УДК 681.3.06

Зубенко В. В.

АЛГЕБРАЇЧНІ ЗАСОБИ СПЕЦИФІКАЦІЇ ІНФОРМАЦІЙНИХ МОДЕЛЕЙ. I

Уроботі розглядаються алгебраїчні засоби специфікації інформаційних моделей.

Щоб ставити задачу пошуку тих чи інших засобів специфікації певного класу об'єктів, необхідно мати чітке уявлення про самі об'єкти. Як відомо, питання типу «що таке алгоритм», «інформаційна модель» є полем гострих дискусій у математиці, і особливо в інформатиці. Спектр пропонованих відповідей на них досить широкий. Від математичних до інженерно-технічних і навіть... теологічних. Принциповим тут у багатьох відношеннях є питання В. Успенського, яке було запропоновано для обговорення учасникам Міжнародного симпозіуму «Алгоритми в сучасній математиці і її застосуваннях», присвяченого Аль-Хорезмі (Ургенч, 1979 р.): «Можно ли понятие алгоритма сформулировать в терминах других стандартных (скажем, теоретико-множественных) математических понятий или же оно по существу является независимым и первичным»? [1]. Ця проблема тісно пов'язана з іншою - загальноматематичною: «Що таке функція - множина пар вигляду (аргумент, значення) чи правило для визначення значень функції за аргументами?». У першому випадку говорять про екстепсіональний підхід до визначення функції, у другому - про інтенсіональний. Як добре відомо, первісно поняття функції формувалось саме як «правило». Тотальний перехід на теоретико-множинні рейки відбувся тільки у 19 ст. Він довів свою ефективність у багатьох застосуваннях. Але проблеми обґрунтування основ сучасної математики на початку 20 ст. змусили знову повернутись до її витоків. Так виникли λ-числення, комбінаторна логіка та інші системи, в яких функція - знову «правило». Саме на цих засадах Черч на початку 30-х років 20 ст. запропонував перше уточнення алгоритмічне обчислюваної функції (так звані λ-визначені функції) і навів перший приклад алгоритмічне нерозв'язної проблеми. Поява ж ЕОМ та розвиток інформатики значно прискорили та поглибили цей інтерес. Зазначимо, що для багатьох відомих алгоритмічних систем існують обидві платформи. Візьмемо, наприклад, основу з основ сучасної математики та інформатики - натуральні числа. Перші системи їх подання були, як відомо, чисто інтенсіональними, наприклад індуктивна система з операцією «збільшення на 1» та нулем. Потім з появою теорії множин виробилось екстенсіональне уявлення про натуральне число як про кардинальне число. Ті ж частково-рекурсивні функції можуть подаватись як інтенсіонально (у вигляді вже згадуваних λ -термів), так і в звичайних алгебраїчних теоретико-множинних термінах (алгебра Кліні). Якщо повернутися до запитання В. Успенського, то, на наш погляд, без залучення мети, контексту уточнення поняття алгоритму віднайти загальну відповідь на нього дуже проблематично, якщо взагалі можливо. Так, коли мова йде про основи інформатики або про формальний опис операційної семантики мов програмування, то стає дедалі очевидніше, що алгоритм тут повинен виступати первинним, а значить, за необхідністю інтенсіональним поняттям [2, 3]. Тоді як, наприклад, у математиці та метаматематиці плідно використовуються обидві платформи [4]. Аналогічна ситуація має місце і в програмуванні. Якщо звернутись до відомих робіт В. М. Глушкова з мікропрограмних алгебр, Т. Хоара з верифікації програм чи робіт Д. Скотта з математичної семантики мов програмування, то всі вони побудовані на екстенсіональних засадах. Знову ж таки, на ранніх стадіях життєвого циклу програм переважають, як правило, екстенсіональні моделі, і тільки на заключних уже - алгоритмічні. Приклади можна продовжити. Як бачимо, кожен з підходів має свою історію і свою сферу застосування, а значить, і свою наукову і практичну цінність. І абсолютизувати будь-який з них було б помилкою.

У [5, 6] зроблено спробу на рівні семантики «чорного ящика» дати загальне визначення алгоритму та інформаційної моделі, інваріантне відносно моделей обчислень та специфіки предметних областей. Ми свідомо не розглядаємо складніші (ті ж номінативні [2, 3]) структури, що потребують нижчих рівнів абстракції. Підхід спирається на поняття Δ-процедури та конструктивного об'єкта. При цьому, враховуючи гранично можливий рівень абстракції, конструктивність об'єктів та процедур трактується чисто алгебраїчне. Тобто конструктивними вважаються будь-які об'єкти (у тому числі і функції), що можуть бути побудовані за скінченну кількість кроків зі своїх складових за допомогою певних операцій - конструкторів. У роботі розглядаються можливості традиційних алгебраїчних структур для подання алгоритмів та інформаційних моделей.

1. Конструктивні об'єкти та інформаційні системи

Зупинимося спочатку детальніше на деяких аспектах поняття інформаційної моделі. Як і всяка модель, інформаційна модель є певною системою об'єктів та співвідношень між ними. Співвідношення ми трактуємо як часткові і багатозначні функції. Коли говорять, що інформаційна система моделює певну систему об'єктів (вхідну систему), то під цим розуміють, що її об'єкти та функції копіюють, імітують об'єкти та функції вхідної системи. Математично це означає, що інформаційна система є гомоморфним образом вхідної системи. Інформаційні моделі відрізняє від інших їх конструктивність, а точніше конструктивність їх об'єктів та функцій. Як уже зазначалось, конструктивність об'єктів означає, що вони можуть бути побудовані зі своїх складових за допомогою спеціальних операцій - конструкторів. Отже, кожен елемент інформаційної системи є або атомним, або складеним, побудованим з атомних елементів за допомогою скінченної кількості застосувань конструкторів. Типовими складеними інформаційними об'єктами є натуральні числа, кортежі фіксованої довжини (однорідні та неоднорідні), скінченні послідовності, різні теоретико-множинні та функціональні структури. Сукупності атомних елементів та конструкторів, що дають змогу побудувати всі об'єкти та співвідношення даної системи, будемо називати системами подання. Принциповим є те, що на відміну від класичних алгебраїчних систем, де атомні елементи (твірні) та конструктори (основні операції) належать самій системі, системи подання інформаційних моделей, як правило, є зовнішніми щодо них. Наприклад, вони можуть належати певній мові програмування. При такому підході конструктивність того чи іншого об'єкта (у тому числі і функції) вже не є абсолютною субстанцією, а тільки відносною і залежить від вибору системи подання. До речі, це яскраво ілюструє і класична теорія алгоритмів, яку було б точніше назвати «теорією окремих класів алгоритмів» (кожний зі своєю (!) системою подання). Взагалі, до конструктивних ми будемо відносити будь-які індуктивне визначені об'єкти. Структура індуктивного визначення **(ІВ)** певної множини об'єктів M має такий вигляд:

(Б) База індукції. Фіксується певна підмножина апріорі вибраних базових (атомних) об'єктів.

(І) Індуктивний перехід. Вибирається певна сукупність F конструкторів - операцій на M, замкнених відносно M. Замкненість конструктора відносно M означає, що результат його застосування гарантовано належить множині M за умови, що він був застосований до аргументів з M.

(П) Повнота. Конструктори дозволяють за скінченну кількість кроків отримати з базових елементів кожен з об'єктів множини M і тільки їх.

Оскільки вся сукупність об'єктів в індуктивних визначеннях повністю описується пунктами (Б) і (І), то, як правило, пункт (П) Повнота явно не формулюється. Індуктивно визначені множини будемо називати просто індуктивними і позначати $M = M(M_0, F)$. Іноді принцип Повноти послаблюють, замінюючи на пункт

(ПС) Повнота Слабка. Конструктори дозволяють за скінченну кількість кроків отримати з базових елементів кожен з об'єктів множини M (але, можливо, і не тільки їх).

Тоді потрібні конструктивні елементи відбираються за допомогою певного предиката $P(x)$. Так, добре відома роль в теорії формальних граматик предиката $P(x)$ = «x - слово в основному алфавіті». Наведемо кілька важливих прикладів конструктивних об'єктів.

1. Вже згадуване класичне індуктивне визначення натуральних чисел **($ІВ_1$)**.

($Б_1$) $0 \in N$ - натуральне число.

($І_1$) Конструктор $succ : N \to N$ породжує наступне за чергою число в N, тобто $\forall n$ ($succ(n) = n+1$).

Дійсно,

$0 \in N$ — **$Б_1$**

$1 = succ(0) \in N$ — **$І_1$**

$2 = succ(1) = succ(succ(0) \in N$ — **$І_1$**

і т. д.

2. Індуктивне визначення слів певного алфавіту **($ІВ_2$)**. Нехай $\Sigma = \{a_1, a_2, ..., a_n\}$ - довільна лінійно упорядкована сукупність, елементи якої назвемо символами, а саму її алфавітом. Для $n \in N$ покладемо

$$\Sigma^n = \prod_{i=1}^{n} \Sigma = \{\langle b_1, ..., b_n \rangle : \forall i (1 \le i \le n \Rightarrow b_i \in \Sigma)\}.$$

Кортежі з Σ^n будемо називати словами над алфавітом Σ довжиною n.

Сукупністю всіх слів над Σ будемо називати множину $\Sigma^* = \bigcup_{n \ge 0} \Sigma^n$. Порожнє слово має довжину 0. Будемо позначати його ε.

$ІВ_2$ слів над Σ має вигляд:

($Б_2$) $\varepsilon \in \Sigma^*$ – слово над Σ.

($І_2$) Для кожного $a \in \Sigma$ визначимо конструктор $app_a : \Sigma^* \to \Sigma^*$

$$app_a(\langle b_1, ..., b_n \rangle) = \langle b_1, ..., b_n, a \rangle, \; app_a(\varepsilon) = \langle a \rangle.$$

Нескладно перевірити, що будь-яке слово над Σ може бути побудоване за допомогою відповідних конструкторів з порожнього слова. Наприклад, для символів $b, c, d \in \Sigma$ слово $\langle b, c, d \rangle = app_d(app_c(app_b(\varepsilon)))$.

Наведемо інше індуктивне визначення **(ІВ$_{\bar{2}}$)** *слів над* Σ:

(Б$_{\bar{2}}$) $\varepsilon \in \Sigma^*$ – слово над Σ.

(І$_{\bar{2}}$) Для кожного $a \in \Sigma$ визначимо конструктор $pre_a : \Sigma^* \to \Sigma^*$

$$pre_a(\langle b_1,...,b_n\rangle) = \langle a, b_1,...,b_n\rangle,\ pre_a(\langle\ \rangle) = \langle a\rangle.$$

У цьому випадку слово $\langle b, c, d\rangle$ будується у зворотному порядку $pre_b(pre_c(pre_d(\varepsilon)))$.

3. Індуктивне визначення **(ІВ$_3$)** множини S_F структурних схем над сигнатурою унарних функціональних символів $F = \{false^0, f_1, f_2,..., f_n\}$.

(Б$_3$) Сигнатурні символи з F— базові структурні схеми.

(І$_{3,a}$) Для структурних схем P, S, S_1, S_2 визначимо нові структурні схеми: *begin* S_1; S_2 *end, if* P *then* S_1 *else* S_2 *fi, while* P *do* S *od.*

(І$_{3,b}$) Для структурних схем P, S, S_1, S_2 визначимо нові структурні схеми: S_1S_2, $_P(S_1 \vee S_2)$, $_P\{S\}$, $(S_1 \vee S_2)$, $(S_1 \,\&\, S_2)$, $\neg S$.

Структурні схеми вигляду **(І$_{3,a}$)** використовуються для подання структурних *While-програм,* а вигляду **(І$_{3,b}$)** - для подання елементів алгоритмічних алгебр [7].

Важливою рисою конструктивних об'єктів є те, що вони допускають математичні доведення методом структурної індукції. Нехай $P(x)$ - певний предикат на множині M конструктивних об'єктів. Правило структурної індукції для M має такий вигляд:

(Б$_4$) База індукції: $\forall x \in M_0 : P(x)$.

(І$_4$) Індуктивний перехід: для всіх конструкторів c

$$\forall x_1,...,x_n \in M : \bigwedge_{i=1}^{n} P(x_i) \Rightarrow P(c(x_1,...,x_n)).$$

(П$_4$) Повнота: $\forall x \in M : P(x)$.

З Бази індукції **(Б$_4$)** та Індуктивного переходу **(І$_4$)** випливає Заключення - Повнота **(П$_{10}$)**. Дійсно, покладемо $R = \{x \in M : P(x)\}$. З **(Б$_4$)** випливає, що $M_0 \subseteq R$, а з **(І$_4$)** – що $M \subseteq R$. Оскільки $R \subseteq M$, то маємо $R = M$.

Таким чином, щоб встановити **(П$_4$),** достатньо перевірити базу індукції **(Б$_4$)** та правило Індуктивного переходу **(І$_4$)**. Звичайна математична індукція для натуральних чисел є частковим випадком структурної індукції.

Структурна індукція для слів має вигляд:

(Б$_5$) $P(\varepsilon)$.

(І$_5$) $\forall w \in \Sigma^*, \forall a \in \Sigma : P(w) \to P(wa)$

або

(І$_5$) $\forall w \in \Sigma^*, \forall a \in \Sigma : P(w) \to P(aw)$.

(П$_5$) $\forall x \in M : P(x)$.

Приклад застосування методу структурної індукції наведемо в наступному розділі.

2. Індуктивні визначення функцій

Конструктивні об'єкти допускають індуктивні визначення функцій та операцій **(ІВФ).** На таких визначеннях, як відомо, побудована значна частина конструктивної математики та програмування. Вони теж складаються з бази індукції **(БФ),** індуктивного переходу **(ІФ)** та повноти **(ПФ).** Розглянемо спочатку **(ІВФ)** на прикладі основних операцій на словах.

Операція $conc : \Sigma^* \times \Sigma^* \to \Sigma^*$ - конкатенація слів:

(БФ$_6$) $conc(\langle\ \rangle, w_2) = w_2$.

(ІФ$_6$) $conc(pre_a(w_1), w_2) = pre_a(conc(w_1, w_2))$

для кожного $a \in \Sigma$.

Визначення коректне. Дійсно, знайдемо, наприклад, значення

$$\begin{aligned} conc(\langle c, b\rangle, \langle d, d\rangle) &= \\ = conc(pre_c(b), \langle d, d\rangle) &= && \mathbf{I}_{\bar{2}} \\ = pre_c(conc(pre_b(\varepsilon), \langle d, d\rangle)) &= && \mathbf{I}_6 \\ = pre_c(pre_b(conc(\varepsilon, \langle d, d\rangle))) &= && \mathbf{I}_6 \\ = pre_c(pre_b(\langle d, d\rangle)) &= && \mathbf{Б}_6 \\ = pre_c(\langle b, d, d\rangle) &= && \mathbf{I}_{\bar{2}} \\ = \langle c, b, d, d\rangle. && && \mathbf{I}_{\bar{2}} \end{aligned}$$

Домовимось про деякі спрощення позначень. Так, слово $\langle b_1,...,b_n\rangle$ будемо позначати традиційно $b_1...b_n$. Замість запису $conc(w_1, w_2)$ будемо вживати $w_1 \cdot w_2$ або w_1w_2. Замість $pre_a(w)$ будемо писати $a\hat{}w$, замість $app_a(w)$ – $w\hat{}a$ або просто aw та відповідно wa.

Операція $head : \Sigma^* \to \Sigma$ повертає перший символ (голову) слова, операція $bottom : \Sigma^* \to \Sigma$ – останній символ слова:

(БФ$_7$) $head(\varepsilon) = undef$, $bottom(\varepsilon) = undef$.

(ІФ$_7$) $head(a\hat{}w) = a$, $bottom(a\hat{}\varepsilon) = a$,

$w \neq \varepsilon \Rightarrow bottom(a\hat{}w) = bottom(w)$.

Операція - $tail : \Sigma^* \to \Sigma^*$ повертає хвіст слова, тобто слово без його голови:

(БФ$_8$) $tail(\varepsilon) = undef$.

(ІФ$_8$) $tail(a\hat{}w) = w$.

Операція – $lead : \Sigma^* \to \Sigma^*$ повертає слово без останнього символу:

(БФ$_9$) $lead(\varepsilon) = \varepsilon$.

(ІФ$_9$) $lead(a \wedge \varepsilon) = \varepsilon$, $w \neq \varepsilon \to lead(a \wedge w) = a \wedge lead(w)$.

Функція $// : \Sigma^* \to N$ обчислює довжину слова:

(БФ$_{10}$) $/\varepsilon/ = 0$

(ІФ$_{10}$) $/a \wedge w/ = 1 + /w/$.

Позначимо $Bool = \{ff, tt\}$ множину істинісних значень, відповідно «хиба» та «істина». Визначимо операцію $\to : Bool \times A \times A$ вибору так, що $(ff \to a, b) = b$ і $(tt \to a, b) = a$ для будь-яких *a, b.*

Предикат $\underset{lex}{>} : \Sigma^* \times \Sigma^* \to Bool$ визначає лексикографічний порядок (нестрогий) на словах над алфавітом $\Sigma = \{a_1, a_2, ..., a_n\}$. Нагадаємо, що символи алфавіту Σ лінійно впорядковані: $a_i < a_j \Leftrightarrow i < j$ для будь-яких $1 \le i, j \le n$. Тоді для довільних $v, u \in \Sigma^+$

(БФ$_{11}$) $(\varepsilon \underset{lex}{<} v) = tt$, $(u \underset{lex}{<} \varepsilon) = ff$, $(\varepsilon \underset{lex}{<} \varepsilon) = tt$.

(ІФ$_{11}$) $u \neq \varepsilon \wedge v \neq \varepsilon \to \big(head(u) = head(v) \to$

$\to \big(tail(u) \underset{lex}{<} tail(v), head(u) < head(v)\big)\big)$.

Далі лексикографічний порядок будемо позначати просто «$\le$».

Наприклад, для довільних символів *a, b* таких, що $a < b$:

$$(\varepsilon \le a) = tt,\ (b \le \varepsilon) = ff,$$
$$(ab \le ab) = (b \le b) = (\varepsilon \le \varepsilon) = tt,$$
$$(ab \le aba) = (b \le ba) = (\varepsilon \le a) = tt,$$
$$(aaba \le aba) = (aba \le ba) = (a < b) = tt.$$

Нехай *M*- довільна індуктивна множина з базою (Б) та індуктивним переходом (І). В загальному випадку індуктивне визначення унарної функції $f : M \to M$ має такий вигляд:

(БФ) База індукції. $\forall x \in M_0 : f(x) = g(x)$, де $g(x)$ – «відома» функція, що задана апріорі на множині базових елементів M_0.

(ІФ) Індуктивний перехід. Для кожного конструктора $c : M^n \to M$ елементів з M існує конструктор значень функції $h_c : M^{2n} \to M$ такий, що $f(x) = h_c(x_1, ..., x_n, f(x_1), ..., f(x_n))$ для будь-якого $x = c(x_1, ..., x_n)$ з *M.*

(ПФ) Повнота. Функція g та конструктори h_c дають змогу за скінченну кількість кроків отримати кожне із значень функції f на Мі тільки їх.

Зазначимо, що деякі аргументи в конструкторі h_c можуть бути фіктивними. Як і у випадку індуктивних множин, пункт Повноти **(ПФ)** можна послаблювати, замінюючи на Повноту Слабку.

(ПФС) Повнота Слабка. Функція g та конструктори h_c дають змогу за скінченну кількість кроків отримати кожне із значень функції f на *M.*

При Повноті Слабкій значення функції відбираються за допомогою певного предиката *P(x).* Слід зазначити, що індуктивні визначення приводять, взагалі кажучи, до багатозначних часткових функцій. У цьому випадку за допомогою Повноти Слабкої можна детермінізувати функцію, тобто відібрати той єдиний елемент, що буде значенням функції для даного аргументу. Однозначність індуктивно визначеної функції гарантують такі умови: 1) області значень будь-яких двох конструкторів c_1 та c_2 системи подання множини M не перетинаються і 2) для кожного $x \in M$ Існує тільки одна сукупність складових $x_1, ..., x_n$ така, що $x = c(x_1, ..., x_n)$. Зрозуміло, що при цьому самі функція g та конструктори h_c повинні бути однозначними. Нескладно впевнитись, що всі індуктивні визначення (ІВ$_1$), (ІВ$_2$), (ІВ$_{\bar{2}}$) та (ІВ$_3$) задовольняють умови 1) та 2).

Аналогічний вигляд мають індуктивні визначення функції від кількох змінних та систем взаємоіндуктивнихфункцій.

Прикладами таких узагальнених ІВФ є схеми ітерування та примітивної рекурсії для числових функцій. Вони використовують для подання натуральних чисел систему ІВ1. Для ітерування, наприклад, відповідне ІВФ має такий вигляд:

(БФ) $f(0) = 0$,

(ІФ) $f(x) = h\ (f(x-1))$

для будь-якого натурального $x = s(x-1)$. Конструктором значень функції f виступає тут функція $h(x)$. Якщо взяти бінарну операцію додавання $x + y$, то ІВФ для неї виглядає так:

(БФ) $x + 0 = x$,

(ІФ) $(x + y) = s(x + (y - 1))$.

Індуктивні визначення функції є основним засобом побудови програм у функціональному програмуванні (наприклад, у мовах програмування Ліспі, ML, FP-програмах Дж. Бекуса) та мовах специфікацій (Meta IV, Z, RSL тощо).

Розглянемо тепер приклад доведення методом структурної індукції, про який згадувалось вище. А саме, доведемо асоціативність операції конкатенації. Покладемо

$$P(u) \Leftrightarrow \forall w, v \in \Sigma^* : (u \cdot v) \cdot w = u \cdot (v \cdot w).$$

Перевіримо спочатку базу індукції (Б$_5$). Нехай w, v – довільні слова над Σ. Тоді з (Б$_6$) випливає, що $(\varepsilon \cdot v) \cdot w = v \cdot w$ і $\varepsilon \cdot (v \cdot w) = v \cdot w$.

А це означає, що $P(\varepsilon)$ виконується. Нехай u, v, w - довільні слова над Σ, а a – довільний символ. Перевіримо правило індуктивного переходу ($I_{5,b}$). У нашому випадку конструктор один - операція *рге.*

$$(au \cdot v) \cdot w = (a(u \cdot v)) \cdot w = \quad (I_6)$$
$$= a((u \cdot v) \cdot w) = \quad (I_6)$$
$$= a(u \cdot (v \cdot w)) = \quad P(u)$$
$$= (au \cdot (v \cdot w)). \quad (I_6)$$

Отже, індуктивний перехід ($I_{5,b}$) має місце. А за ним - І повнота (Π_5).

3. Функціональні та регулярні алгебри функцій

Нехай A - довільна непорожня множина. Щоб не вводити окремо булевий тип, виділимо в A певний елемент *ff* як «хибу». Всі решта елементів A у відповідних контекстах будуть трактуватись як «істина». Виберемо один з них для стандартного подання елемента «істина» і позначимо «*tt*». Покладемо $Bool = \{ff, tt\}$. Таким чином, $Bool \subseteq A$ і кожна однозначна функція $f : A \to A$ подає певний предикат $\Pr(f) : A \to Bool$. Нехай $Nil = \{?\}$ – спеціальний тип з «невизначеним» значенням. Визначимо на A та Nil булеві операції диз'юнкцію $\vee$, кон'юнкцію $\&$ та заперечення $\neg$. Для довільних де, *у,* що належать $A \cup Nil$, покладемо:

$$(x \vee y) = (x \neq ff \to y \# x),$$
$$(x \,\&\, y) = (x = tt \to y \# x),$$
$$\neg x = (x = tt \to ff \# (x = ff \to tt \# x)).$$

Позначимо $A \to A$ (або A^A) та $A \to Bool$ сукупності відповідно всіх часткових багатозначних функцій з A в A та предикатів на A. Для предиката $P : A \to Bool$ *P(A)* означає область його істинності. Для спрощення позначень будемо (там, де це можливо) замість $A \cup Nil$ вживати просто A.

Алгебраїчні засоби специфікації функцій поділяються на явні та неявні. Перші базуються на певній системі подання функцій як конструктивних об'єктів. Тобто фіксується певна сукупність атомних функцій та сукупність конструкторів (композицій), що дають змогу будувати з атомних функцій складені функції. Наприклад, композиція суперпозиції дає змогу будувати складені тригонометричні функції з базових тригонометричних функцій $\sin(x)$, $\cos(x)$ І Інших та операцій: +, –, ×, /. Абстрактною композицією рангу 1 над множиною A будемо називати будь-яку операцію вигляду $K : A \times ... \times A \times (A \to A) \times ... \times (A \to A) \to (A \to A)$. У цьому сенсі звичайні n -арні операції на A будемо відносити до композицій рангу 0. Абстрактність композиції означає, що її визначення є чисто теоретико-множинним і не залежить від засобів подання та іменування аргументів. Нехай Θ -довільна сукупність абстрактних композицій рангу 1 над A. Алгебру вигляду $\langle A^A; \Theta \rangle$ будемо називати функціональною.

Визначимо традиційні абстрактні композиції на A^A.

Множення. Ця композиція парі функцій (*f*, g) ставить у відповідність функцію

$$f \circ g = \{(a, b) \in A^2 : \exists c (f(a) = c \,\&\, g(c) = b)\}.$$

Результат множення двох функцій будемо позначати також *fg.*

Об'єднання. Ця композиція парі функцій (f, g) ставить у відповідність їх теоретико-множинне об'єднання

$$f \cup g = \{(a, b) \in A^2 : \exists b (f(a) = b \vee g(a) = b)\}.$$

Звуження. Нехай $B \subseteq A$. Звуженням функції f на підмножину B називається функція

$$f_B = f \cap B \times A.$$

Обхід. Композиція обходу 2-ці (p, f), де p предикат на *А,* ставить у відповідність функцію

$$f_p = f_{p(A)} \cup E_{\neg p(A)} =$$
$$= \{(a, b) \in A^2 : f(a) = b \,\&\, p(a) \vee b = a \,\&\, \neg p(a)\}.$$

Розгалуження. 3-ці (p, f, g), де p - предикат на *А,* дана композиція ставить у відповідність функцію

$$(p \to f \# g) = f_{p(A)} \cup g_{\neg p(A)}.$$

Обмеження. Обмеженням функції $f : A \to A$ на пїдмножину $B \subseteq A$ називається функція $f^B : A \to B$ така, що $f^B = f \cap A \times B$. Композиція обмеження 2-ці (p, f), де p - предикат на *А,* ставить у відповідність функцію $f^p = f^{p(A)}$.

Ітерація. Позначимо f_B^* – замкнення функції f відносно пїдмножини $B \subseteq A$, **тобто** $f_B^* = \bigcup_{n=0}^{\infty} f^n$, де $f^n = f^{n-1} f_B$, $f^0 = E$. Парі функцій (p, f), де p - предикат на A, $f : A \to A$, композиція ітерації ставить у відповідність функцію $\{p \to f\} = ((f_{p(A)})^*)^{\neg p(A)}$.

Наприклад, функція $Gcd = \{q \to (p \to f_1 \# f_2)\}\phi$ повертає найбільший спільний дільник двох

натуральних чисел для $f_1(\langle a,b\rangle)=\langle a-b,b\rangle$, $f_2(\langle a,b\rangle)=\langle a,b-a\rangle$, $p(\langle a,b\rangle)=(a>b)$, $q(\langle a,b\rangle)=(a\neq b)$, $\phi(a,a)=a$ для всіх $a,b\in N$.

Функція $mirror=(\{p\to f\}\ \phi)_{(\Sigma^*\times\{\varepsilon\})}$ повертає дзеркальне відображення слова в певному алфавіті, якщо $f(\langle x,y\rangle)=\langle tail(x),head(x)y\rangle$, $p(\langle x,y\rangle)=(x\neq\varepsilon)$ для всіх слів $x,y\in\Sigma^*$.

Повторення. Нехай $f^+=f^*-E$**.** Пара функцій (p,f), де ρ - предикат на A, $f:A\to A$, композиція повторення ставить у відповідність функцію

$$[p\to f]=(f^{+}_{\neg p(A)})^{p(A)}=$$
$$=\left\{(a,b)\in A^2:\exists b_0=a,b_1,...,b_n=b\ \&\ \bigwedge_{i=1}^{n-1}\neg p(b_i)\ \&\right.$$
$$\left.\&\,p(b_n)\&1\le i\le n\left(b_i=f\left(b_{i-1}\right)\right)\right\}.$$

Вибір. Функціям $f_1,...,f_n$ та $p_1,...,p_n$ ставить у відповідність функцію $(p_1\to f_1\ \#\ ...\ \#\ p_n\to f_n)=$
$$=\bigcup_{i=1}^{n}f_{ip_i}=\left\{(a,b)\in A^2:\exists i(p_i(a)\&b=f_i(a))\right\}.$$

Наприклад,

$$\mathrm{sign}(x)=(x<0\to -1\#x=0\to 0\#x>0\to 1),$$
$$\pm\sqrt{x}=(x>0\to+\sqrt{x}\ \#\ x>0\to-\sqrt{x}).$$

Предикат p_i в композиції вибору будемо називати за Е. Дейкстрою охороною функції f_i.

Детермінізація. Багатозначній функції $f:A\to B$ композиція детермінізації det ставить у відповідність однозначну функцію

$$\det f=\left\{(a,s)\in A\times 2^B:s=\{b:f(a)=b\}\right\}.$$

Обернення. Ставить у відповідність функції / функцію

$$f^{-1}=\left\{(b,a)\in A^2:b=f(a)\right\}.$$

Покладемо Θ_0 сукупність композицій множення, об'єднання, розгалуження, обмеження, обходу, звуження, вибору, ітерації та повторення. Поповнимо Θ_0 також композиціями диз'юнкції $\vee$, кон'юнкції $\&$ та заперечення $\neg$. Функціональну алгебру $R=\langle A^A;\Theta_0\rangle$ будемо називати регулярною. Нехай $F=\{ff^0,f_1,...,f_n\}$ – довільна сукупність функцій на A. Елементи підалгебри $R(F)$ алгебри R з множиною твірних F називаються регулярними функціями (мікропрограмами) над базисом F [7]. Якщо зафіксувати певну сигнатуру $\Omega(\Theta_0)$ композицій регулярної алгебри, наприклад, доповнивши відповідним чином визначення структурних схем в $(\mathbf{I}_{3,b})$, то кожну регулярну функцію можна подати (специфікувати) у вигляді певної узагальненої структурної схеми ($\Omega(\Theta_0)$ - регулярного виразу).

Для довільної множини A покладемо $A^*=\bigcup_{n\ge 0}A^n$, де $A^n=A\times...\times A$ – n-й декартовий степінь A. Функції з сукупності $A^n\to A$ (або A^{*A}) називаються n-арними. Нехай Ξ – довільна сукупність композицій рангу 1 над множиною A, тобто операцій вигляду $K:A\times...\times A\times\times(A^*\to A)\times\times...\times(A^*\to A)\to(A^*\to A)$. Алгебру вигляду $\langle A^*\to A;\Xi\rangle$ будемо називати функціональною алгеброю n-арних функцій. Визначимо тепер регулярну алгебру n-арних функцій.

Перестановка аргументів. Нехай $\alpha=(i_1,i_2,...,i_n)$ - довільна перестановка чисел $(1,2,...,n)$. Ця композиція довільній функції $f:A^n\to A$ ставить у відповідність функцію $f^\alpha:A^n\to A$ таку, що

$$f^\alpha=\{(<a_1,...,a_n>,b)\in A^n\times A:b=f(a_{i_1},...,a_{i_n})\}.$$

Ототожнення аргументів. Ця композиція довільній функції $f:A^n\to A$ ставить у відповідність функцію $f^\tau:A^n\to A$ таку, що

$$f^\tau=\{(<a_1,a_2,...,a_n>,b)\in A^n\times A:b=f(a_1,a_2,...,a_n)\}.$$

Параметризація аргументів. Параметру $a\in A$ та функції $f:A^n\to A$ ставить у відповідність функцію $f_a:A^{n-1}\to A$ таку, що

$$f_a=\{(<a_2,...,a_n>,c)\in A^{n-1}\times A:f(a,a_2,...,a_n)=c\}.$$

Введення фіктивних аргументів. Ця композиція довільній функції $f:A^n\to A$ ставить у відповідність функцію $f^\varphi:A^{n+1}\to A$ таку, що

$$f^\varphi=\left\{(<a_1,a_2,...,a_n,a_{n+1}>,b)\in A^{n+1}\times A:b=\right.$$
$$\left.=f(a_2,...,a_{n+1})\right\}.$$

Підстановка (суперпозиція). Функції $g\in A^m\to A$ та функціям $g_1,...,g_m\in A^n\to A$ ставить у відповідність функцію $S(g,g_1,...,g_m)\in A^n\to A$:

$$\left\{(<a_1,...,a_n>,b)\in A^n\times A:\exists c_1,...,c_m\left(g_1(a_1,...,a_n)=\right.\right.$$
$$\left.\left.=c_1,...,g_m(a_1,...,a_n)=c_m\ \&\ g(c_1,...,c_m)=b\right)\right\}.$$

Нехай $I_k^n : A^n \to A$, $1 \le k \le n$ – функції проектування по k-му компоненту, тобто $I_k^n(a_1,..., a_n) = a_k$ для $\forall < a_1,..., a_n > \in A^n$. Підстановки $S(g, g_1,..., g_m)$ та $S(g, g_1, I_2^n ..., I_n^n)$ умовимось позначати коротко $g(g_1,..., g_m)$ та $g(g_1)$.

Ітерація. Позначимо f_B^* – замкнення функції $f : A^n \to A$ за першим компонентом відносно підмножини $B \subseteq A^n$, тобто $f_B^* = \bigcup_{k=0}^{\infty} f^k$, де $f^0 = E^{n+1} =$ $= \{(< a, a_2,..., a_n >, a) \in A^n \times A : a \in A\}$, $f^n = f_B(f^{n-1})$, $n > 0$.

Парі функцій (p, f), де p – предикат на A^n, $f : A^n \to A$, композиція ітерації ставить у відповідність функцію

$$\{p \to f\} = (f_p)^* \cap (\neg p(A^n)) \times A =$$

$$= \left\{ \begin{array}{l} (< a_1,..., a_n >, b) \in A^n \times A : \\ \exists a_1 = b_0, b_1,..., b_k = b \mathop{\&}_{i=0}^{k-1} p(b_i, a_2,..., a_n) \, \& \end{array} \right.$$

$$\left. \& \neg p(b_k, a_2,..., a_n) \, \& \, 1 \le \forall i \le k \left(b_i = f(b_{i-1}, a_2,..., a_n) \right) \right\}.$$

Повторення. Для функції $f : A^n \to A$ покладемо $f^+ = f^* - E^{n+1}$. Парі функцій (p, f), де p – предикат на A, $f : A^n \to A$, композиція повторення ставить у відповідність функцію

$$[p \to f] = S(\{\neg p \to f\}, f) =$$

$$= \left\{ \begin{array}{l} (< a_1,..., a_n >, b) \in A^n \times A : \\ \exists b_0 = f(a_1,..., a_n), b_1,..., b_k = b \mathop{\&}_{i=1}^{k-1} \neg p(b_i, a_2,..., a_n) \, \& \end{array} \right.$$

$$\left. \& \, p(b_k, a_2,..., a_n) \, \& \, 1 \le \forall i \le k \left(b_i = f(b_{i-1}, a_2, ..., a_n) \right) \right\}.$$

$\exists_1$-квантифікація. Предикату $p : A^n \to Bool$ композиція $\exists_1$-квантифікації ставить у відповідність предикат $\exists_1 p : A^{n-1} \to Bool$ такий, що для $< a_1, ..., a_{n-1} > \in A^{n-1}$:

$$\exists_1 p(a_1, ..., a_{n-1}) \Leftrightarrow p(A^n) \cap A \times \{a_1\} \times ... \times \{a_{n-1}\} \neq \varnothing.$$

$\forall_1$-квантифікація. Предикату $p : A^n \to Bool$ композиція $\forall_1$-квантифікації ставить у відповідність предикат $\forall_1 p : A^{n-1} \to Bool$ такий, що для $< a_1, ..., a_{n-1} > \in A^{n-1}$:

$$\forall_1 p(a_1, ..., a_{n-1}) \Leftrightarrow A \times \{a_1\} \times ... \times \{a_{n-1}\} \subseteq p(A^n).$$

Покладемо Ξ_0 - сукупність композицій перестановки, ототожнення та параметризації аргументів, введення фіктивних аргументів, підстановки, розгалуження, обходу, обмеження, звуження, вибору, ітерації, повторення, диз'юнкції, кон'юнкції, заперечення, $\exists_1$-квантифікації та $\forall_1$-квантифікації. Функціональну алгебру $R = \langle A^* \to A; \Xi_0 \rangle$ будемо називати регулярною алгеброю л-арних функцій. Елементи підалгебри $R(F)$ алгебри R з множиною твірних F, де $F = \{f_1,..., f_n\}$ - довільна сукупність функцій на A^n, будемо називати регулярними n-арними функціями над базисом F.

Якщо звернутись до числових чи словарних функцій, то регулярних алгебр достатньо, щоб подати будь-яку частково-рекурсивну (загально-рекурсивну чи примітивно-рекурсивну) функцію в цих областях. При цьому атомними (базовими) функціями можуть бути вибрані дуже прості функції та предикати, а в якості композицій - тільки композиції множення, розгалуження та ітерації [7, 8, 9]. Для частково-рекурсивних функцій це випливає ще з роботи [10], де доведено, що будь-яка така функція може бути отримана з базових функцій $succ(x)$, I_k^n, $1 \le k \le n$, $x + y$, $x \dot{-} y$, $x \times y$ за допомогою композицій підстановки та одного простого випадку рекурсії, що фактично є ітерацією. Це ж справедливо і для функціональних алгебр з композицією обернення. Для унарних числових функцій базовими у цьому випадку можна взяти, наприклад, функції $succ(x)$ та $q(x) = x \dot{-} [\sqrt{x}]^2$, а до композиції обернення долучити ще множення та композицію додавання (підстановку в операцію додавання). Це так звані алгебри Робінсон [11].

Неявні алгебраїчні засоби подання функцій полягають у визначенні функцій за допомогою рівнянь або системи рівнянь у певній алгебрі. Це можуть бути як звичайні рівняння з предметними змінними, які описують зв'язок між аргументами та значеннями функції, так і, що важливіше з точки зору застосувань у програмуванні, функціональні рівняння, тобто рівняння, невідомими яких виступають функції. Наприклад, звичайне рівняння XY-1 з невідомим Y (предметна змінна) визначає функцію-гіперболу, а рівняння $Y^2 - X^2 = 1$ відносно Y – двозначну функцію $Y = \pm\sqrt{X^2 + 1}$ у полі дійсних чисел. Якщо взяти фунціональні рівняння, то вони, як правило, мають не один, а багато розв'язків. Тому окрім проблеми пошуку розв'язків взагалі, існує також проблема вибору підходящого у тому чи іншому відношенні розв'язку. Найбільш вивчені функціональні рівняння вигляду $F = \Phi(F)$ (*) та їх системи [12]. Так, у випадку неперервності функціоналу Φ розв'язки рівняння (*) (їх ще називають нерухомими точками) не тільки завжди існують,

але й можуть бути структуровані і, головне, алгоритмізовані [13, 14]. Наприклад, функціональне рівняння $F(x)=(x>100\to x-10, F(F(x+11)))$, де x набуває цілих значень, а невідоме F - функції вигляду $f: Z\to Z$, має кілька розв язків, найменшим серед яких (відносно теоретико-множинного включення) є функція $f_p(x)=(x>100\to x-10, 91)$, відома як 91-ша функція Мак-Карті. Інше рівняння $F(n)=(n=0\to 1, n\times F(n-1))$, де n - натуральні числа, а F - функціональна змінна типу $f: N\to N$, найменшим розв'язком має функцію - факторіал $n!$.

До неявних засобів специфікації інформаційних моделей слід також віднести і так звані логіко-алгебраїчні специфікації [15]. Вони базуються на традиційних логіках першого порядку з рівністю та на їх ініціальних моделях. Докладніше про цей підхід можна довідатись в огляді [16].

4. Процедуризація алгебраїчних специфікацій

Оскільки в програмуванні кінцевою метою є все ж побудова процедурних специфікацій систем, а розглянуті нами методи є чисто екстенсіональними, то виникає законне питання їх процедуризації та алгоритмізації. В теорії автоматів цю проблему називають проблемою синтезу. Наприклад, добре відома проблема синтезу скінченного автомата за даним регулярним виразом або проблема синтезу дискретного перетворювача за даною мікропрограмою.

Процедурний та алгоритмічний засоби подання функцій будемо трактувати згідно з [5, 6]. Нагадаємо основні визначення. Нехай S- довільна множина, елементи якої будемо називати станами. Виділимо підмножини $S_0\subseteq S$, $S_{fin}\subseteq S$ відповідно початкових та заключних станів. Нехай $\Delta=\{f_1, f_2, ..., f_n\}$ - довільна множина певних багатозначних функцій на станах $f_i: S\to 2^S$. Будемо називати їх базовими або елементарними. Часткову функцію вигляду $\delta: N\times(S-S_{fin})\to 2^{\Delta}$ назвемо функцією керування. Процедурою над базисом Δ (Δ-процедурою) будемо називати п'ятірку $P=<S, S_0, S_{fin}, \Delta, \delta>$. Послідовність $s_0, s_1, ..., s_n, ...$, де $s_i=f_{k_i}(s_{i-1})$, а $f_{k_i}=\delta(i, s_{i-1})$ для всіх $i\geq 1$, називається обчисленням за процедурою P на початковому стані s_0. Стан $s_n\in S_{fin}$ для певного $n\geq 0$ називається заключним станом (результатом) обчислення $s_0, ..., s_n$. Результат обчислення будемо позначати $P^*(s_0)$ і говорити, що процедура P повертає його на початковий стан s_0, а про саме обчислення говорити як про результативне. Всі нескінченні обчислення назвемо безрезультатними. В деяких випадках слушно вважати результативним і будь-яке скінченне обчислення, що закінчується певним станом $s_n\in S-S_{fin}$ і таким, що значення $\delta(n+1, s_n)$ не визначене. Результативні та безрезультатні обчислення називаються повними, решта - проміжними. Кожна процедура P визначає певну багатозначну функцію на станах $P^*: S_0\to 2^S$, а саме таку, що $P^*(s_0)=s$, де s – результат певного результативного обчислення на початковому стані s_0. Області визначення та допустимих значень функції P^* позначаються відповідно $In(P^*)$ та $Out(P^*)$. Процедури з однозначними функціями керування називаються детермінованими. Δ-процедура є Δ-алгоритмом відносно певної системи подання, якщо її стани та функція керування є конструктивними в цій системі. Функція $P^*: In(P^*)\to Out(P^*)$, що обчислюється Δ-алгоритмом P, називається Δ-обчислюваною.

Проблема процедуризації конструктивних функцій даної функціональної алгебри полягає в побудові класу Δ-процедур, що обчислюють (моделюють) всі функції алгебри. Ця процедуризація прямо спирається на індуктивну природу конструктивних функцій. Остання дає змогу звести правило обчислення значень складеної функції до правил обчислення її базових функцій та до правил маніпуляції з ними у відповідності з семантикою композицій-конструкторів. Для подання композиційної структури складених функцій використовуються спеціальні синтаксичні структури - терми. Розглянемо спочатку індуктивне визначення термів у найпростішій і найпоширенішій їх формі, що відповідає композиції підстановки (суперпозиції). Це класичні типізовані терми 1-го ступеня в так званих префіксній, постфіксній та інфіксній формах. Терми - це слова, що базуються на певних сигнатурних символах. Сигнатура - це трійка $\Omega=\langle S, C, F\rangle$, де $S\neq\varnothing$, елементи якої називаються сортами або типами; кожному $c\in C$ поставлено у відповідність певний сорт $s\in S$; c називається константним символом типу s; кожному $f\in F$ поставлено у відповідність певне слово $w\in S^+$ та певний сорт $s\in S$. Слово w визначає сорти аргументів, а s - сорт результатів функціонального символу f. Вираз $w\mapsto s$ будемо називати типом функціонального символу/сигнатури Ω.

Приклади сигнатур.

1) $\Omega_1=\langle S, C, F\rangle$ – «мінімальна» сигнатура для натуральних чисел, де

$$S=\{nat, bool\},\ K=\{zero: nat\},$$

$$F=\{succ: nat\to nat,\ pred: nat\to nat,\ =: nat, nat\to nat\};$$

2) $\Omega_2 = \langle S, C, F \rangle$ - словарна сигнатура, де

$$S = \{char, string, bool, nat\},$$

$$K = \{c_1 : char, ..., c_n : char, empty : string\},$$

$$F = \begin{Bmatrix} conc : string, string \mapsto string \\ head : string \mapsto char \\ bottom : string \mapsto char \\ pre : char, string \mapsto string \\ app : char, string \mapsto string \\ tail : string \mapsto string \\ lead : string \mapsto string \\ length : string \mapsto nat \\ \leq : string, string \mapsto bool \end{Bmatrix}.$$

Нехай X - довільна сукупність типізованих змінних, тобто кожній з них поставлено у відповідність певний сорт сигнатури Ω. Зафіксуємо певний сорт $s \in S$. Визначимо сукупність $T_\Omega^s(X)$ термів 1-го ступеня (далі - термів) типу s сигнатури Ω над X у префіксній формі (IB_{12}):

($Б_{12}$) константи $c \in K$ та змінні $x \in X$ типу s є термами типу s ;

(I_{12}) нехай $f \in F$ символ типу $s_1 ... s_k \mapsto s$, $t_i \in T_\Omega^{s_i}(X)$ терм типу s_i для $1 \leq i \leq k$, $g \in F$ – бінарний символ типу $s_1 s_2 \mapsto s$.

Тоді

$$(t_1 g t_2),\ f(t_1, ..., t_k),\ (t_1, ..., t_k) f \in T_\Omega^s(X).$$

Домовимось надалі терми з **($Б_{12}$)** та **(I_{12})** називати відповідно атомними та структурними, а останні, в свою чергу, термами в інфіксній, префіксній та постфіксній формах відповідно. Терм t_i називається підтермом структурних термів з **(I_{12})**, а t_1 - їх лівим підтермом.

Позначимо $T_\Omega(X) = \bigcup_{s \in S} T_\Omega^s(X)$ множину всіх Ω-термів. Ω-терми з сукупності $T_\Omega = T_\Omega(\varnothing)$ будемо називати замкненими, або базовими. Прикладами Ω_1- та Ω_2-термів можуть бути: $zero$, $succ(succ(zero))$, $conc(x, app(a_1, conc(empty, a_3)))$ і т. д. Перші два є базовими термами. Типовим прикладом Ω-термів можуть слугувати арифметичні та алгебраїчні вирази елементарної математики. Наприклад, $((a^n + b^n) = c^n)$, $(2 \times (\sin(x) \times \cos(y)))$ тощо. З метою спрощення нотації термів широко використовують правила замовчування. Для цього фіксують певний пріоритет сигнатурних операцій та предикатів, що дає змогу в тих випадках, коли це можливо, не писати дужки. Наприклад, 5-34*2 означає $(5-(34*2))$, $NOTxORy$ – $(NOT(x)ORy)$ і т. д.

З точки зору семантики кожен з Ω-термів задає певну складену функцію, побудовану з базових функцій за допомогою композиції підстановки. Щоб визначити строго, що це за функція, звернемося до основного поняття сучасної алгебри – поняття багатосортної Ω-алгебри. Нехай $\Omega = \langle S, C, F \rangle$ – довільна сигнатура. Ω-алгеброю називається трійка $A = (S^A, K^A, F^A)$, де

1) кожному сорту $s \in S$ поставлена у відповідність певна множина значень даного сорту s^A і $S^A = \{s^A : s \in S\}$;

2) кожному константному символу $c \in C$ сорту s поставлено у відповідність його значення c^A з множини s^A і $C^A = \{c^A : c \in C\}$;

3) кожному функціональному символу $f \in F$ типу $s_1 ... s_k \mapsto s$ поставлено у відповідність певну функцію $f^A : s_1^A \times ... \times s_k^A \to s^A$ і $F^A = \{f^A : f \in F\}$.

Розглянемо кілька прикладів Ω-алгебр.

1. Нехай $\Omega_3 = \langle S, C, F \rangle$ – інша сигнатура для натуральних чисел.

$$S = \{nat, bool\},\ K = \{zero : nat\},$$

$$F = \begin{Bmatrix} succ : nat \mapsto nat, \\ pred : nat \to nat, \\ + : nat, nat \to nat, \\ - : nat, nat \to nat, \\ \times : nat, nat \to nat, \\ < : nat, nat \to bool, \\ = : nat, nat \to bool \end{Bmatrix}.$$

Визначимо Ω_3-алгебру $N = (S^N, C^N, F^N)$, де $S^N = \{N, \{ff, tt\}\}$, $C^N = \{0\}$, $F^N = \{s(x) = x+1, pd(x) = x-1, x+y, x-y, x \times y, x/y, a \bmod b, x < y, x = y\}$. Ця Ω_3-алгебра N називається арифметикою натуральних чисел.

2. Нехай $\Omega_4 = \langle S, C, F \rangle$ – один з варіантів словарної сигнатури,

$$S = \{char, string, bool\},$$

$$C = \{c_1 : char, ..., c_n : char, empty : string\},$$

$$F = \begin{Bmatrix} conc : string, string \mapsto string \\ head : string \mapsto char \\ tail : string \mapsto string \\ \leq : string, string \mapsto bool \end{Bmatrix}.$$

Визначимо словарну Ω_4-алгебру $W = \left(S^W, C^W, F^W\right)$, де

$$S^W = \left\{\Sigma = \{a_1, ..., a_n\}, \Sigma^*, \{ff, tt\}\right\},$$
$$C^W = \{a_1, ..., a_n, \varepsilon\},$$
$$F^W = \{conc(x,y), head(x), tail(x), \le\}.$$

Оцінкою типізованих змінних сукупності X в Ω-алгебрі A називається довільна функція $\sigma: X \to \bigcup_{s \in S} s^A$, що зберігає типи змінних. Тобто для типізованої змінної $x:s$ $\sigma(x)$ завжди належить множині s^A. Значення $I[t]^A_\sigma$ довільного Ω-терму $t \in T_\Omega(X)$ в Ω-алгебрі A при оцінці змінних σ визначається індуктивно (**ІВ**$_{13}$). Для скорочення доведень обмежимось далі тільки префіксними термами:

(**Б**$_{13,a}$) $I[c]^A_\sigma = c^A$ для всіх констант $c \in C$;

(**Б**$_{13,b}$) $I[x]^A_\sigma = \sigma(x)$ для всіх $x \in X$;

(**І**$_{13}$) $I[f(t_1,...,t_k)]^A_\sigma = f^A\left(I[t_1]^A_\sigma, ..., I[t_k]^A_\sigma\right)$ для всіх $f \in F$ $1 \le i \le k$.

Має місце важлива Лема.

Лема. Нехай σ_1 та σ_2 – довільні оцінки змінних в Ω-алгебрі A і такі, що $\sigma_1(x) = \sigma_2(x)$ для всіх змінних x, що входять у терм $t \in T_\Omega(X)$, тоді

$$I[t]^A_{\sigma_1} = I[t]^A_{\sigma_2}.$$

Доведення. Структурна індукція за індуктивним визначенням термів (ІВ$_{12}$).

(Б). Нехай t є константою $c \in C$. Оскільки значення констант в Ω-алгебрі не залежить від оцінок змінних, то за (**Б**$_{13,a}$) $I[t]^A_{\sigma_1} = c^A = I[t]^A_{\sigma_2}$.

Нехай t є змінною $x \in X$. Тоді за умовою леми і за (**Б**$_{13,b}$)

$$I[t]^A_{\sigma_1} = \sigma_1(x) = \sigma_2(x) = I[t]^A_{\sigma_2}.$$

(І). Нехай терм t має вигляд $f(t_1,..., t_k)$ і для всіх t_i, $1 \le i \le k$, лема має місце. Тоді

$$\begin{aligned} &I[f(t_1,..., t_k)]^A_{\sigma_1} = \\ &= f^A\left(I[t_1]^A_{\sigma_1}, ..., I[t_k]^A_{\sigma_1}\right) = && (\mathbf{І}_{15}) \\ &= f^A\left(I[t_1]^A_{\sigma_2}, ..., I[t_k]^A_{\sigma_2}\right) = && \text{Інд. припущення} \\ &= I[t]^A_{\sigma_2}. \end{aligned}$$

Лема показує, що значення термів визначається виключно їх композиційною будовою та значеннями нефіктивних змінних. Тепер можна визначити Δ-процедуру для обчислення значення довільного структурного Ω-терму $t = t(x_1..., x_m)$ в алгебрі A при заданих значеннях його змінних $x_1 = a_1, ..., x_m = a_m$. Ще раз зазначимо, що змінні можуть бути і фіктивними, але, як свідчить Лема, вони не впливають на значення терму. Станами процедури слугують Ω-терми, заключними станами – терми-константи. Елементарні операції та функцію переходу сформулюємо неформально у вигляді такого загального **Правила**. Позначимо $t = t(x_1 \mapsto u_1, ..., x_m \mapsto u_m)$ терм, утворений шляхом заміни в ньому всіх входжень кожної змінної x_i, $1 \le i \le m$, на терми u_i. Поповнимо сигнатуру Ω типізованими константами c_a для кожного $a \in \bigcup_{s \in S} s^A$. Правило полягає в побудові так званого обчислення в алгебрі A за термом t, що відповідає значенням змінних $x_1 = a_1, ..., x_m = a_m$, а саме скінченної послідовності базових Ω-термів t_0, t_1,..., t_n такої, що

1) $t_0 = t(x_1 \mapsto c_{a_1}, ..., x_m \mapsto c_{a_m})$;

2) для кожного $i \ge 0$ t_{i+1} отримано з t_i шляхом заміни в ньому найлівішого і найбільш внутрішнього структурного підтерму на константу c_b, де b – значення цього підтерму в алгебрі A;

3) $t_n = c_a$, де $a = I[t]^A_\sigma$ для оцінки $\sigma(x_i) = a_i$, $1 \le i \le m$.

Перед тим як довести, що елемент a дійсно є потрібним значенням, розглянемо кілька прикладів обчислень за термом.

1. Для Ω_3-терму $t = succ(succ(succ(x)))$ та $x = 2$ обчислення у відповідності з Правилом за термом t в Ω_3-алгебрі N має такий вигляд:

$$t_0 = succ(succ(succ(2))),$$
$$t_1 = succ(succ(3)),$$
$$t_2 = succ(4),$$
$$t_3 = 5.$$

2. Для Ω_4-терму $t = head(conc(x,y))$ та $x = a_2$, $y = a_1a_3$ обчислення в Ω_4-алгебрі W у відповідності з Правилом за термом t має такий вигляд:

$$t_0 = head(conc(a_2, a_1a_3)),$$
$$t_1 = head(a_2a_1a_3),$$
$$t_2 = a_2.$$

Доведемо тепер коректність Правила. Структурна індукція за (ІВ$_{12}$).

(Б). Очевидно.

(І). Нехай терм t має вигляд $f(t_1,...,t_k)$, $\sigma(x_i)=a_i$ для $1\le i\le m$. Нехай далі для $1\le i\le m$ $I[u_i]^A_\sigma = b_i$, $u^i{}_0, u^i{}_1, ..., u^i_{n_i}$ – обчислення за Правилом значення $I[u_i]^A_\sigma$ і всі вони коректні. Тобто для всіх $1\le i\le m$ має місце $u^i_{n_i}{}^A = b_i$. Покладемо також $I[t]^A_\sigma = b$. Тоді

$$\begin{aligned} &I[f(t_1,...,t_k)]^A_\sigma = \\ &= f^A\left(I[t_1]^A_{\sigma_1},...,I[t_k]^A_{\sigma_1}\right)= && (\mathbf{I}_{15})\\ &= f^A\left(c_{b_1}{}^A,...,c_{b_k}{}^A\right)= && \text{Інд. припущення}\\ &= I[t]^A_{\sigma_2} = \\ &= c_b{}^A. \end{aligned}$$

Розглянемо таку послідовність термів

$$t_0 = f\left(t_1\left(x_1\mapsto c_{a_1},...,x_m\mapsto c_{a_m}\right),\right.$$
$$\left.\ldots,t_k\left(x_1\mapsto c_{a_1},...,x_m\mapsto c_{a_m}\right)\right)= f\left(u^1_0,u^2_0,...,u^k_0\right);$$
$$t_1 = f\left(u^1_1,u^2_0,...,u^k_{0_i}\right);$$
$$t_2 = f\left(u^1_2,u^2_0,...,u^k_{0_i}\right);$$
$$\ldots\ldots\ldots\ldots$$
$$t_{n_1} = f\left(u^1_{n_1},u^2_0,...,u^k_{0_i}\right)= f\left(c_{b_1},u^2_0,...,u^k_{0_i}\right);$$
$$\ldots\ldots\ldots\ldots$$
$$t_r = f\left(c_{b_1},...,c_{b_{k-1}},u^k_{0_i}\right);$$
$$t_2 = f\left(c_{b_1},...,c_{b_{k-1}},u^k_{1_i}\right);$$
$$\ldots\ldots\ldots\ldots$$
$$t_{n-1} = f\left(c_{b_1},...,c_{b_{k-1}},u^k_{n_k}\right)= f\left(c_{b_1},...,c_{b_{k-1}},c_{b_k}\right);$$
$$t_n = c_b.$$

Нескладно перевірити, що дана послідовність дійсно відповідає **Правилу** обчисленням за термом t для значень змінних $x_1=a_1,...,x_m=a_m$. Для п. 1), 3) це очевидно. А в зв'язку з п. 2) зауважимо, що для t_0 найлівіший і внутрішній його структурний підтерм – це найлівіший і внутрішній структурний підтерм в u^1_0, для t_1 найлівіший і внутрішній його структурний підтерм – це найлівіший і внутрішній структурний підтерм в u^1_1 і т. д.

Процедуризація функцій довільних функціональних алгебр здійснюється аналогічно. Розглянемо, наприклад, регулярні функції, побудовані за допомогою композицій підстановки, розгалуження та ітерації. У цьому випадку поняття структурного Ω-терму узагальнюється за рахунок композицій розгалуження та ітерації. А саме, до сигнатури Ω долучаються предикатні символи, а до пункту $(\mathbf{I}_{12})$ додаються Ω-терми вигляду $(p\to t_1\#t_2)$ та $\{p\to t_1\}$, де t_1,t_2 – терми певного типу u, а p – предикатний Ω-терм. Нехай $t = t(x_1...,x_m)$ – терм типу s і $x_1=a_1,...,x_m=a_m$ значення змінних x_i в довільній оцінці σ. При цьому x_1 має, як і t, тип s. Тоді визначення $(\mathbf{IB}_{13})$ значення Ω-терму t в алгебрі A при оцінці змінних σ доповнюється пунктами:

$(\mathbf{I}_{13,a})$ $I[(p\to t_1\#t_2)]^A_\sigma \to \left(I[p]^A_\sigma \to I[(t_1)]^A_\sigma \# I[(t_2)]^A_\sigma\right)$

$(\mathbf{I}_{13,b})$ $I\left[\{p\to t\}\right]^A_\sigma = \left(I[p]^A_\sigma \to I\left[\{p\to t\}\right]^A_\sigma \# I\left[I^m_1\right]^A_\sigma\right)$,

де p – оновлена оцінка змінних σ, в якій $x_1 = I[t_1]^A_\sigma$, а I^m_1 – функція проектування за 1-м компонентом.

Як і у випадку підстановки, має місце Лема про фіктивні змінні та Правило для обчислення значення $I[t]^A_\sigma$ Ω-терму $t = t(x_1...,x_m)$ в Ω-алгебрі A. Єдина відмінність, що найлівіший внутрішній підтерм вибирається тепер з урахуванням семантики також і композицій розгалуження та ітерації.

Проблема Процедуризації нерухомих точок неперервних функціоналів докладно розглядається в [13], а логіко-алгебраїчних специфікацій – у [15, 16].

5. Процедуризація індуктивно визначених функцій

У загальному випадку така Процедуризація здійснюється аналогічно за допомогою Правила, що задає певне числення Ω-термів, сигнатура яких містить також конструктори значень функції h_c та функцію g з бази функціональної індукції. Докладніше на цьому ми зупинимось у другій частині роботи.

6. Терми як засіб синтаксичного подання інформаційних моделей

Покажемо тепер, що довільний конструктивний об'єкт завжди може бути поданий у вигляді певного Ω-терму. Це означатиме, що інформаційні моделі завжди можуть бути специфіковані за допомогою певної сукупності Ω-термів, що відповідають сигнатурам їх систем подання.

Нехай a належить індуктивній сукупності об'єктів $M(M_0,F^M)$. Кожній такій сукупності можна поставити у відповідність певну Ω-алгебру $M=(S^M,C^M,F^M)$. Сигнатура $\Omega=\langle S,C,F\rangle$ вибирається, таким чином, з урахуванням семантики

конструкторів. Кожний конструктор $f^M \in F^M$ є певною операцією, визначеною на кортежах фіксованої довжини $k = k\left(f^M\right)$. Тобто $f^M : M_1 \times ... \times M_k \to M_{k+1}$ для певних множин $M_1, ..., M_k$, M_{k+1}. Позначимо конструктор f^M якимось функціональним символом f, кожній з множин M_i поставимо у відповідність певний сорт s_i. Звісно, якщо певні M_i та M_j при цьому збігаються, то $s_i = s_j$. Покладемо S та F – сукупності всіх так вибраних сортів s_i та функціональних символів для всіх конструкторів $f^M \in F^M$. Зіставимо далі з кожним базовим елементом $a \in M_0$ константу c_a і присвоїмо їй сорт s_i залежно від того, якому саме M_i належить a. Позначимо C сукупність всіх таких констант c_a. Отже, сигнатура Ω повністю визначена, як і відповідна Ω-алгебра $M = \left(S^M, C^M, F^M\right)$. За побудовою, кожний елемент індуктивної сукупності M є елементом Ω-алгебри M. Визначимо тепер поняття Ω-алгебри термів $T_\Omega = \left(S^{T_\Omega}, C^{T_\Omega}, F^{T_\Omega}\right)$ для нашої сигнатури Ω. А саме:

1) кожному сорту $s \in S$ поставимо у відповідність сукупність s^{T_Ω} всіх Ω-термів типу сорту s і покладемо $S^{T_\Omega} = \left\{s^{T_\Omega} : s \in S\right\}$;

2) кожному константному символу $c \in C$ сорту s поставимо у відповідність його самого у якості значення c^{T_Ω}. Отже, $C^{T_\Omega} = C$;

3) кожному функціональному символу $f \in F$ типу $s_1 ... s_k \mapsto s$ поставимо у відповідність операцію $f^{T_\Omega} : s_1^{T_\Omega} \times ... \times s_k^{T_\Omega} \to s^{T_\Omega}$ таку, що для будь-яких термів $t_i \in T_\Omega^{s_i}$ типу s_i, $1 \le i \le k$: $f^{T_\Omega}(t_1, ..., t_k) = f(t_1, ..., t_k)$, і покладемо $F^{T_\Omega} = \left\{f^{T_\Omega} : f \in F\right\}$.

Алгебру термів T_Ω називають вільною Ω-алгеброю над базисом C. Сформулюємо тепер поняття гомоморфізму (ізоморфізму) для Ω-алгебр A і B. Відображення $h : A \to B$ будемо називати Ω-відображенням алгебри A в алгебру B, якщо воно зберігає константу-хибу ff та сорти елементів, тобто для кожного $s \in S : h\left(s^A\right) \subseteq s^B$. Надалі ми будемо розглядати тільки Ω-відображення алгебр і називати їх просто відображеннями. Відображення $h : A \to B$ Ω-алгебри A в Ω-алгебру B називається гомоморфізмом, якщо для кожного функціонального символу $f \in F$ типу $s_1 ... s_k \mapsto s$ і будь-яких $a_i \in s_i^A$, $1 \le i \le k$, виконується:

$$h\left(f^A(a_1, ..., a_k)\right) = f^B\left(h(a_1), ..., h(a_k)\right).$$

Має місце наступна теорема.

Теорема. Відображення $h : C \to M_0$, $h(c_a) = a$, $c \in C$, може бути єдиним чином розширене до гомоморфізму «на» $\overline{h} : T_\Omega \to M$ вільної алгебри $T_\Omega = \left(S^{T_\Omega}, C^{T_\Omega}, F^{T_\Omega}\right)$ в Ω-алгебру $M = \left(S^M, C^M, F^M\right)$.

Доведення. Покладемо $\overline{h}(c) = h(c)$ для кожного $c \in C$ і $\overline{h}\left(f(t_1, ..., t_k)\right) = f^M\left(\overline{h}(t_1), ..., h(t_k)\right)$ для кожного функціонального символу $f \in F$ типу $s_1 ... s_k \mapsto s$ і всіх термів $t_i \in s_i^{T_\Omega}$, $1 \le i \le k$. Покажемо, що $\overline{h}$ є гомоморфізм. Дійсно,

$$\begin{aligned} &\overline{h}\left(f^{T_\Omega}(t_1, ..., t_k)\right) = \\ &= \overline{h}\left(f(t_1, ..., t_k)\right) = && \text{def. } f^{T_\Omega} \\ &= f^M\left(\overline{h}(t_1), ..., h(t_k)\right). && \text{def. } \overline{h} \end{aligned}$$

Нехай g: $T_\Omega \to M$ – інший гомоморфізм і $g(c) = h(c)$ для кожного $c \in C$. Покажемо, що $g = \overline{h}$. Структурна індукція за індуктивним визначенням термів (ІВ_{12}). База індукції виконується за визначенням g. Нехай для довільного функціонального символу $f \in F$ типу $s_1 ... s_k \mapsto s$ і довільних $t_i \in s_i^A$ виконується: $g(t_i) = \overline{h}(t_i)$ (Інд. припущення). Тоді

$$\begin{aligned} &g\left(f^{T_\Omega}(t_1, ..., t_k)\right) = \\ &= f^M\left(g(t_1), ..., g(t_k)\right) = && g\text{-гомоморфізм} \\ &= f^M\left(\overline{h}(t_1), ..., \overline{h}(t_k)\right) = && \text{Інд. припущення} \\ &= \overline{h}\left(f^{T_\Omega}(t_1, ..., t_k)\right). && \overline{h}\text{-гомоморфізм} \end{aligned}$$

Покажемо тепер, що $\overline{h} : T_\Omega \to M$ є відображенням «на». Нехай $a \in M$. Виводом елемента a в алгебрі T_Ω називається послідовність Ω-термів $t_0, t_1, ..., t_n$ така, що

1) для всіх $i \ge 0$ $b_i \in C$ або $b_i = f^{T_\Omega}\left(t_{i_1}, ..., t_{i_k}\right)$ для певних $f \in F$ типу $s_{i1} ... s_{ik} \mapsto s$ і термів t_{i_l} типу s_{i_l} для $1 \le i_l < i$, $1 \le l \le k$;

2) $h(t_n) = a$.

Заключний терм t_n виводу a будемо називати його синтаксичною формою або поданням у вільній алгебрі T_Ω. Доведемо, що для будь-якого $a \in M$ існує його подання в алгебрі T_Ω. Знову звернемось до структурної індукції. База індукції **(Б)** виконується за визначенням T_Ω. **(І)**. Нехай $f \in F$ – довільний символ типу $s_1 ... s_k \mapsto s$ і для довільного $a_i \in M$ типу s_i, $1 \le i \le k$, існує вивід в

алгебрі T_Ω, що закінчується термом t_i таким, що $\bar{h}(t_i) = a_i$. Розглянемо елемент $a = f^M(a_1, ..., a_k)$. Безпосередньо з визначень (зокрема гомоморфізму) випливає

$$\bar{h}(f^{T_\Omega}(t_1, ..., t_k)) = \bar{h}(f(t_1, ..., t_k)) = f^M(\bar{h}(t_1), ..., \bar{h}(t_k)) = a.$$

Тоді одним з виводів a в алгебрі T_Ω буде об'єднання виводів для $a_1, ..., a_k$ із заключним елементом a.

Зазначимо, що конструктивний елемент може мати кілька (а в загальному випадку і зліченну кількість) термальних синтаксичних форм. Наприклад, слово

$$\langle b,c,d \rangle = pre_b(pre_c(pre_d(\varepsilon))) = app_d(app_c(app_b(\varepsilon))).$$

Друга частина роботи присвячена нетрадиційним алгебраїчним системам, зокрема так званим Прикладним Програмним Алгебрам (ППА), орієнтованим на специфікацію програмних структур та даних [17].

1. *Клини С. В.* Алгоритмы в различных аспектах // Алгоритмы в современной математике и ее приложениях.- Ч. 2.- Новосибирск: ВЦ СО АН СССР, 1982.- С. 139-146.
2. *Редько В. Н.* Основания программологии // Кибернетика и системныйанализ-2000-№1-С.33-57.
3. *Никитченко Н. С.* Композиционно-номинативный подход к уточнению понятия программы // Проблемы программирования- 1999.-№ 1.-С. 16-31.
4. *Расева Е. Р., Сикорский Р.* Математика метаматематики.- М: Наука, 1972.-591 с.
5. *Зубенко В. В.* Про деякі загальні питання інформатики // Наукові записки НаУКМА.- Т. 16. Комп'ютерні науки, 1999.-С. 17-23.
6. *Зубенко В. В.* Про одне уточнення поняття алгоритму в довільній області // Комп'ютерна математика. Оптимізація обчислень.-К.-2001.-С. 171-175.
7. *Глушков В. М., Цейтлин Г. Е., Ющенко Е. Л.* Алгебра, языки, программирование.- К.: Наукова думка 1974. - 327 с.
8. *Bohm C., Jacopini G.* Flow Diagrams, Turing Machines and Languages with Only Two Formations Rules // Comm. Of ACM .- 1966 .-V. 9.-№ 5.-P. 366-371.
9. *БуйД. Б., Редько В. Н.* Примітивні програмні алгебри. I, II // Кибернетика- 1984-№ 6- С. 24-32; Кибернетика- 1985-№1.-С. 19-25.
10. *McCarthy J. A* Basis for a Mathematical Theory of Computation // Computer Programming and Formal Systems / P. Braffort and D. Hirshberg (eds.).-Amsterdam: North-Holland, 1963-P. 33-70.
11. *Мальцев А. И.* Алгоритмы и рекурсивные функции- М.: Наука.-1964.-388 с.
12. *Д. Скотт.* Области в денотационной семантике // Математическая логика в программировании.- М.: Мир, 1991.- С. 58-118.
13. *Манна З.* Теория неподвижной точки // Кибернетический сборник. Вып. 15-М.: Мир, 1978-С. 38-100.
14. *Буй Д. Б., Редько В. Н.* Программологические аспекты метода неподвижной точки // Кибернетика и системный анализ.- 1994.-№ 5.- С. 158-167.
15. *Goguen J. A., Thatcher J. W., Wagner E.* An Initial Algebra Approach to the Specification, Correctness and Implementation of Abstract Data Types // Current Trends in Programming Methodology (R. Yeh ed.).- Englewwood Cliffs, NJ.: Prentice Hall, 1978.-P. 80-149.
16. *В. М. Антимиров, А. А. Воронков, А. И. Дехтярев, М. В. Захарьящев.* Математическая логика в программировании. Обзор // Математическая логика в программировании.- М.:Мир, 1991.-С. 331-407.
17. *Zubenko V. V.* Applaid Program Algebras // Technical Report 8504.-Kiev.:CAU, 1985-51p.

V. V. Zubenko

ALGEBRAIC TOOLS FOR INFORMATION SYSTEMS SPESIFICATION. I

Traditional algebraic tools for information systems spesiflcation is discussed.