$\pi_2^{/}\left[\left(1-\pi_1-\pi_2\right)+\frac{1}{2}\pi_2\right]d\alpha$, и таким образом суммарное значение прироста выигрыша составляет

$$dw(\alpha)=\left[\pi_1^{/}\left(1-\frac{1}{2}\pi_1\right)+\pi_2^{/}\left(1-\pi_1-\frac{1}{2}\pi_2\right)\right]d\alpha\,.$$

Исходя из определения равновесия по Нэшу, игроку не выгодно отклоняться от выбранной стратегии, поэтому $dw(\alpha)\le 0$. Поскольку отклонение dα может быть как положительным, так и отрицательным, то выражение, стоящее в квадратных скобках должно быть рано нулю. После элементарных преобразований это условие сводится к уравнению

$$-\alpha\left(1-\frac{1}{2}\frac{1+\alpha-\alpha^2}{(1+\alpha)^2}e^{\alpha-1}\right)+\left(1-\frac{1+(3/2)\alpha-\alpha^2}{(1+\alpha)^2}e^{\alpha-1}\right)=0$$

Левая часть данного уравнения монотонно убывает на интервале [0;1], и данное уравнение имеет единственный корень, примерно равный 0.71.

Таким образом, при отсутствии информации об исходе выбора друг друга, игроки должны придерживаться оптимальной стратегии выбора с параметром $\alpha\approx 0.71$. При этом значения порогов в оптимальной стратегии выбора составляют: $t_1(0.71, t_2(0.71))\approx\left(0.438, 0.585\right)$.

Найдем цену анархии. В найденной равновесной ситуации оба игрока не получат выигрыша с вероятностью $(1-\pi_1(0.71)-\pi_2(0.71)))^2\approx 0.260$, значит, с дополнительной вероятностью, равной 0.74 игроки разделят каким-то образом единичный выигрыш между собой. Таким образом, цена анархии составляет $\frac{0.75}{0.74}\approx 1.014$.

Выводы. Были рассмотрены два варианта игры- симметричный и несимметричный. В несимметричной игре выигрыш второго игрока превосходит выигрыш первого вследствие того, что он более информирован. В симметричной игре оба игрока находятся в одинаковых условиях, и их выигрыши равны. Однако, такое установление справедливости порождает и большее значение анархии.

Список использованных источников

1. Е.Б.Дынкин, А.А.Юшкевич. Теоремы и задачи о процессах Маркова. Москва, изд-во "Наука", с. 91-102.
2. С. М. Гусейн-Заде. Задача выбора и оптимальное правило остановки последовательности независимых испытаний, ТВП, 11:3 (1966), с. 534–537.
3. В.В.Мазалов. Математическая теория игр и приложения. Изд-во "Лань", СПБ., 2010. 446 с.
4. В.В.Мазалов. Игровые моменты остановки. Изд-во "Наука", Новосибирск, 1987. 191 с.
5. С.Доценко. Задача выбора наилучшего объекта как игра двух лиц. Кибернетика и Вычислительная техника, вып. 164, 2011, с. 43-53.
6. С.Доценко. Игровые ситуации в модифицированной задаче оптимального выбора. Кибернетика и Вычислительная техника, вып. 168, 2012, с. 3-8.
7. Melissa de Carvalho, Lucas M. Chaves, Ricardo Martins. Variations of secretary problem via game theory and linear programming. Journal of computer science, vol. 7, N 3, September 2008.
8. О справедливом разделе выигрыша в одной игровой задаче оптимального выбора. Кибернетика и вычислительная техника, вып. 167, 2012, с. 87-96.

Надійшла до редколегії 18.01.13

С. Доценко, канд. фіз.-мат. наук, ст. наук. співроб.
КНУ імені Тараса Шевченка, Київ

ІГРОВІ СИТУАЦІЇ В ЗАДАЧІ ВИБОРУ НАЙКРАЩОГО ЧИ НАЙГІРШОГО ОБ'ЄКТА

В статті розглянута ігрова модифікація задачі оптимального вибору, пов'язана з вибором найкращого або найгіршого елементу. Розглянуто модифікацію задачі, в якій кожен з гравців здійснює вибір на своїй множині елементів, а наслідок вибору одного з гравців стає відомим іншому. Рівновага за Нешем знайдено в явній формі.

Ключові слова: рівновага Неша, оптимальний вибір, гра симетрична, гра несиметрична.

S. Dotsenko, PhD. Sci. Sciences, Senior Research Fellow.
Kyiv National Taras Shevchenko University, Kyiv

GAME OF THE SITUATION IN THE PROBLEM OF THE BEST CHOICE OR OBJECT OF THE WORST

The article considers secretary problem game modification, associated with the best or the worst element choice. Each of the players makes his choice at separate set and choice outcome of one of the players became known to the other. Nash equikibrium is found at explicit form.

Ключові слова: Nash equilibrium, the optimal choice, the game is symmetric, asymmetric game.

УДК 519.9

В. Зубенко, канд. фіз.-мат. наук, доц.
КНУ імені Тараса Шевченка, Київ

КОМПОЗИЦІЙНА МОДЕЛЬ КОМУНІКАТИВНИХ ІНФОРМАЦІЙНИХ СИСТЕМ

Розглядається алгебрична модель комунікативних інформаційних систем.

Ключові слова: композиційна модель,комунікативна система, конструктор, правило декларації, еквітонне розширення.

В [1-3] введено поняття комунікативної платформи інформатики. Проголосивши предметом комунікативної інформатики конструктивні моделі комунікативних процесів та систем, ми хочемо піти далі і запропонувати певну алгебричну їхню модель. При такому підході суб'єкти комунікативних систем – суб'єкт-ініціатор та суб'єкт-виконавець – конкретизуються як спеціальні формалізовані композиційні $\Omega-$системи, в яких операції та предикати визначаються не тільки на кортежах, а й на даних і можуть повертати дані, а композиція суперпозиції доповнюється сукупністю інших композицій (у тому числі і логічних) для утворення похідних операцій та предикатів.

1. Нагадаємо деякі поняття, пов'язані з комунікативною платформою і насамперед центральне в ній – комунікативної системи. У *комунікативних системах (КС)* обов'язковою є наявність:

1) предметної області (ПрО) з певною сукупністю інформаційних об'єктів та співвідношень між ними;

2) суб'єкта-ініціатора, який формує і передає суб'єкту-виконавцю певну вхідну та приймає від нього певну вихідну інформацію;

3) суб'єкта-виконавця(обробника), який приймає вхідну інформацію, аналізує її, обробляє та повертає як вихідну суб'єкту-ініціатору.

Усі інформаційні об'єкти КС – структуровані, а саме: в них явно виділяють повідомлення v, його значення a та зв'язок між ними α. Перше є формою, завдяки якій інформацію ідентифікують і передають, друге відповідає за зміст інформації і враховуються при її перетворенні, а третій подає спосіб, яким встановлюється (визначається) відношення між першими двома, та, можливо, з іншими інформаційними об'єктами. Будемо ототожнювати інформаційний об'єкт з вказаними трьома його елементами і зображати його діаграмою вигляду $v \underset{\alpha}{\mapsto} a$, а суб'єкти-ініціатори та суб'єкти-виконавці називатимемо скорочено *ініціаторами* та *виконавцями (обробниками)*.

Головне призначення комунікативних систем полягає у здійсненні *комунікативних процесів*, кожний з яких розгортається в певному часовому просторі і в результаті встановлює зв'язок між певними об'єктами предметної області системи. За своєю роллю в процесі ці об'єкти розподіляються на вхідні та вихідні. З перших розпочинається процес, другими закінчується. Комунікативний процес складається з етапів, які разом утворюють його життєвий цикл. Останній розпочинається ініціатором, який формує й передає виконавцю за допомогою засобів зв'язку повідомлення, що містить *запит*[1] на обробку певних вхідних інформаційних об'єктів. Запит окрім останніх містить інформацію про мету обробки, яка може бути сформульована неявно – як вимоги до очікуваних вихідних об'єктів або явно – як детальний опис їх отримання за вхідними об'єктами. В обох випадках мета декларує певне співвідношення між об'єктами і сама подається у вигляді спеціального інформаційного об'єкту – програми. Підкреслимо, що програма не обов'язково є імперативною, на відміну від *внутрішніх процедур* обробника, які за означенням є імперативними, призначеними для виконання. Вихідні об'єкти з'являються як раз в результаті виконання їх обробником.

Зазвичай виконавець має власні інформаційні об'єкти – *внутрішні* (сукупності яких утворюють його стани), на відміну від *зовнішніх* – тих, що фігурують в предметній області, Тому сам запит містить не вхідні об'єкти виконавця, а тільки їхні прообрази, які після кодування утворюють початковий стан виконавця. Це стосується й вихідних внутрішніх обєктів поданих у заключному стані процесу обробки. Вони потребують декодування.

Комунікативна платформа накладає певні обмеження на КС та їхні моделі, а саме: вони мають бути обовязково дескрипитивними і не просто дескриптивними – а конструктивними. Дескриптивність КС означає, що до складу її елементів входять дескриптивні системи (ДС), в яких подаються (описуються) усі її інформаційні об'єкти: вхідні і вихідні дані, запити та внутрішні процедури. Мова йде не про одну, а, як правило, про декілька ДС, тому що інформаційні об'єкти різних суб'єктів системи потребують різних дескриптологічних засобів. Як мінімум – мову специфікації запитів для ініціатора і внутрішню мову для виконавця. Остання описує внутрішні інформаційні об'єкти та внутрішні процедури, які задають правила для реалізації кожної з програм мови специфікацій. Внутрішні мови на відміну від мов специфікацій обовязково є імперативними.

Конструктивність КС означає, що її мова (мови) специфікацій та внутрішня мова є фінітними, тобто усі об'єкти в них – скінченно подані. Мови специфікацій конструктивних КС самі називаються *конструктивними,* їхня внутрішня мова – *алгоритмічною*, а внутрішні процедури – *алгоритмами.* Наголосимо, що для конструктивності довільної мови специфікацій недостатньо тільки її фінітності – необхідна також наявність алгоритмічної мови, яка задає реалізацію її запитів.

Важливим прикладом конструктивних КС є обчислювальні комп'ютерні системи. В таких системах внутрішніми процедурами (алгоритмами) виступають машинні програми.

Сформулюємо тепер, що таке композиційна та формалізована композиційна $\Omega-$ системи. Зафіксуємо певну багатосортну сигнатуру $\Omega = (\Omega_s, \Omega_v, \Omega_f, \Omega_p, \Omega_F, \Omega_P, \Omega_c)$, де $\Omega_s, \Omega_v, \Omega_f, \Omega_p, \Omega_F$, Ω_P та Ω_c – сукупність імен відповідно типів, типізованих предметних змінних, основних операцій та предикатів, типізованих функціональних та предикатних змінних та символів композицій. Деякі з цих сукупностей можуть бути порожніми.

Виділемо серед імен типів ім'я логічного типу $Bool$. Щоб проінтерпретувати сигнатуру Ω на певному універсумі U, $\{0,1\} \subseteq U$, необхідно вибрати в ньому конкретні підмножини для символів-типів (для типу $Bool$ - підмножину істиностних значень $\{0,1\}$), предметним змінним поставити у відповідність їхній тип, іменам операцій та предикатів – конкретні типізовані операції та предикати, функціональним та предикатним змінним – відповідні типи операцій та предикатів, символам композицій – операції на функціях та предикатах. Таке співставлення I сигнатурним символам їхніх значень називається *інтерпретацію* сигнатури Ω з основою U. Значення сигнатурного симола c при інтерпретації I будемо позначати c^I. Інтерпретація I може бути частковою. Наприклад, ім'я типу $Bool$ можна не інтерпретувати, якщо сигнатура основних предикатів $\Omega_p = \varnothing$. В інших впадках можуть залишатись неінтерпретованими деякі змінні, а також деякі основні операції та предикати.

Проінтерпретовані сигнатури називаються *композиційними* $\Omega-$*системами* і позначають $C = (U, \Omega, I)$ або Ω_U^I. Фактично композиційна $\Omega-$система C складається з двох алгебр – *функціональної* $C_{f,p}$ та *предметної* C_U. Операціями першої є композиції системи і вона є породженою основними операціями та предикатами сигнатур Ω_f та Ω_p. Елементи її носія називаються *похідними* операціями та предикатами системи C і є класичними в тому сенсі, що вони визначені на типізованих кортежах фіксованої довжини, значення яких теж типізоване. Інтерпретація I дозволяє проінтерпретувати всі похідні операції та предикати функціональної алгебри $C_{f,p}$. Носієм предметної алгебри C_U є сукупність типів, а операціями та предикатами – перетворення типів, похідні операції та предикати системи.

[1] Не змішувати з запитами в інформаційно-пошуковими системах. Тут термін "запит" трактується як загальне поняття з більш широким змістом.

Коли композиції в сигнатурі Ω_c відсутні або не проінтерпретовані в системі C, то вона називається *простою.* Композиція називається *логічною*, якщо її аргументи (не обов'язково усі) та результат є предикатами. Прикладами логічних композицій є логічні зв'язки та типізовані композиції суперпозиції з першим аргументом предикатного типу. Композиційна Ω – система називається *логічною*, якщо серед її композицій присутні логічні композиції.

Система C називається *системою з внутрішніми перетвореннями типів*, якщо до її сигнатури долучено сукупність імен таких перетворень $\Omega_\chi = \{\chi_{s,t} : s,t \in \Omega_s\}$ з інтерпретацією вигляду $\chi^I_{s,t} : s^I \to t^I$ для $s,t \in \Omega_s$. Для деяких типів $s,t \in \Omega_s$ перетворення $\chi^I_{s,t}$ може бути не визначено в інтерпретації I. Відомим прикладом внутрішніх перетворень типів є стандартне перетворення числових типів у логічний тип $Bool$.[2]

З'ясуємо тепер, що таке фінітарні операції та неокласичні композиційні Ω – системаи. Класична n – арна операція визначена на кортежах $\overline{a} = (a_1,...,a_n)$ довжини n, кожний з яких може трактуватись як функція $\overline{a}$ яка, з натуральним $1 \le i \le n$ пов'язує значення a_i, що належить основі A, тобто $\overline{a}(i) = a_i$. Можна піти далі і пов'язувати з елементами не числа, а довільні індекси (імена) з певної множини $X = \{v_1,...,v_n\}$. Результат такого співставлення будемо записувати $\{v_1 \mapsto a_1,...,v_n \mapsto a_n\}$ і називати *індексованою* множиною або X – множиною. Функції визначені на X – множинах називаються *фінітарними* (X – *арними*), а композиційні Ω – системи з фінітарними основними операціями, предикатами та композиціями – *неокласичними.* Як приклад фінітарних операцій можна навести об'єднання та перетину інденксованих сімей множин: $\bigcup\limits_{i \in I} A_i$, $\bigcap\limits_{i \in I} B_i$.

Неокласичні прості Ω – системи зберігають усі загальні властивості притаманні звичайним простим Ω – системам [4]. Це ж стосується і окремих класів неокласичних композиційних Ω – систем, зокрема регулярних та рекурсивних [5,6].

Але перехід від нумерації елементів до індексації їх значно збільшує дескриптивні можливості неокласичних Ω – систем.

Щоб отримати формалізований варіант довільної композиційної системи C, необхідно вибрати формальну мову L_Ω для подання всіх її похідних операцій та предикатів. Алфавіт мови L_Ω обов'язково включає сигнатуру Ω та правила для побудови основних синтаксичних конструкцій – *термів* та *формул*. Перші подають похідні операції, а другі – такі ж предикати системи. Двійка $S = (C, L_\Omega)$ називається *формалізованою композиційною* Ω – *системою*.

2. Вхідні системи представляють алгебричну модель суб'єкта-ініціатора. Якщо абстрагуватись від несуттєвих деталей і зосередитися на семантиці, то вхідні системи можна обмежити описом предметної області та сукупності запитів. І тут необхідно вирішити дилему: привносити чи ні інтенсіональні функціональні елементи в модель ПрО. Наша відповідь буде – ні. Щоб не перевантажувати цю модель деталями, будемо переносити усі можливі такі інтенсіональні елементи у запити.

Багатосортність реальних ПрО, а відтепер і апріорі їхня екстенсіональність логічно приводять до застосування формалізованих композиційних Ω-систем S для уточнення вхідних систем. При такому підході композиційна Ω – система представляє ПрО, а формальна мова (а більш точно її формули) – запити.

У вхідних системах у якості інформаційних об'єктів виступають проінтерпретовані та означені елементи сигнатури Ω. Особливістю таких інформаційних об'єктів є те, що зв'язок в них сугубо формальний, "призначений", результат інтерпретації та формальної оцінки, а не виник природно в результаті тих чи інших процесуальних дій. Проінтерпретовані предметні, функціональні та предикатні змінні сигнатури Ω мають свої екстенсіонали – типи і їх можна формально означувати (конкретизовувати) допустимими значеннями. Конкретизація змінних називається їх *оцінкою.* З іменами основних операцій та предикатів теж можна пов'язати екстенсіонали – це ті типізовані відношення, які є їхніми значеннями в інтерпретації, і тому вони також можуть оцінюватись. Оцінка їх полягає у формальному співставленні імені операції (предикату) певного конкретного варіанту аргументів та його значення на ньому. Таке формальне зв'язування імен, аргументів та значень операції (предикату) приводить до утворення інформаційного об'єкту і називається *формальною аплікацією* функції (предикату), на відміну від аплікацій-дій функцій та предикатів в процесах обчислень.

Нехай $v \in \Omega_v$, $f \in \Omega_f$ та $p \in \Omega_p$, – довільні предметна, функціональні та предикатні символи, а тип T, f^I та p^I – їхні екстенсіонали в інтерпретації I відповідно. Оцінка змінної v та символів f, p в інтерпретації I – це довільне їхнє відображення α у свої екстенсіонали. Нехай $a \in T$, $(a_1,...,a_n,a_{n+1}) \in f^I$ та $(b_1,...,b_m,b_{m+1}) \in p^I$ – значення змінної v та символів f, p при оцінці α, тобто $\alpha(v) = a$, $\alpha(f) = (a_1,...,a_n,a_{n+1})$ та $\alpha(p) = (b_1,...,b_m,b_{m+1})$. Тут a_{n+1} та b_{m+1} – значення операції f^I та предиката p^I на аргументах відповідно $(a_1,...,a_n)$ та $(b_1,...,b_m)$. Такі конкретизовані змінні та символи будемо подавати діаграмами $v \underset{I}{\mapsto} a$, $f \underset{I}{\mapsto} (a_1,...,a_n,a_{n+1})$ та $p \underset{I}{\mapsto} (b_1,...,b_m,b_{m+1})$ і трактувати як зовнішні інформаційні об'єкти. Якщо із контексту зрозуміло, про яку інтерпретацію йде мова, то індекс I в діаграмах можна опускати, а формальну аплікацію функціонального та предикатного символів зображати скорочено Ω – термами $f(a_1,...,a_n)$ та $p(a_1,...,a_m)$ відповідно. Для заданої інтерпретації I обидва терми дозволяють однозначно відновити відповідний інформаційний об'єкт[3].

Оцінені функціональні та предикатні змінні сигнатур Ω_F та Ω_P теж можуть конкретизуватися так, як і основні функціональні та предикатні символи сигнатури Ω.

---

[2] При ньому ненульові числа відображаються в 1, а число нуль – в 0.

[3] Традиційно терми подають тільки значення відповідних проінтерпретованих операцій та предикатів на заданих аргументах.

В предметній алгебрі C_U як інформаційній моделі ПрО усі співвідношеннями між інформаційними об'єктами визначаються функціями та предикатами цієї алгебри. Вони є сугубо семантичними і залежать тільки від значень об'єктів. Якщо взяти довільну $n-$арну операцію f алгебри C_U, то вона визначає відношення f^I не тільки на кортежах значень змінних, а й переносить його на кортежі самих конкретизованих змінних так, що $(v_1 \mapsto a_1,..., v_n \mapsto a_n, v_{n+1} \mapsto a_{n+1}) \in f^I \Leftrightarrow (a_1,...,a_n,a_{n+1}) \in f^I$ для будь-яких змінних $v_1,...,v_n$. Аналогічно і для предикатів.

Якщо запити у вхідних системах подаються формулами формальної мови L_Ω, то для подання вхідних об'єктів використовують оцінки вхідних змінних (це можуть бути як предметні, так і функціональні та предикатні змінні).

В класичних простих $\Omega-$системах похідні операції та предикати породжуються за допомогою композицій суперпозиції. Для отримання інших використовують логічні надбудови простих $\Omega-$систем різного штабу. Найбільш відомою логічною $\Omega-$системою є прикладне числення предикатів (ПЧП) – композиційна $\Omega-$система з тотальними основними операціями та предикатами та з класичними логічними звя'зками та композиціями квантифікації. Головними синтаксичними конструкціями ПЧП є терми та формули. Розрізнюють ПЧП першого, другого та вищих порядків. У перших квантори застовуються тільки до предметних змінних, у других – також і до функціональних та предикатних[4].

Як відомо, ПЧП обох порядків у цілому неконструктивні. Причиною тому є неконструктивність композицій квантифікації [6]. Щоб зробити вхідну систему на базі ПЧП конструктивною, використовують лише частину їхніх формул та обмежують інтерпретації.

Наведемо кілька прикладів запитів в ПЧП. Безкванторні формули $x^2 = 1$ та $z = 2 \times x + 3 \times y$ подають в натуральній арифметиці похідні предикати з областями істинності відповідно $\{x \mapsto 1, x \mapsto -1\}$ та $\{(x \mapsto a, y \mapsto b, z \mapsto 2 \times a + 3 \times b) : a,b \in N\}$, а формула другого порядку

$\forall x(F(x) = (x = 0 \to 1, x \times F(x-1))) \,\&\, \forall G(\forall x(G(x) = (x = 0 \to 1, x \times G(x-1))) \to F \subseteq G)$ – оцінку $F \mapsto n!$, де $n!$ – факторіал числа n. Для формули Φ, предметних u,v,w та функціональних F,G змінних рівність $w = (\Phi \to u, v)$ та формула $F \subseteq G$ є скороченнями відповідно формул $\exists y(w = y \,\&\, \Phi \to y = u \,\&\, \neg\Phi \to y = v)$ та $\forall x(\exists y(y = F(x) \to y = G(x))$.

Аналогічно визначається неокласичний варіант ПЧП.

Можна збагатити сукупність похідних операцій в $\Omega-$системах, допустивши до їх утворення опріч суперпозиції й інші композиції $n-$арних та фінітарних функцій. *Регулярному* збагаченню відповідає долучення усіх регулярних композицій, а *рекурсивному* – додатково композиції рекурсії за одним із обчислювальних правил. До регулярних, опріч суперпозиції, належать композиції розгалуження, недетермінованого вибору, перестановки та ототожнення аргументів та ітерація по компоненті. Регулярно (рекурсивно) збагачені композиційні Ω-системи будемо називати *регулярними (рекурсивними)* як і відповідні ПЧП та їхні терми та формули. Якщо останні містять композицію ітерації чи рекурсії, то їх називають (на наш погляд невдало) ще *динамічними.* Відносно динамічний характер регулярних та рекурсивних термів пов'язаний з тим, що в процесі обчислення їхніх значень що відповідає заданій оцінці змінних може виникнути потреба у залученні інших їхніх оцінок (як, правило, не одноразовій). Назвемо цю динаміку, пов'язану з розширенням об'єму оцінок, *динамікою об'єму* або *об'ємною.*

Якщо до основних операцій $\Omega-$системи додати функціональні константи–проекції $pr_i^n(x_1,...,x_n) = x_i$ для кожного $1 \le i \le n$, то в таких ПЧП усі регулярні композиції можуть бути зведені до суперпозиції та рекурсії. А якщо основа U є ще й індуктивною[5], то в збагаченому ПЧП другого порядку можна елімінувати і композицію рекурсії. Щоправда не для усіх правил її обчислення, а тільки для тих, які обчислюють найменшу нерухому точку композиції [7].

Деталі ПЧП першого та другого порядків можна знайти в [9], формалізованих композиційних систем – [4,10], динамічних логік в – [11].

Нехай S – класична (неокласична) рекурсивна формалізована логічна Ω-система.

Двійка $\mathrm{S}_{in} = (S, \mathrm{O})$, де $\mathrm{O} \subseteq (\Omega_v^I)^U$ – сукупність допустимих оцінок вхідних змінних, називається класичною (неокласичною) **вхідною системою** комунікативних інформаційних систем. За означенням, не всі можливі оцінки вхідних змінних є допустимими, а тільки ті, що належать сукупності O**.**

Така універсальна алгебрична модель ініціатора виявляється достатньо змістовною й має свої переваги. З одного боку, високий рівень абстракції дозволяє при уточненні предметних областей використовувати весь потужний арсенал екстенсіональних алгебричних засобів, а з другого, робить можливим використання логічних засобів для специфікації запитів. Запити у цьому випадку мають вигляд логічних формул $\Phi = \Phi(x_1,...,x_n,y_1,...,y_m)$, де $X = \{x_1,...,x_n\}$ – сукупість вхідних змінних, а $Y = \{y_1,...,y_m\}$ – вихідних, зображених у мові L_Ω.

3. Вихідні системи є алгебричною семантичною моделлю суб'єктів-виконавців. Як і вхідні системи, вони належать до класу формалізованих композиційних $\Omega-$систем. Наша задача – з'ясувати специфіку цих систем і, насамперед, специфіку внутрішніх інформаційних об'єктів, якими оперують виконавці КС. Тому що композиційна складова вихідних систем – та ж, що і в нелогічній частині вхідних систем і, як правило, нерекурсивна.

Відразу відзначимо, що вихідні системи, на відміну від об'ємно динамічих вхідних систем, є динамічними по суті, тобто в них властивості об'єктів (їхні значення і навіть зв'язки між іменами та значеннями) можуть в процесі обчислень змінюватись, а самі об'єкти утворюватися і руйнуватися в процесі обчислень[6]. Щоб підкреслити цю обставину,

[4] У ПЧП третього порядку квантори застосовуються до змінних-композицій другого порядку і т.д. Але в цій роботі ми їх торкатися не будемо.

[5] Часково впорядкована множина є індуктивною, якщо кожий незпадний ланцюг елементів в ній має найменшу вержню границю.

[6] У цьому сенсі ще раз наголосимо, що об'єкти вхідних систем тільки за назвою є змінними й динамічними, а по суті – це іменовані константи. Тут змінність означає плюралізм, паралельне існування в часі кількох оцінок однієї і тієї ж змінної (кількох одноіменнних об'єктів), при якому поява нової її оцінки не відміняє вже існуючих її оцінок.

операції та функції з динамічними об'єктами, будемо називати *діями* [12], а застосування їх до конкретних об'єктів – *аплікацією,* але вже не формальною, а *операційною*.

Імена, значення та зв'язки. Повідомлення у вихідних системах представляють *імена*. Нехай $V=\{v_1,v_2,...\}$ – певна сукупність імен, $V_0 \subseteq V$ – певна сукупність *атомних* імен. На природу атомних імен не будемо накладати жодних обмежень, вважається тільки, що серед них присутні всі натуральні числа. Таке припущення дозволить використовувати там, де це доцільно, кортежі і послідовності при структуруванні інформаційних об'єктів і класичні операції та предикати в діях над ними. Неатомні імена називаються *структурованими* – вони певним чином будуються з атомних імен. Правила їх побудови визначаються у формальній мові L_Ω вихідної системи.

Нехай $U=\{a_1,a_2,...\}$ – довільний універсум *значень* об'єктів, серед яких присутні як імена, тобто $V \subset U$, так і зв'язки. Як бачимо, у вихідних системах самі імена при деяких умовах можуть виступати значеннями об'єктів. Об'єкти з такими значеннями отримали назву *покажчиків.*

Роль імен у вхідних і вихідних системах суттєво різниться. Якщо в перших вони є лише сигнатурними елементами, то у других імена не тільки позначають значення, а й самі у якості значень стають активними учасниками обчислювальних процесів. Їх можна обчислювати за допомогою дій, пересилати тощо.

Усі значення розподіляються на дві категорії: *предметні* та *імперативні*. Перші подають предмети, другі – певну конкретну дію над предметами, наприклад, арифметичну або певний зв'язок. Як і імена, значення теж можуть мати певну структуру і розподіляються на *атомні* (неструктуровані) та *складені* (структуровані). Елементами останніх можуть бути і самі інформаційні об'єкти (зокрема покажчики) та їхні сукупності. Тобто в загальному вигляді інформаційні об'єкти мають ієрархічну будову[7].

Імена та значення об'єднуються в об'єкти за допомогою *зв'язків*. Структуру зв'язків ми конкретизувати не будемо, але деякі важливі їхні характеристики обговоримо. Позначимо Σ – сукупність усіх зв'язків в системі.

Об'єкти та інформаційні поля. Для утворення об'єктів існує спеціальна дія – *конструктор* об'єктів $\mapsto$, яка встановлює заданий зв'язок між іменами та їхніми значеннями. Об'єкт, який є результатом застосування конструктора до імені v, зв'язку $\alpha \in \Sigma$ та значення $a \in U$, подається діаграмою $v \underset{\alpha}{\longmapsto} a$[8]. Покладемо Θ сукупність усіх потенційних об'єктів вихідної системи. Сукупності об'єктів можуть об'єднуватися в *інформаційні поля*. Останні не є множини – вони мають інтенсіональну, а не екстенсіональну природу, наприклад, об'єкти в них можуть збігатися. Розрізнюють інформаційні поля зі *слабким* і *сильним* іменуванням. У першому випадку імена об'єктів у полі можуть повторюватись, у другому – ні.

Інформаційні поля утворюються спеціальним конструктором $[...]$ і позначаються $[v_1 \underset{\alpha_1}{\longmapsto} a_1,...,v_n \underset{\alpha_n}{\longmapsto} a_n,...]$ або $[v_1 \mapsto a_1,...,v_n \mapsto a_n,...]$. Ми використовуємо квадратні дужки у записі інформаційного поля, щоб підкреслити їхню відмінність від множин. Останній запис використовують, коли відомо (наприклад, з контексту) які саме зв'язки маються на увазі чи, навпаки, це не має значення. Покладемо $\varnothing$ – порожнє поле. За означенням, об'єкт $v \underset{\alpha}{\longmapsto} a$ і одноелементне інформаційне поле $[v \underset{\alpha}{\longmapsto} a]$, не збігаються. Зазначимо, що ми не виключаємо з розгляду і нескінченні поля, але вони теж мають бути результатом певних дій.

Коли усі зв'язки всередині об'єктів інформаційного поля збігаються зі зв'язком α, то його можна утворити *груповим* конструктором $\mapsto$, який зображається $v_1,...,v_n \underset{\alpha}{\longmapsto} a_1,...,a_m$ і повертає поле $[v_1 \underset{\alpha}{\longmapsto} a_1,...,v_n \underset{\alpha}{\longmapsto} a_n, v_n \underset{\alpha}{\longmapsto} a_{n+1},...,v_n \underset{\alpha}{\longmapsto} a_m]$, якщо $n \le m$ і поле $[v_1 \underset{\alpha}{\longmapsto} a_1,...,v_m \underset{\alpha}{\longmapsto} a_m, v_{m+1} \underset{\alpha}{\longmapsto} a_m,...,v_n \underset{\alpha}{\longmapsto} a_m]$, якщо $n > m$.

Інформаційні поля будемо трактувати як передмножини [13], які складаються з елементів-об'єктів. На передмножинах не визначено відношення рівності елементів, тому елементи в них можуть повторюватись. Основними діями на них є: *об'єднання* $\cup$, *вибір* ch, *вибір з вилученням* chd, *опустошення* $\overline{\varnothing}$ та *обчислення потужності* – $card$. Перша дія за двома полями повертає поле, яке складається з усіх без винятку їхніх об'єктів (немає склеювання однакових елементів), друга – реалізує недетермінований вибір об'єкта поля, третя – те саме, але ще й з вилученням вибраного елементу з поля, четверта – перетворює поле в порожнє і п'ята – повертає кількість елементів у полі.

Наступні чотири дії над полями пов'язані зі специфікою об'єктів. Дія *розіменування* $\backslash$ – параметризована і за фіксованим іменем v на полі $d=[v_1 \underset{\alpha_1}{\longmapsto} a_1,...,v_n \underset{\alpha_n}{\longmapsto} a_n,...]$ повертає значення a_i, якщо $v=v_i$ для деякого $i \ge 1$.

Перевірка приналежності $v!d$ за іменем v та полем d повертає поле d, коли об'єкт з іменем v присутній в d та порожнє поле $\varnothing$ у супротивному.

Іменування в полі $d_1 \cup d_2$ може виявитись слабким навіть коли об'єднуються два поля із сильним іменуванням. Це не так у випадку дії *оновлення* $d_1 \nabla d_2$, результатом якої є поле d_2, до якого долучено всі об'єкти поля d_1 з іменами, які відсутні в d_2. Дія оновлення не- комутативна – вважається, що інформація в об'єктах поля d_2 "свіжіша" ніж в одноіменних об'єктах поля d_1.

[7] У даній роботі ми її деталізувати не будемо.

[8] Якщо нас цікавить тільки сам факт існування певного зв'язку між іменами та їхніми значеннями, а його інтенсіонал – ні, то подання інформаційних об'єктів можна традиційно обмежити упорядкованими парами (v,a), в яких перший компонент – ім'я, а другий – його значення.

Дія *деструктора* $des(v,d)$ полягає у вилученні з поля d об'єкта з іменем v, якщо v - статична змінна, та – і його декларації, якщо – динамічна.

Типізація. У конкретних вихідних системах існують обмеження на встановлення зв'язків між іменами та значеннями. За цими обмеженнями усі зв'язки системи типізовані – кожний з них має справу не из усіма значеннями універсуму U, а тільки з деякими, що належать певному фіксованому типу $T \subset U$. Сукупність таких прив'язаних до типу T зв'язків будемо позначати $T\uparrow$. Вона сама є типом і її зв'язки – унікальні і не можуть належати іншим типам. Таким чином, універсум U вихідної системи апріорі розбивається на систему типів Θ, так що $U = \bigcup_{T\in\Theta} T$. Вона структурована відносно типів-зв'язків. За означенням, $\Theta = \bigcup_{i\geq 0} \Theta_i$, де Θ_0 сукупність усіх типів-незв'язків системи, $\Theta_i = \{T\uparrow : T \in \Theta_{i-1}\}$, $i \geq 0$. Вважається, що порожньому типу T_0 відповідає тільки один ("фіктивний") зв'язок – *nil*. тобто $T_0\uparrow = \{nil\}$ і цей зв'язок, як виключення, входить до усіх типів-зв'язків системи Θ.

У зв'язку з типізацією універсуму U з'являються нові дії – *конструктури типів* $\underset{c}{\Longrightarrow}$ та *типізованих змінних* $\rightarrow$. Перший – параметризований, за типами $T_1,...,T_n$ і іменем v утворює новий тип T і пов'язує його з іменем v. Другий – утворює зв'язок α типу T і пов'язує його з іменем v. Ці дії називаються *деклараціями* і позначаються (як і їхні результати) діаграмами $v \underset{c}{\Longrightarrow} (T_1,...,T_n)$ та $v \rightarrow T$ відповідно. Говорять, в процесі цих декларацій ім'я v набуває статусу *імені типу* T та *змінної типу* T і стає *задекларованим*. Перший статус дозволяє за іменем v ідентифікувати тип T і використовувати його в інших конструктурах для побудови нових типів та в деклараціях змінних (в деклараціях типів фігурують насправді не самі типи $T_1,...,T_n$, а їхні імена). Він не може бути змінений під час обробки полів.

Для того щоб за допомогою конструктурів типів породжувати нові типі, деякі з них – *первинні* – мають бути визначені апріорі. В противагу первинним, породжені типи отримали назву – *похідних.* Введемо спеціальний конструктор $v \Rightarrow T\uparrow$, що утворює за типом T відповідний іменований тип $T\uparrow$ зв'язків. Усі такі іменовані типи-зв'язки є похідними.

Для кожного типу T серед зв'язків типу $T\uparrow$ розрізнюють статичні та динамічні їхні варіанти і відповідно такі ж варіанти змінних. Змінна після набуття свого *статичного* статусу типу T не може змінити його на інший – типу T', а *динамічна* – може. За означенням, статус імен типів – теж статичний.

Долучимо до складу інформаційних полів декларації типів та змінних. Таким чином, відтепер до складу інформаційних полів можуть входити не тільки інформаційні об'єкти, а й декларації типів та змінних при умові, що всі імена типів та змінних в них попарно різні.

Введемо спеціальне "неконкретизоване" значення $\#$, яке належить усім типам $T \in \Theta$. Будемо вважати, що в процесі декларації змінної $v \rightarrow \alpha$ паралельно утворюється ще й "неконкретизований" інформаційний об'єкт $v \underset{T}{\longmapsto} \#$.

Покладемо $pt(d)$, $st(d)$, $\mathit{var}(d)$, $sn(d)$, $dn(d)$, $nm(d)$ – множини усіх імен відповідно типів, задекларованих типів[9], змінних, статичних та динамічних змінних та об'єктів, які фігурують в полі d. У кожному інформаційному полі d виконується **правило декларації:**

1) кожне з задекларованих імен типів, як і змінних, є унікальним (поле містить тільки одну декларацію типу чи змінної з таким іменем);[10],

2) $pt(d) = st(d)$, $nm(d) = \mathit{var}(d)$, $sn(d) \cap dn(d) = \varnothing$, $\mathit{var}(d) = sn(d) \cup dn(d)$.

За правилом декларації усі імена похідних типів та об'єкти поля мають бути обов'язково задекларовані в ньому.

Покладемо V^U (X^U) сукупність усіх інформаційних полів з іменами із множини V (X) та значеннями з U, які задовольняють правилам декларації.

Конкретизація. Задекларована змінна в полі може бути *конкретизована* в ньому відповідним конструктором, але тільки значеннями свого типу. Якщо значення належить іншому типу, то воно має бути перетворене до типу змінної. Цей тип можна змінити, якщо оновити її декларацію. Конкретизація змінної типу T полягає в її означенні, тобто в утворенні об'єкту $v \underset{T}{\longmapsto} a$, $a \in T \setminus \{\#\}$ і долученні його до поля. Говорять, що після цього змінна отримала або має значення a. Якщо змінна була вже конкретизована в полі, то в процесі нової конкретизації чи зміні її типу попередній зв'язок втрачається. Перша конкретизація змінної називається її *ініціалізацією.* Іноді ініціалізація відбувається разом з декларацією змінної, при цьому в деяких випадках – за замовчуванням (наприклад, при обов'язковому початковому обнулінні арифметичної змінної). До ініціалізації вважається, що змінна має "неконкретизоване" значення $\#$. Позначимо $ini(d)$ сукупність усіх ініціалізованих змінних поля d. Тоді $ini(d) \subseteq var(d)$.

Рівні доступу. У зв'язку з конкретизацією важливими є можливість та характер доступу до значень змінних. Тому характеризуючи зв'язки, враховують не тільки тип значень, а й ті можливості (рівень) доступу до об'єктів, які вони (зв'язки) забезпечують. За рівнем доступу зв'язки розподіляються на такі, що дозволяють доступ: 1) тільки для дій *читання* значення об'єкту, 2) тільки для дій *запису* (заміни значення об'єкту), 3) *повний* доступ для читання і запису, 4) *повний* доступ до дубліката дозволяє дії читання та запису з дублікатом об'єкту, 5) *обмежений* доступ – доступ тільки для окремих дій, 6) *блокування* – неможливість будь-яких дій.

За рівнем доступу серед змінних виділяють *публічні* (за умовчуванням), *приватні* та *змінні-константи*. До перших (або їхніх частин) є повний доступ, до других – обмежений, треті доступні для читання. Дії читання та запису називаються *інтерфейсними.*

[9] Усі первинні типи, за означенням, належать до задекларованих.

[10] Щоправда, це не розповсюджується на імена типів та змінних – вони між собою збігатися можуть.

Оператори. У внутрішніх мовах вихідних систем інформаційні поля називаються даними, дії, що перетворюють дані – операторами, а операції та функції є діями, визначеними на даних. Результат застосування оператора до даного позначається як звичайними термами, так і термами вигляду. Говорять, що одне дане включає (містить) більше інформації, ніж інше дане або що воно розширює дане.

Зазвичай при роботі з даними увага фокусується не на всій їхній інформації, а тільки на актуальній на даний момент її частині, що міститься в певному фрагменті поля. Під фрагментом поля будемо розуміти його частину, яка відповідає певній сукупності його імен. Це стосується й операторів – вони змінюють, як правило, не всю інформацію в полі, а тільки ту, що належить певному скінченному його фрагменту. Такі актуальні фрагменти даних отримали назву $X-$*фреймів,* де *X* – фіксована сукупність імен об'єктів, яка ідентифікує ці фрагменти[11]. Наприклад, класичні $n-$арні операції визначені на кортежах $(a_1,...,a_n)$, які подаються у вихідних системах $X-$фреймами вигляду $[1\mapsto a_1,...,n\mapsto a_n]$, $X=\{1,...,n\}$. Області визначення (D_φ) та значень (E_φ) операторів вихідних систем складають певні відповідно $X-$та $Y-$фрейми та їхні розширення. Змінні з $X-$фрейму обов'язково мусять мати доступ як мінімум для читання, а з $Y-$фрейму – як мінімум для запису.

Найчастіше в даних змінюється значення якоїсь однієї змінної. Таку змінну здійснює оператор **присвоювання** $:=$, який має два аргументи: вираз le- задає ім'я змінної, функція re обчислює нове її значення. Нехай $d=[v_1\underset{T_1}{\longmapsto}a_1,...,v_n\underset{T_n}{\longmapsto}a_n]$ дане з області визначень функцій le та re і $v=le(d)$ для деякого імені $v\in V$, $a=re(d)$, $a\in T'$. Позначимо d_{-v} результат вилучення з даного d конкретизованої змінної v, якщо вона статична в d, а також і її декларації, якщо вона динамічна в d. Зазначимо, що $d_{-v}=d$, коли $v\notin var(d)$. Дія оператора присвоювання на даному d залежить від того, чи є ім'я v серед змінних даного d чи немає, а також від її типу T і статусу – статична вона чи динамічна. Якщо $v\in sn(d)$, то $le:=re|_d=[v\underset{T}{\longmapsto}a']\cup d_{-v}$, де 1) $a'=a$, коли $T'=T$, 2) $a'=\chi_{T',T}(a)$, коли $T'\neq T$ і $\chi_{T',T}:T'\to T-$ внутрішнє перетворення типу T' в тип T, 3) невизначено, коли $T'\neq T$ і такого перетворення не передбачено. У решті випадків $le:=re|_d=[v\to T',v\underset{T'}{\longmapsto}a]\cup\alpha_{-v}$.

Вираз le та функція re називаються відповідно *l-* та *r-виразами*[12]. Оператор присвоєння може використовуватись і для ініціалізації змінних.

Серед *r*-виразів виділяють найпростіші – розіменування $\backprime v$ змінних та змінних-констант. За означенням, якщо змінна v задекларована, але не ініціалізована у полі d, то $\backprime v|_d=\#$.

Нехай $X=\{x,y\}\subset V$ та $d=[x\to int\uparrow,y\to int\uparrow,z\to real\uparrow,x\mapsto 2,y\mapsto 5,z\mapsto\#]$. Розглянемо оператор-присвоювання $x:=x+2*y$. *l*-виразом у ньому є ім'я змінної x, *r*-виразом – функція, яка з довільного стану X -фрейму читає значення змінних x та y, підставляє їх у терм $x+2*y$ та обчислює його значення. Результатом застосування оператора $x:=x+2*y|_d$ буде поле $d'=[x\to int,y\to int,z\to real,x\mapsto 12,y\mapsto 5,z\mapsto\#]$.

У мовах програмування функції розіменування змінних подаються скорочено – просто як їхні імена, а відрізняють їх від імен змінних за контекстом. Наприклад, у попередньому операторі перше входження x є іменем змінної (*l*-виразом), а друге, як і входження інших змінних – операціями читання, при цьому $x|_d=2,y|_d=5,z|_d=\#$.

Змінити значення $Y-$фрейму $Y=\{v_1,...,v_n\}$ можна за допомогою *групового* оператора присвоєння $v_1,...,v_n:=re_1,...,re_n$. Дія цього оператора на даному d зводиться до одночасного застосування до нього операторів присвоєння $v_i:=re_i$, $1\le i\le n$. Наприклад, обміняти значеннями змінні x та y можна груповим присвоєнням $x,y:=y,x$.

Оператори присвоювання належать до фундаментальних засобів перетворення інформації в системах обробки інформації та їхніх моделях. Вони є прикладом важливого класу, так званих, операторів оновлення.

Нехай $X=\{x_1,...,x_n\}\subset V$ –довільний $X-$фрейм. Функція φ називається $X-$*функцією,* якщо $D_\varphi\subseteq X^U$. X -функція природним чином поширюється на всі дані, що містять даний $X-$фрейм.

Функція φ називається **еквітонним розширенням** $X-$функції ϕ, якщо $\forall d\in D_\phi\forall d'\in V^U(d\subseteq d'\Rightarrow\varphi(d')=\phi(d))$.

Еквітонно розширені $X-$функції зі значеннями в множині U будемо називати $X-$*арними операціями* (або просто *узагальненими* операціями, якщо фрейм X не фіксовано). Значення $X-$арної операції φ на довільному даному d' залежить лише від стану $X-$фрейму в ньому, тобто інформаційні об'єкти поза цим $X-$фреймом не є актуальними і жодним чином не впливають на значення $\varphi(d')$. Узагальнені операції обчислюють нові значення змінних в ході перетворень даних.

Нехай X,Y – певні сукупності імен. За областю значень серед еквітонно розширених $X-$функцій φ виділяють: 1) $X-Y-$*оператори*, якщо $E_\varphi\subset Y^U$; 2) $X-$*предикати*, якщо $E_\varphi\subseteq\{0,1,\#\}$; 3) $l-$*операції,* якщо $E_\varphi\subseteq V$.

Нехай ϕ – довільний $X-Y$ -оператор. Розглянемо еквітонно розширену $X-$функцію φ таку, яка для довільного поля d $\varphi(d)\overset{def}{=}d\nabla\phi(d')$. Функцію φ будемо називати $X-Y-$оператором **оновлення**.

[11] Іноді X-фреймом називають саму таку сукупність імен X.

[12] Від положення їх в операторі присвоювання: англ. – left (лівий) і – right (правий).

Зазначимо, що $X-\varnothing$ - оператори оновлення залишають інформаційне поле без змін ($d_{-\varnothing}=d$). Вже згадувані оператори присвоювання $x:=x+2*y$ та $x,y:=y,x$ є $\{x,y\}-\{x\}-$ та $\{x,y\}-\{x,y\}-$ операторами оновлення.

$X-Y-$ оператори оновлення здійснюють усі перетворення даних у вихідних системах[13]. Надалі $X-Y-$ оператори оновлення будемо називати просто $X-Y-$ операторами. Загальна схема перетворення інформаційного поля певним $X-Y-$ оператором виглядає так: 1) береться поточна інформація (значення інформаційних об'єктів) з $X-$ фрейму поля; 2) на її підставі отримується нова інформація за допомогою звичайних n -арних операцій; 3) нова інформація заноситься у $Y-$ фрейм. Дану схему можна дещо конкретизувати:

1) спочатку l -функції формують імена змінних із $X-$ та $Y-$ фреймів та можливо інших необхідних змінних поля;

2) операції читання за отриманими іменами знаходять поточні їхні значення в інформаційному полі і за допомогою n -арних операцій за цими значеннями виробляють нові значення змінних;

3) оператори присвоювання та більш складні оператори оновлення виробляють новий стан $Y-$ фрейму й усього інформаційного поля.

Таким чином, для обробки інформації у вихідній системах мають бути наявні такі елементи:

1) сукупність V^U інформаційних полів з іменами з V і значеннями з універсуму U ;

2) сукупності базових інтерфейсних l - та r -операцій та операцій запису для кожного типу об'єктів;

3) базова багатосортна система (U,Ω,I) з внутрішніми перетвореннями типів;

4) сукупність регулярних (рекурсивних) композицій для породження складених $X-$ арних операцій та таких же предикатів, а також $X-Y-$ операторів з елементів пунктів 2)-3).

Усі чотири наведені елементи об'єднуються в композиційну регулярну (рекурсивну) $\Omega-$ систему D *маніпулювання даними*. Вона описує модель станів виконавця та сукупність $X-Y-$ операторів, які він здатен обчислити. Незважаючи на свою апріорі функціональність, ця модель є алгоритмічною, що випливає з того, що всі регулярні та рекурсивні композиції зберігають алгоритмічність своїх аргументів, тобто якщо їхні аргументи конструктивні і обчислюються певними процедурами, то і результат є конструктивним і обчислюється певною процедурою. Усі похідні елементи функціональної алгебри системи D є алгоритмами відносно базових елементів з пунктів 2)-3), для подання яких використовується формальна мова L'_Ω звичайних та регулярних (рекурсивних) $\Omega-$ термів з сигнатурою Ω . Мова L'_Ω – аналогічна формальній мові вхідних систем L_Ω , але не використовує кванторів і основними її конструкціями є терми, а не формули. Вона збагачена синтаксичними правилами для побудови структурованих імен, системи типів, типізованих об'єктів, декларацій типів, статичних та динамічних змінних з різними рівнями доступу, інформаційних полів (даних). Сигнатура Ω розширена символами для подання базових інтерфейсних операцій для кожного з типів та сигнатурою Ω_χ операцій внутрішнього перетворення типів.

Двійка $\boldsymbol{S}_{out}=(D,L'_\Omega)$ називається **вихідною системою.**

Елементами КС є також функції кодування й декодування об'єктів, які пов'язують між собою вхідну й вихідну системи. Нехай S_{in} та S_{out} – певні вхідна й вихідна системи. Довільні типізовані однозначні часткові відображення $c_1:\mathrm{O}\to V^U$ та $c_2:V^U\to(\Omega_v)^U$ називаються відповідно *функціями кодування* й *декодування*. Функція кодування відображає допустимі оцінки вхідних змінних запитів КС у відповідні інформаційні поля, які підлягають обробці у вихідній системі, а функція декодування повертає у вхідну систему результати цієї обробки у вигляді певної оцінки вихідних змінних запитів. Дані функції можуть включати також процеси шифрації та дешифрації даних.

Ми готові дати загальне означення композиційної моделі комунікативних інформаційних систем.

Зафіксуємо певні функції кодування c_1 й декодування c_2 . Вихідна система S_{out} називається *коректною* відносно запиту $\Phi=\Phi(x_1,...,x_n,y_1,...,y_m)$ вхідної системи S_{in} , де $\{x_1,...,x_n\}$ – сукупність його вхідних змінних, а $\{y_1,...,y_m\}$ – сукупність вихідних, якщо у вихідній системі S_{out} знайдеться $X-Y$ -оператор φ такий, що

$$\forall\alpha\in\mathrm{O}\exists\beta\in(\Omega_v^I)^U\left(\left(\beta=c_2\left(\varphi\mid_{c_1(\alpha)}\right)\&_{i=1}^{n}x_i\alpha=a_i\&_{j=1}^{m}y_j\beta=b_j\right)\supset\Phi(a_1,...,a_n,b_1,...,b_m)\right)\ (*).$$

Умова (*) означає: якщо закодувати будь-який варіант вхідних даних запиту Φ і застосувати до отриманого інформаційного поля оператор O , а потім узяти результат роботи O і розкодувати його, то після підстановки в запит Φ вхідних і отриманих вихідних значень змінних x_i та y_j запит буде задоволено. Назвемо оператор O *реалізацією* запиту Φ . Вихідна система S_{out} називається *коректною* по відношенню до вхідної системи S_{in} , якщо вона коректна відносно всіх запитів S_{in} .

Четвірка $\mathrm{S}=(\mathrm{S}_{in},\mathrm{S}_{out},c_1,c_2)$, де вихідна система S_{out} є коректною по відношенню до вхідної системи S_{in} , називається **композиційною моделлю** комунікативних інформаційних систем.

Висновки. В роботі введено поняття композиційної моделі комунікативних систем. Ця модель є сугубо алгебричною і описує функціональну семантику КС. Базовими її поняттями є поняття формалізованої композиційної рекурсивної $\Omega-$ системи з внутрішніми перетвореннями типів, інформаційного об'єкту, інформаційного поля (даного) та $X-Y-$ оператора оновлення даних. Для конструктивізації цієї моделі необхідно доповнити вихідні системи конструктивною операційною семантикою.

Список використаних джерел

1. Зубенко В. В, Омельчук Л. Л. Програмування: Навчальний посібник. – К., 2011.
2. Зубенко В.В. Про комунікативну платформу інформатики // В кн.: Теоретичні та прикладні аспекти побудови програмних систем. – К., 2012.

[13] Це справедливо тільки для вихідних систем без, так званих, побічних ефектів.

3. Зубенко В.В. Про комунікативну інформатику // Вісник КНУ ім.Тараса Шевченка, сер. фіз.-мат.наук, вип.4. – 2012.
4. Нікітченко М.С. Шкільняк С. С. Математична логіка та теорія алгоритмів. – К.: ВПЦ "Київ. ун-т", 2008.
5. Зубенко В.В. Элементарные программные алгебры: Автореф. дисс. на соиск. степ. канд.физ.-мат. наук / Зубенко В.В. – К., 1986.
6. Зубенко В.В. Элементарная схематология //В кн.: Системное и теоретичское програмирование. – Кишинев, 1983.
7. Клини С. Математика метаматематики. – М., 1956.
8. Манна З. Теория неподвижной точки программ // Кибернетический сборник. Новая серия. – М.: Мир, 1978. – Вып. 15.
9. Мальцев А. И. Алгебраические системы. – М.: Наука, 1970.
10. Нікітченко М.С.Теорія програмування. Частина І: Навчальний посібник. – Ніжин, 2010.
11. Плюшкявичус Р.А., и др. О прикладных логиках // Кибернетика, №2. – 1979.
12. Редько В.Н., Редько І.В., Гришко Н.В. Дескриптивні основи сутнісної платформи//Проблеми програмування, № 2-3.
13. Nikitchenko M. Chentsov A. Basics of Intensionalized Data: Presets, Sets and Nominats // Computer Science Journal of Moldova, vol. 20, no.3 (60). – 2012.

Надійшла до редколегії 08.04.13

В. Зубенко, канд. физ.-мат. наук, доц.
КНУ имени Тараса Шевченко, Киев

КОМПОЗИЦИОННАЯ МОДЕЛЬ КОММУНИКАТИВНЫХ ИНФОРМАЦИОННЫХ СИСТЕМ

Рассматривается алгебраическая модель коммуникативных информационных систем.
Ключевые слова: композиционная модель, коммуникативная система, конструктор правило декларации, эквитонное расширения.

V. Zubenko, PhD. Sci. Sciences,
Kyiv National Taras Shevchenko University, Kiev

COMPOSITIONAL COMMUNICATIVE MODEL INFORMATION SYSTEMS

The algebraic model of communicative informations systems is considered.
Keywords: composite model communication system designer usually Declaration equitone expansion.

УДК 519.852:519.876

В. Кудін, д-р техн. наук
КНУ імені Тараса Шевченка, Київ

ПРО СХЕМИ МЕТОДУ БАЗИСНИХ МАТРИЦЬ АНАЛІЗУ ЛІНІЙНИХ СИСТЕМ

Проаналізовано варіанти методу базисних матриць для розв'язання двоїстої пари задач лінійного програмування з однотипними обмеженнями. На прикладі матричної гри у змішаних стратегіях.
Ключові слова: метод базисних матриць пряма и двоїстна задача.

Вступ. Для багатьох практичних задач встановлення існування та знаходження оптимального розв'язку, наприклад, задачі лінійного програмування (матричної гри) є лише однією із задач дослідження [1-6]. Виникає і ряд інших задач такі як, дослідження властивостей розв'язків. Від вибору варіанту симплекс-методу та його алгоритму залежить ефективність розв'язання таких задач вцілому. Слід зазначити, що ряд задач потребують аналізу як прямої, так і двоїстої задачі, наприклад, матричні ігри у змішаних стратегіях [5-6].

Встановлено [1,5], як побудувати пару двоїстих задач лінійного програмування з однотипними обмеженнями, розв'язок яких визначає оптимальні стратегії гравців заданої матричної гри. Параметри задач лінійного програмування, що відповідають заданій матричній грі, вибираються в процесі конструктивного доведення основної теореми теорії ігор [1].

Постановка задачі. В даній роботі розглянуто варіанти алгоритмічних схем методу базисних матриць [7] для задач з однотипними обмеженнями (саме до таких відносяться матричні ігри у змішаних стратегіях, як задачі лінійного програмування).

В процесі доведення основної теореми теорії ігор для матричної гри з платіжною матрицею $A = \left\| a_{ij} \right\|_{i=1,\,j=1}^{m,n}$ $(a_{ij} > 0)$ було встановлено зв'язок з парою двоїстих задач лінійного програмування типу (1)-(3) та (4)-(6) з однотипними обмеженнями, які наведені нижче.

Пряма задача:

$$\max \sum_{j=1}^{n} c_j x_j, \tag{1}$$

за умов:

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n \le b_1; \\ a_{21}x_1 + a_{22}x_2 + \ldots + a_{2n}x_n \le b_2; \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ a_{m1}x_1 + a_{m2}x_2 + \ldots + a_{mn}x_n \le b_m; \end{cases} \tag{2}$$

$$x_j \ge 0, \;\; j = \overline{1, n} \;\; . \tag{3}$$

Або в матричному вигляді введемо в розгляд задачу [1] лінійного програмування вигляду:

$$\min Cx,$$
$$Ax \le B^T,$$
$$x \ge 0,$$

де $C = (c_1, c_2, \ldots, c_n)$, $x = (x_1, x_2, \ldots, x_n)^T$, $B = (b_1, b_2, \ldots, b_m)$. Вважаємо, що модель виду (1)–(3) має матрицю обмежень А в якій кількість стовпців більш ніж рядків ("довга").