцюжки тотожностей виконують тотожні перетворення, а ланцюжки односпрямованих імплікацій разом з тотожними перетвореннями – відповідні імплікативні доведення. В наведених нижче формулах збільшені знаки ‘≡’, ‘⇐’ і ‘⇒’ визначають тотожність та імплікативність ланцюжків перетворень і мають нижчий пріоритет, ніж звичайні логічні операції з такими ж назвами, що можуть зустрітися в специфікаціях і виконавчих кодах.

$$S_0 \equiv S_1 \equiv \ldots \equiv S_j \quad \equiv \quad S_0 \equiv S_j \qquad (1)$$

$$S_0 \equiv S_1 \Leftarrow \ldots \equiv S_j \quad \equiv \quad S_0 \Leftarrow S_j \qquad (2)$$

$$S_0 \equiv S_1 \Rightarrow \ldots \equiv S_j \quad \equiv \quad S_0 \Rightarrow S_j \qquad (3)$$

$S_0 \equiv S_1 \Rightarrow \ldots \Leftarrow S_j$ або $S_0 \equiv S_1 \Leftarrow \ldots \Rightarrow S_j$ – ланцюжки, що не мають сенсу.

Інша група ключових дій доказових перетворень протоколює тотожні перетворення специфікацій та їх перетворення імплікаціями на коди програм у комп’ютерній мові або в зворотному напрямку. Таблиці таких перетворень для специфікацій, кодів і тверджень відносно елементарних та складних об’єктів включають з одного боку, специфікації даних та зв’язків і обмежень, а з іншого боку відповідні коди виконавчих операторів та специфікації даних.

Для скорочення витрат на виконання доведень та зберігання їх результатів обґрунтована доцільність використання даних, інкапсульованих в об’єктно-орієнтованих типах. Якщо організувати традиційне успадкування разом з обмеженням множин значень окремих полів додержанням і визначити обмеження на правила композиції типів, то виникає можливість успадкування доведень відповідності типів об’єктів їх специфікаціям. Аксіоматичний базис доказових аналітичних перетворень повинен включати типи, що відповідають числовим математичним абстракціям, механізми для доведення відповідності та впорядкованості домену значень і комплексу методів класу об’єктів специфікаціям відповідного типу [4, 6, 7]. Це створює можливість поширення доведення коректних методів обробки загальних та математичних типів даних на похідні типи даних з обмеженими та успадкованими множинами значень їх елементів.

Можливі базові шляхи для первинних імплікативних доказових перетворень наведено в Табл. 2. Напрямки комплексування типів окремих полів при створенні класу визначаються обмеженнями: 1) односпрямованої впорядкованості всіх полів комплексного типу; 2) однакових наборів операцій батьківських типів; 3) доведеними наборами операцій похідного класу пересіченням наборів операцій батьківських типів. Для показаних напрямків можна довести імлікативність множин даних та коректність всіх методів типу або класу (абстрактні математичні типи, які не мають комп’ютерної реалізації виділено сірим кольором).

Табл. 2. Напрямки перетворення типів зі скороченням кардинальних чисел доменів за правосторонньою імплікацією $T_a \Rightarrow T_i$

Континуальні ⇒	Цілі необмежені ⇒	Натуральні необмежені ⇒	Необмежені без знака ⇒	Перенумеровані (кодовані)
Континуальні ⇒	Обмежені з плаваючою точкою ⇒	Цілі обмежені ⇒	Обмежені без знака ⇒	Перенумеровані (кодовані)
Континуальні ⇒	Обмежені з плаваючою точкою ⇒	З обмеженням порядку плаваючої точки ⇒	З обмеженням порядку і мантиси ⇒	Перенумеровані (кодовані)
Дискретні з довільними доменами ⇒	Дискретні з обмеженими доменами ⇒	Дискретні з впорядкованими доменами ⇒	Впорядковано кодовані ⇒	Перенумеровані (кодовані)

Для визначення діапазону числових або впорядкованих множинних даних в позначеннях [4] x_{imin},.. x_{imax}: T_i необхідно задати нижню x_{imin} і верхню x_{imax} межі значень конкретизованого типу T_i. Ці межі повинні гарантувати або імплікативно доводити адекватну роботу всіх методів (операцій і функцій) типу T_i відносно типу T_a в рамках обмежень, визначених аналітиком. Тобто, якщо для цілих числових типів границі їх діапазонів відповідають відношенням: $x_{imin} \le x_{amin}$ та $x_{amax} \le x_{imax}$, всі сусідні значення x_i відрізняються на 1, а для всіх операцій та функцій, зв'язаних з цими типами, можна довести за [4, 6, 7], що $f_a \Rightarrow f_i$, то $T_a \Rightarrow T_i$, і коректність типу T_i слідує з коректності типу T_a. Тобто, всі специфікації, операції, дії, функції, оператори, підпрограми, тощо, коректні для типу T_a, що відрізняються лише заміною типу даних на тип T_i, також будуть коректними.

Континуальні або дійсні числові типи повинні забезпечувати задану точність наближеного відтворення $x_i \approx x_a$ даних так, щоб $|x_a: T_a - x_i: T_i| \le \varepsilon$ та необхідний діапазон припустимих значень, а також додержання таких самих або не гірших ніж задані в специфікаціях точності та діапазонів значень при виконанні базових операцій над даними відповідного типу. В таких випадках конкретизовані специфікації доводяться правосторонньою імплікацією зі специфікацій базових абстрактно математичних типів з додаванням умов точності.

З позицій об'єктно-орієнтованого програмування будь-які класи об'єктів C повинні бути реалізовані, як структурне об'єднання полів даних F базового типу T з комплексом методів $f=(f_o, f_f, f_s, f_p)$, до яких відносяться операції (оператори [1]) f_o, функції f_f (як математичні, так і програмні), підпрограми f_s, предикати f_p (в тому числі, предикати обмежень і відповідності специфікаціям). Розширення такого типу або класу за допомогою позначень визначення структур [4] може бути записане, як $C = F \rightarrow T \mid f_1$

$\rightarrow T\,|\ldots|\,f_k \rightarrow T\,|$, де k – кількість методів в типі. Відповідність первинних методів розширених типів або класів їх специфікаціям доводиться базовими методами доведення, що і було зроблено Е.Хехнером для базових абстрактних математичних типів [4].

Для обґрунтування подальших доказових перетворень важливо довести набір теорем для переходу по ієрархії типів від абстрактно математичних типів до поширених в сучасних мовах програмування комп'ютерних типів або класів даних:

1. ***Теорема обмеження доменів значень типів даних***, яка визначає імпликативне доведення коректності типів зі скороченими доменами з коректності базового типу при використанні такого самого набору методів.

2. ***Теорема скорочення кількості полів структурованих типів даних***, яка визначає імпликативне доведення коректності з вилученням наборів полів з коректності базового типу при незмінному наборі методів.

3. ***Теорема побудови композиційних методів в рамках одного типа***, яка визначає умови імпликативного доведення коректності композиційних методів з коректності методів базового типа.

4. ***Теорема побудови композиційних типів з окремих полів***, яка визначає імпликативне або тотожне доведення коректності композиційних типів з коректності базових типів при незмінному наборі методів.

5. ***Теорема доведення функціональних розширень типів***, яка визначає умови імпликативного доведення коректності композиційних методів з методів базового типа та операцій розширення, взятих з зовнішніх типів.

Для спрощення доказів важливо, щоб в обмеженому типі T_{i} використовувались ті самі значення, що і в базовому типі T_{a}, тоді в типі T_{i} залишаться справедливими всі відношення порядку. В Табл. 3 зібрані основні алгебраїчні методи ручної мінімізації, формальної верифікації та синтезу виразів і кодів засобів розв'язання задач (ЗРЗ) і визначено основні шляхи, правила і шаблони різних функцій та операцій $\text{F}_{\text{a}}(G_{\text{Sln}}, G_{a\text{C}\bullet})$ семантичних перетворень. Аналітичні дані специфікацій G_{Sln} і правила перетворень $G_{\text{C}\bullet lj}$ задаються кореневими вузлами відповідних СГ.

Базовий порядок обробки встановлюється за правилами управління Cnt, поданими через кореневі вузли $G_{\text{C}\bullet lj}$ графів спрямованих перетворень відповідно з послідовністю їх використання в ланцюжках перетворень. Спочатку вони найчастіше перетворюють вхідні вирази на проміжну внутрішню форму Inout, а потім – на заключну цільову форму перетворення.

Табл. 3. Базові типові шляхи семантичних перетворень специфікацій, моделей та програм

Тип	Спосіб перетворення	Внутрішня інформаційна база	Ланцюжки
Мінімізація подання кодів	1. Перетворення фрагмента на спеціальну форму ⇒ виключення повторень ⇒ перетворення на мінімальну форму	Cnt: Правила перетворення: 1.1.G_{Cntlj} – вхідних виразів G_{In} на часткову спеціальну форму G_{Int}; 1.2.G_{Cmnlj} – обмеженої спеціальної форми G_{Int} на мінімальну G_{Mn}. Inout: поточне внутрішнє подання $G_{In} \Rightarrow G_{Int} \Rightarrow G_{Mn}$;	$G_{In} \Rightarrow$ $G_{Int}= F_{Cn}(G_{In}, G_{Cntlj}) \Rightarrow$ $G_{Mn}= F_{Mi}(G_{Int}, G_{Cmnlj})$
Мінімізація з наближеннями	2. Спосіб 1 ⇒ імплікативне перетворення мінімальної форми	Cnt: Правила перетворення способу 1 (1.1 і 1.2) з доданням: 1.3. Правил перетворення G_{Cmalj} мінімальної форми G_{Mn} на мінімальну наближену форму G_{Mna}. Inout: $G_{In} \Rightarrow G_{Int} \Rightarrow G_{Mn} \Rightarrow G_{Mna}$;	Ланцюжки способу 1 ⇒ $G_{Mna}=F_{Ma}(G_{Mn}, G_{Cmalj})$
Формальна верифікація	3. Формування виразів для кодів визначення цільових змінних ⇒ підстановка до специфікації ⇒ доведення істинності	Cnt: 3.1.Правила підстановки G_{Csslj} виразів програм G_{ECd} до специфікацій G_{Sp}; правила перетворення: 3.2.(1.1)G_{Cntlj} – результатів підстановки G_{Rs} на обмежену спеціальну форму G_{Int}; 3.3.(1.2)G_{Cmnlj} – часткової спеціальної форми G_{Int} на мінімальну G_{Mn} з результатом *true* Inout: $G_{Sp} \cup G_{ECd} \Rightarrow G_{Rs} \Rightarrow G_{Int} \Rightarrow$ *true*;	$G_{Sp} \cup G_{ECd} \Rightarrow$ $G_{Rs}=F_{Sb}(G_{ECd}, G_{Sp}, G_{Sbnlj}) \Rightarrow$ $G_{Int}=F_{Cn}(G_{Rs}, G_{Cntlj}) \Rightarrow$ $G_{Mn}= F_{Mi}(G_{Int}, G_{Cmnlj}) \equiv$ *true*
	4. Декомпозиція задачі ⇒ доведення відповідності окремих фрагментів ⇒ доведення відповідності композиції	Cnt: Правила: 4.1.Розширення G_{Gelj} специфікацій проміжними цілями; 4.2.G_{Dtslj} – декомпозиції специфікацій відносно проміжних цілей G_{Gin}; Доведення частин за способом 3. Inout: $G_{Sp} \cup G_{ECd} \Rightarrow G_{Sp} \cup G_{ECd} \cup G_{Gin} \Rightarrow G_{Rsn} \Rightarrow G_{Int} \Rightarrow$ *true*;	$G_{Sp} \Rightarrow G_{Gin}$; ⇒ $G_{Rsn}= F_{Dc}(G_{Sp} \cup \cup G_{ECd} \cup G_{Gin})$ $G_{Int}=F_{Sb}(G_{Rsn}, G_{Sp}, G_{Cntlj}) \Rightarrow$ $G_{Mn}= F_{Mi}(G_{Int}, G_{Cmnlj}) \equiv$ *true*
Формальний синтез	5. Декомпозиція специфікацій задачі на послідовність ⇒ синтез окремих фрагментів коду	Cnt: Правила декомпозиції 4.1 і 4.2. 5.1. Правила перетворення G_{Cvcnlj} специфікацій G_{Sp} на коди G_{ECdn}. Inout: $G_{Sp} \Rightarrow G_{Sp} \cup G_{Gin} \Rightarrow G_{Rsn} \Rightarrow G_{ECdn}$;	$G_{Sp} \Rightarrow G_{Gin}$; ⇒ $G_{Rsn}= F_{Dc}(G_{Sp} \cup \cup G_{ECd} \cup G_{Gin})$ $G_{ECdn}= F_{Cv}(G_{Rsn}, G_{Cvcnlj})$

	6. Пошук аналогів серед наявних ЗРЗ ⇒ визначення різниці типів і зв'язків специфікацій ⇒ синтез окремих фрагментів коду	Cnt: 6.1.Критерії близькості $G_{\text{Cnre}lj}$ специфікацій задач G_{Sp} до наявних ЗРЗ; правила $G_{\text{Sd}n}$ визначення різниці між специфікаціями і кодом ЗРЗ Inout: $G_{\text{Sp}} \Rightarrow G_{\text{Sp}} \cup G_{\text{ECp}} \Rightarrow G_{\text{Sp}} \cup G_{\text{ECp}} \cup \cup G_{\text{ECd}n} \Rightarrow G_{\text{ECp}} \cup G_{\text{ECd}n}$	$G_{\text{Sp}} \Rightarrow G_{\text{ECp}}=\text{R}_{\text{Cv}}(G_{\text{Sp}}, G_{\text{Cnre}lj})$; $G_{\text{Sd}n}= G_{\text{Sp}} \setminus G_{\text{ECp}}$; $G_{\text{ECd}n}= \text{F}_{\text{Cv}}(G_{\text{Sd}n}, G_{\text{Cvcn}lj})$

В задачах **мінімізації або оптимізації внутрішнього подання** вхідними даними є вузли СГ G_{In} внутрішнього подання, утвореного синтаксичним аналізатором. Потім формуються проміжні дані внутрішньої обмеженої спеціальної форми як вузли СГ G_{Int} за правилами $\text{F}_{\text{Cn}}(G_{\text{In}}, G_{\text{Cnt}lj})$ перетворення на цю форму. Вони також зберігають інформацію управління у формі вузлів СГ $G_{\text{Mn}}= \text{F}_{\text{Mi}}(G_{\text{Int}}, G_{\text{Cmn}lj})$ послідовність використання яких визначається режимами перетворення. Насамкінець вирази мінімальної наближеної форми в форматі вузлів СГ G_{Mna} за правилами перетворення обмеженої спеціальної форми на мінімальну $\text{F}_{\text{Ma}}(G_{\text{Mn}}, G_{\text{Cma}lj})$, які також зберігають інформацію управління у формі вузлів СГ $G_{\text{Cma}lj}$, послідовність використання яких визначається режимом наближеного перетворення.

Один з підходів до **формальної верифікації** методом аналітичних перетворень технологічно дуже схожий на базовий підхід до мінімізації внутрішнього подання, але використовує дещо іншу послідовність дій. Спочатку вузли СГ об'єднання $G_{\text{Sp}} \cup G_{\text{ECd}}$ специфікацій та виконавчих кодів цілеспрямовано реконфігуруються на подання формул результатів G_{Rs} виразами вхідних даними за правилами підстановки $\text{F}_{\text{Sb}}(G_{\text{In}}, G_{\text{Sp}}, G_{\text{Sbn}lj})$ виразів присвоювань та інших операторів виконавчої мови. Далі необхідно виконати співставлення або доведення відповідності результатів G_{Rs} специфікації за правилами $\text{F}_{\text{Cn}}(G_{\text{Int}}, G_{\text{Cnt}lj})$, сформувавши спеціальну форму подання кодів G_{Int} як проміжну. Після двох заключних кроків, повністю аналогічних останнім крокам мінімізації 1.2 та 1.3 (за табл. 3) ми повинні одержати або тотожне логічне значення *true*, або вираз, що визначає умови відповідності кодів специфікаціям.

Більш загальний підхід до **формальної специфікації** як і задачі **формального синтезу** відрізняються від попередніх задач семантичних перетворень необхідністю декомпозиції задач на частини, починаючи з текстів початкових специфікацій. Ці перетворення виконуються за допомогою загальних правил декомпозиції $G_{\text{Rs}n}=\text{F}_{\text{Dc}}(G_{\text{Sp}} \cup G_{\text{ECd}} \cup G_{\text{Gi}n})$ через визначення проміжних цільових змінних $G_{\text{Gi}n}$ або на базі пошуку ЗРЗ, найближчих за специфікацією, і розміщених в базі знань СПМ як обґрунтовані ЗРЗ. Для формального синтезу ключове значення має повнота правил еквівалент-

них перетворень елементів специфікації на коди $G_{ECdn} = F_{Cv}(G_{Sdn}, G_{Cvcnlj})$ [7], де всі ключові та функціональні елементи подаються у вигляді СГ.

Розв'язання задач семантичних перетворень можна полегшити за допомогою створення аналітичного типу даних. ***Аналітичний тип даних*** (як вбудований тип даних комп'ютерної мови) з комплексом параметрів, який визначає таблиці передувань і таблиці синтаксичних та семантичних правил, дозволяє одержати узагальнений механізм, доступний для використання на етапах проектування, аналізу та виконання операторів програм і математичних виразів в програмах і книжкових текстах. Поняття аналітичного типу визначає шлях реалізації узагальнення формальних і формалізованих математичних та комп'ютерних моделей, яке полегшує обробку виразів і всіх видів операторів програм на всіх етапах проектування.

Зовнішні властивості використання ***аналітичного типу*** спрощують використання і обробку **математичних виразів будь-яких типів і складності** на етапі проектування програмних об'єктів та накопичення баз знань. Схожим чином перетворюються окремі конструкції і фрагменти програм, процедур, функцій, специфікацій будь-яких об'єктів та інших модулів програм, моделей та запитів як інтегрованих об'єктів і забезпечують можливість їх модифікації, контроль спрямованості модифікацій. Крім того, за рахунок спеціального кодування полів вони можуть дозволити стандартизувати всі базові види семантичної обробки СМП.

Додаткові особливості власної поведінки аналітичного типу та даних такого типу визначають реакцію середовища обробки на етапі проектування або перестроювання і спирається на можливість їх використання для програмування та задач аналітичної обробки, а також для виконання різних видів навчання шляхом накопичення правил і уточнення коефіцієнтів.

Таким чином, дані ***аналітичного типу*** формуються на етапах лексичного та синтаксичного аналізу специфікацій проблемної галузі аналізу і реконфігурації конструкцій. Вони представляють різні типи текстів на комп'ютерних і природних мовах і використовуються в процесі перетворень на етапі оптимізації та таких інших етапах семантичної обробки, як генерація виконавчих кодів або формальне доведення коректності та складання відповідних протоколів.

Прагматична корисність аналітичного типу визначається **іменуванням** або **кодуванням** аналітичних об'єктів, як єдиного цілого, що здатне відтворювати семантичні характеристики, призначені для аналізу можливості та доцільності використання аналітичних моделей даних. Інший напрямок використання прагматичної корисності полягає в можливості вживання аналітичного типу не тільки в комп'ютерних мовах моделювання, програмування та запитів для цілей специфікації. Прагматична корисність проявляється і в формально визначеному розширенні мови математики на кшталт мови специфікації Z, яка може розглядатися як розширення формалізованої природної мови.

Особливі зовнішні **властивості** ***аналітичного типу*** даних пов'язані з поліморфізмом і можливістю супроводження еквівалентних і наближених формул, виразів і правил. Вони забезпечують визначення та дотримання напрямків перетворень аналітичних виразів і задаються наборами правил та послідовностями їх застосування.

Аксіоматика ***аналітичного типу*** повинна визначати правила або властивості для забезпечення повноти функціональних та багатозначних перетворень в межах цього типу. Це можна зробити на основі використання аксіоматики теорії множин та теорії наближених обчислень для комплексів операцій та функцій перетворення, еквівалентності, імплікативності та/або наближеності. Крім того, необхідно визначити аксіоми підтримки структурованості подання, кодування або іменування, цільової спрямованості перетворень, можливості доказового (обґрунтованого) обчислення при наявності необхідних даних. Включення наборів аксіом для оцінок характеристик реалізації перетворень за різними види складності дозволяє попередньо оцінити можливий час розв'язання задач доведення і ліквідації фундаментальних суперечностей визначень, неоднозначності спрямувань в перетвореннях.

Власна поведінка та реакції на середовище даних ***аналітичного типу*** визначаються операціями та функціями, які в основному схожі на множини операцій та операторів алгебри множин і мови маніпуляції даних в SQL. Крім того до набору операцій повинні включати правила підстановок та перетворень, зібрані в Табл. 3. На основі послідовностей таких операцій, визначених для кожної задачі семантичної обробки, виконуються еквівалентні перетворення різних напрямків.

Аналітичний тип природно розширюються виразами на метамовах типами, які визначають комп'ютерні та природні мови. При цьому похідні функціональні або операційні типи від аналітичного типу визначають базові механізми внутрішніх перетворень СПМ. Вони можуть задавати механізми автоматизованої генерації методів та структур даних будь-яких інших типів та класів об'єктів, що вбудовуються в мову або реалізуються за її допомогою.

Область визначення та припустимих значень даних аналітичного типу при реалізації визначається областями значень окремих полів вузла графу виразів мов і метамов. Ряд полів визначає покажчики на підлеглі або батьківські об'єкти цього ж типу, які можуть бути рекурсивними. Значення кодів полів операцій, відношень, зв'язків або термінальних позначень і полів типів їх результатів повинні однозначно кодуватися в СПМ і у випадках можливості різної довжини результатів включати додаткові коди довжини.

Висновки

Сучасні узагальнення внутрішнього подання моделей і програм дозволяють зберігати не тільки проміжні результати їх взаємних перетворень, але і протоколи обґрунтувань та доведень при перетвореннях. Викорис-

тання схожих форматів внутрішнього подання для даних, виразів і операторів широкого спектру задач дозволяє створити набори базових методів аналітичного типу для внутрішніх перетворень при мінімізації кодів, формальній верифікації та формальному синтезі кодів. Механізми перетворень мають однакову структуру правил роботи ядра при різних підходах та методиках побудови та перетворень програм і моделей. Однак результати автоматизованих доказових перетворень істотно залежать числа варіантів заповнення вбудованих баз знань, а час вибору найкращого варіанту перебудови – від числа варіантів послідовності застосування правил доведення. Тому можливість та доцільність використання наведеного підходу визначається характеристиками часової складності окремих задач доведення і попереднього накопичення в базі знань найбільш корисних варіантів перетворень для уникнення повного перебору варіантів перетворень за можливими напрямками.

Список посилань

1. Ахо А., Сети Р., Ульман Дж. Компиляторы: принципы, технологии, инструменты: Пер. с англ. – М.: Издательский дом Вильямс, 2001. – 768 с.
2. Лисков Б., Гатэг Дж. Использование абстракций и спецификаций при разработке программ. Пер. с англ. М.: Мир, 1989. 424 с.
3. Пустоваров В.І. Особливості трансформаційного підходу до реалізації мов програмування і моделювання. Вісник Національного технічного університету “Київський політехнічний інститут”. Інформатика, управління та обчислювальна техніка. К., «Век+», 2002, 38, с. 73-80.
4. Hehner E.C.R. Practical theory of programming. Springer-Verlag, New York, 1993 – 243 p.
5. Muchnick S.S. Advanced compiler design and implementation – San Francisco, Morgan Kaufmann Publishers, 1997. - 856 p.
6. Pierce Benjamin C. Types and Programming Languages. MIT Press, Cambridge, 2002. 620 p
7. Woodcock J., Davies J. Using Z. Specification, Refinement, and Proof. C.A.R. Hoare Series editor, 1995 – 390 p.