НАЦІОНАЛЬНА АКАДЕМІЯ НАУК УКРАЇНИ
ІНСТИТУТ ПРОГРАМНИХ СИСТЕМ

ISSN 1727-4907

П Р О Б Л Е М И ПРОГРАМУВАННЯ

НАУКОВИЙ ЖУРНАЛ

P R O B L E M S IN PROGRAMMING

SCIENTIFIC JOURNAL

2017

№ 4

Теми випуску:

- ***Моделі та засоби паралельних і розподілених програм***
- ***Інструментальні засоби та середовища програмування***
- ***Формальні методи розробки програмного забезпечення***
- ***Експертні та інтелектуальні інформаційні системи***
- ***Програмні системи захисту інформації***
- ***Прикладні засоби програмування та програмне забезпечення***

НАЦІОНАЛЬНА АКАДЕМІЯ НАУК УКРАЇНИ
ІНСТИТУТ ПРОГРАМНИХ СИСТЕМ

П Р О Б Л Е М И ПРОГРАМУВАННЯ

науковий журнал

Головний редактор
Андон Пилип Іларіонович
академік НАН України,
директор Інституту програмних систем
НАН України
✉ Інститут програмних систем
НАН України
проспект Академіка Глушкова, 40, корп. 5
03187, Київ-187
☎ Тел.+380 (44) 526 5507
💻 E-mail: andon@isofts.kiev.ua
http://www.pp.isofts.kiev.ua

Адреса для кореспонденції

✉ Інститут програмних систем
НАН України
Проспект Академіка Глушкова, 40
03187, Київ-187

☎ Тел.: +380 (44) 526 5065
Факс: +380 (44) 526 6263
💻 E-mail: iss@isofts.kiev.ua

Редактор *В.П. Замула*
Комп'ютерна верстка *В.П. Замула*

Підписано до друку 20.11.2017. Формат 60x84/8. Папір офс. Ум. друк. арк. 15,11.
Обл.-вид. арк. 12,41. Тираж 120 прим. Ціна договірна. Замовл.

Віддруковано ВД «Академперіодика» НАН України
вул. Терещенківська, 4, м. Київ, 01004

Свідоцтво суб'єкта видавничої справи ДК № 544 від 27.07.2001

НАЦІОНАЛЬНА АКАДЕМІЯ НАУК УКРАЇНИ
ІНСТИТУТ ПРОГРАМНИХ СИСТЕМ

ПРОБЛЕМИ ПРОГРАМУВАННЯ

науковий журнал

№ 4 жовтень-грудень 2017

Заснований у березні 1999 р.

ЗМІСТ

Свідоцтво про державну реєстрацію КВ № 7490 від 01.07.2003

Науковий журнал "Проблеми програмування" занесений до переліку наукових фахових видань України, в яких можуть публікуватися основні результати дисертаційних робіт.

ISSN 1727-4907

NATIONAL ACADEMY OF SCIENCES OF UKRAINE
INSTITUTE OF SOFTWARE SYSTEMS

PROBLEMS IN PROGRAMMING

scientific journal

№ 4 October – December 2017

Founded in March, 1999

CONTENTS

УДК 681.3.06

С.В. Єршов, Р.М. Пономаренко

МЕТОД ПОБУДОВИ ПАРАЛЕЛЬНИХ СИСТЕМ НЕЧІТКОГО ЛОГІЧНОГО ВИВЕДЕННЯ НА ОСНОВІ ГРАФІЧНИХ ПРИСКОРЮВАЧІВ

Розроблено та обґрунтовано метод побудови паралельних систем нечіткого логічного виведення Такагі – Сугено на основі графічних прискорювачів Nvidia. Розроблено інтелектуальну систему оцінювання якості стартапів на основі ієрархічних систем нечіткого логічного виведення Такагі – Сугено. Встановлено оцінки прискорення систем нечіткого виведення, що побудовані за зазначеним методом, здійснено порівняльну характеристику з варіантами нечітких систем, що реалізовані на базі технології паралельного програмування MPI.

Ключові слова: інтелектуальна система, ярусно-паралельна форма обчислень, графічні прискорювачі, система Такагі – Сугено, нечітка логіка, CUDA.

Вступ

У даний час нечітка логіка широко застосовується у багатьох сферах при створенні інтелектуальних програмних систем та технологій, а саме, діагностичних систем, систем розпізнавання образів, інтелектуальних мультиагентних систем, що забезпечують підтримку прийняття складних рішень в умовах невизначеності [1, 2]. Методи нечіткого виведення, що використовуються у різних інтелектуальних системах, грунтуються на базах знань, які задаються у вигляді сукупності нечітких предикатних правил. Запропоновано кілька методів нечіткого логічного виведення, які відрізняються операторами застосування та композиції нечітких множин, що містяться у правилах, методами приведення отриманих нечітких значень до чітких чисел, таких як методи Мамдані, Ларсена, Такагі – Сугено та інші [2]. Зокрема, нечіткий метод Такагі – Сугено ґрунтується на базі знань, що представляється наборами правил «якщо-то», які можуть бути узагальнені наступним чином:

R_j : якщо $x_1 \in A_{1j}$ та $x_2 \in A_{2j}$ та …

… та $x_n \in A_{nj}$ то $y = g_j(x_1, x_2, ..., x_n)$,

$j = 1,2,...,N$,

де g_j – чітка функція від x_i.

Проте, при побудові інтелектуальних систем існує проблема, що заважає будувати нечіткі системи великої розмірності, а саме, при збільшенні кількості вхідних значень у системі значно збільшується кількість правил. Система з нечіткими правилами (СНП) з n вхідними змінними та l лінгвістичними термами на одну змінну містить l^n правил [3]. Очевидно, що побудувати та перевірити таку кількість правил для систем, що мають кілька десятків вхідних змінних, не під силу навіть досвідченим експертам. Тому для вирішення задач великої складності та розмірності запропоновані ієрархічні системи нечіткого логічного виведення, що покликані зменшити число правил системи та відповідно збільшити складність систем логічного виведення на основі нечіткої математики.

Запропоновано різні варіанти введення ієрархічних структур для систем з нечіткими правилами [3–7]. Правила або змінні можуть бути розподілені на різних рівнях відповідно до їх специфічності, деталізації, релевантності тощо. Визначення ієрархічних нечітких систем як методу вирішення задач нечіткого логічного виведення з вищим рівнем складності, ніж ті, для яких зазвичай призначені звичайні СНП, базується на застосуванні певного способу декомпозиції, що породжує ієрархію систем нижчої складності [3].

Перший метод побудови ієрархічних структур визначає такі структури на основі пріоритетності правил таким чином,

що правила з неоднаковим рівнем специфічності отримують неоднаковий пріоритет. При цьому, правила, що мають більш високий пріоритет, є більш специфічними [4, 5]. Загальне правило застосовується лише в тих випадках, коли не існує відповідного специфічного правила. У цьому випадку ієрархія утворюється в результаті застосування конкретного механізму імплікації, який застосовує правила, беручи до уваги їх пріоритет. Для розробки ієрархічних СНП правила мають бути згруповані за певними рівнями пріоритетів.

Інший варіант полягає у тому, щоб розглянути ієрархію розбиття вхідного простору змінних з різною деталізацією [6]. Нечіткі розбиття описують набори лінгвістичних термів, пов'язаних з кожною змінною в лінгвістичних правилах, а також функції належності, що визначають семантику зазначених лінгвістичних термів. З цієї точки зору, СНП може бути структуровано за шарами, де кожен шар містить нечіткі розбиття з окремою деталізацією, а також правила, що використовують ці нечіткі розбиття. Зазвичай, кожне розбиття у певному шарі має однакове число нечітких термів. У цьому випадку правила, що належать до різних шарів мають різний рівень деталізації, що певним чином пов'язано з вищезазначеною ідеєю специфічності.

Зовсім інший метод – це введення декомпозиції на рівні змінних. У цьому випадку вхідний простір розбивається на підпростори нижчої розмірності, і кожна вхідна змінна розглядається тільки на певному рівні ієрархії. Результатом є каскадна структура СНП, в якій, крім підмножини вхідних змінних, вихід кожного рівня розглядається як один з можливих входів наступного рівня [7]. У результаті нечітка система розкладається на скінчене число підсистем зменшеної складності, усуваючи необхідність застосування надскладної машини логічного виведення. Така декомпозиція зазвичай виражається як спосіб усунути проблеми, що виникають внаслідок так званого зростання розмірності, тобто експоненціального зростання кількості правил, пов'язаних з кількістю змінних нечіткої системи.

У такому підході до ієрархічних СНП змінні (і правила) поділяються на різні рівні таким чином, що найбільш впливові змінні обрані як вхідні змінні на першому рівні, наступні найбільш важливі змінні вибираються як вхідні змінні на другому рівні тощо. Вихідні змінні окремих рівнів вводяться як вхідні змінні на наступному рівні.

З такою структурою правила на першому рівні СНП мають структуру аналогічну будь-якій СНП типу Такагі – Сугено або Мамдані, але на k-му рівні ($k > 1$), правила містять як вхід вихідні значення попереднього рівня

$$IF\ X_{Nk+1}=LT_{Nk+1}\ and\ .\ .\ .\ and\ X_{Nk+nk}=LT_{Nk+nk}$$

$$and\ Ok\text{-}1=LTO_{k\text{-}1}\ THEN\ O_k=LT_{Ok},$$

де значення N_k визначає кількість вхідних змінних, що розглядаються на попередніх рівнях

$$N_k = \sum_{t=1}^{k-1} n_t ,$$

де n_t – кількість змінних системи, застосованих на рівні t. Змінна Ok представляє вихідне значення k-го рівня ієрархії. Всі вихідні значення – це проміжні змінні, за винятком виходу останнього рівня Y (загальне вихідне значення нечіткої системи).

За допомогою зазначеної структури показано [7], що кількість правил у повній базі правил може бути представлена як лінійна функція від числа змінних, тоді як у звичайних (неієрархічних) СНП достатня кількість правил визначається експоненціальною функцією від числа змінних.

Однак, застосування вищезазначеного підходу представлено в [2] тільки для систем нечіткого виведення, для яких кількість рівнів не перевищує 2–3, а кількість вхідних змінних – не більше кількох десятків відповідно. Подальше збільшення складності ієрархічних систем нечіткого виведення вимагає розв'язання проблеми, що пов'язана із значним збільшенням часу виконання логічного виведення

в складних моделях з великою кількістю ієрархічних нечітких систем.

Для подолання зазначеної проблеми запропоновано методи паралельної роботи нечітких ієрархічних систем для зменшення часу їх виконання з використанням технології MPI [8, 9].

Метою даної роботи є розробка методу побудови паралельних алгоритмів нечіткого логічного виведення на базі сучасної архітектури графічних прискорювачів та технології CUDA для отримання суттєво більшого прискорення, ніж на основі MPI, та встановлення показників ефективності складних ієрархічних систем нечіткого логічного виведення Такагі – Сугено, побудованих на основі ярусно-паралельних моделей обчислень.

Ярусно-паралельна модель обчислень

У роботах [8, 9] розроблено та обґрунтовано алгоритми ярусно-паралельної схеми обчислень для ієрархічних систем нечіткого логічного виведення.

В даній роботі для здійснення нечіткого логічного виведення на графічних прискорювачах, за основу було взято саме ярусно-паралельну форму обчислень, проте, як буде показано далі, в даній модифікації буде реалізовано внутрішній паралелізм кожної окремої вершини графа залежностей ієрархічної системи.

Побудова графа залежностей є основою аналізу паралельних алгоритмів. Будь-який паралельний алгоритм можна представити у вигляді орієнтованого ациклічного мультиграфа $G=(V,E)$, де V – множина вершин, а E – множина ребер графа. Вершини представляють множину операцій алгоритму, кожна операція алгоритму має власні вхідні аргументи, та не може бути виконана до моменту виконання всіх операцій-постачальників аргументів для неї.

Нехай u та v – пара вершин графа. Наприклад, припустимо, що вершина v має постачати обчислені аргументи вершині u. Тоді існує ребро графа від вершини v до вершини u. Якщо ж вершина v не постачає аргументів (змінних) для вершини u, то дугу не проводимо. Вершини, що не мають вхідних (вихідних) дуг називаються вхідними (вихідними) вершинами відповідно, а весь граф – графом алгоритму. У випадку, коли дуги (залежності між операціями) відсутні, аргументи операцій постачаються із вхідних даних. Під операцією розуміємо множину логічно взаємозв'язаних інструкцій програми [10].

Введемо поняття ярусу паралельної форми алгоритму. Нехай задано орієнтований ациклічний граф, який має n вершин. Кожну вершину даного графа можна помітити таким числом, меншим за n, що якщо з вершини з індексом i іде дуга з індексом j, то $i<j$ [10]. Група вершин, які мають однаковий індекс, називається ярусом паралельної алгоритму, а число вершин i-ї групи – шириною ярусу. Наступний ярус не розпочне роботу до тих пір, поки не закінчиться виконання останньої операції попереднього ярусу, тобто не буде проведена ярусна синхронізація.

Паралельні алгоритми, які досліджуються в даній роботі, відповідають ярусно-паралельній формі алгоритмів [10] (рис. 1). У даному випадку N=7, $s=3$. Алгоритм ярусно-паралельної форми виконується наступним чином:

1) обчислюються результати вершин першого ярусу (i =1) до тих пір, поки не відпрацюють всі до останньої вершини даної групи, тобто не буде проведена синхронізація роботи вершин ярусу 1;

2) попередній ярус відкидається, повторюються операції з першого пункту, але вже для ярусу $i=i+1$;

3) повторюється (2) до тих пір, поки $i<=s$. При досягненні значення $i>s$ процес обчислень за ярусно-паралельною формою вважається завершеним.

Процес синхронізації за ярусами є обов'язковою умовою ярусно-паралельної форми обчислень. Проте, якщо блоки правил у вершинах графа залежностей мають приблизно однаковий час роботи, на прискорення всієї ієрархічної системи синхронізація суттєво не впливає.

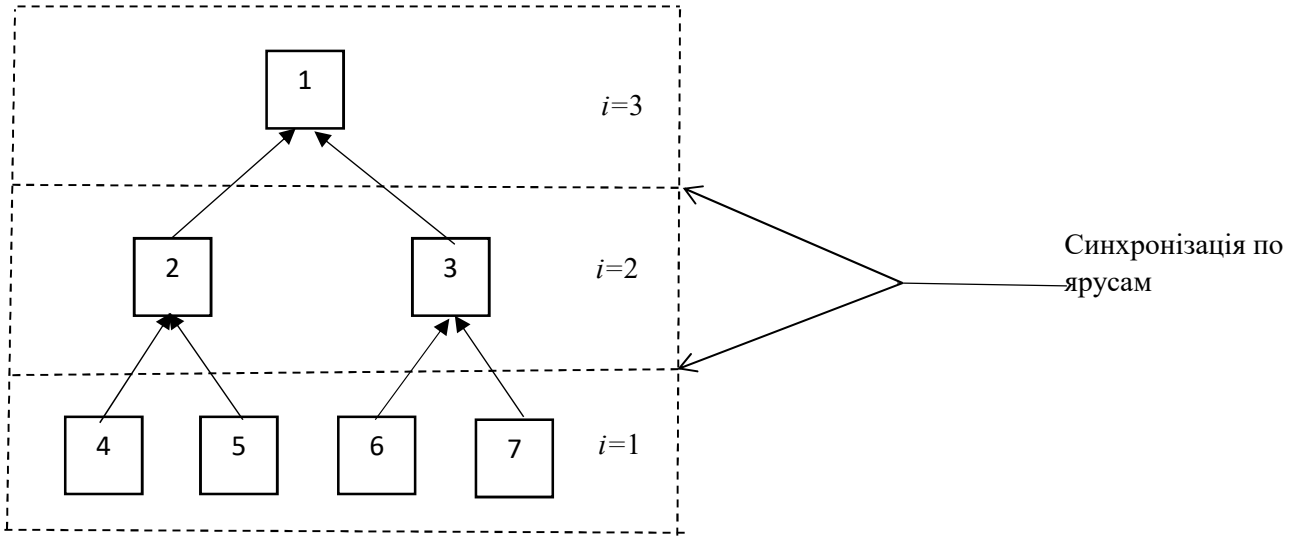


Рис. 1. Схема ярусно-паралельного алгоритму

Паралельний метод нечіткого логічного виведення Такагі – Сугено на основі архітектури CUDA

Технологія CUDA [11] – це технологія, що дозволяє використовувати відеокарту як обчислювальний пристрій з метою прискорення виконання обчислень. Керування всією програмою залишається за процесором, на GPU виконуються тільки окремі функції, команду до запуску яких надає центральний процесор. Завантаження та вивантаження даних із пам'яті GPU до оперативної пам'яті також покладено на неї.

Серед труднощів роботи з технологією CUDA можна виділити обмеження розмірності вкладених циклів. Методи усунення даної проблеми запропоновано в [12].

Паралельний метод нечіткого логічного виведення Такагі – Сугено грунтується на ярусно-паралельній схемі обчислень.

Побудова ярусно-паралельної схеми обчислень. Першим етапом є побудова ярусно-паралельної схеми обчислень для ієрархічної нечіткої системи. У випадку ієрархічних нечітких систем множиною вершин вважатимемо множину нечітких блоків правил. Зв'язки між вершинами (залежності між нечіткими блоками правил, елементарними системами нечіткого виведення), або ребра графу залежностей задаються на етапі введення вхідних даних.

Ярусно-паралельна схема містить наступні етапи обчислень (i – номер ярусу, s – кількість ярусів, n – кількість вершин у графі залежностей):

1) знаходимо всі вершини, що не мають дуг (висячі), та записуємо їх індекси до списку групи вершин першого ярусу (i=1). Змінну s інкрементуємо на одиницю;

2) відкидаємо з графу вершини, знайдені на попередньому етапі та шукаємо за попереднім принципом вже нову групу вершин, але індексуємо дану групу як i=i+1 та s= s+1;

3) повторюємо операцію (2) доти, поки $n > 0$, тобто поки не будуть помічені всі вершини графу.

В глобальну пам'ять відеокарти мають бути скопійовані наступні дані:

- кількість вершин графа N;
- одномірний масив об'єктів вершин графа $V = \{v_i\}, i = \overline{1,N}$. У даному випадку вершинами є об'єкти, що моделюють поведінку кожної елементарної системи нечіткого логічного виведення Такагі – Сугено;
- кількість ярусів графа s;
- одномірний масив значень ширини кожного ярусу $L = \{l_i\}, i = \overline{1,s}$;
- одномірний масив номерів вершин у тому порядку, в якому вони були розділені на групи $M = \{m_i\}, i = \overline{1,N}$, як показано на рис. 2.

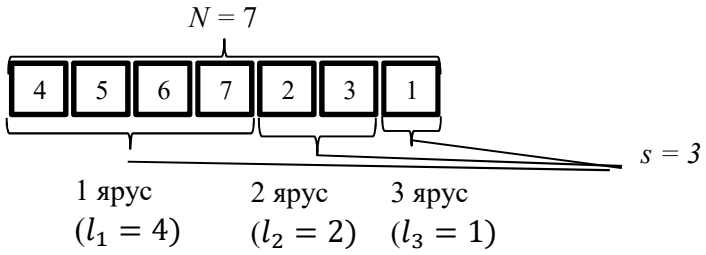


Рис. 2. Масив послідовності номерів вершин у порядку слідування груп

Індексація вершин графу. Для копіювання вхідних даних в пам'ять GPU, об'єкти-вершини ярусно-паралельної схеми зберігаються в одномірному масиві. Тому виникає проблема індексації вершин з урахуванням номеру яруса. Далі представлено метод індексації вершин у масивах, в яких зберігаються графи залежностей блоків правил нечітких систем та, відповідно, спосіб визначення індексів вершин кожного ярусу.

Нехай N – кількість вершин графа (кількість блоків правил в ярусі), s – кількість ярусів. Тоді, якщо кількість систем в i-му ярусі – l_i, $i = \overline{1,s}$, то маємо наступне співвідношення:

$$\sum_{i=1}^{s} l_i = N .$$

Нехай V – масив вершин, M – масив номерів вершин. Визначимо для кожного ярусу i групу вершин $D_i = \left\{d_p^{(i)}\right\}$, $p = \overline{1,l_i}$ у масиві V. Тоді кожна вершина i-го ярусу в масиві V індексується за наступною формулою:

$$d_p^{(i)} = M_{z_p^{(i)}}, \; z_p^{(i)} = p + \sum_{k=1}^{i} l_k,$$

де i – номер ярусу, p – номер вершини в i-му ярусі, $z_p^{(i)}$ – індекс номера вершини p i-го ярусу в масиві M. $M_{z_p^{(i)}}$ – це обчислений індекс вершини $d_p^{(i)}$ в масиві V.

Внутрішній паралелізм. Кожна вершина (елементарна нечітка система) містить множину об'єктів для здійснення нечіткого виведення за певною множиною вхідних значень. Особливістю обчислення кожного правила є можливість їх активізації у будь-якій послідовності, тобто вихідні значення правил можуть обчислюватися незалежно одне від одного. Таким чином, обчислення значень за всіма правилами в межах кожної елементарної нечіткої системи (кожної вершини) можна виконувати паралельно. В цьому випадку кожна вершина v_i являє собою окремий підграф, а кожне правило можна розглядати як окрему вершину підграфа v_i. Нехай кожна v_i вершина графа G – орієнтований ациклічний мультиграф $R = \{H, Y\}$, де H – множина вершин, а Y – множина ребер графа R. $H = \{h_j\}, j = \overline{1, M_i + 1}$, де M_i – кількість правил i-ї вершини графа G, а $M_i + 1$ вершиною є функція нечіткого логічного вводу. $Y = \{y_d\}$, $d = \overline{1, M_i}$, де M_i – кількість правил i-ї вершини графа G. У кожному графі R від кожної вершини $h_j \ldots h_{M_i}$ проведені дуги до вершини h_{M_i+1} (рис. 3). Такий граф має назву канонічної паралельної форми [10] і відповідно значення правил у вершинах $h_j \ldots h_{M_i}$ можуть бути обчислені паралельно в межах одного ярусу.

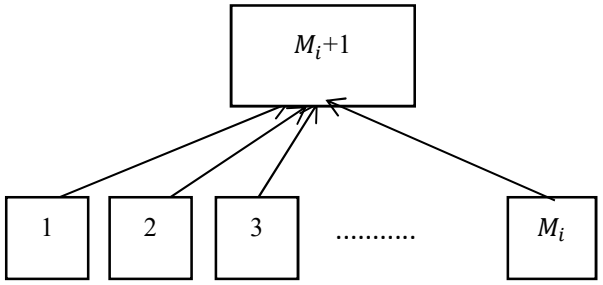


Рис. 3. Представлення графа R

Паралельне виведення в ієрархічних нечітких системах. Найважливішим етапом є здійснення обчислень результуючих значень ієрархічної системи на графічних прискорювачах Nvidia. Графічний процесор складається з сітки Grid, яка поділяється на блоки (blocks). Кожний блок, у свою чергу містить нитки (threads). Розподіл обчислень в ієрархічних нечітких системах між блоками та нитками показано на рис. 4.

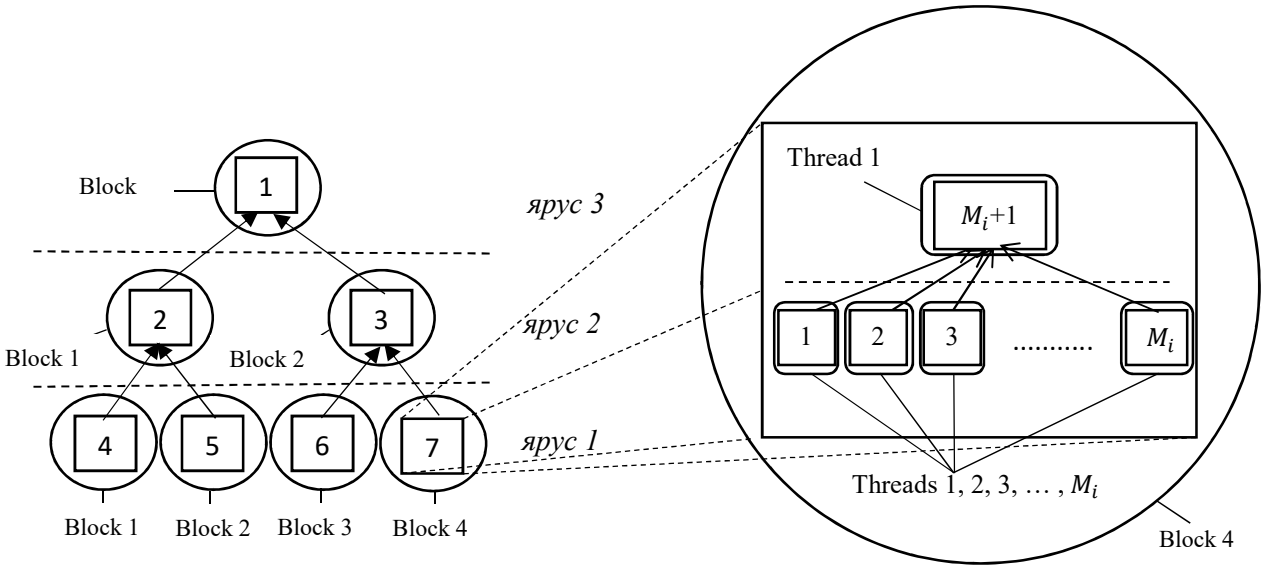


Рис. 4. Схема розподілу обчислень в ієрархічних нечітких системах на основі GPU

Основна ідея застосування технології CUDA при побудові ієрархічних систем нечіткого логічного виведення полягає у тому, що результуюче значення кожної елементарної нечіткої системи (кожної вершини v_i графа G) обчислюється окремим блоком правил, у якому, в свою чергу, обробка кожного правила (кожної вершини h_j графа R) системи розподіляється за нитками зазначеного блоку.

Таким чином, для досягнення більшого прискорення логічного виведення в ієрархічних нечітких системах Такагі – Сугено ефективно реалізовано як паралельне обчислення результатів блоків правил, так і можливість внутрішнього паралелізму за правилами в кожному окремому блоці (рис. 4).

Відеокарта архітектури 6.1. Pascal, яка використовувалась для експериментальних обчислень побудованої програмної системи, має 768 процесорних ядер, що обчислюють виділені блоки та нитки в них, що дозволяє ефективно реалізувати внутрішній паралелізм при нечіткому виведенні. Внутрішній паралелізм, у свою чергу, мінімізує втрати процесорних ресурсів, максимально навантажуючи ядра графічного процесора.

Залучення великої кількості процесорів на CPU вимагає залучення суперкомп'ютера, проте має суттєві обмеження, що пов'язані з відсутністю спільної пам'яті та додатковими витратами часу на обмін даними між процесорами розподіленої системи.

Оцінка прискорення нечіткого логічного виведення в ієрархічних системах. При використанні графічних прискорювачів досягається значне прискорення обчислень для ієрархічних нечітких систем, яке набагато більше ніж для паралельних обчислень таких саме систем на основі MPI (рис. 5). Такий результат отримано саме за рахунок масового паралелізму, використання загальної пам'яті, а особливо за рахунок розпаралелювання всередині кожної системи за правилами, тобто внутрішнього паралелізму. Зазначений результат експериментально підтверджено для ієрархічних нечітких систем, залежності входів-виходів між окремими блоками правил якого задаються графом великої розмірності (N=1000 блоків) та генеруються випадковим чином. Кількість ребер q задається як $q = 4 * n$, де n – кількість згенерованих вершин графа.

Очевидно, що зі збільшенням кількості систем збільшується і прискорення (рис. 5). Для ієрархічних нечітких систем, що складаються із 1000 елементарних систем, прискорення на основі технології CUDA – більше ніж у 500 разів у порівнянні з послідовним варіантом. Варіант програми на базі MPI на 24 процесорах прискорює нечітке виведення не більше, ніж у 20 раз, але в 25 разів менше за GPU.

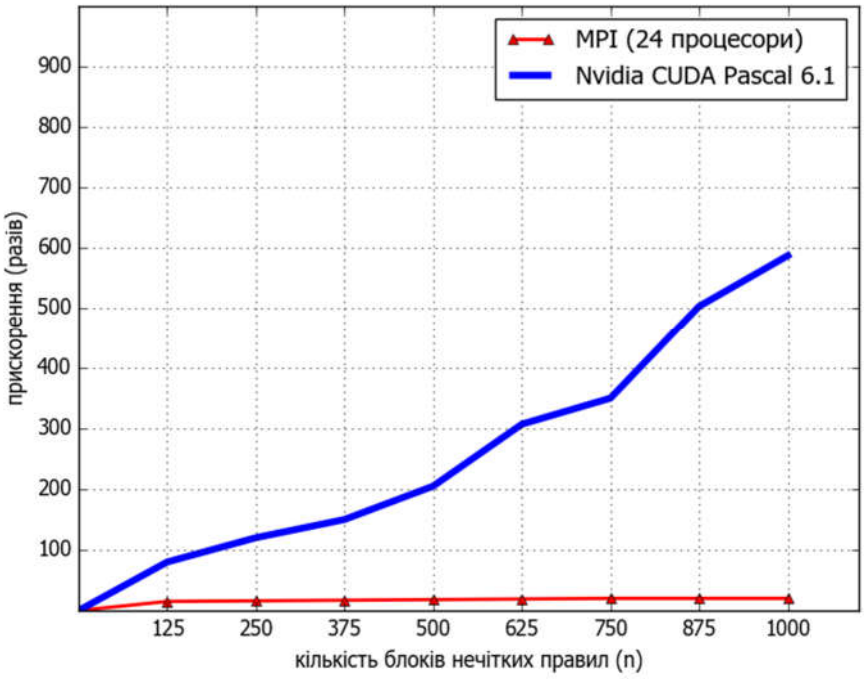


Рис. 5. Прискорення нечіткого виведення для ієрархічних нечітких систем

Побудова інтелектуальної системи оцінки привабливості стартапів на основі ієрархічної нечіткої системи

Метою даної інтелектуальної системи є аналіз та оцінювання якості стартапів за різними показниками та отримання комплексної оцінки привабливості даних стартапів. Під привабливістю стартапу часто розуміють доцільність його подальшого запуску, фінансування та роботи над ним [13].

На сьогоднішній день вищезазначена задача є дуже актуальною. Невеликі інноваційні проекти все більше займають місце в малому бізнесі, проте виникає необхідність попередньої оцінки проекту на предмет прибутковості та зменшення ризиків для потенційних інвесторів та розробників таких проектів.

Базу знань у вигляді нечітких правил розробляє експерт або група експертів з даної предметної області. Програмна система ґрунтується на лінгвістичних правилах, таких як, *if x is high then y is good.* Спеціальний програмний модуль системи обробляє лінгвістичні вирази, переводить їх до представлення у вигляді нечітких множин.

Всього ієрархічна нечітка система оцінки привабливості стартапів містить 162 правила. Враховуючи, що кожна з 15 вхідних змінних може бути представлена одним з трьох лінгвістичних значень, побудова відповідної повної бази для системи без ієрархії вимагає створення 3^{15} нечітких правил. Очевидно, що побудувати та перевірити таку кількість правил для аналогічної, але однорівневої системи не під силу навіть досвідченим експертам.

Зазначимо, що розробка ієрархічних систем нечіткого логічного виведення дозволяє значно збільшити кількість факторів, що впливають на оцінку привабливості стартапів, дозволяючи прийняти більш об'єктивне рішення щодо подальшого фінансування. При цьому кількість правил, які має написати експерт, за рахунок декомпозиції, значно зменшується. Граф залежностей між блоками нечітких правил, їх вхідні та вихідні змінні показано на рис. 6.

На рис. 7 показано лінгвістичні терми (Високий, Середній, Низький) та відповідні функції належності, які використовуються в інтелектуальній системі оцінки привабливості стартапів. Використані трикутні функції належності на інтервалі можливих значень від одного до п'яти (рис. 7).

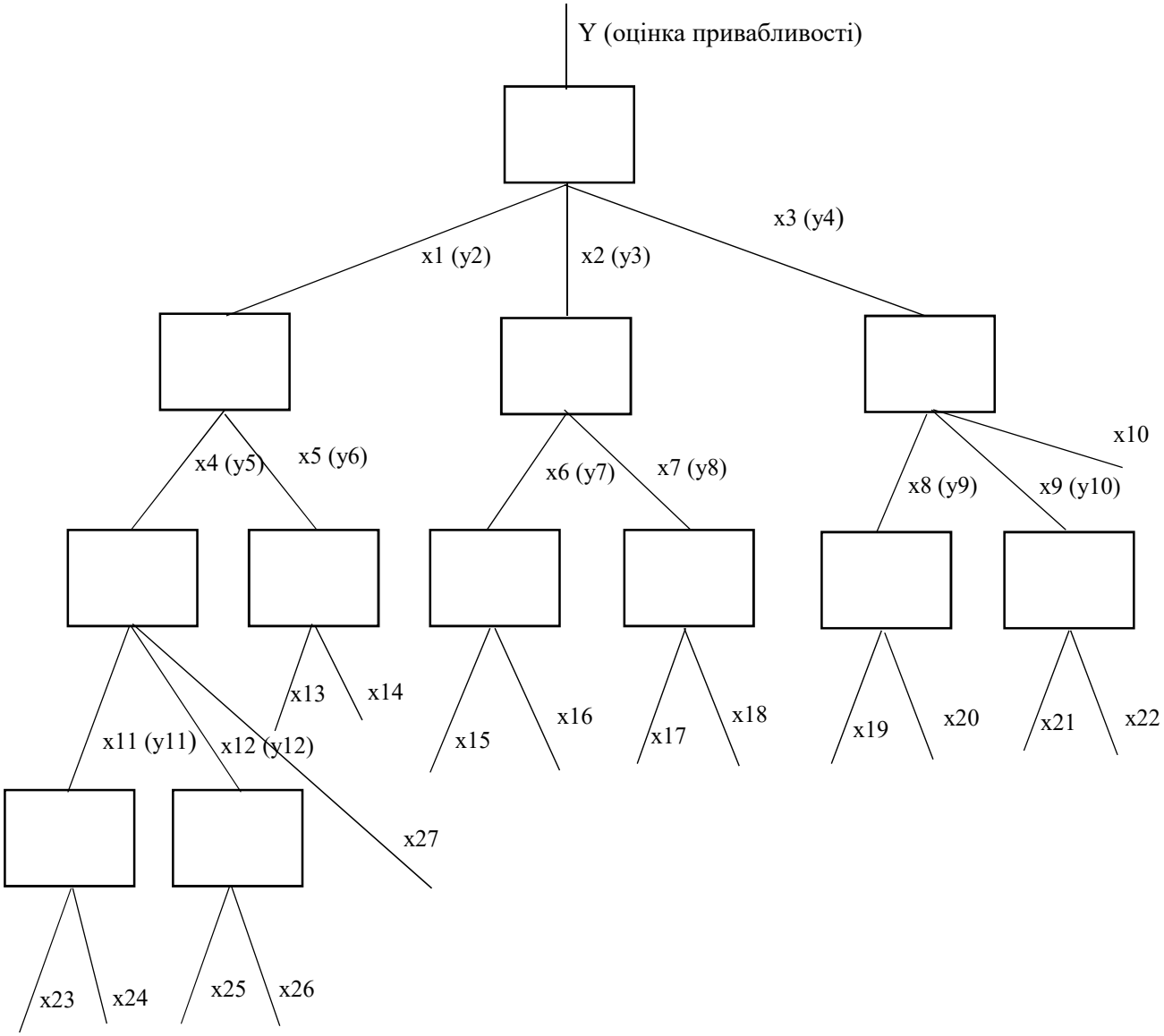


Рис. 6. Граф залежностей між блоками нечітких правил в інтелектуальній системі оцінки привабливості стартапів

Для побудови нечіткої ієрархічної системи оцінки привабливості необхідно визначити простір вхідних факторів $X = \{x_i\}$, $i = \overline{1,n}$ (табл. 1), та простір вихідних значень системи $Y = \{y_j\}$, $j = \overline{1,m}$ (табл. 2), що являють собою різні показники проекту. На основі критеріїв, що представлені в табл. 1, можна визначити вхідні змінні, що необхідні для нечіткого виведення в інтелектуальній системі.

Представлена система є ієрархічною (рис. 6), тобто всі вхідні змінні задаються користувачем до початку нечіткого виведення, а інші змінні обчислюються з попередньо визначених змінних нечітких блоків правил від яких вони залежать. Для здійснення нечіткого виведення користувачу необхідно задати наступні фактори: *x23* – стійкість проектних рішень (чи змінюються рішення під впливом зовнішніх факторів); *x24* – менеджмент проекту (якість керування проектом); *x25* – можливість виводу коштів у разі провалу проекту або за власним бажанням; *x26* – технічний ризик; *x13* – стартовий капітал (наявність власних коштів для започаткування справи); *x14* – подальше власне фінансування (чи є можливість у подальшому підтримувати стартап власними коштами); *x15* – досвід керівника за тематикою подібних проектів; *x16* – рівень освіти керівника; *x17* – готовність команди працювати повний робочий день (суміс-

ники або штатні працівники); *x18* – якість колективної праці (результативність роботи); *x19* – масштаб інновацій (рівень визнання інновацій); *x20* – тип інновацій (оригінальність проектних рішень); *x21* – пріоритет ринку (обмеження за типом споживачів продукції стартапу); *x22* – розмір ринку (обмеження за кількістю споживачів продукції стартапу); *x10* – ступінь розробки (на якій стадії знаходиться розробка даного проекту). Результуюча змінна *y* є остаточною оцінкою рейтингу стартапу на основі результатів обчислень підсистем нижчих рівнів та кінцевої обробки цих даних. Змінна *y* може приймати такі лінгвістичні значення: схвалити стартап, схвалити після доробки або відхилити.

Таблиця 1

Специфікація вхідних змінних інтелектуальної системи (фрагмент)

Змінна	**Найменування фактору**	**Значення**	**Зміст**
x1	Фінансування	Низьке (Н)	Недостатнє фінансування для даного проекту
		Середнє (С)	Фінансування проекту пов'язане з деякими труднощами
		Високе (В)	Фінансування повністю покриває вартість проекту
x2	Кадри	Низьке (Н)	Кадри є профнепридатними
		Середнє (С)	Потрібна додаткова професійна підготовка
		Високе (В)	Кадри повністю задовольняють вимогам проекту
x3	Проект	Низький (Н)	Проект приречений на провал, вимагає повного перегляду
		Середній (С)	Має слабкі сторони, вимагає деякої переробки
		Високий (В)	Проект може зайняти гідне місце на ринку
x4	Інвестиції	Низькі (Н)	Відсутні перспективи інвестування
		Середні (С)	Деякі інвестори готові почати співпрацювати
		Високі (В)	Присутні один або більше інвесторів, з перспективою подальшого капіталовкладення
…	…	…	…
x24	Менеджмент проекту	Низький (Н)	Слабке планування або відсутність планування і моніторингу
		Середній (С)	Планування і моніторинг з виконання завдань
		Високий (В)	Планування системи і моніторинг процесів проводяться своєчасно
x25	Можливість виводу коштів	Низька (Н)	Виведення коштів з проекту неможливий
		Середня (С)	Виведення коштів можливий тільки з певним відсотком втрат
		Висока (В)	Можливість повного виведення коштів
x26	Технічний ризик	Низький (Н)	Вимоги до проекту відсутні
		Середній (С)	Вимоги до проекту сформульовані нечітко
		Високий (В)	Вимоги до проекту чітко сформульовані

Таблиця 2

Специфікація результуючих та проміжних змінних інтелектуальної системи (фрагмент)

Змінна	**Найменування фактору**	**Значення**	**Зміст**
Y	Оцінка привабливості	Низька (Н)	Не варто займатися даним проектом
		Середня (С)	Проект має ризик провалу
		Висока (В)	Проектом варто займатися
y2	Фінансування	Низьке (Н)	Не варто займатися даним проектом
		Середне (С)	Проект має ризик провалу
		Високе (В)	Проектом варто займатися
…	…	…	…
y11	Ефективність інвестицій	Низька (Н)	Потенційний прибуток інвестора 10 % в рік
		Середня (С)	Потенційний прибуток інвестора 50 % в рік
		Висока (В)	Потенційний прибуток інвестора 100 % в рік
y12	Безпека	Низька (Н)	90 і більше відсотків, що інвестор втратить капітал
		Середня (С)	До 60 відсотків ймовірності збереження капіталу
		Висока (В)	Імовірність збереження капіталу понад 95 відсотків

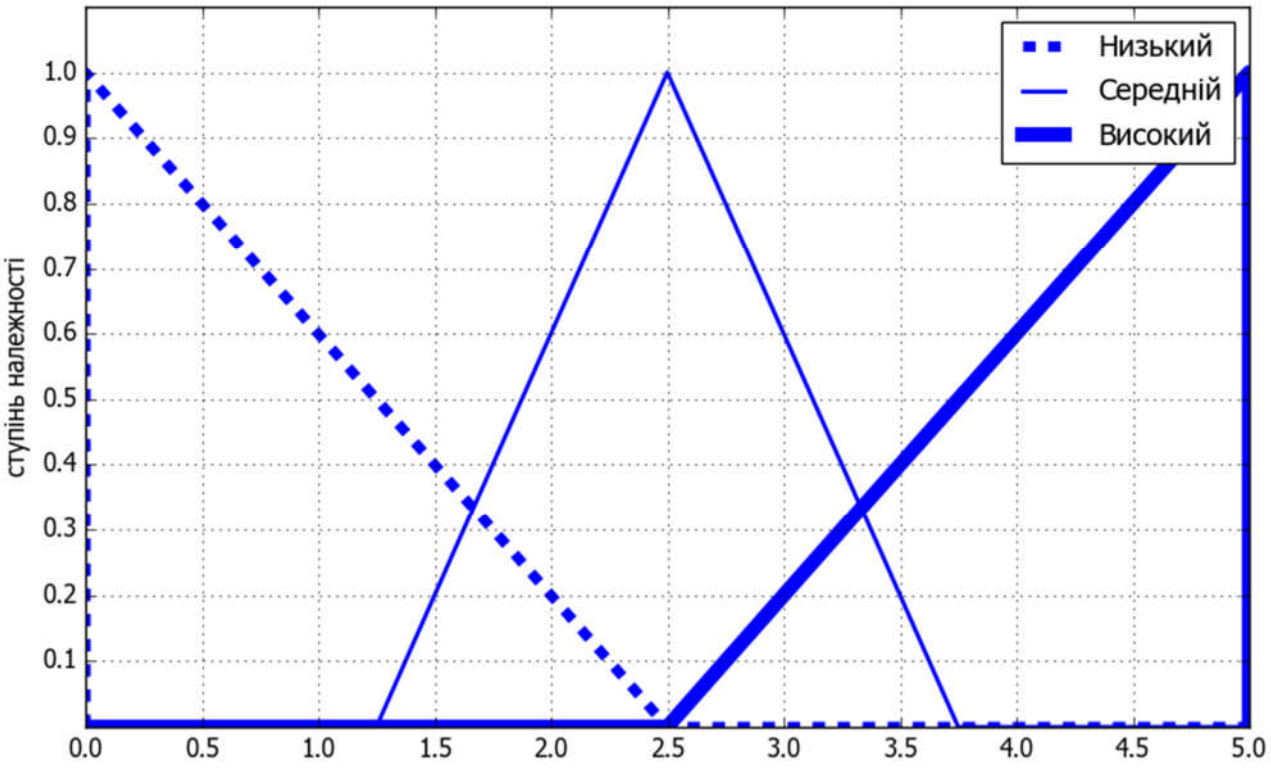


Рис. 7. Графічне представлення функцій належності

В табл. 3 представлено отримані рішення інтелектуальної системи для різних факторів у п'яти різних випадках. Очевидно, що проекти стартапів № 1–3 рекомендовано відправити на доробку, стартап № 5 визнаний привабливим для інвестування, а стартап № 4 вимагає повного перегляду та для інвестування не рекомендований.

Зазначимо, що така ієрархічна інтелектуальна система паралельно обчислює результуючі значення вершин графа залежностей (елементарних нечітких систем) в межах кожного ярусу. Завдяки внутрішньому паралелізму обробка всіх правил кожної вершини виконується також паралельно, для досягнення більш високих показників прискорення. Наприклад, вершина графу залежностей (блок правил), що розраховує остаточну оцінку привабливості стартапу, містить наступні правила:

if x1 is low and x2 is average and x3 is low then y1 is low
if x1 is low and x2 is average and x3 is average then y1 is average
……..
if x1 is average and x2 is high and x3 is high then y1 is high,

де *low, average, high* відповідають значенням *Низький, Середній, Високий* відповідно. Обробку кожного правила в межах всього блоку можна проводити незалежно одне від одного. Саме за рахунок цього кожний блок правил виконується паралельно, тобто обчислення кожного правила розпочинається одночасно окремою ниткою (thread) графічного процесору відеокарти (рис. 4).

Прискорення нечіткого логічного виведення для інтелектуальної системи оцінки привабливості стартапів на основі технологій MPI та CUDA у порівнянні з однопроцесорним варіантом такої системи, представлено на рис. 8. Зазначимо, що технологія CUDA дає прискорення приблизно в 100 разів порівняно з послідовним варіантом, а MPI дає прискорення, для цієї системи нечіткого виведення тільки в 3 рази. Для ієрархічної нечіткої системи з 12 блоками правил технологія CUDA дає прискорення більше, ніж в 30 разів, порівняно з технологією MPI на однакових вхідних даних. Це пов'язано, в першу чергу, з використанням внутрішнього паралелізму.

Таблиця 3

Результати роботи інтелектуальної системи за різних вхідних даних

Стартап №	Вхідні фактори стартапів для оцінки привабливості																Y	
	Порядкові номери вхідних змінних ієрархічної системи (**X**)																	
	23	***24***	***25***	***26***	***13***	***14***	***15***	***16***	***17***	***18***	***19***	***20***	***21***	***22***	***10***	***27***		
1	3	2	4	5	1	3	2	3	5	2	2	1	2	4	3	5	**3,0001**	**C**
2	3	4	3	1	4	2	2	3	2	3	2	5	4	2	3	3	**3,0007**	**C**
3	3	4	4	4	3	4	2	2	4	5	5	4	3	4	3	4	**3,0005**	**C**
4	1	2	2	1	3	3	3	3	2	1	4	3	1	1	1	2	**1,5986**	**H**
5	4	5	5	5	5	4	3	4	5	5	5	5	4	5	4	5	**4,7320**	**B**

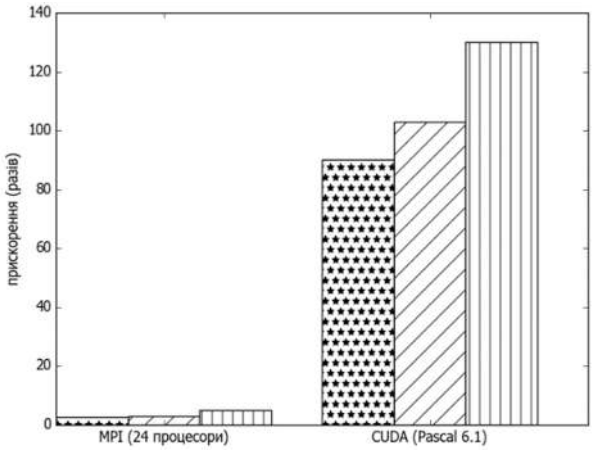


Рис. 8. Прискорення нечіткого виведення для MPI та CUDA

Висновки

В роботі розглянуто метод розпаралелювання систем нечіткого логічного виведення на основі багатопотокового програмування графічних прискорювачів Nvidia CUDA. Обґрунтовано переваги даного методу, встановлено оцінки прискорення систем нечіткого виведення, що побудовані за зазначеним методом, здійснено порівняльну характеристику з варіантами нечітких систем, що реалізовані на базі технології паралельного програмування MPI. В роботах [3,4] для побудови нечітких систем застосовувалася технологія паралельного програмування MPI, що має такі переваги, як відносна простота використання та висока переносимість коду. Дослідження та експерименти, проте, виявили ряд обмежень технології MPI при розпаралелюванні ієрархічних систем нечіткого логічного виведення. Зокрема, це необхідність налагоджувати механізми передачі даних між процесами, що ускладює код програми та уповільнює роботу програми за наявності великих обсягів даних, що передаються. Суттєвим є обмеженість процесорними ресурсами, які водночас є досить дорогими, оскільки для отримання кількох десятків паралельних процесів потрібно задіювати суперкомп'ютер або розподілену комп'ютерну мережу. Для подолання зазначених обмежень авторами розроблено та обгрунтовано метод нечіткого логічного виведення на основі багатопотокового програмування Nvidia CUDA. Застосування апаратної архітектури останнього покоління Pascal 6.1 надає у розпорядження досить велику кількість потоків для паралельної обробки нечітких правил, що повністю задовольняє потреби при нечіткому виведенні для систем дуже великої розмірності (кількість вхідних та вихідних змінних). Однією з головних переваг розробленого метода на основі графічних прискорювачів CUDA є можливість застосувати внутрішнє розпаралелювання за правилами для кожного блоку нечітких правил, що неможливо ефективно здійснити з використанням технології MPI через необхідність створення та координації великої кількості потоків. Це дозволяє отримати вагомі оцінки прискорення нечіткого логічного виведення у сотні разів (для окремих графів залежностей) у порівнянні з MPI.

Розроблено програмну систему оцінки привабливості стартапів на основі ієрархічних систем нечіткого логічного виведення Такагі – Сугено. На базі ярусно-паралельної форми обчислень та з використанням технології CUDA 8.0 в середовищі програмування Visual Studio 2012 побудовано програмну систему для оцінки привабливості стартапів.

1. Парасюк И.Н., Ершов С.В. Мультиагентные модели на основе нечеткой логики высшего типа для высокопроизводительной среды. *Проблеми програмування*. 2012. № 2-3. С. 260–269.
2. Штовба С.Д. Проектирование нечетких систем средствами MATLAB. М.: Горячая линия. Телеком. 2007. 288 с.
3. Torra V. A. review of the construction of hierarchical fuzzy systems. Int. J. Intell. Syst. 2002. N 17. P. 531–543.
4. Yager R.R. On a hierarchical structure for fuzzy modeling and control. IEEE Trans. Syst. Man Cybern. 1993. N 23. P. 1189–1197.
5. Yager R.R. On the construction of hierarchical fuzzy systems models. IEEE Trans. Syst. Man Cybern. 1998. N 28. P. 55–66.
6. Cordon O., Herrera F., Zwir I. Linguistic modeling by hierarchical systems of linguistic rules. IEEE Trans. Fuzzy Syst. 2002. N 10. P. 2–20.

7. Raju G.V.S., Zhou J., Kisner R.A. Hierarchical fuzzy control. Int. J. Control. 1991. N 54. P. 1201–1216.
8. Єршов С.В., Пономаренко Р.М. Паралельні моделі багаторівневих нечітких систем Такагі – Сугено. *Проблеми програмування.* 2016. № 1. С. 141–149.
9. Єршов С.В., Пономаренко Р.М. Ярусно-паралельна модель обчислень для логічного виведення у нечітких багаторівневих системах. Комп'ютерна математика. 2016. № 1. С. 28–36.
10. Воеводин В.В. Параллельные вычисления. СПб: БХВ-Петербург. 2002. 608 с.
11. http://www.nvidia.com.ua/object/cuda-parallel-computing-ru.html
12. Дорошенко А.Ю., Бекетов О.Г. Метод паралелізації циклів сіткових обчислювальних задач для графічних прискорювачів. *Проблеми програмування.* 2017. № 1. С. 59–66.
13. Киселева Е.М., Притоманова О.М., Журавель С.В. Оценка инвестиционной привлекательности стартапов на основе нейронечетких технологий. *Проблемы управления и информатики*. 2016. № 5. С. 123–143.

References

1. Parasyk I.N, Yershov S.V. (2012) Multilevel models base on fuzzy logic of the highest type for high-performance environment // Problems in programming. – N 2-3. P. 260–269.
2. Shtovba S.D. (2007) Fuzzy systems design by MATLAB mean // M.: Goryachaya liniya – Telecom – 288 p.
3. Torra V. (2002) A review of the construction of hierarchical fuzzy systems // Int. J. Intell. Syst. – Vol.17, N 5. – P. 531–543 .
4. Yager R.R. (1993) On a hierarchical structure for fuzzy modeling and control // IEEE Trans. Syst. Man Cybern. – Vol. 23, N 4. – P. 1189–1197.
5. Yager R.R. (1998) On the construction of hierarchical fuzzy systems models // IEEE Trans. Syst. Man Cybern. – Vol. 28, N 1. – P. 55–66.
6. Cordon O., Herrera F., Zwir I. (2002) Linguistic modeling by hierarchical systems of linguistic rules // IEEE Trans. Fuzzy Syst. – N 10. – P. 2–20.
7. Raju G.V.S., Zhou J., Kisner R.A. (1991) Hierarchical fuzzy control // Int. J. Control. – Vol. 54, N 5. – P. 1201–1216.
8. Yershov S.V., Ponomarenko R.M. (2016) Parallel models for Takagi-Sugeno's fuzzy multilevel systems // Problems in programming. – N 1. – P. 141–149.
9. Yershov S.V., Ponomarenko R.M. (2016) Tier parallel computing model for logical inference on fuzzy multilevel systems // Mathematic for computers. – N 1. – P. 28–36.
10. Voevodin V.V. (2002) Parallel inference // SPb: BHV-Peterburg. – 608 p.
11. Nvidia CUDA. – http://www.nvidia.com.ua/object/cuda-parallel-computing-ru.html.
12. Doroshenko A.U., Beketov O.G. (2017) The cycle paralellization method of the mesh tasks for graphic accelerator // Problems in programming. – N 1. – P. 59–66.
13. Kiseleva E.M., Prytomanova O.M., Zhuravel S.V. (2016) Evaluation of the start-ups investment attractiveness base on the neuro-fuzzy technologies // Problems in management and informatics. – N 5. – P. 123–143.

Одержано 04.10.2017

Про авторів:

Єршов Сергій Володимирович,
доктор фізико-математичних наук,
завідувач відділу інтелектуальних програмних систем Інституту кібернетики імені В.М. Глушкова НАН України.
Кількість наукових публікацій в українських виданнях – 67.
Кількість наукових публікацій в зарубіжних виданнях – 9.
http://orcid.org/0000-0002-9895-777X

Пономаренко Роман Миколайович,
аспірант Інституту кібернетики імені В.М. Глушкова НАН України,
Кількість наукових публікацій в українських виданнях – 3.
http://orcid.org/0000-0001-9681-2297

Місце роботи авторів:

Інститут кібернетики
імені В.М. Глушкова НАН України,
проспект Академіка Глушкова, 40,
Київ, 03187, Україна.
Тел.: (044) 526 41 78.
E-mail: sershv@ukr.net.

Тел.: (044) 526 64 22.
E-mail: ponomarenko_roman@ukr.net

УДК 004.94; 004.4; 004.62

Е.В. Жаріков

КЕРУВАННЯ РЕСУРСАМИ ХМАРНИХ ЦЕНТРІВ ОБРОБКИ ДАНИХ НА ОСНОВІ ЕВРИСТИЧНОГО ПОШУКУ

Проаналізовано особливості хмарного центру обробки даних (ЦОД) з точки зору керування ресурсами. Для вирішення задачі керування ресурсами хмарного центру обробки даних запропоновано і досліджено двостадійний метод консолідації віртуальних машин на базі використання локального променевого пошуку. В статті проаналізовано роботу евристики першої та другої стадій запропонованого методу, розроблений алгоритм променевого пошуку для вирішення задачі керування ресурсами. Для аналізу роботи методу використані дані про надходження завдань в кластер Google. Запропонований метод дозволяє переключити в режим зниженого енергоспоживання в середньому 56 відсотків фізичних серверів, що потенційно визначені для переключення в режим сну за допомогою верхньої оцінки необхідної ємності ресурсів. Перерозподіл віртуальних машин виконується з урахуванням обмеження допустимої кількості міграцій на один фізичний сервер.

Ключові слова: віртуалізація, керування ресурсами, хмарні обчислення, евристичний пошук.

Вступ

Керування ресурсами хмарного ЦОД – важлива задача, вирішення якої в сучасних умовах забезпечує роботу таких сервісних моделей хмарних обчислень, як інфраструктура як сервіс (англ. IaaS), платформа як сервіс (англ. PaaS) та застосунки як сервіс (англ. SaaS). Для досягнення бажаних показників енергоефективності та продуктивності роботи IT інфраструктури в центрах обробки даних застосовуються технології віртуалізації апаратного забезпечення, програмних засобів, мереж, сховищ даних, робочих місць та інші.

Сервісна модель IaaS дозволяє більш ефективно використовувати апаратне забезпечення фізичного сервера (ФС) за рахунок віртуалізації його локальних ресурсів. ФС надає такі ресурси, як процесорний час, пам'ять, локальну підсистему зберігання даних та підсистему доступу до мережі. При цьому, клієнтові надається частина ресурсів ФС у вигляді віртуальної машини (ВМ) або контейнеру. Для реалізації сучасних інформаційних послуг клієнт розгортає одну або декілька ВМ необхідної конфігурації, яка визначена провайдером хмарних послуг. Кожній ВМ гіпервізор надає частку ресурсів ФС. З точки зору керування ресурсами хмарного ЦОД, ресурси всіх ФС об’єднуються в *пул* і надаються віртуальним машинам для використання. Таким чином, виникає комплекс задач, пов’язаних з керуванням ресурсами хмарного ЦОД. Один із варіантів перерозподілу ресурсів пулу між віртуальними машинами полягає в реалізації процесу *консолідації віртуальних машин* (англ. virtual machine consolidation). Консолідація віртуальних машин – це розміщення віртуальних машин на фізичних серверах на базі технологій віртуалізації з метою досягнення певних показників ефективності використання ресурсів ЦОД. Фактично, керуючі впливи виробляються для віртуальних машин, фізичних серверів, мережевих пристроїв, сховищ, застосувань та інших підсистем.

Для роботи кожного екземпляру застосування в хмарному ЦОД зазвичай створюється окрема ВМ. На відміну від *монолітних* застосувань, які потребують нарощування ресурсів *вертикально* (англ. scale-up) при зростанні навантаження, застосування для хмарних ЦОД (англ. cloud-native applications) потребують нарощування ресурсів горизонтально (англ. scale-out). Горизонтальне нарощування ресурсів полягає у створенні додаткових екземплярів застосування у вигляді ВМ для обслуговування зростаючого навантаження (запитів клієнтів). Останнім часом, для забезпечення роботи хмарних застосувань широко використовуються так звані *контейнери* (англ. containers). Контейнер – це середовище для виконання екземпляра застосування. Контейнери ізо-

льовані один від одного і виконуються в спільній віртуальній машині.

В статті розглядається робота застосувань на базі ВМ. Виходячи з певних бізнес-потреб клієнт може динамічно змінювати конфігурацію ВМ або налаштовує механізми балансування навантаження, відмовостійкості та резервного копіювання. В процесі роботи хмарного сервісу клієнти створюють множину ВМ. В результаті зміни навантаження, зміни кількості запитів клієнтів та кількості пакетних задач, кількість віртуальних машин змінюється. Відповідно, змінюється кількість ВМ, що виконуються на окремому ФС, тому деякі ФС виявляються незавантаженими до максимально можливого порогу і витрачають зайву енергію.

З метою більш ефективного використання ресурсів хмарного ЦОД засоби віртуалізації надають можливість "живої" міграції (англ. live migration) ВМ з одного ФС на інший. При цьому, вплив на роботу ВМ є мінімальним, і для клієнта міграція ВМ не впливає на виконання задач. Але навантаження на фізичні сервери, які обмінюються цією ВМ, зростає.

Таким чином, однією з головних задач при керуванні ресурсами хмарного ЦОД є розміщення і перерозміщення віртуальних машин таким чином, щоб задіяти меншу кількість фізичних серверів та зменшити кількість міграцій віртуальних машин. Процес перерозподілу віртуальних машин серед фізичних серверів, консолідацію ВМ, можливо виконувати як неперервно, так і дискретно. Вирішення проблеми консолідації віртуальних машин представлено в багатьох публікаціях [1]. Але дослідження виконувалися для наявних на той час технологій віртуалізації та хмарних технологій. В сучасних ЦОД впроваджуються все нові і нові апаратні та системні засоби, які співіснують з технологіями попередніх поколінь. Таким чином, актуальною є розробка нових алгоритмів і методів для керування обчислювальними ресурсами ЦОД в цілому, та розміщенням віртуальних машин, зокрема.

Для вирішення задачі керування ресурсами хмарного ЦОД пропонується двостадійний метод на базі використання алгоритму променевого пошуку. Запропонований метод призначений для вирішення однієї з підзадач керування ресурсами шляхом консолідації віртуальних машин. В статті проаналізовано роботу евристики першої та другої стадій запропонованого методу, розроблений алгоритм променевого пошуку, уточнені функція оцінювання та умови закінчення роботи алгоритму. Для аналізу роботи методу використані дані про надходження завдань у кластер Google.

1. Аналіз публікацій

Останнім часом запропоновано багато методів та алгоритмів для вирішення проблеми керування ресурсами ЦОД [1–3]. Зокрема, задача консолідації віртуальних машин розглядається як оптимізаційна задача з різними цільовими функціями. Також, задача консолідації віртуальних машин розглядається як багатокритеріальна оптимізаційна задача [4, 5]. Складність цієї задачі полягає у наявності великої кількості станів середовища та обмежень. Крім того, складність виявляється при формулюванні цільової функції, до якої входять декілька показників, які треба оптимізувати. В результаті аналізу існуючих рішень з’ясувалося, що досягти ефективності деяких показників одночасно виявляється неможливим. Наприклад, неможливо досягнути одночасно високої швидкості розгортання ВМ та енергозбереження, або одночасно високої продуктивності та енергозбереження.

Для промислових ЦОД з хмарними інфраструктурами використовуються прості алгоритми керування, такі як first-fit, best-fit та їх модифікації (Eucalyptus [6], Microsoft [7], Google [8]). Це обумовлено вимогами робастності застосувань клієнтів та участю адміністраторів в автоматизованому процесі керування ресурсами ЦОД при постійному моніторингу якісних показників.

В дослідженнях використовуються такі цільові функції, як мінімізація споживання електроенергії, мінімізація порушень угод про якість сервісу (SLA), мінімізація мережевого трафіку,

максимізація продуктивності та використання ресурсів. Однією з основних умов для сучасних кластерів ФС є можливість роботи методів керування ресурсами в режимі онлайн [3]. Ці методи використовують потоки керування, що працюють *паралельно*. Разом з цим допускається застосування методів керування ресурсами, які спрацьовують при появі певних умов роботи кластеру, або через певний проміжок часу. Такі методи використовують потоки керування, що працюють *послідовно*. Для такого випадку нові завдання зазвичай розміщуються на ФС, які не задіяні в процесі консолідації ВМ.

Крім традиційних евристик та детермінованих алгоритмів при керуванні ресурсами ЦОД використовуються і алгоритми локального пошуку, такі як еволюційні алгоритми [9], алгоритми оптимізації мурашиних колоній [10], табу пошук [11], пошук з емуляцією відпалу [12]. Більш ефективним на нашу думку є використання локального пошуку на другій стадії оптимізації, після підготовки відповідного набору станів детермінованими алгоритмами з метою покращити рішення, знайдене на першій стадії.

Таким чином, в цій статті проаналізовано застосування алгоритму променевого пошуку в складі двостадійного методу для керування ресурсами хмарного ЦОД з метою зменшити кількість міграцій ВМ та збільшити кількість ФС, переключених у режим сну.

2. Модель системи

На теперішній час більшість послуг клієнти отримують на базі хмарних центрів обробки даних. Хмарні ЦОД – це складні системи, що складаються з серверних підсистем, підсистем зберігання даних, мережевих підсистем та підсистем інженерного забезпечення.

В статті розглянуто задачу керування ресурсами окремої підсистеми ЦОД у вигляді кластера з використанням віртуалізації. Для побудови кластеру в сучасних ЦОД використовують два підходи компоновки: з використанням гетерогенної або гомогенної конфігурації. Обидва підходи мають свої недоліки та переваги, однак уникнути розміщення гетерогенних конфігурацій в масштабі ЦОД в більшості випадків не вдається. Конфігурації кожного кластеру можуть відрізнятися з причин еволюції елементної бази серверів, сховищ та мережевих пристроїв, а також появи нових вимог користувачів до IT-інфраструктури. Однак має місце і найгірший випадок, коли конфігурації фізичних серверів у кластері відрізняються.

Дослідження роботи запропонованого двостадійного методу, що базується на алгоритмі променевого пошуку, виконані для кластеру, який складається з фізичних серверів різної конфігурації. Кожен ФС надає для локальних ВМ такі ресурси як: процесорний час (CPU), обсяг пам’яті (RAM), доступ до підсистеми зберігання даних (IOPS), доступ до мережевої підсистеми (NET) та інші. В залежності від наявної ємності ресурсів ФС, вимог до ресурсів з боку ВМ, часу виконання завдань всередині її та інтенсивності надходження завдань кількість ВМ, що виконуються на ФС, постійно змінюється. Зміна кількості локальних ВМ відбувається в таких станах: розгортання нової ВМ, завершення роботи ВМ, міграція ВМ.

В сучасних умовах, при використанні промислових гіпервізорів та швидкісних локальних мереж, середній час міграції віртуальної машини в середині ЦОД складає приблизно півхвилини, в залежності від обсягу пам’яті, що використовує ВМ.

Крім того, на процеси керування ресурсами також впливає час включення фізичного сервера, який складає 3–4 хвилини в залежності від конфігурації. Таким чином, краще перемикати фізичний сервер в режим сну. Переключення фізичного сервера з режиму сну в режим роботи відбувається значно швидше, чим холодний старт сервера.

Аналіз роботи запропонованого методу виконано з урахуванням двох ресурсів, що надаються для ВМ: CPU та RAM. Однак метод може бути доповнений іншими ресурсами, які потребує ВМ.

Вибір саме двох ресурсів обумовлено використанням вхідних даних з набору Google cluster-usage traces (GCT) [13]. Для аналізу роботи алгоритму променевого пошуку використано дві таблиці з набору GCT, а саме "Machine events" та "Task events table". Випадковим чином з першої таблиці обрано 6000 ФС, з другої таблиці обрано 70000 завдань. З таблиці "Machine events" для кожного ФС використано такі атрибути: machine ID, capacity: CPU, capacity: memory. З таблиці "Task events table" для кожного ФС використано такі атрибути: "task index within the job", "machine ID", "resource request for CPU cores", "resource request for RAM". Дані в таблицях нормовані відносно ФС з найбільшим значенням ємності відповідного ресурсу серед його кластеру.

Процеси циклу керування можуть повторюватись через певні проміжки часу при наявності двох вимог: міграції ВМ, визначені на попередньому кроці завершені та є передумови для переключення ФС в режим сну. Таким чином, запропонований двостадійний метод керування ресурсами використовує потоки керування, що працюють *послідовно.*

3. Постановка задачі

Кластер керування складається з множини P з M фізичних серверів та множини V з N віртуальних машин, $N, M \in \mathbb{N}$. В процесі розробки та аналізу роботи алгоритму променевого пошуку та в процесі циклу керування міграціями кількість ФС та ВМ не змінюється.

Для кожного завдання з таблиці "Task events table" розгортається окрема ВМ. В загальному випадку, між запусками алгоритму променевого пошуку кількість ФС та ВМ може змінюватись.

Задана ємність *j*-ї ВМ для ресурсу *k*, що позначена як $c_j^k \in (0,1]$, $k \in \{CPU, RAM\}$, визначається вимогами завдання і нормовано відносно ФС з найбільшою ємністю ресурсу k. Ємність фізичного сервера i для ресурсу k, що позначена $C_i^k \in (0,1]$, визначається типом ФС і нормовано відносно ФС з найбільшою ємністю ресурсу k.

Множина P складається з множини A фізичних серверів, які визначені для вимикання, та множини B фізичних серверів, які надають ресурси для ВМ, що будуть мігрувати з ФС, які належать до множини A, $A \cup B = P$, $A \cap B = \varnothing$.

Міграцію віртуальної машини j на фізичний сервер і позначимо як $U_{ij} \in \{0,1\}$. Міграція відбувається якщо $U_{ij} = 1$. Кожна ВМ з множини V має свій ID, який в процесі роботи алгоритму пов'язаний з номером j. Кожний ФС теж має свій ID, який в процесі роботи алгоритму пов'язаний з номером i.

Основною метою розробки і застосування двостадійного методу керування ресурсами хмарного ЦОД є зменшення кількості міграцій ВМ під час циклу керування та збільшення кількості ФС, що переключені в режим сну.

4. Двостадійний метод керування ресурсами на основі променевого пошуку

В основу метода покладено алгоритм променевого пошуку та евристики для оцінки стану кластера ЦОД. Робота методу складається з двох стадій. На першій стадії відбувається підготовка даних, на другій стадії визначається план міграцій ВМ.

Вхідними даними алгоритму променевого пошуку є: n – ширина променя, A – список ФС, які є претендентами для перемикання в режим сну, B – список ФС для обміну ВМ, в якому є вільні ресурси і є можливість розмістити додаткові завдання у вигляді ВМ.

Ідея алгоритму: перегляд i-го ФС зі списку A та пошук таких або такого ФС зі списку B, куди можливо мігрувати j-ту ВМ з i-го ФС. Якщо вдалося звільнити всі або частку ФС зі списку A, видаємо результат у вигляді матриці U_{ij}, яка є планом міграцій.

Опис роботи алгоритму променевого пошуку.

Перша стадія: підготовка вхідних даних для другої стадії. Формування списку A фізичних серверів, які треба звільни-

ти від ВМ, та списку B фізичних серверів для визначення плану міграцій.

Друга стадія виконується для кожного ФС із списку A.

1. На кожному кроці обираємо одну ВМ, назначену на *i*-й ФС і розглядаємо наступні варіанти обміну (міграцій):

a) мігрувати ВМ на інший ФС, в якого залишилось достатньо вільних ресурсів CPU та RAM;

b) перевірити можливість обміну з іншим ФС віртуальною машиною, яка вимагає менше ресурсів (так ми розглядаємо лише стани з кращою оцінкою та уникаємо можливих зациклювань) і, в результаті, ФС не буде перенавантаженим після обміну.

2. З усіх можливих обмінів обирається *n* обмінів з найвищою оцінкою.

3. Завершуємо пошук, якщо *i*-й ФС вдалося звільнити від віртуальних машин або, якщо не можна побудувати нові стани (тобто не залишилось жодного варіанта для реалізації допустимого обміну a) або b)).

Порівняння станів виконується за допомогою критерію:

$$J = \sum_{i=1}^{m} u_i^2 + \sum_{i=1}^{n} f_i^2 , \qquad (1)$$

де u_i – кількість використовуваних ресурсів на *i*-му ФС, f_i – кількість вільних ресурсів на *i*-му ФС, m – кількість ФС у списку B, n – кількість ФС у списку A.

Таким чином, алгоритм поступово зменшує використовувані ресурси на ФС, які належать до списку A, та завантажує ФС зі списку B.

Умови завершення алгоритму

Для завершення роботи алгоритму пропонуються умови.

1. Вичерпання списку A.

2. Неможливість мігрувати всі ВМ з певної визначеної кількості ФС Th_A поспіль.

Якщо другу умову не застосовувати, цикли пошуку виконуються занадто довго, порівняно з часом на створення нової ВМ та часом на міграцію для її з середніми вимогами до ресурсів пам'яті. Для визначення Th_A пропонується застосувати евристику, яка полягає у розгляді певного відсотку фізичних серверів із списку A, але не менше, ніж α фізичних серверів. У реалізації досліджуваного алгоритму прийнято $Th_A = 0.05$, $\alpha = 10$. З меншими значеннями α алгоритм пропускає значну кількість ФС, що могли бути переключені в режим сну, але в результаті завершення алгоритму були не звільнені від локальних ВМ. Даний критерій завершення вводиться для зменшення часу виконання алгоритму.

Опис першої стадії роботи методу.

Отримання списку фізичних серверів, з яких треба мігрувати всі ВМ з метою подальшого переключення ФС в режим сну пропонується здійснювати за допомогою двох методик: *нижньої границі* та *порогу вільних ресурсів*. Розглянемо кожну з методик більш детально.

Методика нижньої границі полягає у визначенні такої кількості фізичних серверів, які вимкнути в результаті міграції усіх ВМ, що на них працює, не уявляється можливим. Таке уявлення виникає з гіпотез, що використовуються в роботі алгоритму визначення нижньої границі.

Пошук нижньої границі виконується таким чином:

1) знаходимо середній обсяг наявних ресурсів за всіма ФС, на яких працюють ВМ,

$$T_{av}^k = \frac{1}{Q} \sum_{i=1}^{Q} C_i^k , \; 0 < Q \le M .$$

Значення для кожного ресурсу *k* розраховуються окремо;

2) по кожному ресурсу окремо рахуємо суму необхідних ресурсів для виконання всіх наявних ВМ,

$$D^k = \sum_{j=1}^{R} c_j^k , \; 0 < R \le N ;$$

3) окремо по кожному ресурсу рахуємо відношення необхідних ресурсів до середнього обсягу ресурсів та округляємо значення до більшого цілого, $E^k = \lceil D^k / T_{av}^k \rceil$;

4) нижньою границею буде найбільше число E^k з отриманих на кроці 3;

5) сортуємо фізичні сервери одним з наступних способів:

a) за обсягом ресурсів ФС, потім за відношенням використаних ресурсів до кількості працюючих ВМ;

b) за обсягом ресурсів ФС, потім за кількістю назначених завдань, потім відношенням використаних ресурсів до кількості локальних ВМ.

В результаті досліджень роботи алгоритму з'ясувалося, що варіант b) виявився значно ефективнішим. При його використанні на одному й тому самому наборі даних вдалося вимкнути в середньому 82 ФС, тоді як при використанні варіанту a) вдалося вимкнути в середньому лише 33 ФС.

В результаті, знаходимо різницю між кількістю ФС, що використовуються, та отриманою нижньою границею. Знайдене число – це кількість ФС списку A на вимкнення, які є першими у відсортованому списку. Решта ФС потрапляє у список B. Таким чином, методика нижньої границі дозволяє отримати оцінку наявної вільної ємності фізичних ресурсів кластеру.

Ідея методики порогу вільних ресурсів полягає у такому формуванні списку A, при якому до цього списку потрапляють ФС, загальна кількість невикористаних ресурсів яких перевищує обсяг ресурсів одного з ФС кластеру.

Позначимо поріг вільних ресурсів як β. Побудова списку A з використанням порогу вільних ресурсів виконується наступним чином:

1) встановлюємо значення β;

2) з множини P відбираємо ФС, у яких кожного вільного ресурсу більше за значення β;

3) по кожному ресурсу окремо рахуємо суму вільних ресурсів,

$$F^k = \sum_{i=1}^{Q} C_i^{k,free},\ 0 < Q \le M;$$

4) сортуємо обрані ФС аналогічно до варіанту b) з методики нижньої границі;

5) проходимо за відсортованим списком. Поки сума ресурсів більша за обсяг поточного ФС віднімаємо від суми цей обсяг, додаємо поточний ФС в список A та переходимо до наступного ФС, інакше завершуємо проходження списку. Таким чином, отримуємо список A з ФС, які треба переключити в режим сну, та список B з ФС для обміну віртуальними машинами.

5. Оцінка результатів моделювання

Дослідження роботи двостадійного методу виконане на фрагментах набору даних GCT [13]. Дані для тестування і дослідження методу обрані з GCT і являють собою частину за певний діапазон часу. Це необхідно, щоб врахувати всі дані за один і той самий період часу в журналах GCT. З набору випадковим чином відібрано 70000 завдань з певними вимогами до ресурсів k. Для кожного завдання буде розгорнута окрема ВМ в процесі моделювання. Обираємо 6000 фізичних серверів, з яких на 5832 серверах розміщено ВМ, та на 168 серверах завдань немає, але вони доступні для розміщення ВМ. Моделювання виконане на комп'ютері з процесором i5-3570 та обсягом системної пам'яті 4024 Мб.

Якісними показниками роботи алгоритму є кількість вимкнених ФС PM_{sleep} та кількість міграцій ВМ VM_{mig}, за допомогою яких ФС вивільнюються від ВМ. Крім того, враховано час роботи методу, t.

В таблиці 1 представлені результати роботи запропонованого методу з консолідації ВМ для різних значень β. В таблиці 2 представлені результати роботи методу при консолідації ВМ з використанням методики нижньої границі.

Таблиця 1

Результати моделювання консолідації ВМ з використанням методики порогу вільних ресурсів

β	B	A	PM_{sleep}	VM_{mig}	t, s
0.005	199	3	0	0	0.53
0.004	299	3	0	0	0.69
0.003	1098	9	0	0	4.05
0.002	1393	10	0	0	6.55
0.001	1868	28	13	219	38.08
0.0009	1943	31	15	263	42.29
0.0008	2002	39	22	349	67.93
0.0007	2081	43	24	399	81.02
0.0006	2204	48	28	432	94.14
0.0005	2302	55	32	504	105.48
0.0004	2407	63	36	554	141.43
0.0003	2565	73	43	683	143.16
0.0002	2764	82	48	782	166.54
0.0001	3706	95	59	992	238.06
0.00005	4323	106	68	1199	330.18
0.00001	5095	116	75	1336	463.36
0	5328	125	84	1459	518.34

Таблиця 2

Результати моделювання консолідації ВМ з використанням методики нижньої границі

B	A	PM_{sleep}	VM_{mig}	t, s
5707	125	82	1460	508.09

Вплив значення порогу β на якісні показники роботи алгоритму є суттєвим (рис. 1 та 2), залежність виявилася майже лінійною. Чим менший поріг β тим більше ФС потрапляють на вхід алгоритму.

При $\beta = 0$ деякі ФС (а саме ті, у яких хоча б один ресурс повністю зайнятий) не потрапляють на вхід алгоритму, тому що при перевірці наявності ресурсів використовується строга нерівність. При $\beta < 0.0002$ алгоритм з методикою порогу вільних ресурсів починає працювати більш ефективно і на граничних значеннях працює краще чим з методикою нижньої границі.

Після формування списку A ефективність пошуку кінцевого стану кластера можливо оцінити за допомогою відношення кількості ФС, що заплановані для переключення у режим сну, до кількості ФС, що фактично визначені для перемикання в режим сну після роботи другої стадії методу. Ефективність перемикання ФС в режим сну зростає з ростом значення порогу β і знаходиться в діапазоні від 45 % до 65 %.

Для перевірки роботи алгоритму з різними методиками та їх порівняння створено декілька наборів даних з однаковою кількістю ФС та завдань за інші періоди часу в наборі даних GCT. В результаті моделювання отримані показники ефективності, які розрізняються не більше ніж на 7 %.

Перевагою використання методики з порогом вільних ресурсів є можливість адаптуватися до стану кластера шляхом

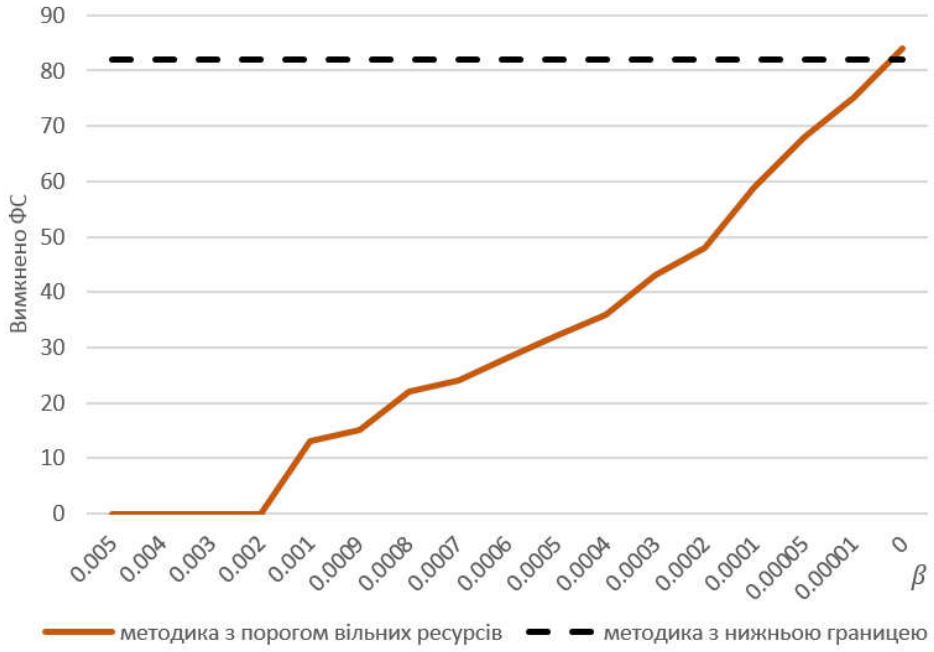


Рис. 1. Вплив порогу на кількість ФС, переключених в режим сну

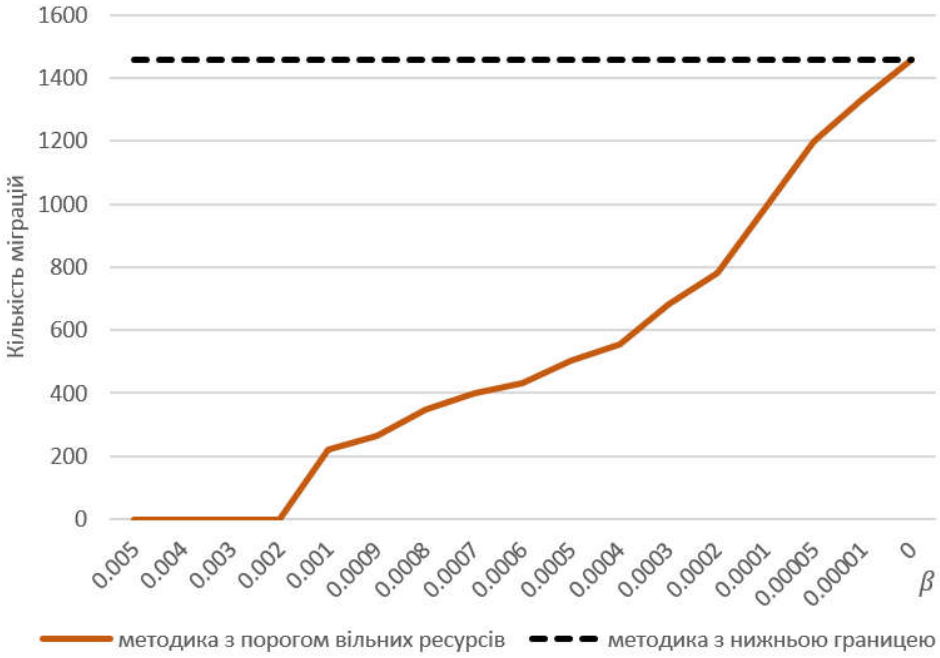


Рис. 2. Вплив порогу на кількість міграцій

зміни порогу перед циклом керування, враховуючи інші ресурси, час міграції ВМ та інтенсивність надходження заявок на створення нових ВМ.

В процесі дослідження роботи методу, з метою встановлення впливу ширини променя на якісні показники роботи алгоритму, ширина променю для алгоритму променевого пошуку змінювалась в діапазоні від 1 до 15 (табл. 3 та 4).

При використанні методики нижньої границі збільшення ширини променя алгоритму призводить до суттєвого збільшення часу пошуку рішення. Так, при ширині променя n=10 час виконання алгоритму значно зростає і стає неприпустимим для використання в кластері з високою динамікою процесів створення ВМ (табл. 3).

Таблиця 3

Результати моделювання консолідації ВМ з використанням методики нижньої границі

n	PM_{sleep}	VM_{mig}	t, с
1	23	423	66.72
2	23	435	101.33
3	23	433	137.75
4	23	450	173.57
5	82	1460	508.09
6	82	1457	812.05
7	83	1471	832.64
8	81	1418	894.11
9	83	1479	1169.78
10	80	1417	-
11	82	1428	-
12	82	1449	-
13	85	1520	-
14	83	1501	-
15	84	1487	-

Таблиця 4

Результати моделювання консолідації ВМ з використанням методики з порогом вільних ресурсів

n	PM_{sleep}	VM_{mig}	t, с
1	27	422	53.51
2	35	548	116.5
3	57	986	186.26
4	57	959	204.49
5	57	940	247.78
6	57	918	331.03
7	55	922	383.36
8	59	969	466.55
9	57	943	492.38
10	56	957	471.13
11	56	944	549.1
12	58	972	584.52
13	60	1063	718.3
14	58	1021	791.87
15	61	1042	844.12

Крім того, зміна ширини променя при використанні кожної методики може не призводити до суттєвого покращення якісних показників, при суттєвому збільшенні часу пошуку. Так, для методики з нижньою границею зміна ширини променя з 5 до 15 не призводить до покращення якісних показників, а для методики з порогом вільних ресурсів покращення якісних показників не відбувається при ширині променя від 3 до 15 (рис. 3 та 4).

Рис. 3. Вплив ширини променя на кількість ФС, переключених в режим сну

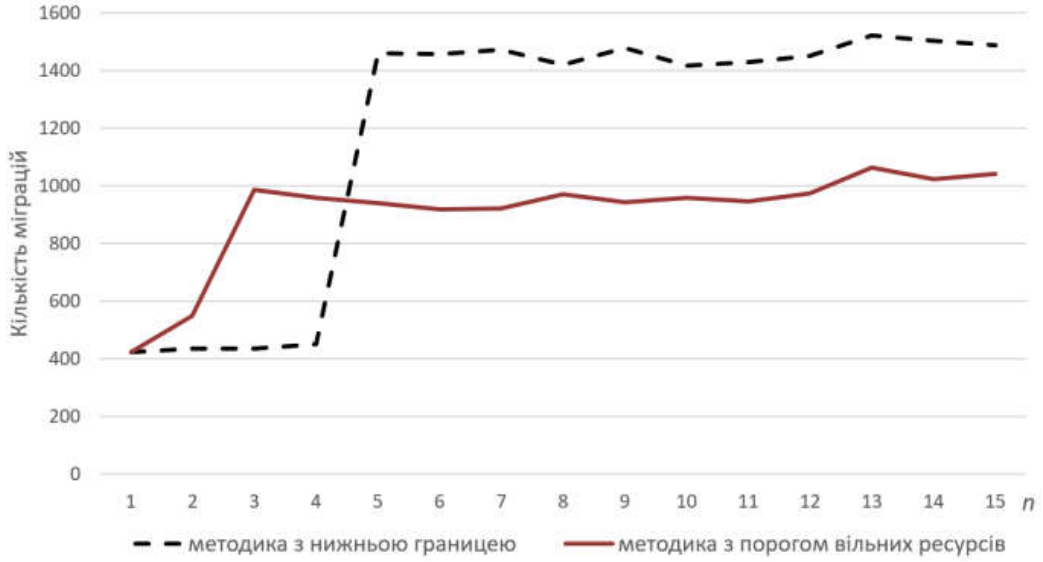


Рис. 4. Вплив ширини променя на кількість міграцій

Таким чином, треба звертати увагу на кількість міграцій, забезпечуючи їх допустиму кількість. З урахуванням висновків, рекомендується обирати ширину променя від 5 до 8, в залежності від умов роботи методу.

Висновки

Для керування ресурсами хмарного ЦОД на рівні кластера в статті запропоновано і досліджено двостадійний метод на базі алгоритму променевого пошуку. В статті проаналізовано роботу евристики першої та другої стадій запропонованого методу, розроблений алгоритм променевого пошуку для вирішення задачі керування ресурсами. Для аналізу роботи методу використані дані про надходження завдань в кластер Google. Запропонований метод дозволяє переключити в режим сну в середньому 56 відсотків фізичних серверів, що потенційно визначені для переключення в режим сну за допомогою верхньої оцінки необхідної ємності ресурсів.

Перерозподіл віртуальних машин виконується з урахуванням обмеження допустимої кількості міграцій. Завдяки урахуванню обмеження кількості міграцій на один фізичний сервер запропонований метод може бути застосований в реальних умовах ЦОД.

В результаті дослідження пропонується використовувати методику з порогом вільних ресурсів, що показала більш якісні

результати консолідації ВМ. Перевагою використання методики з порогом вільних ресурсів є можливість адаптуватися до стану кластера шляхом зміни порогу перед циклом керування, враховуючи інші ресурси, час міграції ВМ та інтенсивність надходження заявок на створення нових ВМ.

Також встановлено, що рекомендована ширина променя в алгоритмі променевого пошуку складає від 5 до 8, в залежності від умов роботи методу, обмежень на кількість міграцій ВМ та обмежень на час виконання алгоритму.

Для перевірки роботи алгоритму з різними методиками та їх порівняння створено декілька наборів даних з однаковою кількістю ФС та завдань за інші періоди часу в наборі даних GCT. Отримані показники ефективності розрізняються не більше ніж на 7 %.

1. Pires F.L., & Barán, B. (2015, May). A virtual machine placement taxonomy. In Proc. of the 15th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid). P. 159–168.
2. Ahmad R.W., Gani A., Hamid S.H.A., Shiraz M., Yousafzai A., & Xia F. A survey on virtual machine migration and server consolidation frameworks for cloud data centers. Journal of Network and Computer Applications. 2015. 52. P. 11–25.
3. Telenyk S., Zharikov E., & Rolik O. An approach to virtual machine placement in cloud data centers. In Radio Electronics & Info Communications (UkrMiCo), 2016 International Conference. IEEE. September, 2016. P. 1–6.
4. Pires F.L., & Barán B. Multi-objective virtual machine placement with service level agreement: A memetic algorithm approach. In Proceedings of the 2013 IEEE/ACM 6th International Conference on Utility and Cloud Computing. IEEE Computer Society. 2013, December. P. 203–210.
5. Saber T., Ventresque A., Brandic I., Thorburn J., & Murphy L. Towards a multi-objective vm reassignment for large decentralised data centres. 8th International Conference on Utility and Cloud Computing (UCC), 2015 IEEE/ACM. IEEE. 2015, December. P. 65–74.
6. Eucalyptus community [Online] – Available from: http://open.eucalyptus.com/
7. Lee S., Panigrahy R., Prabhakaran V., Ramasubramanian V., Talwar K., Uyeda L., & Wieder U. Validating heuristics for virtual machines consolidation. Microsoft Research, MSR-TR-2011-9. 2011. P. 1–14.
8. Sharma B., Chudnovsky V., Hellerstein J.L., Rifaat R., & Das C.R. Modeling and synthesizing task placement constraints in Google compute clusters. In Proceedings of the 2nd ACM Symposium on Cloud Computing. ACM. (2011, October). p. 3.
9. Mark, C. C. T., Niyato, D., & Chen-Khong, T. Evolutionary optimal virtual machine placement and demand forecaster for cloud computing. IEEE International Conference on Advanced Information Networking and Applications (AINA), IEEE. 2011, March. P. 348–355.
10. Gao Y., Guan H., Qi Z., Hou Y., & Liu L. A multi-objective ant colony system algorithm for virtual machine placement in cloud computing. Journal of Computer and System Sciences. 2013. 79(8). P. 1230–1242.
11. Ferreto, T., De Rose, C. A., & Heiss, H. U. Maximum migration time guarantees in dynamic server consolidation for virtualized data centers. In European Conference on Parallel Processing Springer, Berlin, Heidelberg. 2011, August. P. 443–454.
12. Wu Y., Tang M., & Fraser W. A simulated annealing algorithm for energy efficient virtual machine placement. In IEEE International Conference on Systems, Man, and Cybernetics (SMC), IEEE. 2012, October. P. 1245–1250.
13. Reiss C., Wilkes J., & Hellerstein J.L. Google cluster-usage traces: format+ schema. Google Inc., White Paper. 2011. P. 1–14.

References

1. Pires, F. L., & Barán, B. (2015, May). A virtual machine placement taxonomy. In Proc. of the 15th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid). pp. 159-168.
2. Ahmad, R. W., Gani, A., Hamid, S. H. A., Shiraz, M., Yousafzai, A., & Xia, F. (2015). A survey on virtual machine migration and

server consolidation frameworks for cloud data centers. Journal of Network and Computer Applications, 52, pp. 11-25.
3. Telenyk, S., Zharikov, E., & Rolik, O. (2016, September). An approach to virtual machine placement in cloud data centers. In Radio Electronics & Info Communications (UkrMiCo), 2016 International Conference (pp. 1-6). IEEE.
4. Pires, F. L., & Barán, B. (2013, December). Multi-objective virtual machine placement with service level agreement: A memetic algorithm approach. In Proceedings of the 2013 IEEE/ACM 6th International Conference on Utility and Cloud Computing (pp. 203-210). IEEE Computer Society.
5. Saber, T., Ventresque, A., Brandic, I., Thorburn, J., & Murphy, L. (2015, December). Towards a multi-objective vm reassignment for large decentralised data centres. 8th International Conference on Utility and Cloud Computing (UCC), 2015 IEEE/ACM (pp. 65-74). IEEE.
6. Eucalyptus community [Online] – Available from: http://open.eucalyptus.com/
7. Lee, S., Panigrahy, R., Prabhakaran, V., Ramasubramanian, V., Talwar, K., Uyeda, L., & Wieder, U. (2011). Validating heuristics for virtual machines consolidation. Microsoft Research, MSR-TR-2011-9, pp. 1-14.
8. Sharma, B., Chudnovsky, V., Hellerstein, J. L., Rifaat, R., & Das, C. R. (2011, October). Modeling and synthesizing task placement constraints in Google compute clusters. In Proceedings of the 2nd ACM Symposium on Cloud Computing (p. 3). ACM.
9. Mark, C. C. T., Niyato, D., & Chen-Khong, T. (2011, March). Evolutionary optimal virtual machine placement and demand forecaster for cloud computing. IEEE International Conference on Advanced Information Networking and Applications (AINA), (pp. 348-355). IEEE.
10. Gao, Y., Guan, H., Qi, Z., Hou, Y., & Liu, L. (2013). A multi-objective ant colony system algorithm for virtual machine placement in cloud computing. Journal of Computer and System Sciences, 79(8), pp. 1230-1242.
11. Ferreto, T., De Rose, C. A., & Heiss, H. U. (2011, August). Maximum migration time guarantees in dynamic server consolidation for virtualized data centers. In European Conference on Parallel Processing (pp. 443-454). Springer, Berlin, Heidelberg.
12. Wu, Y., Tang, M., & Fraser, W. (2012, October). A simulated annealing algorithm for energy efficient virtual machine placement. In IEEE International Conference on Systems, Man, and Cybernetics (SMC), 2012 (pp. 1245-1250). IEEE.
13. Reiss, C., Wilkes, J., & Hellerstein, J. L. (2011). Google cluster-usage traces: format+ schema. Google Inc., White Paper, 1-14.

Одержано 01.10.2017

Про авторів:

Жаріков Едуард В'ячеславович,
кандидат технічних наук,
доцент, докторант Національного технічного університету України "КПІ імені Ігоря Сікорського".
Кількість наукових публікацій в українських виданнях – 86.
Кількість наукових публікацій в зарубіжних виданнях – 17.
Індекс Хірша – 1.
http://orcid.org/0000-0003-1811-9336.

Місце роботи автора:

Національний технічний університет України "КПІ імені Ігоря Сікорського".
Тел.: 38044 204 86 10.
E-mail: zharikov.eduard@acts.kpi.ua

УДК 004.4'24

А.Ю. Дорошенко, О.А. Яценко, О.Г. Бекетов

АЛГОРИТМ АВТОМАТИЗОВАНОГО РОЗПАРАЛЕЛЮВАННЯ ЦИКЛІЧНИХ ОПЕРАТОРІВ ДЛЯ ГРАФІЧНИХ ПРИСКОРЮВАЧІВ

Розроблено програмний засіб для оптимізації обчислень, що дозволяє в напівавтоматичному режимі здійснити паралелізацію циклічних операторів програми для виконання обчислень на графічних прискорювачах. Здійснено буферизацію даних, синхронізовану із виконанням основного циклу, та побудований за допомогою системи переписувальних правил TermWare засіб інтегровано з інструментарієм проектування та синтезу програм (ІПС). Проведено випробування розробленої системи на гетерогенному мультиядерному кластері.

Ключові слова: методи паралелізації, оптимізація циклів, обчислення загального призначення на графічних процесорах, проектування та синтез програм.

Вступ

Розпаралелювання циклічних операторів є давно відомою проблемою паралельного програмування. З широким використанням графічних прискорювачів для обчислювальних задач виникла нова постановка цієї проблеми для цього класу мультиядерних систем. Мета даної статті – вдосконалення механізму перетворень операторів циклу для його паралелізації для виконання на графічному прискорювачі, наведеного в попередній статті [1]. Недоліком попередньої схеми перетворень було використання попередньої лінеаризації даних перед їх буферизацією, що суттєво ускладнювало подальшу процедуру передачі даних, оскільки потребувало окремого механізму контролю синхронізації. В даній роботі натомість реалізовано буферизацію даних, синхронізовану із виконанням основного циклу. Крім того, з метою підтвердження концепції було створено автоматизований інструментарій перетворення циклів за допомогою системи переписувальних правил TermWare і інтегрування його з інструментарієм проектування та синтезу програм ІПС. Наведено порівняння властивостей програм, отриманих при застосуванні наведеного перетворення та за допомогою системи автоматизованої паралелізації Par4All [2].

1. Механізм перетворення циклу

Нехай задане гніздо циклу, складене з циклів із лічильником, таке, що його ітерації незалежні між собою, а області значень ітераторів задані статично.

Графічний прискорювач є багатопотоковим обчислювальним пристроєм з SIMD архітектурою, тобто окремі паралельні потоки виконують один спільний набір команд, але над різними даними. Оскільки ітерації циклу незалежні, можливе їх виконання паралельними потоками. Отже, слід розбити цикл на окремі ітерації так, щоб кожна ітерація виконувалась окремим потоком, і кожному потоку надати набір даних, що відповідає виконуваній ітерації.

Наявність власної оперативної пам’яті призводить до необхідності обміну даними з центральним пристроєм. Обсяг даних, що обробляються, може перевищувати обсяг пам’яті графічного прискорювача, тому дані слід передавати порціями. У тому випадку, коли дані, що обробляються, не вміщуються у пам’ять графічного прискорювача, або у випадку обробки неперервного потоку даних, постає необхідність розбиття початкового циклу на окремі підцикли.

Розглянемо складений цикл, що має наступний вигляд:

$$\begin{array}{l} for \quad i_0 = 0...\#I_0 \\ \quad for \quad i_1 = 0...\#I_1 \\ \quad\quad \ldots \\ \quad\quad for \quad i_N = 0...\#I_N \\ \quad\quad\quad F(\bar{i}, \overline{p^{in}(\bar{i})}, \overline{p^{out}(\bar{i})}); \end{array} \qquad (1)$$

де $I_0, I_1, ..., I_N$ – множини значень індексів $i_0, i_1, ..., i_N$; $N+1$ – глибина вкладеності циклу, символом $\bar{i}$ позначено множину $\bar{i} = \{ i_j \mid j = 0, ..., N \}$;

$\overline{p^*(\bar{i})} = P^* = \{ p_j^*(\bar{i}) \mid j = 0 ... \# P^* \}, * \in \{in, out\}$ – множини значень параметрів вхідних та вихідних даних, F – відображення, що здійснює перетворення даних. Для спрощення припустимо, що множини вхідних та вихідних параметрів не перетинаються.

Позначимо T кількість потоків, що будуть задіяні при виконанні ядра *kernel*. Після перетворень цикл набуде наступного вигляду:

$$
\begin{aligned}
&for\ \ e = 0 : L \\
&\quad fill(inBuf); \\
&\quad push(inBuf); \\
&\quad kernel(e, inBuf, outBuf); \\
&\quad pull(outBuf); \\
&\quad unpack(outBuf);
\end{aligned}
\qquad (2)
$$

де L – кількість викликів ядра, що обирається таким чином, щоб $L \cdot T$ не перевищувало G – загальної кількості ітерацій початкового циклу; *fill* – функція заповнення буферу початкових даних; *push* – переміщення буферу початкових даних у пам'ять прискорювача; *kernel* – виклик ядра; *pull* – переміщення оброблених даних із пам'яті прискорювача до буферу оброблених даних; *unpack* – копіювання даних із буферу оброблених даних у відповідні змінні. Функція *fill* має наступний вигляд:

$$
\begin{aligned}
&for\ \ t = 0 : T \\
&\quad id = e \cdot T + t; \\
&\quad inbuf_t = p^{in}(g(id)).
\end{aligned}
$$

Таким чином, *id* визначає номер поточної ітерації. Тут g – функція, що співставляє номеру ітерації відповідний набір значень ітераторів циклу [1], $inbuf_t$ – буфер вхідних даних ітерації *t*. Функція *kernel* складається із виклику вмісту початкового циклу:

$$F(\bar{i}, \overline{inbuf_{id}(\bar{i})}, \overline{outbuf_{id}(\bar{i})}),$$

де *id* – номер потоку графічного прискорювача, що виконує ітерацію. Функція *unpack* аналогічна *fill*:

$$
\begin{aligned}
&for\ \ t = 0 : T \\
&\quad id = e \cdot T + t; \\
&\quad p^{out}(g(id)) = outbuf_t .
\end{aligned}
$$

Таким чином, при паралелізації окремого циклу структура основної програми залишається незмінною.

На рис. 1 наведено діаграму послідовності паралельної програми для одного прискорювача. Обчислення виконуються у три потоки, що одночасно виконують функції *fill*, *kernel* та *unpack*.

2. Реалізація методу

За допомогою системи символьних обчислень TermWare [3] було реалізовано експериментальний інструмент LoopRipper, що дозволяє, маючи списки вхідних та вихідних змінних, згенерувати функції *kernel, fill* та *unpack*, із яких паралельна програма компонується за допомогою інструментарію проектування та синтезу програм [4]. Генерація функцій здійснюється шляхом заміни параметрів вхідним та вихідним буферами даних та перерозрахунком ітераторів у заданому початковому циклі. Таким чином, від користувача вимагається надати цикл та списки параметрів. Подальша автоматизація процесу потребує аналізу дерева залежностей параметрів.

3. Інтеграція з системою ІПС

З метою автоматизації проектування та перетворення (паралелізації) програми у даній роботі спільно з системою TermWare [3] застосовується інструментарій проектування та синтезу програм (ІПС) [4].

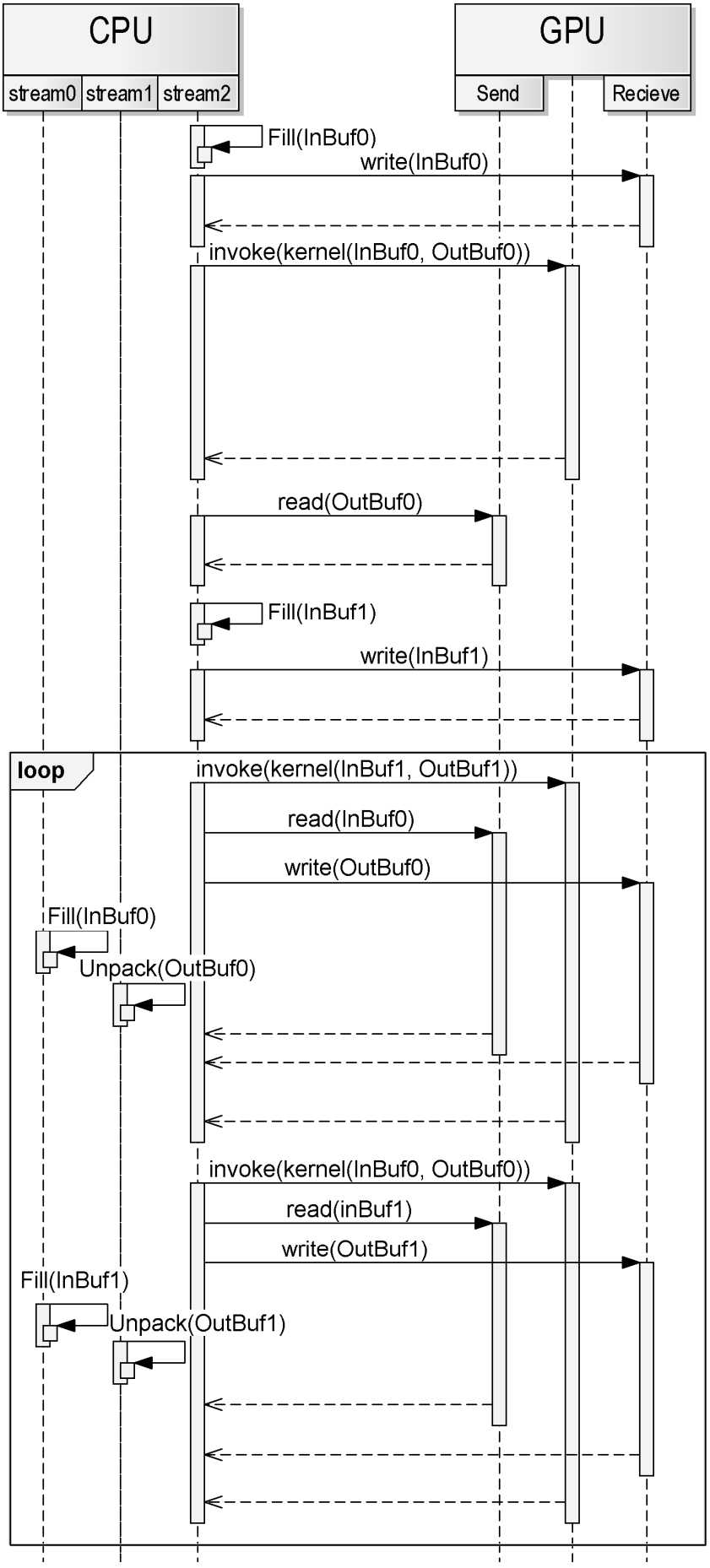


Рис. 1. Діаграма послідовності паралельної програми для одного прискорювача

Система ІПС призначена для конструювання послідовних і паралельних САА-схем алгоритмів, поданих в модифікованих системах алгоритмічних алгебр В.М. Глушкова (САА-М), а також генерації програм мовами C++, Java, C for CUDA та ін. Основним компонентом системи є діалоговий конструктор синтаксично правильних програм (ДСП-конструктор), за допомогою якого виконується порівневе проектування САА-схем зверху вниз шляхом деталізації мовних конструкцій САА-М. На кожному кроці проектування система надає користувачу на вибір лише ті конструкції, підстановка яких у текст схеми, що формується, не порушує її синтаксичну правильність. ДСП-конструктор використовує список конструкцій САА-М і дерево конструювання алгоритму. Специфікації конструкцій та їх відображення в цільові мови програмування зберігаються в базі даних інструментарію.

Основними етапами процесу розробки та трансформації програми (у рамках задачі паралелізації циклів, що розглядається у даній роботі) за допомогою ІПС та TermWare є такі:

1) конструювання САА-схеми послідовного алгоритму за допомогою ІПС та генерація відповідного коду цільовою мовою програмування;

2) трансформація коду послідовного циклу у функцію-ядро та генерація додаткових структур даних та макровизначень за допомогою TerwWare;

3) конструювання САА-схеми шаблону виклику ядра у системі ІПС та заміна послідовного циклу на виклик ядра;

4) вставка коду, згенерованого в TerwWare, у вихідну програму за допомогою системи ІПС;

5) генерація паралельної програми цільовою мовою програмування (C for CUDA) на основі отриманої САА-схеми в ІПС.

Приклад. Запропонований метод застосовано для розпаралелювання послідовного циклу з глибиною гнізда рівною чотирьом, що входить до складу програми чисельного прогнозування погоди, розглянутої в [5], а саме циклу інтерполяції початкових даних у вузлах локальної сітки. Далі наведено фрагмент послідовної САА-схеми, сконструйований в системі ІПС.

```
FOR (h FROM 0 TO Pk - 1)
LOOP
  FOR (k FROM 0 TO Lmz - 1)
  LOOP
    FOR (j FROM 0 TO Mmz - 1)
    LOOP
      FOR (i FROM 0 TO Nmz - 1)
      LOOP
        "(a) := ((WZZ + US[h][k][j][i] /
          0.321) * Rs * VS[h][k][j][i])";
        "(Tp) := (TS[h][k][j][i] *
          pow(1000.0 / HS[h][k][j][i],
           2.0/7.0))";
        "(Tv) := (Tp * (1.0 + 0.6078 *
          QS[h][k][j][i]))";
        "(Qc[h][k][j][i]) := (a - (0.5 * Tv +
          (1.0 - Zmz[k]) * g * F_X[j][i] /
          0.321))"
      END OF LOOP
    END OF LOOP
  END OF LOOP
END OF LOOP;
```

На основі наведеної САА-схеми в ІПС виконана генерація програмного коду мовою С. За допомогою TerwWare здійснене перетворення тіла послідовного циклу у функцію-ядро (Kernel) та генерація додаткових даних і макровизначень. В системі ІПС спроектовані САА-схеми шаблону виклику ядра (складеного оператора GPU) та функції-ядра (складеного оператора Kernel), що наведені далі.

```
"GPU" ====
= "Declare a variable (threadNum) of
    type (int) = (threadNum_def)";
  "Declare a variable (threadsPerBlock) of
    type (int) = (threadsPerBlock_def)";
```

```
"Declare a variable (gIterNum) of
 type (int) = (gIterNum_def)";
"Declare a variable (launchNum) of
 type (int)";
 (launchNum :=
  ceiling(gIterNum, threadNum));
"Declare a variable (blocksPerGrid) of
 type (int)";
(blocksPerGrid :=
 ceiling(threadNum, threadsPerBlock));
"Allocate CPU memory for input and output
 buffers (HI0, HI1, HO0, HO1)";
"Allocate GPU memory for input and
 output buffers (DI0, DI1, DO0, DO1)";
FOR (t FROM 0 TO threadNum - 1)
LOOP
  "Declare a variable (id) of type (int) =
   (0*threadNum + t)";
  "Copy parameters to input data buffer
   (HI0)"
END OF LOOP;
"Copy variable (HI0) from CPU memory to
 variable (DI0) of size (threadNum *
 sizeof(inDataChunk)) on GPU";
FOR (t FROM 0 TO threadNum - 1)
LOOP
  "Declare a variable (id) of type (int) =
   (1*threadNum + t)";
  "Copy parameters to input data buffer
   (HI1)"
END OF LOOP;
FOR (e FROM 0 TO launchNum - 1)
LOOP
  ("Copy data from (HI0, HI1) to (buf)
    according to (e)"

  PARALLEL

  (IF 'Even number (e)'
   THEN
     "Copy variable (HI1) from CPU
      memory to variable (DI1) of size
      (threadNum * sizeof(inDataChunk))
      on GPU in the stream (stream1)
      asynchronously";
     Call_GPU_kernel(blocksPerGrid,
                threadsPerBlock, 0, stream2)
     (
       "Kernel(DI0, DO0, threadNum)"
     );
     "Copy variable (DO1) from GPU
      memory to variable (HO1) of size
      (threadNum*sizeof(outDataChunk))
      on CPU asynchronously in the
      stream (stream3)"
   ELSE
     "Copy variable (HI0) from CPU
      memory to variable (DI0) of size
      (threadNum * sizeof(inDataChunk))
      on GPU in the stream (stream1)
      asynchronously";
     Call_GPU_kernel(blocksPerGrid,
                threadsPerBlock, 0, stream2)
     (
       "Kernel(DI1, DO1, threadNum)"
     );
     "Copy variable (DO0) from GPU
      memory to variable (HO0) of size
      (threadNum*sizeof(outDataChunk))
      on CPU asynchronously in the stream
      (stream3)"
   END IF;
   WAIT 'All threads completed work'

   PARALLEL

  "Copy data from (HO0, HO1) to (outbuf)
    according to (e) and unpack outbuf"
   ))
END OF LOOP;
"Copy data from (HO0, HO1) to (outbuf)
 according to (launchNum-1) and unpack
```

```
  outbuf";
 "Free the memory for variable (DI0) on
  GPU";
 "Free the memory for variable (DO0) on
  GPU";

"Kernel(inDataBuf, outDataBuf, threadNum)"
==== "Insert program code from the file
      (kernel.cu)";
```

У вищенаведеному складеному операторі GPU виклики функції-ядра виконуються асинхронно за допомогою операції

```
Call_GPU_kernel(blocksPerGrid,
        threadsPerBlock, 0, stream)
(
  "Kernel(inDataBuf, outDataBuf,
          threadNum)"
),
```

де blocksPerGrid – кількість блоків у сітці; threadsPerBlock – кількість потоків у кожному блоці; stream – черга виконання функцій-ядер; Kernel – функція-ядро; inDataBuf, outDataBuf – вхідний та вихідний буфери даних, відповідно; threadNum – кількість потоків. (Більш детально операції CAA-M, призначені для формалізації обчислень на GPU, розглядаються в роботі [4]).

У складеному операторі Kernel зазначено базисний оператор вставки програмного коду функції-ядра мовою C з файлу, згенерованого за допомогою TerwWare:

```
"Insert program code from the file
 (kernel.cu)"
```

Код згаданої функції-ядра має вигляд:

```
__global__ void Kernel(
    inDataChunk *inDataBuf,
    outDataChunk *outDataBuf,
      int threadNum)
{
  LocalDataInit;

  int myId = blockDim.x*blockIdx.x +
             threadIdx.x;  // cuda grid id

 if (myId < threadNum)
 {
   a = (WZZ + inDataBuf[myId].a0 / 0.321)*
       Rs * inDataBuf[myId].a1;
   Tp = inDataBuf[myId].a2 * pow(1000.0 /
        inDataBuf[myId].a3, 2.0/7.0);
   Tv = Tp * (1.0 + 0.6078 *
        inDataBuf[myId].a4);
   outDataBuf[myId].a0 = a - (0.5 * Tv +
        (1.0 - inDataBuf[myId].a5) * g *
        inDataBuf[myId].a6 / 0.321);
 }
}
```

Вставка згенерованих у TermWare додаткових структур даних та макровизначень у паралельну програму здійснюється також за допомогою базисного оператора вставки коду з файлу.

На основі побудованої паралельної CAA-схеми алгоритму в ІПС було виконано генерацію паралельної програми мовою C for CUDA, результати виконання якої на наведено у наступному розділі.

4. Експеримент

Для визначення ефективності перетвореної програми проведено експеримент, що складається із двох частин.

В першій частині здійснено порівняння результатів виконання програм, отриманих за допомогою описаної системи перетворень та системи автоматичної паралелізації Par4All, що використовує поліедральний метод паралелізації циклів і дозволяє генерувати програми для графічних процесорів на рівні коду CUDA в Linux системах. Система Par4All має суттєві обмеження. По-перше, ця система не дозволяє працювати із обсягом даних, що

перевищує обсяг пам'яті відеокарти. По-друге, як виявилося при застосуванні Par4All до експериментальної програми, система не спроможна виконати обробку вказівників, тому довелося відмовитись від їх використання. При проведенні першої частини експерименту використовувалися лише статичні масиви із залученням стеку великих обсягів (понад 2 Гб). Експеримент виконано на операційній системі Ubuntu 16.04 із використанням компілятора nvcc 8.0.61.

В другій частині порівнюються результати виконання перетвореної програми для однієї та для двох відеокарт. Par4All не передбачає можливості використання більш ніж одного прискорювача одночасно, тому в експерименті задіяна лише система LoopRipper.

Експеримент проводився з використанням апаратної системи, оснащеної процесором Intel Core i5-3570 (4 ядра, 3.8 ГГц), оперативною пам'яттю обсягом 8 Гб та відекартами NVIDIA Tesla M2050 (3 Гб оперативної пам'яті, ширина шини передачі даних 384 біт, 448 ядер CUDA, підключення через інтерфейс PCI Express x16) та NVIDIA GeForce GTX 650Ti (1 Гб оперативної пам'яті, ширина шини передачі даних 128 біт, 768 ядер CUDA, підключення через інтерфейс PCI Express x1). Попри більшу кількість ядер, друга відеокарта має повільнішу шину обміну даних, до того ж, підключення через повільний інтерфейс PCIex1 додатково обмежує швидкість передачі, тому розподіл навантаження між відеокартами встановлено на рівні 4:1. При такому розподілі час на виконання обчислень у обох відеокарт приблизно однаковий.

Масштабування дослідної задачі здійснювалось шляхом вибору розміру локальної сітки і варіювалось у межах доступної пам'яті центрального пристрою. Параметр *T* (кількість потоків відеокарти, задіяних при виклику ядра) підібрано таким чином, щоб мінімізувати час виконання програми. Після виконання результати послідовної та паралельної програм порівнювались між собою.

Як видно із графіка першого експерименту (рис. 2), експериментальна система випереджає Par4All, коефіцієнт відносного прискорення досягає 1.45. При виконанні експерименту з двома відеокартами (рис. 3) використовувалось динамічне

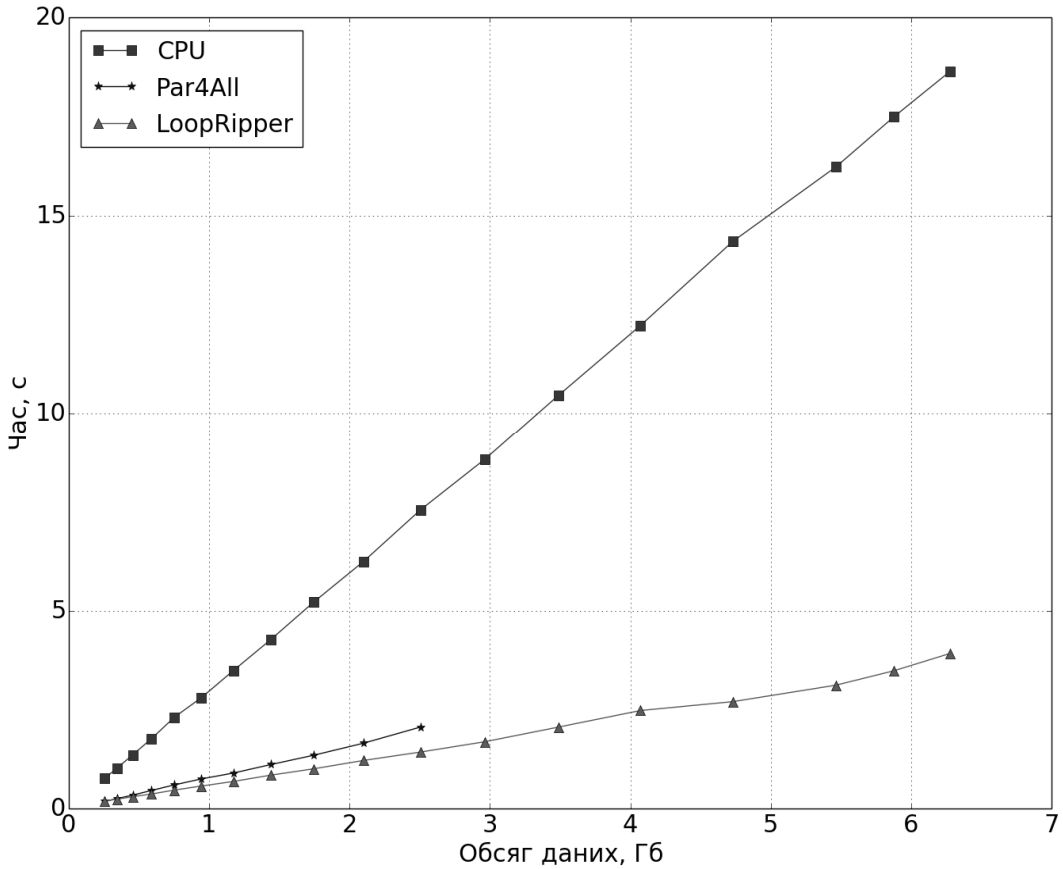


Рис. 2. Графік першого експерименту

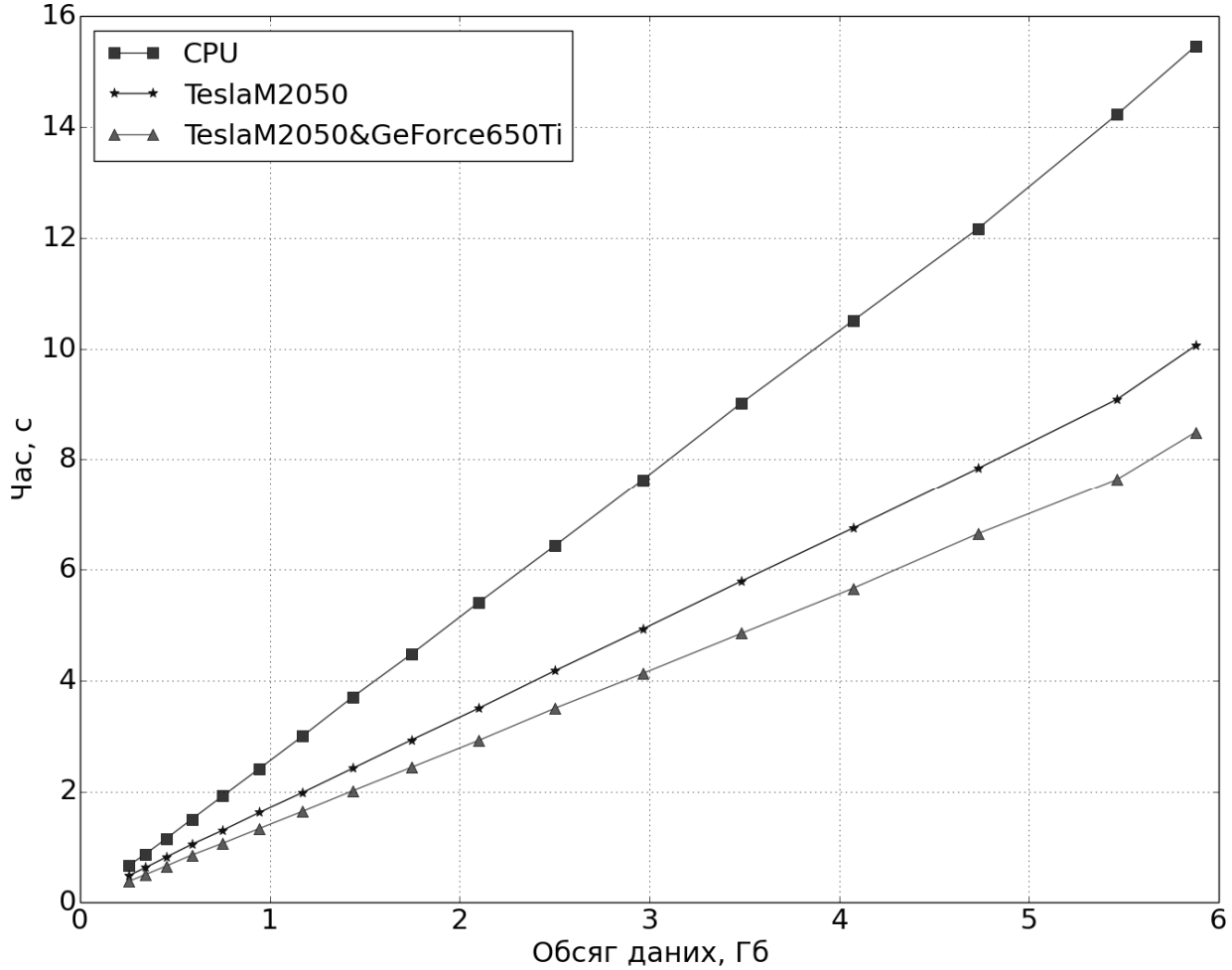


Рис. 3. Графік другого експерименту

керування пам'яттю, експеримент виконано на операційній системі Windows 7 із використанням компілятора nvcc 5.5.0. Оскільки повна паралелізація за схемою із копіюванням даних в окремих потоках потребує трьох потоків на відеокарту, при наявних чотирьох ядрах центрального процесора для уникнення конкуренції між потоками копіювання даних виконувалось послідовно із запуском ядра, тобто із залученням по одному потоку для кожної відеокарти. При залученні додаткової відеокарти зростання швидкості наближається до 1.2 рази, що пропорційно розподілу навантажень. Було виявлено відставання програм, виконуваних в системі Ubuntu, в порівнянні з системою Windows, причини якого наразі не з'ясовано.

Висновки

Реалізовано програмний засіб паралелізації циклів для оптимізації обчислень за допомогою графічних прискорювачів, що дозволяє в напіватоматичному режимі здійснити перехід від послідовної до паралельної програми. Проведено порівняння з відомою автоматизованою системою паралелізації Par4All, виявлено переваги розробленої системи у швидкодії, можливості обробки обсягів даних, що перевищують обсяг пам'яті графічного прискорювача та можливості залучення декількох прискорювачів одночасно.

1. Дорошенко А.Ю., Бекетов О.Г. Метод паралелізації циклів сіткових обчислювальних задач для графічних прискорювачів. *Проблеми програмування.* 2017. № 1. С. 59–66.
2. PIPS: Automatic Parallelizer and Code Transformation Framework [Електронний ресурс]. Режим доступу до ресурсу: http://pips4u.org/

3. Doroshenko A., Shevchenko R. A rewriting framework for rule-based programming dynamic applications. *Fundamenta Informaticae*. 2006. Vol. 72, N 1–3. P. 95–108.
4. Андон Ф.И., Дорошенко А.Е., Жереб К.А., Шевченко Р.С., Яценко Е.А. Методы алгебраического программирования: формальные методы разработки параллельных программ. Киев: Наукова думка, 2017. 440 с.
5. Дорошенко А.Ю., Бекетов О.Г., Прусов В.А., Тирчак Ю.М., Яценко О.А. Формалізоване проектування та генерація паралельної програми чисельного моделювання погоди. *Проблеми програмування*. 2014. № 2–3. С. 72–81.

References

1. Doroshenko A.Yu., Beketov O.G. (2017) Method of parallelization of loops for grid calculation problems on GPU accelerators. Problems in programming. (1). P. 59–66. (in Ukrainian).
2. PIPS: Automatic Parallelizer and Code Transformation Framework [Online]. Available from: http://pips4u.org/
3. Doroshenko A. & Shevchenko R. (2006) A rewriting framework for rule-based programming dynamic applications. Fundamenta Informaticae. 72 (1-3). P. 95–108.
4. Andon P.I. et al. (2017) Methods of algebraic programming: formal methods of parallel program development. Kyiv: Naukova dumka. (in Russian).
5. Doroshenko A.Yu., Beketov O.G., Prusov V.A., Tyrchak Yu.M. & Yatsenko O.A. (2014) Formalized designing and generation of parallel program for numerical weather forecasting task. Problems in programming. (2-3). P. 72–81. (in Ukrainian).

Одержано 10.10.2017

***Про авторів*:**

Дорошенко Анатолій Юхимович,
доктор фізико-математичних наук,
професор, завідувач відділу теорії
комп'ютерних обчислень
Інституту програмних систем
НАН України,
професор кафедри автоматики
і управління в технічних системах НТУУ
"КПІ імені Ігоря Сікорського".
Кількість наукових публікацій в
українських виданнях – понад 150.
Кількість наукових публікацій в
зарубіжних виданнях – понад 50.
Індекс Хірша – 5.
http://orcid.org/0000-0002-8435-1451,

Яценко Олена Анатоліївна,
кандидат фізико-математичних наук,
старший науковий співробітник
Інституту програмних систем
НАН України.
Кількість наукових публікацій в
українських виданнях – 36.
Кількість наукових публікацій в
зарубіжних виданнях – 12.
http://orcid.org/0000-0002-4700-6704,

Бекетов Олексій Геннадійович,
молодший науковий співробітник
Інституту програмних систем
НАН України.
Кількість наукових публікацій в
українських виданнях – 11.
Кількість наукових публікацій в
зарубіжних виданнях – 1.
http://orcid.org/0000-0003-4715-5053.

Місце роботи авторів:

Інститут програмних систем
НАН України,
03187, м. Київ-187,
проспект Академіка Глушкова, 40.
Тел.: (044) 526 3559.
E-mail: doroshenkoanatoliy2@gmail.com,
oayat@ukr.net,
beketov.oleksii@gmail.com

УДК 004.8

А.Ф. Кургаев, С.Н. Григорьев

ОПРЕДЕЛЕНИЕ ФОРМАЛЬНЫХ ЯЗЫКОВ В МЕТАЯЗЫКЕ НОРМАЛЬНЫХ ФОРМ ЗНАНИЙ

Исследованы выразительные возможности метаязыка нормальных форм знаний по отношению к формальным языкам разного уровня. Даны формальные описания множества лингвистических примеров. Разработано графическое метаописание интерпретатора универсальной машины Тьюринга, эквивалентное приведенному текстовому описанию. Представленное определение интерпретатора универсальной машины Тьюринга может быть применено для решения задачи моделирования поведения любой Т-машины (решающей задачу преобразования данных на ее ленте памяти). Дано формальное определение транслятора детерминированной машины Тьюринга. Обоснована и определена возможность описания в метаязыке нормальных форм знаний интерпретирующе-транслирующего и транслирующе-транслирующего процессов постановки и решения произвольной задачи, имеющей решение.

Ключевые слова: метаязык, регулярные языки, контекстно-свободные языки, контекстные языки, 0-языки, машины Тьюринга, текстовое и графическое описания языка.

Введение

Для определения языков используют формальные грамматики, которые задают четверкой

$$G = <\mathbf{V_T}, \mathbf{V_N}, S, \mathbf{P}>,$$

где $\mathbf{V_T}$ и $\mathbf{V_N}$ – непересекающиеся терминальный и нетерминальный алфавиты; S – аксиома, начальный символ; **P** – конечная система правил подстановки. Каждой грамматике G однозначно соответствует анализируемый (или порождаемый) язык $L(G)$.

В зависимости от выразительных возможностей различают четыре уровня грамматик (и языков) с номерами от 3 до 0 в порядке возрастания их общности: автоматные (или регулярные), контекстно-свободные (КС), далее – контекстные (или НС-грамматики) и на вершине иерархии – уровня 0 и соответствующие им машины Тьюринга.

Множество **G** грамматик $G \in \mathbf{G}$ с единообразным набором структур правил множества **P** оформляют в некоторый метаязык для определения синтаксиса объектных (описываемых) языков использованием некоторого количества правил. Среди известных метаязыков наиболее популярны BNF (Backus-Naur Form) и Extended (расширенный) BNF [1].

Однако, в качестве языка представления знаний метаязык EBNF (и все другие известные метаязыки) имеет определенные недостатки. В частности, созданный первично для узкоспециальных целей и такой, что хорошо их обеспечивает, он не является функционально полным и потому не пригоден для представления произвольных знаний.

В работе [2] предложен и в [3, 4] описан метаязык нормальных форм знаний (НФЗ), развивающий выразительные средства EBNF. Для достижения функциональной полноты в метаязык EBNF введены:

- отношение отрицания, применимое к любому понятию в описании (обозначается знаком ^, предшествующим имени отрицаемого понятия);
- три элементарных операции над информационными структурами: распознавание, распознавание со следом и порождение, обозначаются соответственно знаками ?, # и !, присоединяемыми к имени понятия, вводящему некоторую информационную структуру.

Покажем, что для формальных языков каждого уровня в метаязыке НФЗ есть адекватные выразительные средства.

1. База знаний в метаязыке НФЗ

Любое описание предметной области в метаязыке НФЗ составляется из двух частей. Первая часть – это информационная структура множества определений понятий (нетерминалов), связанных базовы-

ми отношениями метаязыка НФЗ. Вторая часть состоит из двух подмножеств терминалов – множества элементарных процедур и множества элементарных структур данных, которые реализуют на одном из языков программирования в форме единой библиотечной системы. Каждый из терминалов имеет собственное имя и в описании информационной структуры используется наравне с нетерминальными понятиями.

Определение 1. База знаний – это любое замкнутое на предметы предметной области ***D*** определение понятия в форме информационной структуры, элементами которой являются понятия, связанные между собой системой отношений метаязыка НФЗ, выполнимых на предметной области ***D***.

Определение 2. Данные – это предметные значения понятий.

2. Структура области данных

Общую предметную область ***D*** значений понятий представим двумя независимыми массивами – входным (INP) и выходным (OUT), каждый из них удовлетворяет аксиомам отношения предшествования [5]:

- никакой предмет не предшествует самому себе;
- если A предшествует B, а B предшествует C, то A предшествует C;
- если A и B – произвольные различные предметы, то либо A предшествует B, либо B предшествует A;
- в любом непустом классе предметов имеется первый предмет, то есть предмет, предшествующий всем остальным предметам класса.

С массивами INP и OUT свяжем переменные m и n, принимающие из множества натуральных чисел значения текущих координат соответствующих массивов, и две библиотеки пар процедур одноименных предикатов – библиотеку анализа над данными INP и библиотеку порождения над данными OUT. Переменные m и n будем считать независимыми, неявно заданными в квантификации всеобщности, аргументами (указателями на предметные значения) именованных понятий базы знаний.

Для каждой процедуры из библиотеки анализа есть одноименная процедура в библиотеке порождения и наоборот. Каждая процедура – это реализация операционального определения некоторого предиката в форме алгоритма обработки данных. Как правило, – это структурно простые действия типа сравнения с константой, записи, чтения, арифметические и другие вычисления.

Каждая процедура библиотеки анализа согласно своему содержанию:

- выполняет некоторое преобразование данных, взятых из массива INP, начиная с координаты, определяемой значением переменной m;
- формирует значение истинности, зависящее от успеха преобразования;
- при успехе (истинность – "истина") модифицирует значение переменной m, переводя указатель на начало следующего фрагмента массива INP.

Каждая процедура библиотеки порождения:

- порождает некоторый фрагмент данных в массив OUT, начиная с координаты, определяемой значением переменной n;
- формирует значение истинности "истина" и модифицирует значение переменной n, переводя указатель на начало следующего свободного фрагмента массива OUT.

Введем также набор системных процедур, управляющих этими элементами пространства данных, в том числе:

RB – всегда истинная процедура переключения библиотек. После ее выполнения вызов процедуры из библиотеки порождения заменяется вызовом процедуры из библиотеки анализа и наоборот;

RIO – всегда истинная процедура переключения массивов данных. После ее выполнения процедуры из библиотеки анализа обрабатывают данные из OUT, а процедуры из библиотеки порождения – данные из INP;

UIO – всегда истинная процедура объединения / разделения массивов данных. После однократного ее выполнения INP обрабатывается как два массива (INP и OUT) с всегда равными координатами анализа и порождения. Повторный ее вызов разделяет массивы.

Процедура RIO переключения массивов объединяет в единый процесс преобразование данных одного из массивов с преобразованием данных другого массива. То, что порождено в OUT, становится доступным для анализа и наоборот, то, что проанализировано из INP, может быть заменено порожденным.

3. Соотношения между языками для основных форм определения понятий в метаязыке НФЗ

Формальным языком, заданным на базовом множестве **U**, называют всякое подмножество свободной полугруппы **U***, т. е. множество цепочек из элементов **U** (подмножество множества **U***), вследствие чего на формальные языки распространяются теоретико-множественные операции объединения, умножения, дополнения, пересечения, возведения в степень [6–8]. Согласно теории понятий [9] любому понятию (наравне с содержанием, смыслом) присуще множество предметных значений, составляющее объем понятия. Основываясь на одинаковой (теоретико-множественной) природе объема понятия и формального языка в [10] обоснован ряд утверждений, принятых здесь в качестве исходных свойств метаязыка НФЗ.

Утверждение 1. Определение в форме альтернативы задает язык определяемого понятия как *объединение* языков понятий определяющей части.

Утверждение 2. Определение понятия в форме последовательности задает его язык как *произведение* языков понятий определяющей части.

Утверждение 3. Определение понятия в форме последовательности двух понятий, одно из которых взято с отрицанием, задает язык определяемого как *дополнение* языка отрицательного до языка положительного понятия.

Утверждение 4.

а) *отрицательное* определение любого понятия в форме *альтернативы отрицательных* понятий задает язык как *пересечение* языков понятий, входящих в альтернативу.

б) определение любого понятия последовательностью двух понятий, одно из которых взято *с отрицанием* и задано *альтернативой отрицательных* понятий, а другое – положительное, задает язык понятия как *пересечение* языков понятий, входящих в определяющую часть альтернативы.

Утверждение 5. Определение любого понятия в форме *итерации* другого понятия задает язык понятия как *итерацию* (операция Клини) языка понятия, входящего в определяющую часть.

Так, если понятие с именем A определено как итерация понятия с именем B:

$$A = (B);$$

то язык L_A определен итерацией языка L_B, т. е. произвольной *степенью* (в том числе *нулевой*) языка L_B.

Утверждение 6. Язык всякого понятия (с именем A), являющийся *ненулевой степенью* языка другого понятия (с именем B), задается рекурсивным определением формы:

$$A = B\,A/B;$$

4. Определение регулярных языков

Язык является регулярным, если он задан грамматикой, правила которой имеют форму:

$A \to x$ и/или $A \to y\,B$, допуская

$A \to x\,A$ ($x, y \in \mathbf{V_T}^+$; $A, B \in \mathbf{V_N}$).

Поскольку любая конечная последовательность слов выразима конечным произведением и/или конечной степенью языков, включающих лишь один символ

алфавита $\mathbf{V_T}$, то в силу утверждений 1, 5 и 6 справедливо

Утверждение 7. Любой регулярный язык может быть определен в метаязыке НФЗ.

Пример 1. Язык [7]:

$$L = \{a^n \mid n \geq 0\} \ ,$$

определяется итерацией

$$L = (\ 'a'\ ).$$

Пример 2. Язык [7]:

$$L = \{a^n \cup b^m \mid n, m \geq 1\},$$

определяется в форме

$$L = \ 'a'\ \ ('a'\ ) \ / \ 'b'\ ('b').$$

В силу замкнутости класса регулярных языков относительно умножения, объединения, итерации, дополнения, пересечения [6–8] (и, добавим, – ненулевой степени), справедливо

Утверждение 8. Язык любого понятия, определение которого дано с использованием лишь форм, указанных в формулировках утверждений 1–6, является регулярным.

5. Определение контекстно-свободных языков

Язык является контекстно-свободным, если задан КС-грамматикой, правила которой имеют форму:

$A \to \varphi$, где $A \in \mathbf{V_N}$, $\varphi \in (\mathbf{V_N} \cup \mathbf{V_T})^*$.

Класс КС-грамматик включает класс регулярных грамматик в качестве собственного подкласса и отличается от него допущением правил с самовставлением нетерминальных символов (теорема 11 [8]).

В самоопределении метаязыка НФЗ [4] нет ограничений на самовставление определяемого понятия, следовательно, справедливо

Утверждение 9. Любой КС-язык можно определить в метаязыке НФЗ.

Пример 3. Язык [6, 12]:

$$L = \{a^n b^n \mid n > 0\},$$

определяется предложением метаязыка НФЗ в форме альтернативы последовательностей:

$$L = \ 'a'\ \ L\ 'b'\ /\ \ 'a'\ \ 'b'\ .$$

Пример 4. Язык [6, 8]:

$$L = \{a^n b^n a^m \mid n, m > 0\}$$

определяется в форме:

1) $L = S_1\ S_2$;
2) $S_1 = \ 'a'\ \ S_1\ 'b'\ /\ 'a'\ 'b'$;
3) $S_2 = 'a'\ \ S_2\ /\ 'a'$.

Утверждение 10. Язык любого понятия, определение которого дано с использованием форм, указанных в утверждениях 1–6, и форм с самовставлением, является КС-языком.

6. Определение контекстных языков

Язык является контекстным (НС-языком), если задан НС-грамматикой, правила которой имеют форму:

$\varphi_1\ A\ \varphi_2 \to \varphi_1\ \omega\ \varphi_2$, где $A \in \mathbf{V_N}$; φ_1, φ_2, $\omega \in (\mathbf{V_N} \cup \mathbf{V_T})^*$ и ω не пусто.

КС-грамматики составляют подкласс НС-грамматик при ограничении правилами с пустым контекстом φ_1 и φ_2.

Вследствие незамкнутости КС-языков относительно операций дополнения и пересечения (теорема 21 [8]), в класс контекстных включаются языки, заданные с использованием этих операций над КС-языками. В силу утверждений 3 – 4 (формулировка которых не ограничивает сложность языков, над которыми определены эти операции) справедливо

Утверждение 11. Существуют НС-языки, для определения которых достаточно теоретико-множественных операций и форм с самовставлением.

Пример 5. Язык [6–8, 11–13]:

$$L = \{a^m b^m a^m \mid m \geq 1\}$$

формально определяется описанием метаязыка НФЗ, имеющим следующий смысл. Если первые две (из трех) подцепочки

равны по числу символов (контекстное условие), то первая подцепочка произвольна по длине, а вторая – равна третьей:

1) $L = \wedge S_1\ ('a')\ S_2$;

2) $\wedge S_1 = 'a'\ \wedge S_1\ 'b' / \ 'a'\ 'b'$;

3) $S_2 = 'b'\ S_2\ 'a' / \ 'b'\ 'a'$.

НС-грамматики занимают промежуточное положение между контекстно-свободными и самыми общими (0-грамматиками): в структуре их правил вместо одного нетерминала слева (как в КС-грамматиках), хотя и в ограниченном виде, но уже есть цепочка, поэтому они, не решая всех проблем формализации языков, приобретают проблемы реализации 0-грамматик.

Кроме теоретико-множественных операций, в теории формальных грамматик для определения языков используется дерево составляющих [6] (структура составляющих, С-маркер [12]) конкретного предложения. С-маркер строится конечным преобразователем (на его ленте памяти) в процессе вывода предложения по заданной грамматике языка. Известно [6, 8], что С-маркер можно построить для любого предложения любого КС-языка.

В описанной [3, 4] реализации метаязыка НФЗ операция построения С-маркера предложения языка некоторого понятия именуется символом "#" (в текстовом описании присоединяется к имени понятия) и выполняется процедурой интерпретации с записью показателей структуры в память следа. С-маркер используется процедурой интерпретации в режиме порождения – как повторение конечного процесса вывода предложения. Операция построения С-маркера является аналогом логического оператора *неопределенной* дескрипции ("эпсилон-оператора" [9, 14]), выделяющего конкретный предмет из объема понятия, к которому он применен.

Множество форм метаязыка НФЗ не включает *определенной* дескрипции, называемой также "йота-оператором" [9] (рассмотренного в [15] определения понятия в форме равенства предметных значений), однако она нужна для ассоциативного поиска, выражения контекстных условий и для других целей. Семантику идентичности значения некоторого понятия (с именем N) текущему значению из массива INP будем выражать (с использованием процедуры RB переключения библиотек) в следующей общей форме:

A = RB N RB! / ^RB.

Ее смысл состоит в следующем: при интерпретации понятия с именем A после выполнения первого терминала RB вызов любого терминала из одной библиотеки заменяется вызовом одноименного терминала из другой. Поэтому при порождении предложения (связанного с понятием N) выполняется анализ из INP в точности такого же предложения. Если этот анализ успешен, то выполнение второго терминала RB восстанавливает исходный порядок вызова терминалов из библиотек и интерпретация определения понятия A оканчивается со значением истинности "истина", иначе – "ложь".

Пример 6. Язык [13]:

$$L = \{a^n b^m a^n b^m c^3 \mid n, m \geq 1\},$$

на примере которого доказано существование НС-языков, не являющихся КС-языками, *формально определяется* описанием в метаязыке НФЗ:

1) $L = S_1\ S_2 \#\ S_3\ 'c'\ 'c'\ 'c'\ \wedge 'c'$;

2) $S_1 = 'a'\ S_1 / \ 'a'$;

3) $S_2 = 'b'\ S_2 / 'b'$;

4) S_3 = RB S_1 S_2 RB! / ^RB.

Пример 7. Язык [12]:

L = {$x\ x \mid x$ – не пустая цепочка из букв "a", "b"} *формально определяется* описанием в метаязыке НФЗ:

1) $L = S_1$RIO $S_2 \#$ RIO $S_1 !\ S_3\ \wedge S_1 / S_1\ L$;

2) $S_1 = 'a' / 'b'$;

3) $S_2 = (S_1)$;

4) S_3 = RB S_2 S_1 RB! / ^RB;

Исходная ситуация – в INP не пустая цепочка из букв "a", "b", OUT – пуст; **результат** – "истина", если в INP – xx, дополнительно – в OUT порождается x.

При интерпретации понятия L с понятием S_1 связывается значение первого символа цепочки из INP, с понятием S_2 – (возможно, пустая) цепочка символов из OUT, далее значение S_1 порождается в OUT, а интерпретацией S_3 проверяется совпадение конкатенации значений S_2 и S_1 с текущим значением цепочки из INP:

- если совпадает (значение S_3 – истина), то интерпретацией $\wedge S_1$ выполняется проверка окончания цепочки и, если цепочка исчерпана, то успешное завершение интерпретации понятия L, иначе выполняется вторая альтернатива определения L – пропускается первая буква текущего значения цепочки и рекурсивное повторение интерпретации понятия L;
- если не совпадает, то формируется значение S_3 – ложь, вследствие чего выполняется вторая альтернатива определения L с пропуском первой буквы текущего значения цепочки и рекурсивным повторением интерпретации понятия L. Если же выполнение второй альтернативы определения L не может быть продолжено из-за исчерпания цепочки букв, то формируется результирующее значение истинности ложь.

Рассмотренные общесистемные операционные средства, определения примеров известных НС-языков и отсутствие контрпримеров *обосновывают*

Утверждение 12. Всякий НС-язык можно определить в метаязыке НФЗ.

7. Определение 0-языков

Язык принадлежит уровню 0 (является 0-языком), если он задан 0-грамматикой с использованием правил "$\varphi \rightarrow \psi$", ограниченных лишь тем, что φ и ψ (цепочки из $\mathbf{V_N \cup V_T}$) не могут быть пустыми.

Общие системы подстановок универсальны: если существует формальная процедура задания языка, то его можно определить 0-грамматикой [6]. Множество правил 0-грамматики включает КС-грамматику, с каждой интерпретацией которой можно связать С-маркер (структуру составляющих [16] или, в другой терминологии, дерево составляющих [6]), множество трансформаций, каждая из которых *преобразует* набор (k, $k \geq 1$) С-маркеров в новый С-маркер, и множество правил, определяющих порядок применения трансформаций. Среди разновидностей трансформаций различают присоединение, перестановки, устранение (или добавление) элементов собственного анализа, любую из которых можно применять только к полностью проанализированной цепочке [16].

8. Универсальная машина Тьюринга

Универсальная машина Тьюринга – это U-машина с фиксированной структурой, имитирующая *поведение* любой машины Тьюринга (Т-машины).

Для Т-машины исходный результат состоит в том, что если язык L определен 0-грамматикой, то по ней можно построить Т-машину, которая *допускает* язык L, и наоборот [16, 17]. Тоесть утверждается однозначное соответствие между 0-языком и Т-машиной. Следовательно, для множества 0-языков *можно* построить множество Т-машин.

U-машина отличается тем, что на ней одной *можно* моделировать поведение всего множества Т-машин, каждая из которых *допускает* некоторый 0-язык, и тем самым *решить произвольную задачу* уровня 0, для которой есть решение.

В теории алгоритмов интерпретатор U-машины исследовался столь детально потому, что составляет отдельную задачу, сложность которой (по определению) не ниже уровня 0, так как в конечном итоге, именно интерпретатор решает произвольную задачу уровня 0 (интерпретируя модель любой Т-машины, решающей задачу уровня 0). В силу этого реализация в некотором языке интерпретатора U-машины гарантирует *достаточность* его выразительных возможностей для постановки и решения произвольной задачи (для которой есть решение) и, тем самым, *универсальность* машины, реализующей этот язык. А. Тьюринг предложил и показал способ построения

U-машины, М. Минский [18] разработал форму представления любой Т-машины на ленте U-машины и описал интерпретатор U-машины в форме диаграммы состояний.

Используя эти результаты, для формализации примем два *допущения* [19].

1. Пусть полубесконечная вправо лента U-машины отождествлена с совмещенными (после выполнения процедуры UIO) массивами INP и OUT пространства данных метаязыка НФЗ и разделена на зоны, расположенные слева-направо в следующем порядке:

- начальное состояние q_t некоторой произвольной Т-машины;
- первый, обозреваемый читающе-записывающей головкой Т-машины, символ a_t;
- конечное описание Т-машины (последовательность взаимосвязанных продукций), представленное пятерками q_i, a_i, q_j, a_j, d_i – текущее состояние Т-машины, считываемый символ, новое состояние Т-машины, записываемый символ и значение сдвига (0 – вправо, 1 – влево) читающе-записывающей головки Т-машины соответственно. Каждая из продукций имеет смысл выражения: "*Если* Т-машина находится в состоянии q_i и на ее ленте обозревается символ a_i, *то* заменить его на символ a_j и перейти в состояние q_j, в котором выполнить преобразование соседнего (слева или справа от a_i) символа ленты Т-машины". Каждое состояние q_i, q_j, q_t из множества Q пусть именуется одним символом латиницы или цифры. Входные символы a_i, a_j, a_t пусть принадлежат алфавиту Σ, составленному из букв латиницы и цифр, а также включают нулевой код;
- разделяющий выделенный символ Код0;
- полубесконечная вправо псевдолента Т-машины содержит конечную цепочку символов (за ней – нулевые коды), одно из ее знакомест занято маркером (тот же выделенный символ Код0), указывающим положение читающе-записывающей головки Т-машины.

2. Положение читающе-записывающей головки U-машины однозначно определяется значением переменной m (координаты INP).

В начальный момент: читающе-записывающая головка U-машины расположена в крайнем левом положении и обозревает символ, именующий начальное состояние q_t Т-машины; читающе-записывающая головка Т-машины установлена также в крайнем левом положении ее ленты (маркер Код0 записан в первой клетке псевдоленты Т-машины).

Семантику интерпретатора U-машины *формально определим* в метаязыке НФЗ:

1) интерпретация = состояние символ# (^Код0 ^правило ^преобразовать);

2) ^правило = та_пятерка? состояние символ сдвиг# / пятерка ^Код0 ^правило;

3) та_пятерка = RB состояние символ RB! / ^RB;

4) пятерка = состояние символ состояние символ сдвиг;

5) ^преобразовать = (^Код0 пятерка) Код0 (символ ^Код0) ^зап_сдвиг заменить ^чтение? Код0!;

6) ^зап_сдвиг = символ? сдвиг! / сдвиг!;

7) сдвиг = '0' / '1';

8) заменить = символ ^если_0? символ! /^если_0? символ! /^записать_символ;

9) ^если_0 = '0';

10) ^записать_символ = символ? символ! / символ!;

11) ^чтение = символ#;

База знаний состоит из 16 понятий, в том числе 11 нетерминалов определяют состояния U-машины (это число можно уменьшить до 10, заменив понятие "пятерка" его определяющей частью) и пяти терминалов: RB – переключения библиотек; "символ", "состояние" – значения из множеств Σ и Q соответственно; "сдвиг" – значения из двузначного множества {'0', '1'} и "Код0" – разделяющий символ или маркер. В ряду известных определений U-машины,

составленных в свое время Икено, Ватанабе и Минским, данное является наиболее *естественным* и *простейшим* по критерию К. Шеннона [20].

Графическое метаописание интерпретатора U-машины показано на рисунке. Очевидно, что оно эквивалентно приведенному текстовому описанию.

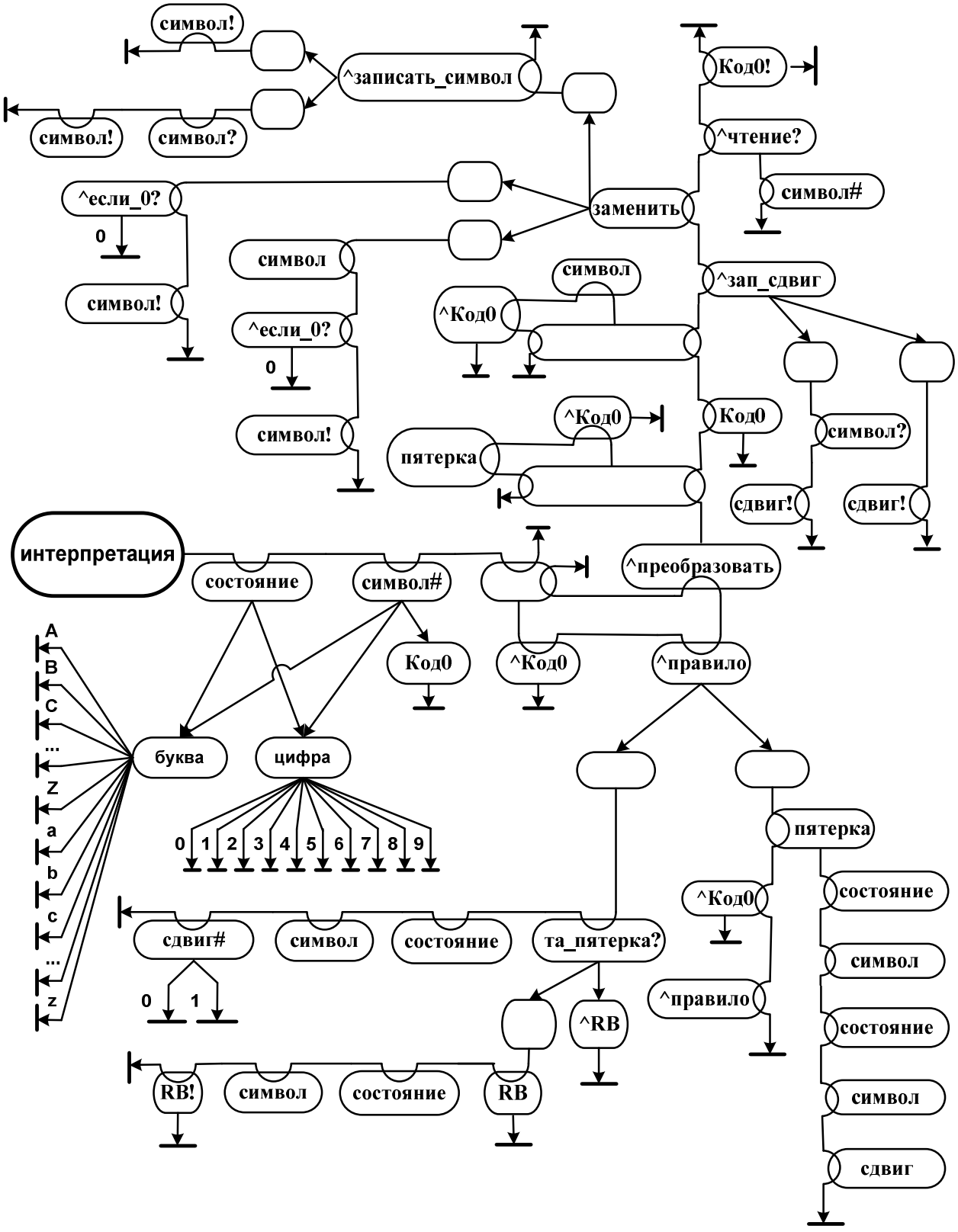


Рисунок. Граф метаописания интерпретатора U-машины в метаязыке НФЗ

Представленное определение интерпретатора U-машины может быть применено для решения задачи моделирования поведения любой Т-машины (решающей задачу преобразования данных на ее ленте памяти), которая формулируется обычным образом – как доказательство общего суждения: "Любое значение данных из INP (начало которых определено значением переменной *m*) суть интерпретация".Вывод этого суждения идет в следующей последовательности. При анализе со следом с понятиями "состояние" и "символ" связываются: начальное состояние Т-машины и первый символ на ее ленте. Затем в итерационном цикле интерпретируется последовательность понятий "^поиск_правила" и "^преобразовать", первое из них определяет ассоциативный поиск пятерки, описывающей очередную смену состояния моделируемой Т-машины, второе – замену на ленте Т-машины ранее взятого символа новым (из соответствующей пятерки), читает очередной символ (согласно указателю о сдвиге головки Т-машины) и записывает на его место маркер Т-машины. Поиск подходящего правила Т-машины выполняется просмотром пятерок описания Т-машины отождествлением текущего и связанного значений. Если проанализированные со следом состояние или символ не совпадают со значениями элементов очередной пятерки, то пятерка пропускается, иначе – анализируются со следом новое состояние, заменяющий символ и значение сдвига головки Т-машины. Если все описание Т-машины уже просмотрено безуспешно и на очереди разделяющий Код0, то формируется значение истинности "ложь", рекурсивный и затем итерационный цикл прерываются и интерпретатор заканчивает свою работу с заключительным значением истинности "истина". Иначе поиск пятерки рекурсивно повторяется до нужной пятерки или пока не выявится, что запомненное состояние – заключительное. Если пятерка найдена, – после последовательной интерпретации четного числа отрицаний головка U-машины возвращается к началу описания Т-машины. Интерпретация понятия "^преобразовать" начинается с пропуска описания Т-машины, затем пропуск разделяющего символа и далее поиск маркера. После того, как положение головки Т-машины найдено, вместо маркера записывается значение сдвига, затем анализ значения сдвига, запись на его место заменяющего символа, чтение очередного символа (слева или справа от маркера) и запись вместо него маркера.

Рассмотренная интерпретация представленного в метаязыке НФЗ определения U-машины подтверждает его семантическую эквивалентность диаграмме состояний, составленной М. Минским [18].

Эта задача – **пример** задачи *поведения* с фиксированной *стратегией*. Ее смысл состоит в преобразовании *данных* (представленных на ленте *любой* Т-машины) согласно *знаниям* (системе продукций в форме диаграммы состояний *конкретной* Т-машины) о преобразовании. Результат (в форме данных на ленте Т-машины) можно использовать как конечный результат или как исходные данные для следующей задачи. Работу U-машины можно возобновить после установки головок в начальное положение и смены данных и/или программы Т-машины.

Утверждение 13. Метаязык НФЗ включает семантические возможности U-машины, следовательно, любой 0-язык в форме интерпретатора может быть определен в метаязыке НФЗ.

Используя этот результат, *интерпретирующе-транслирующий процесс* постановки и решения произвольной задачи представляется следующей последовательностью преобразований:

- на метаязыке описать интерпретатор объектного языка и ввести в машину в качестве данных – разместить в массиве INP;
- выполнить ввод (трансляцию в машинное представление) определения интерпретатора объектного языка;
- на объектном языке описать задачу и ввести в машину в качестве данных – разместить в массиве INP;

- дать задание машине вывести общее суждение в форме "Любое значение данных из INP (чье начало определено значением переменной *m*) суть интерпретация объектного языка".

Результат: значение истинности последнего суждения и преобразованные данные.

9. Формализация в метаязыке НФЗ транслятора Т-языка

Любая детерминированная Т-машина описывается [7] конечным множеством продукций: $q_i\ a_i \rightarrow q_j\ a_j\ d_i$, форма которых совместно с определением множеств **Q** ($q_i, q_j \in$ **Q**), **Σ** ($a_i\ ,\ a_j \in$ **Σ**) и **D** ($d_i \in$ **D**) задает язык Т-машин (Т-язык).

Имя каждого состояния ($q_i\ ,\ q_j$) из конечного множества **Q** пусть принадлежит объему понятия "имя_понятия", алфавит **Σ** пусть составлен как объединение объемов понятий "буква", "цифра" и "знак" (их определение дано в [4]), и **D** – пусть две выделенных константы ('0', '1'), имеющих смысл сдвига читающе-записывающей головки вправо и влево соответственно.

Синтаксис этого языка *формально определим* таким образом:

1) Т-язык = продукция (продукция);

2) продукция = состояние1 пробел символ1 фон '→' фон состояние2 пробел символ2 пробел сдвиг;

3) сдвиг = '0' / '1';

4) состояние1 = имя_понятия;

5) состояние2 = имя_понятия;

6) символ1 = буква / цифра / ^мета_знак знак ;

7) символ2 = буква / цифра / ^мета_знак знак.

Семантика любой продукции состоит в смене одного состояния Т-машины на другое, если текущий символ совпадает с символом посылки; в случае применимости продукции – замена текущего символа символом заключения и сдвиг читающе-записывающей головки.

Семантику Т-языка *формально определим* таким образом:

1) семантика_Т-языка = Т_определения анализ_символов запись_символов;

2) Т_определения = Т_определение (Т_определение);

3) Т_определение = состояние1 Т_определяющее ';' ;

4) Т_определяющее = ' = ' Т_элемент ('/' Т_элемент / '/' 'истина');

5) Т_элемент = '^если_' символ1 пробел заключение;

6) заключение = '^записать_' символ2 пробел определение_сдвига пробел состояние2 ';' пробел;

7) определение_сдвига = символ2 / 'Сдвиг_влево' ;

8) анализ_символов = анализ_символа (анализ_символа);

9) запись_символов = записать_символ (записать_символ);

10) анализ_символа = '^если_' символ1 ' $=_{df}$ ' символ1 ';' пробел;

11) записать_символ = '^записать_' символ2 ' = ' символ2 '!' ';' пробел.

Согласно этому определению семантика любой конкретной Т-машины задается конечным (не пустым) множеством определений семантики всех ее состояний. Любое состояние Т-машины – это альтернатива допустимых вариантов замены текущего символа, сдвига читающе-записывающей головки (вправо, пропуская текущий символ, или влево, выполнив терминал "Сдвиг_влево") и перехода в одно из следующих состояний (не обязательно другое) в зависимости от значения текущего символа. Здесь "Сдвиг_влево" – всегда истинный терминал, определенный в форме семантической процедуры, выполняющей декремент значения m. На множестве состояний любой конкретной Т-машины считается известным (для использования) ее начальное состояние q_i, заключительное – определяется как

успешное завершение работы Т-машины, которое достигается из произвольного состояния в результате выяснения неприменимости преобразований текущего символа, предусмотренных определением состояния. Анализ и запись значений символов (исходно указанных в продукциях Т-языка) представлены как отдельные понятия, объединенные в соответствующие группы, расположенные (в синтаксическом представлении в метаязыке) после определений состояний Т-машины.

Семантику трансляции определения любой Т-машины из Т-языка в метаязык *формально определим* следующим образом:

1) Т-язык_в_метаязык = ^преобр_прод RIO ^определ_вперед ^анализы_вперед ^нормализация RIO;

2) ^преобр_прод = преобр_1 (преобр_1);

3) преобр_1 = состояние1 пробел символ1 фон '→' фон состояние2 пробел символ2 пробел# посылка '^записать_' символ2 пробел! преобр_сдвиг? пробел состояние2 ';' пробел анализ_символа записать_символ!;

4) посылка = состояние1 '$=_{df}$ ' '^если_' символ1 '?' пробел;

5) преобр_сдвиг = '0'? символ2! / '1'? 'Сдвиг_влево'!;

6) ^определ_вперед = Т_определение (^вперед_определение Т_определение);

7) ^вперед_определение = анализ_запись Т_определение# RIO Т_определение анализ_запись! RIO;

8) анализ_запись = (анализ_символа / записать_символ);

9) ^анализы_вперед = Т_определения (анализ_символов ^вначале_анализ);

10) ^вначале_анализ = записать_символ анализ_символа# RIO анализ_символа записать_символ! RIO;

11) ^нормализация = (склеить_определения) (удалить_дубли_анализа) (удалить_дубли_записи);

12) склеить_определения = состояние1 Т_определяющее# состояние1 Т_определяющее! ';' (^поиск_состояния? '/' Т_элемент!)? '/' 'истина' ';'!;

13) ^поиск_состояния = ^взять_определение? RIO хвост RIO! /Т_определение ^поиск_состояния;

14) ^взять_определение = то_же_состояние? '$=_{df}$ ' Т_элемент ';' хвост#;

15) то_же_состояние = RB состояние1 RB! / ^RB;

16) хвост = (Т_определение) (анализ_символа) (записать_символ) Код0;

17) удалить_дубли_анализа = анализ_символа# анализ_символа! (^тожд_анализ);

18) ^тожд_анализ = ^исключить_анализ? RIO хвост RIO! / анализ_символа ^тожд_анализ;

19) ^исключить_анализ = тот_же_анализ? хвост#;

20) тот_же_анализ = RB анализ_символа RB! / ^RB;

21) удалить_дубли_записи = записать_символ# записать_символ! (^тожд_запись);

22) ^тожд_запись = ^исключить_запись? RIO хвост RIO! / записать_символ ^тожд_запись;

23) ^исключить_запись = та_же_запись? хвост#;

24) та_же_запись = RB записать_символ RB! / ^RB.

Это определение, как и любое другое, становится выполнимым (интерпретируемым) после его ввода (трансляции в машинное представление).

Результат его интерпретации: представление в метаязыке НФЗ семантики любой Т-машины размещается с начала массива INP, замещая исходное представление в Т-языке. Логический результат: значение истинности – "истина", значения переменных m и n – исходные.

Согласно данному определению трансляция определения любой Т-машины из Т-языка в метаязык НФЗ состоит в следующей последовательности преобразований.

При интерпретации понятия "^преобр_прод" любая последовательность продукций преобразуется из исходного представления (в Т-языке) в представление в метаязыке, причем каждая продукция преобразуется независимо от других и каждой из них ставится в соответствие три определения взаимосвязанных понятий – состояния (посылки), анализа (символа посылки) и записи (символа заключения). Результат преобразования – в массиве OUT. По общему правилу интерпретации инверсии: логический результат – "истина", значения m и n – исходные.

Недостаток полученного представления (тот же, что и исходного) – при его интерпретации необходим ассоциативный поиск нужного определения текущего состояния из-за возможного наличия в описании Т-машины нескольких, расположенных в произвольном порядке, независимых разных определений всякого состояния. Последующие преобразования направлены на устранение этого недостатка и составляют *упорядочение и нормализацию* определения любой Т-машины.

При интерпретации понятий "^определения_вперед" и "^анализы_вперед" выполняется *упорядочение* в массиве OUT преобразованного определения Т-машины: определения всех состояний перемещаются в начало, анализы символов размещаются вслед за определениями состояний, а записи символов остаются в конце определения Т-машины.

В процессе интерпретации итерации понятия "склеить_определения" выполняется объединение множества различных определений любого состояния в одно – определение состояния в форме *альтернативы* допустимых в данном состоянии преобразований текущего символа и переходов в следующее состояние. В процессе интерпретации итерации понятия "удалить_дубли_анализа" исключаются тождественные анализы символов, а итерации понятия "удалить_дубли_записи" – тождественные записи символов. Результат этих преобразований порождается в массив INP, замещая исходное представление в Т-языке.

Полученное определение (в метаязыке) любой конкретной Т-машины после ввода и присоединения к имеющейся базе знаний может быть применено для решения соответствующей задачи обработки данных (введенных в массив INP), которая формулируется обычным образом: как доказательство общего категорического суждения формы "Любое конкретное значение данных из INP (начало которых определено значением переменной m) суть q_i", где q_i – имя начального состояния Т-машины, определение которой введено в базу знаний.

Представленная формализация транслятора Т-языка обосновывает

Утверждение 14. Любой 0-язык в форме транслятора может быть определен в метаязыке НФЗ.

Используя этот результат, **транслирующе-транслирующий процесс** постановки и решения произвольной задачи некоторой предметной области определим следующей последовательностью преобразований [10]:

- на метаязыке НФЗ описать транслятор языка предметной области и ввести в машину в качестве данных – разместить в массиве INP;
- выполнить ввод (трансляцию в машинное представление) определения транслятора языка предметной области и присоединить к имеющейся базе знаний;
- на языке предметной области описать задачу и ввести в машину в качестве данных – разместить в массиве INP;
- дать задание машине вывести общее суждение в форме "Любое значение данных из INP (начало которых определено значением переменной m) суть транслятор языка предметной области", результат – значение истинности суждения и определение на метаязыке НФЗ конкретной задачи предметной области;
- выполнить ввод (трансляцию в машинное представление) определения

(результат предыдущего пункта) этой конкретной задачи и присоединить к имеющейся базе знаний;

- исходные данные задачи разместить в массиве INP;
- дать задание машине вывести общее суждение в форме "Любое значение данных из INP (начало которых определено значением переменной m) суть начало программы конкретной задачи".

Результат: значение истинности последнего суждения и преобразованные данные, смысл которых определяется семантикой задачи.

Выводы

Исследованы выразительные возможности метаязыка НФЗ по отношению к формальным языкам разного уровня. Показано, что для формального языка каждого уровня в метаязыке НФЗ есть адекватные выразительные средства. Представленные результаты исследования, иллюстрированные множеством примеров, обосновывают вывод об универсальных выразительных возможностях метаязыка НФЗ, о его пригодности в качестве базового инструмента для реализации языков и систем произвольной сложности и назначения.

1. International Standard ISO/IEC 14977 : 1996(E). Электронный ресурс. Режим доступа: http://www.cl.cam.ac.uk/~mgk25/iso-14977.pdf
2. Спосіб представлення і використання знань / О.П. Кургаєв, С.М. Григор'єв / Патент на корисну модель **UA 92484 U**, 2014р., Бюл. №16.
3. Kurgaev A., Grygoryev S. The normal forms of knowledge. Dopov. NAN Ukraine, 2015, № 11. P. 36–43.
4. Кургаев А.Ф., Григорьев С.Н. Метаязык нормальных форм знаний. *Кибернетика и системный анализ*. 2016. Том. 52. № 6. С. 11–20.
5. Ершов А.П. Предварительные соображения о лексиконе программирования. *Кибернетика и вычислительная техника*. Под ред. В.А. Мельникова. 1985. Вып. 1. С. 199–210.
6. Глушков В.М., Цейтлин Г.Е., Ющенко Е.Л. Алгебра. Языки. Программирование. К.: Наук. думка, 1978. 320 с.
7. Гросс М., Лантен А. Теория формальных грамматик. М.: Мир, 1971. 294 с.
8. Хомский Н. Формальные свойства грамматик. *Кибернетический сборник*. Новая серия. М.: Мир, 1966. Вып. 2. С. 122–230.
9. Войшвилло Е.К. Понятие как форма мышления: логико-гносеологический анализ. М.: Изд. МГУ, 1989. 239 с.
10. Кургаев А.Ф. Проблемная ориентация архитектуры компьютерных систем. Киев: Сталь, 2008. 540 с.
11. Тейз А., Грибомон П., Луи Ж. и др. Логический подход к искусственному интеллекту: от классической логики к логическому программированию. Пер. с фр.; Под ред. Г.П. Гаврилова. М.: Мир, 1990. 432 с.
12. Хомский Н. и Миллер Дж. Введение в формальный анализ естественных языков. *Кибернетический сборник*. Новая серия. М.: Мир, 1965. Вып. 1. С. 229–290.
13. Хомский Н. О некоторых формальных свойствах грамматик. *Кибернетический сборник.* М.: Изд-во иностр. лит., 1962. Вып. 5. С. 279–311.
14. Гильберт Д., Бернайс П. Основания математики: Теория доказательств. М.: Наука, 1982. 652 с.
15. Кургаев А.Ф. Логические формы определения понятия. *УСиМ.* 1998. № 2. С. 3–12.
16. Chomsky, N.: Aspects of the Theory of Syntax, MIT Press, Cambridge, MA, 1965.
17. Fu K.S. Syntactic Pattern Recognition and Applications. New Jersey: Prentice-Hall, Inc., Englewood Cliffs, 1982. 596 p.
18. Minsky M.L. Computation: Finite and Infinite Machines. NY: Prentice-Hall, Englewood Cliffs, 1967.
19. Kurgaev A., Grygoryev S. The Universal Turing Machine Interpreter. Dopov. NAN Ukraine, 2016, 10. С. 28–34. https://doi.org/10.15407/dopovidi2016.10.028
20. Shannon C.E. A universal Turing machine with two internal states. Princeton: Automata Studies, 1956.

References

1. International Standard ISO/IEC 14977 : 1996(E). [Electronic resourse]. – Mode of

access: http://www.cl.cam.ac.uk/~mgk25/iso-14977.pdf
2. Kurgaev A., Grygoryev S. Utility model patent UA 92484 U, 2014, Bulletin № 16 (in Ukrainian).
3. Kurgaev, A. The normal forms of knowledge / A.Kurgaev, S.Grygoryev. – Dopov. NAN Ukraine, 2015, № 11. – P. 36-43 (in Russian).
4. Kurgaev, A. Metalanguage of Normal Forms of Knowledge / A.Kurgaev, S.Grygoryev. – Cybernetics and Systems Analysis. – November 2016, 52(6). – P. 11-20. (in Russian).
5. Ershov, A.P. Preliminary considerations on the lexicon of programming // Cybernetics and computer technology / Ed. V.A. Melnikov. - 1985. - Vyp. 1. – P. 199–210. (in Russian).
6. Glushkov, V. Algebra. Languages. Programming / V.Glushkov, G.Zeitlin, E.Yushchenko. – K.: Nauk. Dumka, 1978. – 320 p. (in Russian).
7. Gross M. Theory of formal grammars / M.Gross, A.Lunten. – M.: Mir, 1971. - 294 p. (in Russian).
8. Chomsky, N. Formal properties of grammars. / Cybernetic collection. New episode. – Moscow: Mir, 1966. – Vyp. 2. – P. 122–230. (in Russian).
9. Voishvillo E. Concept as a form of thinking: logical-gnoseological analysis / E. Voishvillo. – Moscow: Izd. MSU, 1989. – 239 p. (in Russian).
10. Kurgaev, A.F. Problem orientation of the architecture of computer systems / A.F. Kurgayev. – Kiev: Steel, 2008. – 540 p. (in Russian).
11. Teiz, A. Logical approach to artificial intelligence: from classical logic to logical programming / A. Teiz, P. Gribomon, J. Louis et al. – Transl. With fr .; Ed. G.P. Gavrilov. – Moscow: Mir, 1990. – 432 p. (in Russian).
12. Chomsky, N. Introduction to the formal analysis of natural languages / N. Chomsky, J. Miller. – Cybernetic collection. New episode. – Moscow: Mir, 1965. – Vyp. 1. – P. 229–290. (in Russian).
13. Chomsky, N. On some formal properties of grammars / Cybernetic collection. – Moscow: Izd-vo inostr. Lit., 1962. - Vyp. 5. – P. 279–311. (in Russian).
14. Gilbert, D. Foundations of Mathematics: Theory of Evidence / D.Gilbert, P.Bernays. – M oscow: Nauka, 1982. – 652 p. (in Russian).
15. Kurgaev A. Logical forms of definition of the concept // Control Systems and Computers. – 1998. – № 2. – P. 3–12. (in Russian).
16. Chomsky, N. Aspects of the Theory of Syntax, MIT Press, Cambridge, MA, 1965.
17. Fu, K.S. Syntactic Pattern Recognition and Applications. – New Jersey: Prentice-Hall, Inc., Englewood Cliffs, 1982. – 596 p.
18. Minsky, M.L. Computation: Finite and Infinite Machines. – NY: Prentice-Hall, Englewood Cliffs, 1967.
19. Kurgaev, A. The Universal Turing Machine Interpreter / A.Kurgaev, S.Grygoryev. – Dopov. NAN Ukraine, 2016, 10. – P. 28-34. https://doi.org/10.15407/dopovidi2016.10.028 (in Russian)
20. Shannon, C.E. A universal Turing machine with two internal states. – Princeton: Automata Studies, 1956.

Получено 23.05.2017

Об авторах:

Кургаев Александр Филиппович,
доктор технических наук,
профессор, ведущий научный сотрудник.
Количество публикаций в
украинских изданиях – более 200.
Количество публикаций в
зарубежных индексированных
изданиях – около 10,
H-індекс (Google Scholar): 4
http://orcid.org/0000-0001-5348-2734,

Григорьев Сергей Николаевич,
соискатель.
Количество публикаций в
украинских изданиях – более 20.
http://orcid.org/0000-0002-7583-3088.

Место работы авторов:

Институт кибернетики
имени В.М. Глушкова НАН Украины.
03187, Киев-187,
проспект Академика Глушкова, 40.
Тел.: 050 881 6218,
050 505 05 75.
E-mail: afkurgaev@ukr.net,
Sergey@Grigoriev.kiev.ua

УДК 519.164:004.14

О.О. Слабоспицька

РАМКОВА МОДЕЛЬ АДАПТИВНОГО КОМПОЗИТНОГО СЕРВІСУ В СЕМАНТИЧНОМУ ВЕБ-СЕРЕДОВИЩІ

Обґрунтовано новий підхід до on-line композиції семантичного Веб-сервісу, який є адаптивним – здатним до змін поведінки для задоволення нових вимог і пристосування до нових (не)передбачених ситуацій – і застосовним третіми сторонами, для довільного методу компонування. Надано рамкову модель цього адаптивного композитного сервісу (АКС) як динамічної лінії змінюваних семантичних сервісів для споживачів у цільовій предметній області. Сформульовано засади побудови АКС за рахунок керування динамічною варіабельністю цієї лінії. Розроблено діагностичну модель варіабельності АКС для виявлення потреб і стратегій його on-line адаптування. Формалізовано операції реалізації стратегій – (не)передбаченого змінення складу атомарних компонентних сервісів, структури й функцій їх проміжних композицій, функцій самого АКС – за рахунок наданого ізоморфізму між моделями його функцій і композитних сервісів. Запровадження підходу сприяє підвищенню ефективності застосування й реінжинірингу ділових процесів з різнорідними та змінними контекстами.

Ключові слова: адаптивний композитний сервіс; семантичний Веб-сервіс, динамічна лінія програмних продуктів, динамічна варіабельність, метод композиції семантичних Веб-сервісів, пошук сервісів, операція адаптування, ізоморфізм, сервіс-орієнтована програмна система.

Постановка проблеми

За умов стрімкого розвитку програмної індустрії у світі та Україні, дефакто стандартом автоматизованої підтримки ділових процесів сучасних організацій стає композиція виконуваних Веб-сервісів, on-line відшуканих в Інтернет. Саме такі сервіс-орієнтовані програмні системи (СоПС), звані *композитними Веб-сервісами*, здатні уможливити декларовані переваги сервісного програмування [1] – скорочення вартості й терміну розробки та контрольованість якості й окупності в життєвому циклі. Ці СоПС найбільш запитані в динамічних предметних областях (ПрО), де параметри інфраструктури доступу до сервісів з їх складу та ролі, взаємодії, потреби й пріоритети зацікавлених сторін різнорідні, змінні й слабко передбачувані під час проектування, оскільки саме тут їх зазначені переваги особливо відчутні.

Але дійсне досягнення згаданих переваг композитних Веб-сервісів неможливе без додаткового забезпечення для них:

1) здатності до змін поведінки під час виконання для задоволення нових вимог і пристосування до нових ситуацій (як передбачених за проектування, так і непередбачених), званої *адаптовністю* [2, 3] та, відповідно, *адаптивністю* [2, 4];

2) ефективного багаторазового on-line відшукування виконуваних сервісів-складників та їх композиції (званої *динамічною*) на підтримку поточних потреб споживачів;

3) постійної застосовності третіми сторонами цільової ПрО на підставі описів у реєстрах Інтернет (UDDI тощо).

Однак аналіз актуального доробку інженерії СоПС висвітлює істотне розмежування підходів до конструювання СоПС з властивостями 1)–3). Зокрема, огляд [5] висвітлює значну різнорідність формалізмів моделювання *адаптивних* СоПС. Показовими прикладами є:

– подання СоПС динамічною лінією проміжних композитних сервісів на підтримку потреб окремих груп споживачів [3, 4] (A_1);

– побудова СоПС на засадах повно-аспектного керування її спеціально визначеною варіабельністю [6] (A_2);

– формальне подання процесу побудови СоПС як вирішення проблеми автоматичного планування через перевірку моделей шляхом формалізації фраг-

ментів підтримуваних ділових процесів [7, 8] (A_3);

– проектування СоПС як віртуальної адаптивної організації сервісів [9] (A_4).

Альтернативою $A_1 – A_4$ є каркас моніторингу й адаптування СоПС під час її проектування й виконання (A_5, проект S-Cube[1]). Він забезпечує проактивні узгод-жені змінення всіх структурних рівнів СоПС – від підтримуваних ділових процесів і потреб її споживачів до операційних систем [1]. Для цього в каркасі передбачено шаблони подій бажаної/небажаної поведінки СоПС і відповідні їм правила добору дій з адаптації, обновлювані згідно з досвідом виконання [2, 5, 10]).

На жаль, підходи A_1, A_4, A_5 передбачають побудову СоПС "для компонувача", а A_2, A_3 – "для компонувача й підписувача", не забезпечуючи властивість 3).

У свою чергу, відшукування й компонування сервісів зазвичай трактують як автономні й розмежовані одноразові дії. Винятком є методи для *семантичних* сервісів, зокрема семантичне зіставлення:

– входів і виходів сервісів (P. Rodríguez-Mier [11], C_1);

– входів, виходів, перед- і постумов (S. Bansal [12], C_2).

Співставлення підходів $A_1 – A_5$, не зорієнтованих на семантику Веб-сервісів, і C_1, C_2, що ігнорують вимоги адаптивності, висвітлює нагальну потребу непротирічної інтеграції переваг $A_1 – A_5$ і C_1, C_2 у *новому підході* до динамічної композиції семантичних Веб-сервісів, результат якої має властивості 1)–3) і тому названий *адаптивним композитним семантичним Веб-сервісом* (АКС). Опис базових рішень такого підходу і побудова рамкової моделі АКС є метою статті.

Стаття підсумовує доробок автора в проекті ДР 0112U002764 ІПС НАН України під керівництвом академіка НАН України, д-ра фіз.-мат. наук П.І. Андона (2013–2016).

Установчі рішення з побудови АКС

Пропонований підхід узагальнює модель процесу адаптивного компонування Веб-сервісів різнорідними споживачами [13] та формальні засоби забезпечення адаптовності програмних продуктів [14] для семантичних Веб-сервісів згідно з рамковим каркасом адаптування СоПС [10].

Підхід охоплює:

а) формування вимог до АКС і його побудови для реалізації властивостей 1)–3);

б) визначення рамкових моделей згідно з вимогами – самого АКС, процесу його побудови (з первинним проектуванням і подальшим поетапним виконанням/адаптуванням до змін потреб споживачів, умов виконання і/або відмов компонентних сервісів), окремого етапу побудови та операцій адаптування АКС на підставі його запропонованої моделі;

в) уточнення рамкових моделей для ефективних методів динамічної композиції, насамперед C_1, C_2;

г) розвиток наданих моделей і методів на підтримку адаптації АКС до проявів незадовільної якості (у цілому або його складників).

Запропоновані вимоги охоплюють:

– уніфікацію засобів забезпечення адаптовності та динамічної контекстно-залежної адаптивності (B_1);

– узгодження дій з адаптації на п'яти де-факто стандартних рівнях АКС [1] (потреб споживачів, ділових процесів, атомарних і композитних сервісів, їх реалізацій за допомогою певних методологій і мов програмування, операційних систем) (B_2);

– урахування залежностей між подіями – підставами адаптації АКС (B_3);

[1]Офіційний сайт: http://www.s-cube-network.eu/

– урахування неповноти й обмеженої вірогідності даних про стан АКС і контекст її виконання (B_4);

– інваріантність формального апарату до методу динамічного компонування семантичних Веб-сервісів (B_5).

На підставі зіставлення вимог B_1–B_5 з рамковим життєвим циклом композитного Веб-сервісу [15], каркасом A_5 [10] і підходами A_1, A_2 й C_1, C_2 зафіксовано базові об'єкти процесу побудови АКС і типові проблеми їх застосування в семантичному Веб-середовищі ($П_1$–$П_4$). Склад і співвідношення об'єктів і проблем показано на рис. 1 (об'єкти позначено потовщеним контуром, а проблеми – "прапорцями").

Рішення з опрацювання проблем $П_1$–$П_4$ підсумовано в таблиці.

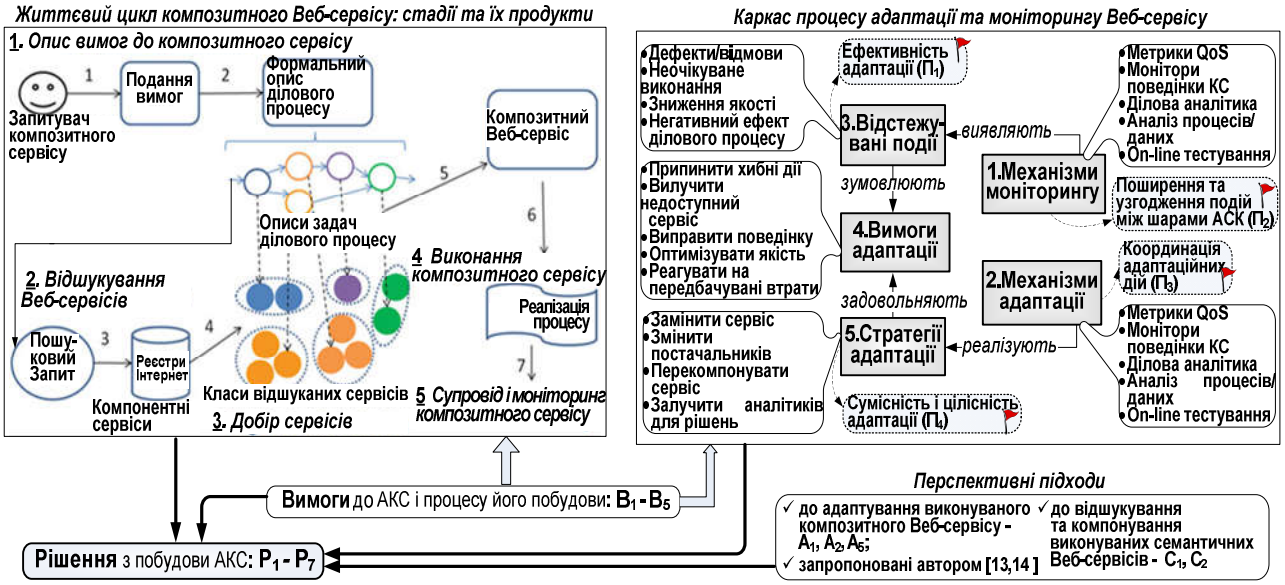


Рис. 1. Підстави та об'єкти рішень з побудови АКС

Таблиця

Сутність пропонованого підходу до побудови АКС

Пропоноване рішення		Опрацьована	
Код	Сутність	Вимога	Проблема
1	2	3	4
P_1	Конструювання АКС у форматі *динамічної лінії* програмних продуктів [16]. Розгляд продукту як композитного семантичного сервісу на підтримку певного вділового процесу в цільовій ПрО	B_1–B_3	$П_3$, $П_4$
P_2	Конструювання АКС за допомогою *трансформаційного* підходу [17] на підставі її взаємопов'язаних моделей у просторах [17,18]: *проблеми* (незалежно від формалізмів опису семантики компонентних Веб-сервісів і методу їх динамічного компонування) та *рішень* (у залежності від цього методу).	B_1, B_2, B_5	$П_2$
P_3	Подання АКС у просторі проблеми за допомогою розвитку *моделі властивостей під час виконання* [16] відповідно до особливостей варіабельності Веб-сервісів [6] і вищезазначеної п'ятирівневої структури АКС як СоПС [1].	B_2	$П_2$

Закінчення таблиці

1	2	3	4
$Р_4$	Розгляд процесу конструювання АКС як процесу управління *динамічною варіабельністю* [6,16] лінії композитних сервісів. Подання цього процесу композицією спеціальних функцій управління варіабельністю в єдиному інформаційному середовищі, структурованому на підставі моделі властивостей під час виконання.	$В_1$, $В_5$	$П_1$–$П_4$
$Р_5$	Заміна шаблонів подій – підстав адаптації [5] рамковими подіями потенційної й фактичної незадовільності очікуваної або поточної варіабельності АКС. Діагностування типу неадекватності й рекомендування адаптаційних дій за допомогою діагностичної підмоделі варіабельності композитних Веб-сервісів.	$В_3$, $В_4$	$П_3$, $П_4$

Згідно з традиційним визначенням Інституту програмної інженерії[2], лінія програмних продуктів у певній ПрО – це їх відкритий набір, елементи якого мають керовану множину спільних *властивостей* на підтримку потреб споживачів у ПрО і розробляються попередньо визначеним способом зі спільної множини *ресурсів повторного використання* [17, 18]. Властивість – потреба, функція, нефункціональна характеристика, запитана споживачами.

ISO/IEC 26550 [18] фіксує особливості процесу розроблення лінії продуктів:

- простори Проблеми та Рішень, утворені передбаченими для ПрО спільними (commonalities) і змінними (variabilities) властивостями продуктів і, відповідно, ресурсами з правилами їх компонування упродукти для реалізації властивостей;
- процеси конструювання *домену* (де розробляють ресурси) і *застосунків* (де їх компонують у продукти із спільними й передбаченими змінними властивостями);
- процес керування варіабельністю – здатністю продуктів лінії до ефективного розвитку, зміни, налаштування або конфігурування для використання в певному контексті [17, 18];
- формалізм *моделі властивостей* (feature model, МВ) [17, 19] – їх дерева з додатковими відношеннями взаємозалежності [13], яка є де-факто стандартом опису простору проблеми.

Динамічна лінія продуктів [16] надає розвиток статичної за рахунок змінності за часом складу й структури просторів проблеми й рішень згідно зі змінами потреб споживачів та умов виконання продуктів. Відповідно, МВ стає змінною за часом, а варіабельність під час виконання замінює канонічну варіабельність.

Отже, далі АКС модельовано як поповнювану систему (змінюваних) композитних семантичних Веб-сервісів на підтримку потреб різнорідних суб'єктів змінних ділових процесів у цільовій ПрО. Ці сервіси названо *цільовими сервісами* в АКС (ЦС). Кожний ЦС має спільні й змінні функції згідно з поточними потребами споживачів, умовами виконання і станом АКС, передбачені на період до зміни потреб і/або умов. У разі такої зміни склад і взаємозв'язки функцій окремих ЦС і/або АКС у цілому та ЦС для них мають змінюватися узгоджено за допомогою операцій адаптування для дотримання відповідності між ними.

Рамкова модель АКС: структура та застосування для його адаптації

Базові визначення. Прийнятий розгляд АКС як динамічної лінії ЦС зумовлює два узгоджені складники його рамкової моделі. Першим є інтенсіональне подання АКС (моделі просторів проблеми й рішень), а другим – його екстенсіональне

[2] http://www.sei.cmu.edu/productlines/

подання (поточний репозиторій опублікованих ЦС).

Формальний опис інтенсіонального подання потребує низки допоміжних визначень.

Визначення 1 [13]. Нехай t_0 – момент початку виконання АКС, $[t_{u-1}, t_u)$, $u \in \mathbf{N}$ – u-й період, в якому склад функцій АКС і залежності між ними відповідають поточним потребам суб'єктів ділових процесів цільової ПрО, F_u – множина однозначних описів функцій АКС (для стислості званих далі просто функціями) в u-му періоді.

Будемо говорити, що функції $f_i, f_j \in F_u$ пов'язано відношенням:

а) *обумовленості* (IM_F), *зіставлення* (EQ_F) або *виключення* (EX_F), якщо з реалізації в довільному ЦС з АКС функції f_i випливає реалізація в ньому й f_j, f_i, f_j можна реалізувати в ЦС лише разом або, відповідно, реалізувати їх разом неможливо;

б) (m, n)-*варіантного підпорядкування* ($C_F{}^{mn}$, $0 \le m \le n \le (|F_u| - 1)$), якщо функцію f_i (звану далі *підпорядковуючою,* або *батьком*) реалізовано в ЦС з АКС тоді й тільки тоді, коли в ньому реалізовано довільну підмножину функцій SV ($\varnothing \subseteq SV \subseteq VF \subset F_u$, $m \le |SV| \le n$, $f_j \in SV$), яка в разі $m > 0$ містить f_j (звану, відповідно, *підпорядкованою*, або *нащадком*) і має своїми елементами від m до n функцій $f_j^* \in VF$.

У разі $m = n = |VF|$ відношення $C_F{}^{mn}$ зветься *обов'язковим* підпорядкуванням, а кожна підпорядкована функція $f_j \in VF$ – *обов'язковою* для реалізації підпорядковуючої функції. Отже, обов'язкові функції спільні для всіх ЦС в АКС. Функції, що не є обов'язковими, звуть *змінними*.

Визначення 2. Модель *функцій* АКС в u-му періоді їх відповідності потребам споживачів цільової ПрО – це п'ятірка

$$FM_u = \langle F_u;\ IM_F,\ EQ_F,\ EX_F,\ C_F{}^{mn} \rangle. \quad (1)$$

Далі будемо вважати АКС коректним впродовж його виконання в сенсі

Визначення 3. АКС є *коректним* в u-му періоді відповідності його функцій потребам споживачів цільової ПрО, якщо одночасно виконано такі умови:

а) поточний метод динамічного компонування семантичних Веб-сервісів (κ_u) має природну властивість [11, 12]: якщо кожна з композицій методом κ_u (названих κ_u-*композиціями*) Веб-сервісів з двох різних множин реалізує певну функцію, то її реалізує й κ_u-композиція елементів об'єднання цих множин;

б) множина F_u містить універсальну функцію $f_0 =$ "Задоволення потреб суб'єктів ділових процесів цільової ПрО", а граф

$$\langle F_u;\ E_F{}^{mn} \rangle,\ 0 \le m \le n \le (|F_t| - 1|), \quad (2)$$

$$E_F{}^{mn} = \{(f,g) \in F_u \mid C_F{}^{mn}(f,g)\}$$

є *деревом* з коренем f_0, де листки ($lf \in LF_u$) відповідають певним елементарним завданням підтримуваних ділових процесів і звуться далі *термінальними,* а вузли ($f \ne f_0$, $f \notin LF_u$) – *проміжними* функціями АКС;

в) функції, разом безпосередньо підпорядковані спільному батьку в дереві (2), не пов'язано відношеннями IM_F, EX_F;

г) кожній термінальній функції lf зіставлено такий опис, згідно з методом κ_u, ЦС її реалізації (ld (lf;κ_u)), для якого послідовно відшукано m_u множин, названих lf *-релевантними*, семантичних Веб-сервісів з описами ($wd_{qs}(\kappa_u)$) того ж типу, що й ld (lf ;κ_u), без доступу до внутрішньої структури, on-line доступних в q-му підперіоді зазначеного u-го періоду, κ_u-композиція яких згідно з описом ld реалізує lf :

$$AW_q(ld(lf))=\{wd_{qs}(\kappa_u), s\geq 1\}\supset\varnothing, \quad (3)$$

$$q=1,\ldots, m_u,\ m_u>0;$$

д) $\forall(i,j)$ lf_i- й lf_j-релевантні множини не є підмножинами одна одної.

Дотриманню умов коректності а)–д) сприяють стандартизовані кращі практики побудови МВ [17, 18] і новітня техніка *Аналізу впливів* (Г. Аджік) [13] для визначення вершин першого рівня в дереві (2), а також значні обсяги on-line доступних семантичних Веб-сервісів з широким спектром функцій і привабливою якістю, оскільки κ_u-компонування їх lf-релевантних підмножин зазвичай значно ефективніше для реалізації lf, ніж традиційне розроблення lf-релевантних сервісів.

Зауваження 1. Виконання умов г), д) з визначення 3 є критерієм завершення декомпозиції функцій АКС, тобто надання поточним вершинам дерева (3) статусу термінальних функцій.

Згідно з трансформаційним підходом до побудови АКС (рішення P_2, таблиця) запровадимо відображення для формального зіставлення функціям коректного АКС (елементам простору проблеми) ЦС їх реалізації (елементів простору рішень).

Зауваження 2. *Нехай* $f\notin LF$ – нетермінальна функція АКС. Тоді множина термінальних функцій, опосередковано варіантно підпорядкованих f у моделі функцій АКС (1), має вигляд:

$$IC(f)=M(f)\cup(\cup_{\nu\geq 0}O_\nu(f))\subseteq LF), \quad (4)$$

$$\varnothing\subseteq M(f),\ \varnothing\subseteq O_\nu(f),\ \cap_\nu O_\nu(f)=\varnothing,$$

$$|IC(f)|\geq 2,$$

де $M(f)$ – обов'язкові функції;

$O_\nu(f)$ – ν-а множина змінних функцій, які реалізуються або не реалізуються в АКС лише разом.

Згідно з (4), функцію f реалізовано в ЦС з АКС тоді й лише тоді, коли в ЦС реалізовано одну з множин термінальних функцій $M(f)\cup O_\nu(f)$.

Визначення 4. Нехай d – деякий опис семантичного Веб-сервісу згідно з методом κ_u, $pr(d)$ – предикат на множині κ_u-композицій із змістом: "κ_u-композиція відповідає опису d". *Перетворенням* функцій коректного АКС в u-му періоді відповідності його функцій потребам споживачів цільової ПрО назвемо відображення $\tau(\kappa_u,\cdot)$: $F_u\to S_u$:

$$\forall lf\in LF_u\ |\ \exists\ (ld(lf), q>0, AW_q(ld))\ |$$

$$\tau(\kappa_u,\ lf\ )=lw(lf)=(ld\,;\ \kappa_u(ld,\ AW_q(ld))\,; \quad (5)$$

$$\forall f\notin LF_u\ \tau(\kappa_u,f)=\{\ (fd;\ \kappa_u(fd,\ \{\tau(\kappa_u,\ lf\ ),\quad (6)$$

$$lf\in M(f)\cup O_\nu(f)\ \}),\ \nu\geq 0\},$$

де $S_u=\{\tau(\kappa_u,\ f),\ f\in F_u\}$ – множина *моделей ЦС* (5), (6), передбачених в АКС для реалізації його функцій у зазначеному u-му періоді (в довільний момент t u-го періоду $t_{(q-1)u}\leq t<t_{qu}$, $t_{0u}=t_{u-1}$, $t_{muu}=t_u$ множина S_u формується із застосуванням q-ї lf-релевантної множини $AW_q(ld)$);

fd – такий опис семантичного Веб-сервісу згідно з методом κ_u, що

$$pr(fd)\Leftrightarrow\wedge_{lf\in M(f)\cup O_\nu(f)}pr(ld(lf));$$

$M(f)$ і $O_\nu(f)$ – множини обов'язкових і змінних термінальних функцій, підпорядкованих f у сенсі зауваження 2.

Образ функції $f\in F_u$ $w(f)=\tau(\kappa_u,f)$ назвемо *моделлю f-релевантного композитного* ЦС, відповідно *термінального* (5), якщо $f\in LF_u$; *проміжного*, якщо $f\notin LF_u$, $f\neq f_0$ і *кореневого*, якщо $f=f_0$ (6).

Зауваження 3. З умови д) коректності АКС випливає, що будь-які два предикати $pr(ld(lf_i))$ і $pr(ld(lf_i))$ не пов'язано відношенням імплікації.

Зауваження 4. З виразу (6) і умов коректності АКС випливає, що перетворення $\tau(\kappa_u,\cdot)$ є ізоморфізмом між множинами функцій і ЦС АКС.

Зауваження 5. Вигляд описів ЦС *ld* і *fd*, перетворення $\tau(\kappa_u,)$ і відповідної їм композиції (5), (6) визначено поточним методом κ_u. Для методу [11] ці вирази та приклади ярусно-паралельних графів *нормалізованої* композиції наведено в [20].

На підставі визначення 1 індукуємо на множині ЦС S_u відношення IM_S, EQ_S, EX_S, $C_S{}^{mn}$, однойменні з відношеннями на множині функцій F_u, так, щоб $\forall u>0$ перетворення (5), (6) було ізоморфізмом між моделлю функцій АКС FM_u (1) і моделлю ЦС їх реалізації

$$SM_u = \langle S_u; IM_S, EQ_S, EX_S, C_S{}^{mn} \rangle \quad (7)$$

як алгебраїчними моделями типу ⟨2,2,2,2,2⟩.

Визначення 5 [13]. Відношення *обумовленості* (IM_S), *зіставлення* (EQ_S), *виключення* (EX_S) і (*m*,*n*)-*варіантного підпорядкування* ($C_F{}^{mn}$, $0 \le m \le n \le (|C_u| - 1)$ пов'язують ЦС в АКС тоді й тільки тоді, коли однойменні відношення пов'язують, у сенсі визначення 1, прообрази цих ЦС за перетворенням $\tau(\kappa_u,\cdot)$: $\forall$ ($w_i, w_j \in S_u$)

$$IM_S(w_i, w_j) \Leftrightarrow IM_F(\tau_t^{-1}(w_i), (\tau_t^{-1}(w_j))); \quad (8)$$

$$EQ_S(w_i, w_j) \Leftrightarrow EQ_F(\tau_t^{-1}(w_i), (\tau_t^{-1}(w_j)));$$

$$EX_S(w_i, w_j) \Leftrightarrow EX_F(\tau_t^{-1}(w_i), (\tau_t^{-1}(w_j)));$$

$$C_S{}^{mn}(w_i, w_j) \Leftrightarrow C_F{}^{mn}(\tau_t^{-1}(w_i), (\tau_t^{-1}(w_j))).$$

Зауваження 5. У моделі ЦС SM_u (7) граф ЦС

$$\langle S_u; E_S{}^{mn} \rangle, \ 0 \le m \le n \le (|S_u| - 1), \quad (9)$$

$$E_S{}^{mn} = \{(w_i, w_j) \in S_u \mid C_S{}^{mn}(w_i, w_j)\}$$

є *деревом,* ізоморфним дереву функцій $\langle F_u; E_F{}^{mn} \rangle$ (2) за перетворенням $\tau(\kappa_u,\cdot)$.

Тому множина моделей ЦС АКС має вигляд

$$S_u = LS_u \cup (\cup_{k=2,\dots,n} IS_{ku}) \cup \{w_0\}, \quad (10)$$

де $LS_u = IS_{1u}$ – множина термінальних композитних ЦС (5), які є листками дерева (9);

IS_{ku} – множина проміжних композитних ЦС (6), які є вузлами *k*-го рівня підпорядкованості в (9) за відношенням $C_S{}^{mn}$;

$w_0 = \tau(\kappa_u, f_0)$ – універсальний (змінний) композитний ЦС реалізації універсальної функції f_0 (тобто всіх функцій АКС), що є коренем дерева (9).

Отже, множина ЦС (5), (6), (10) являє собою *модель* пошарованого *репозиторію* АКС у сенсі [17, 18]. У свою чергу, множина

$$AP_u = \cup_{lf \in LF_u} AW_q(ld(lf)), \quad (11)$$

$$t_{(q-1)u} \le t < t_{qu}, \ t_{0u} = t_{u-1}, \ t_{muu} = t_u$$

є *платформою* АКС у сенсі [17, 18]: усі її елементи розглядаються як атомарні, подані своїми описами ($wd_{qs}(\kappa_u)$) згідно з методом κ_u, а довільний сервіс $w \in S_u$ є їх κ_u-композицією – безпосередньою (5) або дворівневою (6).

Зауваження 6. За виразами для моделі функцій FM_u (1), моделі ЦС SM_u (7), (8) і платформи AP_u (11), змінення АКС для опрацювання його неадекватності потребам споживачів і/або умовам використання, названі *адаптаційними*, стосуються узгоджених між собою:

- складу і/або структури функцій;
- ЦС їх реалізації;
- елементів платформи АКС.

Для діагностування подій – підстав адаптаційних змінень, надання рекомендацій щодо них і відстеження їх впливів на п'яти зазначених рівнях АКС запровадимо.

Визначення 6. *Інтегрована модель варіабельності* АКС у *v*-му періоді між його адаптаційними зміненнями – це трійка

$$VM_v = \langle SV_v, AV_v, DV_v \rangle, \quad (12)$$

де SV_ν і AV_ν – моделі варіабельності структури АКС та його артефактів;

DV_ν – діагностична модель рівня задовільності варіабельності.

Внутрішню структуру SV_ν, AV_ν, DV_ν описано далі визначеннями 9–11.

Поєднання виразів (1), (7), (8) і (12) уможливлює формальний опис інтенсіонального подання АКС за допомогою.

Визначення 7. Інтенсіональне подання АКС у довільний момент t між моментами його останнього здійсненого та наступного адаптаційного змінення (відповідно, t_{n-1} і t_n) – це поповнюваний структурований кортеж

$$IM(t)=\langle r,\langle FM_\nu, SM_\nu, VM_\nu, \kappa_\nu\rangle,$$

$$\nu=1,...,n\rangle, \qquad (13)$$

де r – унікальне ім'я АКС.

Описану структуру інтенсіонального подання АКС показано на рис. 2. Стрілками вгорі тут позначено послідовність адаптаційних змінень АКС. Кожне змінення ($\alpha(t_\nu)$) є композицією операцій над об'єктами неадекватності поведінки АКС потребам споживачів і/або умовам виконання, виявленими в моменти t_ν і приналежними до зазначених у зауваженні 6 типів, аж до досягнення поточного стану АКС в момент t $IM(t)$ (13).

Як показано на рис. 2, складниками подання $IM(t)$ є "фрагменти" просторів проблеми й рішень АКС, формовані в процесах конструювання домену та застосунків у поточному n-му періоді між адаптаційними зміненнями АКС і взаємопов'язані за допомогою перетворення $\tau(\kappa_n,\cdot)$ та елементів інтегрованої моделі варіабельності.

Для визначення цих елементів зафіксуємо особливості варіабельності композитних Веб-сервісів [6]:

а) три її взаємопов'язані типи (k):

– *спостережувана* (exposed), призначена для опису припустимих змін операцій, протоколів і типів повідомлень в інтерфейсах ЦС з АКС ($k=1$);

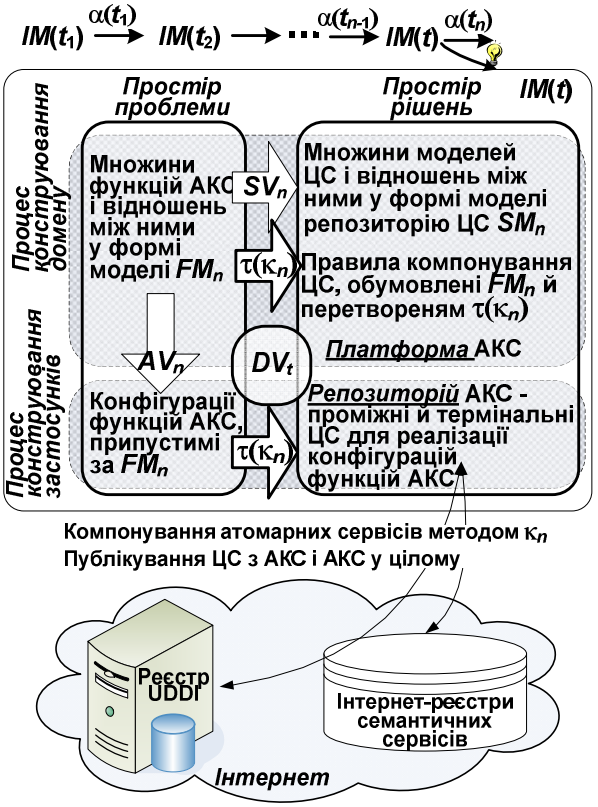


Рис. 2. Внутрішня структура АКС

– *композиційна* (composition), що відображає різноманітність способів компонування компонентних сервісів для реалізації різних функцій АКС ($k=2$);

– *компонентна* (partner), яка описує змінність взаємодій між складниками компонентного сервісу, що сам є композицією (композитних) сервісів з певним рівнем рекурсії ($k=3$);

б) п'ять рівнів (l) реалізації згідно зі структурою АКС як СоПС [1]:

– потреб споживачів у ПрО, відображених у моделі функцій FM_ν ($l=1$);

– ділових процесів, суб'єктами яких є ці споживачі ($l=2$);

– семантичних Веб-сервісів з підтримки ділових процесів, відображених у підмоделі ЦС CSt ($l=3$);

– програмних компонентів, що реалізують функціональність сервісів ($l=4$);

– операційних систем, під керуванням яких функціонують сервіси ($l=5$).

З урахуванням особливостей а) і б), формалізуємо для АКС поняття динаміч-

них проявів її варіабельності (уточнення "динамічний" далі опущено для стислості).

Визначення 8 [6, 13]. *Точка варіабельності* АКС на рівні $l=1,\ldots,5$ типу $k=1,2,3$ – формальне подання робочого продукту процесу конструювання домену на рівні *l*, який можна реалізувати кількома способами в сенсі, відповідному типу k.

Варіант для точки варіабельності на рівні *l* типу *k* – формальне подання припустимого способу реалізації робочого продукту, який вона подає.

Залежність – відношення на декартовому добутку множин точок варіабельності й варіантів усіх рівнів і типів, яке обмежує вибір варіантів для певних точок варіабельності в залежності від їх вибору для інших точок варіабельності.

Обмеження – залежність, яка має місце тільки для точок варіабельності. Типові обмеження – відношення обумовленості/виключення.

Зауваження 7. Точка варіабельності функцій АКС – нетермінальна функція $f\in F_v \mid \exists\, \varnothing\neq VF\subset F_v \;\; C_F{}^{mn}(f, f^*)$, а варіанти f – варіантно підпорядковані їй функції $f^*\in VF$. Цю функцію f реалізує f-релевантний проміжний композитний сервіс $w=\tau(\kappa_v,f)$ (6), варіантами якого є сервіси

$$w^*=\tau(\kappa_v, f^*)\mid C_S{}^{mn}(w,\ w^*).$$

Зауваження 8. Варіанти довільного типу й рівня в АКС можна розподілити за чотирьма основними класами:

1) опціональний $(m=0, n=1)$, що може і реалізуватися в ЦС, і бути відсутнім;

2) альтернативний $(m=n=1)$, який має реалізуватися в ЦС тільки один з множини припустимих;

3) варіантний $(m=1,\ n\geq 1)$ – має реалізуватися хоча б один з припустимих;

4) опціонально-варіантний $(m=0,\ n\geq 1)$ – мають реалізуватися від жодного до всіх припустимих.

Уточнення моделей варіабельності в структурі та в артефактах канонічної лінії продуктів [17, 18] з урахуванням зафіксованих особливостей а) і б) варіабельності Веб-сервісів дозволяє сформулювати

Визначення 9 [13]. Модель *варіабельності в структурі* АКС у *v*-му періоді між його адаптаційними зміненнями – структурований кортеж

$$SV_v=\langle\langle G_{1kv};\langle\langle G_{lk}, TR_{lkv}\rangle, l=2,\ldots,5\rangle;\quad (14)$$

$$CN_{kv};\ DP_{kv}\rangle, k=1,2,3;\ CN_v;\ DP_v\rangle,$$

де $G_{lkv}=(F_{lkv},\ RF_{lkv})$ – граф, множину F_{lkt} вершин якого складають унікальні ідентифікатори точок варіабельності й їх варіантів і безваріантних робочих продуктів типу *k* на рівні *l*, а множину RF_{lkv} дуг – відношення взаємозалежності вершин;

TR_{lkt}, $l=2,\ldots,5$ – двосторонні зв'язки трасовності між вершинами графів $G_{(l-1)kt}$ і G_{lkt}, що індукують відношення їх взаємозалежності як ізоморфне поширення відношень IM_F, EQ_F, EX_F, $C_F{}^{mn}$ між функціями АКС на нижчі рівні її структури;

CN_{kv}, $DP_{kv}:\otimes_{l=1,\ldots,5} F_{lkv}\ \rightarrow\{0;1\}$ – предикати, що подають обмеження й залежності для варіабельності АКС типу *k*;

CN_v, $DP_v:\otimes_{l=1,\ldots,5,\ k=1,2,3}\ F_{lkt}\rightarrow\{0;1\}$ – предикати, що подають обмеження й залежності між точками варіабельності й варіантами АКС різних рівнів і типів.

На підтримку зв'язків трасовності TR_{lkv} в АКС треба застосовувати стандартизовані кращі практики конструювання домену [17, 18].

Модель SV_v (14) обумовлює уніфіковане формальне подання всіх робочих продуктів, створених, створюваних чи передбачених в АКС після її адаптації в момент *t*, відповідно до

Визначення 10 [13]. Модель *варіабельності робочих продуктів (артефактів*) АКС – множина їх подань як структурованих кортежів $av(id_{mkt})$, що є "наскрізними фрагментами" моделі SV_v (14):

$$AV_v = \{av(id_{mkv}),\ m = 1,\ldots,5;\ k = 1,2,3\}; \quad (15)$$

$$av(id_{mkv}) = \langle\langle g_{1kv};\ \langle\langle g_{lkv},\ tr_{lkv}\rangle;\ l = 2,\ldots,m\rangle;$$

$$\langle\langle p_{lkv},\ tr_{lkv}\rangle;\ l = m+1,\ldots,5\rangle;$$

$$cn_{kv};\ dp_{kv}\rangle,\ k = 1,2,3;\ cn_t;\ dp_t\rangle,$$

де id_{mkv} – унікальний ідентифікатор робочого продукту;

g_{ukv} і p_{ukv} – підграфи графа G_{lkt}, а решта елементів – відповідні звуження елементів SV_t (13).

Запровадження моделей SV_v (14) і AV_v (15) надає додаткові можливості адаптації ЦС з АКС і АКС загалом як під час проектування, так і під час виконання:

– точки варіабельності робочих продуктів АКС, насамперед функцій і ЦС, стають "центрами " її локалізації та "міше-нями" змін, передбачених на момент t в просторі проблеми АКС у разі певних подій, сутність яких описують варіанти;

– допоміжна модель робочого продукту АКС $av(id_{mkv})$ визначає для $f \in F_v$ множину f-релевантних ЦС, які на момент t можна згенерувати в АКС, і виокремлює серед них ЦС з точками варіабельності, придатні до передбачених змін під час їх запуску й виконання;

– модель AV_v (15) зіставляє кожній точці варіабельності в певному ЦС множини передбачених варіантів;

– постачальник АКС, брокери й різнорідні споживачі його функцій можуть публікувати моделі (14), (15) (в цілому і по фрагментах за рівнями й типами варіабельності) та багаторазово незалежно застосовувати й налаштовувати їх.

Спираючись на формалізацію структури АКС у моделі SV_v (14), запровадимо останній складник DV_v – діагностичну модель рівня варіабельності.

Визначення 11. *Діагностична модель рівня задовільності* варіабельності АКС у v-му періоді між його адаптаційними зміненнями – це пара

$$DV_v = (EV_v, NT); \quad (16)$$

$$EV_v = \langle il_k; \langle vl_{lk}, rl_{lk}, pl_{lk}, ml_{lk}\rangle; ia_k; \langle va_{lk}, \quad (17)$$

$$ra_{lk}, pa_{lk}, pu_{lk}\rangle; \langle NA_{lik},\ CA_{lik},\ l,i = 1,\ldots,5\rangle;$$
$$k = 1,2,3\rangle,$$

де EV_v – рівень задовільності поточної варіабельності АКС для зацікавлених сторін (у (17) індекс v вилучено для спрощення нотації);

il_k, ia_k – інтегральні рівні автономної варіабельності АКС та її відповідності потребам її постачальника, брокерів і споживачів з різними ролями;

vl_{lk} і va_{lk} – проміжні рівні автономної варіабельності типу k та її відповідності, формовані робочими продуктами l-го рівня АКС;

rl_{lk}, pl_{lk} і ml_{lk} – вкладені рівні варіабельності типу k, передбаченої й реалізованої для робочих продуктів l-го рівня АКС, та ступеню відповідності між ними;

ra_{lk}, pa_{lk}, pu_{lk} – вкладені ступені адекватності передбаченої й реалізованої варіабельності типу k в робочих продуктах l-го рівня АКС потребам її постачальника, брокерів і споживачів з різними ролями, а також ефективності повторного використання робочих продуктів;

NA_{lik} – множина робочих продуктів l-го рівня АКС, що обумовлюють діагностовану неадекватність типу $i \in NT$ в сенсі варіабельності типу k і тому є безпосередніми об'єктами адаптаційних змін;

CA_{lik} – множина операцій адаптаційної зміни, які мають застосовуватися до відповідних елементів NA_{lik} для опрацювання діагностованої неадекватності типу i й розглянуті далі;

NT – перелік типів неадекватності АКС, що зумовлюють незадовільність рівня варіабельності та опрацьовуються за допомогою адаптаційних змін.

На підставі аналізу технік керування варіабельністю Веб-сервісів [6] до складу NT внесено такі типи неадекватності (i):

1) "надлишковість" – наявність робочих продуктів (від функцій до складни-

ків проміжного програмного забезпечення), незастосовних/неокупних під час виконання АКС ($i=1$);

2) "неповнота" – відсутність в поточних просторах проблеми і/або рішень АКС робочих продуктів (насамперед, функцій $f \in F_v$ і/або ЦС їх реалізації $\tau(\kappa_v, f)$ (5), (6) та сервісів платформи $wd(\kappa_v)$, запитаних значущою більшістю її зацікавлених сторін ($i=2$);

3) "клони" – наявність робочих продуктів, повністю взаємозамінних під час конструювання або виконання АКС ($i=3$);

4) "хибна структура" – наявність робочих продуктів, для яких мають місце незадовільні відношення з точками варіабельності певних типів і/або їх варіантами;

5) "хибні взаємозв'язки" – наявність робочих продуктів, для яких мають місце незадовільні відношення обумовленості, зіставлення або, відповідно, виключення.

Для застосування DV_v листки й вузли моделі EV_v (17) слід доповнити експертно формованими вербально-числовими шкалами – наборами пар "опис стану АКС в термінах спостережених значень складників EV; рівень критичності". Ці шкали доцільно формувати шляхом узагальнення експертних суджень відповідальних представників основних рольових груп організації-постачальника АКС за допомогою класичних технік групового експертного оцінювання. Далі їх слід постійно уточнювати за результатами аналізу задоволеності брокерів і споживачів функцій АКС.

Хоча безпосереднім об'єктом адаптаційної зміни може бути робочий продукт довільного рівня l, подальший розгляд цих об'єктів обмежено функціями АКС ($l=1$) і ЦС їх реалізації ($l=3$), найбільш значущими для зацікавлених сторін АКС, і композиційною варіабельністю ($k=2$), найбільш відповідною меті її конструювання.

Застосування рамкової моделі АКС для його адаптування. Формалізуємо поняття ЦС АКС, як самостійного об'єкта адаптаційних змін, за допомогою

Визначення 12. В АКС з інтенсіональним поданням $IM(t)$ (13) *конфігурація* функцій – це зв'язний підграф $g=(SF, RF)$ моделі функцій FM_v (1) з коренем *r*, де вершини не пов'язані виключенням EX_F і зберігають свої відношення з FM_v.

Конфігурація сервісів – підграф $q=(SS, RS)$ моделі ЦС SM_v (7) з коренем w_0, де вершини не пов'язані виключенням EX_S і зберігають свої відношення з SC_t.

Визначення 13. Нехай $C(FM_v)$ і $C(SC_v)$ – множини конфігурацій функцій і, відповідно, сервісів в АКС з інтенсіональним поданням (13) включно з порожньою конфігурацією ∅. Модель ЦС, який реалізує конфігурацію функцій $g=(SF, RF) \in \in C(FM_t)$, – це четвірка

$$tw(g)=\langle ip;\ g;\ \tau(\kappa_v, g),\ pr(ip)\rangle, \quad (18)$$

де ip – унікальний ідентифікатор ЦС;

$\tau(\kappa_v, g)=(\{\tau(\kappa_v, f), f \in SF\}, RS)$ – образ конфігурації функцій g згідно з поточним перетворенням в АКС $\tau(\kappa_v,)$ (5), (6), RS – відношення ЦС згідно з визначенням 5;

$pr(ip)$ – протокол застосування ЦС tw (споживачами функцій АКС і для побудови інших ЦС) під час виконання АКС.

Зауваження 9. Оскільки $\tau(\kappa_v, \cdot)$ – ізоморфізм FM_v і SC_v як алгебраїчних моделей, $\tau(\kappa_v, g)) \in C(SM_v)$, $v=1,\ldots,n$.

Отже, згідно зі сформульованим визначенням, ЦС в АКС є системою атомарних семантичних сервісів зі складу платформи АКС AP_v (11) або, з іншого погляду, проміжних композитних ЦС $icw \in \in \cup_{k=2,\ldots,n} IS_{kn}$ (10). Ці сервіси on-line взаємодіють під час виконання згідно з методом κ_v їх компонування для надання функцій g.

Прийняте визначення уможливлює екстенсіональне подання АКС. Разом з інтенсіональним поданням (13) воно утворює його *рамкову модель*.

Визначення 14. Екстенсіональне подання АКС у певний момент t між моментами його останнього здійсненого й наступного адаптаційного змінення (відповідно, t_{n-1} і t_n) – це множина поточно виконуваних ЦС і необов'язкових технічних сервісів динамічної композиції (sc), моніторингу задовільності варіабельності АКС (vm), редагування моделі функцій (fe), постійно доступних у разі їх наявності:

$$EM(t) = \langle\langle ip\,; g\,; \tau(\kappa_n, g),\, pr(ip)\rangle,$$

$$g \in C(FM_n)\rangle; [fe]\,; [sc]\,; [vm]\rangle. \quad (19)$$

Сервіси fe, vm і sc необов'язково надаються програмно-апаратною платформою виконання АКС і тому позначені у виразі (19) як необов'язкові.

Функціями технічного сервісу vm є:

а) підтримка експрес-оцінювання рівня задовільності варіабельності АСК за підмоделлю діагностичною моделлю EV_t (17) і регламентом, заданим під час її проектування або останньої адаптації;

б) діагностування незадовільності цього ступеня та виявлення її джерел NA_{lik};

в) вироблення рекомендацій щодо адаптаційних дій з опрацювання виявленої незадовільності й надання необхідної інформації адаптеру va.

Сервіси fe і sc ер підтримують операції адаптаційного змінення з необхідним залученням архітектора АКС.

Слід зазначити, що змінний ЦС $tw(g)$ (18) з точками варіабельності являє собою "вкладену" АКС, де підмоделями функцій і, відповідно, сервісів стають конфігурації g і $\tau(\kappa_v, g)$.

Отримана рамкова модель АКС $\langle IM(t); EM(t)\rangle$ дозволяє формалізацію операцій адаптаційної зміни: як окремого ЦС без впливу на інші, так і АКС загалом. Наведені далі визначення є результатом уточнення для семантичних Веб-сервісів формального апарату [14], розробленого автором для канонічної лінії продуктів.

Визначення 15. Множина SA операцій адаптаційної зміни ЦС tw (18) в АКС $\langle IM(t); EM(t)\rangle$ (13),(19) має вигляд:

$$SA = PV \cup NV;\ PV = \{\delta^{p}, \alpha^{p}, \kappa^{p}\}; \quad (20)$$

$$NV = \{\delta^{ut}; \delta^{dn}; \delta^{mn}; \alpha^{n}; \kappa^{n}; \chi^{n}\},$$

де PV – операції передбаченої зміни, ініційовані з боку як функцій, так і ЦС та сервісів платформи, а саме узгоджене *вилучення* (δ^{p}) й *долучення* (α^{p}) в точці варіабельності функцій або ЦС певного її варіанта чи його *заміна* (κ^{p}) іншим варіантом;

NV – операції непередбаченої зміни:

– узгоджене *долучення* (α^{n}) й *заміна* (κ^{n}) окремого варіанта з їх множини в точці варіабельності функцій або ЦС і сервісів платформи у tw новою функцією або сервісом її реалізації як новим варіантом (із збереженням складу обов'язкових функцій і сервісів), ініціюване як з боку фунцій, так і з боку сервісів;

– реалізація операцій α^{n} і κ^{n} для довільної вершини графів g, $\tau(\kappa_v, g)$ у (18);

– узгоджене *коректне* (у сенсі зв'язності графу $\tau(\kappa_v, g)$ з (18)) *вилучення* (δ^{ut}) обов'язкового сервісу $w \in \tau(\kappa_v, g)$ і функції $f = \tau^{-1}(\kappa_v, g)$, реалізовуваної w, "згори" (тобто з необхідними змінами підпорядковуючого сервісу), за якого склад обов'язкових функцій і сервісів змінюється;

– *вилучення* "знизу" (δ^{dn}) – навпаки, з потрібними змінами підпорядкованого сервісу для цілісності $\tau(\kappa_v, g)$;

– *вилучення "зсередини"* (δ^{mn}) із взаємним узгодженням ЦС, "суміжних" у $\tau(\kappa_v, g)$, і зміненнями конфігурації функцій;

– узгоджене *перепідпорядкування* (χ^{n}) функцій і сервісів, ініційоване як з боку функцій, так і з боку сервісів.

Зауваження 10. Операції PV можливі тільки для ЦС з точками варіабельності. Вони забезпечують пристосування ЦС

до змін, передбачених в поточних продуктах процесу конструювання домену, а змінений ЦС залишається в межах АКС. Натомість, операції NV реалізують адаптацію ЦС до непередбачених змін потреб споживачів і умов виконання поширенням функції змінених ЦС поза межі АКС.

Користуючись рамковою моделлю АКС (13), (19), сформулюємо

Визначення 16. Множина FA операцій адаптаційного змінення АКС містить:

а) множину SA (20) операцій адаптаційної зміни ЦС з АКС , застосованих до графів підмоделей FM_v (1) і SM_v (7);

б) операції *вилучення* з платформи AP_v (11) і/або з репозиторію S_v (10) *сервісу-клону* чи сервісу, що став недоступним/ не-прийнятним за показниками якості (QoS);

в) операції *долучення* до AP_v семантичного сервісу з описом $wd(\kappa_v)$ у форматі методу κ_v, – безпосереднього та опосередкованого, у композиціях для термінальних і проміжних композитних сервісів;

г) операції *еволюції* сервісів з платформи AP_v і ЦС з репозиторію АКС S_v разом з долученнням описів нових набутих функцій до моделі функцій FM_v (1).

Висновки

Обґрунтовано актуальність нового підходу до автоматизованого відшукування релевантних сервісів у семантичному Веб-середовищі під час виконання та їх компонування на підтримку складних розподілених ділових процесів сучасних організацій. Підхід уможливлює адаптивність композитного сервісу (його здатність до зміни поведінки для задоволення нових вимог і пристосування до нових передбачених і непередбачених умов виконання) та його застосовність у незалежних композиціях третіх сторін. Він є також інваріантним до методу компонування.

Базовими конструктами підходу є рамкові моделі: самого композитного сервісу, процесу його поетапного конструювання, окремого етапу цього процесу та операцій адаптування композитного сервісу на підставі його наданої моделі. Останню формалізовано за допомогою розгляду композитного сервісу як динамічної лінії проміжних і термінальних композитних семантичних сервісів реалізації припустимих підмножин його поточних функцій.

Вироблено установчі рішення з конструювання такого композитного сервісу (з первинним проектуванням і подальшим поетапним адаптуванням до змін потреб користувачів, умов виконання і/або відмов), модельованого як процес керування його динамічною варіабельністю в концептуальному середовищі, структурованому на підставі моделі властивостей. Запропоновано діагностичну модель варіабельності для виявлення потреб і стратегій адаптування. Визначено операції реалізації стратегій – зміни окремого складника композитного сервісу та його самого в цілому відповідно до передбачених (під час проектування) й непередбачених (під час виконання) змін потреб користувачів, параметрів середовища виконання і/або відмов сервісів-складників.

Запровадження запропонованого підходу в сучасних організаціях сприятиме підвищенню ефективності використання й реінжинірингу ділових процесів з різнорідними й нестабільними контекстами. Необхідний для цього розвиток рамкових моделей з метою підтримки оптимізації та адаптування композиції до проявів її незадовільної якості та їх уточнення для окремих ефективних методів компонування семантичних Веб-сервісів – предмет подальших досліджень автора.

1. Андон П.І., Бабенко Л.П. Проблеми і можливості програмування в середовищі SEMANTIC WEB. *Проблеми програмування*. 2012. № 2-3. С. 363–373.
2. Kazhamiakin R., Benbernou S., Baresi L. et al. Adaptation of Service-Based Systems. In: M. Papazoglou et al. (Eds.): Service Research Challenges and Solutions, LNCS 6500. Springer-Verlag Berlin Heidelberg, 2010. P. 117–156.

3. Alférez G.H., Pelechano V. Facing uncertainty in Web service compositions. *Int. J. of Services Computing.* 2014. Vol. 2. N. 2. P. 1–16.
4. Alférez G., Pelechano V., Mazo R. Dynamic Adaptation of Service Compositions with Variability Models. *J. of Syst. and Software*. Vol. 91. 2014. P. 24–47.
5. Bucchiarone A. et al. Design for Adaptation of Service-Based Applications:Main Issues and Requirements. Dan A., Gittler F., and Toumani F. (Eds.): ICSOC/ServiceWave 2009. LNCS 6275. Springer-Verlag Berlin Heidelberg, 2010. P. 467–476.
6. Nguyen T., Colman A., Han J. Comprehensive Variability Modeling and Management for Customizable Process-Based Service Compositions. In: Bouguettaya A. Web Services Foundations. Springer: Science & Business Media, 2013. P. 507–534.
7. Bucchiarone A., De Sanctis M., Pistore M. Domain Objects for Dynamic and Incremental Service Composition. In: Villari M.et al. (Eds.) Proc. ESOCC 2014, Manchester, UK – LNCS 8745. 2014. P. 62–80.
8. Kazhamiakin R., Marconi A., Pistore M. Data-Flow Require-ments for Dynamic Service Composition. Proc. IEEE 20th International Conference on Web Sevices. 2013. P. 243–250.
9. Kapuruge M., Han J., Colman A. Service orchestration as organization: Building multi-tenant service applications in the Cloud. 2014. 363 p.
10. Zeginis C., Plexousakis D. Веб Service Adaptation: State of the art and Research Challenges. Technical Report 410. ICS-FORTH, 2010. 66 p.
11. Rodriguez-Mier P., Pedrinaci C., Mucientes M., Lama M. An Integrated Semantic Web Service Discovery and Composition Framework. 2015. [Electronic resourse]. Mode of access https://arxiv.org/pdf/1502.02840.pdf.
12. Bansal S., Bansal A., Gupta G., Brian Blake M. Generalized semantic Web service composition. *Service Oriented Computing and Applications*. 2016. Vol. 10. Is. 2. P. 111–133.
13. Слабоспицька О.О. Технологічна модель процесу побудови та використання адаптивної композиції Веб-сервісів. *Проблеми програмування.* 2015. № 2. С. 52–62.
14. Slabospickaya O. Feature Model of Software Product Line Enhancing to Enable Product Adaptability In: Bulletin of University of Kiev. Series: Physics & Mathematics, special issue, 2014. P. 151–158. (in Ukrainian).
15. Moghaddam M., Davis J. Service Selection in Web Service Composition: A Comparative Review of Existing Approaches. In: Bouguettaya A. Веб Services Foundations. Springer: Science & Business Media, 2013. P. 321–346.
16. Capilla R. et al. An overview of Dynamic Software Product Line architectures and techniques: Observations from research and industry. *J. Syst. Software*. 2014. P. 1–21.
17. Van der Linden F., Schmid K., Rommes E. Software product lines in action: the best industrial practice in product line engineering. Heidelberg: Springer, 2007. 340 p.
18. ISO/IEC 26550:2015 Software and systems engineering. Reference model for product line engineering and management. Ed. 2. 2015. 35 p.
19. Lee K. Concepts and Guidelines of Feature Modeling for Product Line Software Engineering. 7th Int. Conf. on Software Reuse: proceedings. Springer-Verlag, 2002. P. 62–77.
20. ДР 0112U002764 Розробка теоретичних основ та прикладних питань побудови сервіс-орієнтованих прикладних програмних систем у семантичному Веб-середовищі. Звіт про НДР (заключний). К.:, ІПС НАНУ, 2016. 280 с.

References

1. Andon P.I. The Problems and Opportunities of Programming in SEMANTIC WEB / P.I.Andon, L.P.Babenko // Problems of Programming. – 2012. – N 2-3. – P. 363–373. (In Ukrainian).
2. Kazhamiakin R. Adaptation of Service-Based Systems / R.Kazhamiakin, S.Benbernou, L.Baresi et al. // In: M. Papazoglou et al. (Eds.): Service Research Challenges and Solutions, LNCS 6500. Springer-Verlag Berlin Heidelberg, 2010. – P. 117–156
3. Alférez G.H. Facing uncertainty in Веб service compositions. / G.H.Alférez, V.Pelechano // Int. J. of Services Computing – 2014. – V.2. – N. 2. – P. 1–16.
4. Alférez G. Dynamic Adaptation of Service Compositions with Variability Models / G.Alférez, V.Pelechano, R.Mazo // J. of Syst. and Software. – V. 91. – 2014. – P.24–47.
5. Bucchiarone A. et al. Design for Adaptation of Service-Based Applications:Main Issues and Requirements // A. Dan, F. Gittler, and F.

Toumani (Eds.): ICSOC/ServiceWave 2009. – LNCS 6275. – Springer-Verlag Berlin Heidelberg, 2010. – P. 467–476.
6. Nguyen T. Comprehensive Variability Modeling and Management for Customizable Process-Based Service Compositions / T.Nguyen, A.Colman, J.Han // In: Bouguettaya A. Веб Services Foundations – Springer: Science & Business Media, 2013. – P. 507–534
7. Bucchiarone A. et al. Domain Objects for Dynamic and Incremental Service Composition / A.Bucchiarone, M. De Sanctis, M. Pistore // In: Villari M.et al. (Eds.) Proc. ESOCC 2014, Manchester, UK – LNCS 8745, 2014. – P. 62–80.
8. Kazhamiakin R. et al. Data-Flow Requirements for Dynamic Service Composition / R.Kazhamiakin, A.Marconi, M.Pistore // Proc. IEEE 20th International Conference on Веб Sevices – 2013. – P. 243-250.
9. Kapuruge M. et al. Service orchestration as organization: Building multi-tenant service applications in the Cloud. / M.Kapuruge, J.Han, A.Colman – 2014 – 363 p.
10. Zeginis C. Веб Service Adaptation: State of the art and Research Challenges. Technical Report 410 / C.Zeginis, D.Plexousakis – ICS-FORTH, 2010. – 66 p.
11. Rodríguez-Mier P. An Integrated Semantic Web Service Discovery and Composition Framework / P.Rodriguez-Mier, C.Pedrinaci, M. Mucientes, M.Lama, 2015. [Electronic resourse]. – Mode of access https://arxiv.org/pdf/1502.02840.pdf.
12. Bansal S. Generalized semantic Web service composition / S.Bansal, A.Bansal, G.Gupta, M.Brian Blake// Service Oriented Computing and Applications. – 2016. – V.10. – Is. 2. – P. 111–133
13. Slabospitskaya O.A. Technological model for the process of adaptive Web service composition engineering and exploiting / O.A. Slabospitskaya // Problems in Programming – 2015. – № 2. – P. 52– 62. (in Ukrainian).
14. Slabospickaya O. Feature Model of Software Product Line Enhancing to Enable Product Adaptability In: Bulletin of University of Kiev. Series: Physics & Mathematics, special issue, 2014. – P. 151–158. (in Ukrainian).
15. Moghaddam M. Service Selection in Web Service Composition: A Comparative Review of Existing Approaches / M. Moghaddam, J. Davis // In: Bouguettaya A. Веб Services Foundations – Springer: Science & Business Media, 2013. – P. 321–346
16. Capilla R. et al. An overview of Dynamic Software Product Line architectures and techniques: Observations from research and industry / R.Capilla // J. Syst. Software. – 2014. – P. 1–21.
17. Van der Linden F. Software product lines in action: the best industrial practice in product line engineering. / F.Van der Linden, K.Schmid, E.Rommes – Heidelberg: Springer, 2007. – 340 p.
18. ISO/IEC 26550:2015 Software and systems engineering. Reference model for product line engineering and management. Ed. 2. – 2015. – 35 p.
19. Lee K. Concepts and Guidelines of Feature Modeling for Product Line Software Engineering / K. Lee // 7th Int. Conf. on Software Reuse: proceedings. –Springer-Verlag, 2002. – P. 62–77.
20. DR 0112U002764 Theoretical Fundamentals and Applied Issues Investigation of Applied Service-oriented Information Systems Engineering in Semantic Web. Research report (final).). – К.: SSI NASU, 2016. – 280 p. (in Ukrainian).

Одержано 15.04.2017

Про автора:

Слабоспицька Ольга Олександрівна,
кандидат фізико-математичних наук,
старший науковий співробітник.
Кількість наукових публікацій в українських виданнях – більше 50.
Кількість наукових публікацій в зарубіжних виданнях – 5.
http://orcid.org/0000-0001-6556-0947

Місце роботи автора:

Інститут програмних систем
НАН України,
03187, Київ-187,
проспект Академіка Глушкова, 40.
Тел.: +38(044) 526 4286.
E-mail: olsips2017@gmail.com

УДК 004.94

О. Захарова

ВИЗНАЧЕННЯ ТА ВИРІШЕННЯ ЗАДАЧІ ВИЯВЛЕННЯ ВЕБ-СЕРВІСІВ ЗА ДОПОМОГОЮ АПАРАТУ ДЕСКРИПТИВНИХ ЛОГІК

На сьогодні Веб-сервіси дозволяють вирішувати конкретні бізнес-задачі, що реалізують бізнес процеси у різних галузях життєдіяльності людини. Але для того, щоб отримати виконуваний Веб-сервіс, треба вміти ефективно вирішувати цілу низьку задач самих Веб-сервісів на всіх етапах їх життєвого циклу. Апарат дескриптивних логік, завдяки своїм механізмам міркувань та можливостям логічного виводу та надання описам семантичного змісту, є ефективним та потужним інструментом для вирішення задач Веб-сервісів. Мета данної роботи полягає у визначенні ланцюжка задач Веб-сервісів на функціональному рівні та підходів до вирішення цих задач із застосуванням апарата дескриптивних логік.
Ключові слова: семантичний Веб-сервіс, дескриптивна логіка, задача виявлення, пошук веб-сервісу, композиція веб-сервісів, семантичний опис, онтологія домену, онтологія сервісу.

Вступ

Згідно визначення W3C [1–5], під *сервісом* розуміють таку програмну систему, що ідентифікується URI, публічні інтерфейси, прив'язки якої визначені та описані мовою XML. Опис цієї програмної системи може бути знайдено іншими програмними системами, які можуть взаємодіяти з нею відповідно до цього опису з використанням повідомлень, що базуються на XML та передаються за допомогою Інтернет-Протоколів. Веб-сервіси базуються на чотирьох ключових технологіях [5–7]: eXtensible Markup Language (XML), Simple Object Access Protocol (SOAP), Web Services Description Language (WSDL) та Universal Description, Discovery and Integration (UDDI). Ці технології використовуються для забезпечення функціонування Веб-сервісів.

Хоча Веб-сервіси базуються на широко прийнятих стандартах, відсутність формального опису значень їх функціональності та обміну даних є значною перешкодою у реалізації інтеграціонних перспектив. Так як кількість Веб-сервісів зростає, важливо мати автоматизовані засоби для виявлення та композиції Веб-сервісів. Ступінь опису, що є доступною в існуючому стандарті WSDL, залишає місце для неоднозначних інтерпретацій функціональності та даних Веб-сервісів. Неоднозначність інтерпретації ускладнює автоматизацію таких задач, як виявлення, композиція, виклик сервісу тощо. Семантичні описи таких Веб-сервісів відкривають шлях до автоматизації їх композиції.

На практиці існує два рівні представлення сервісів, а саме: функціональна та процесна модель сервісів [8]. На функціональному рівні сервіси розглядаються як окремі існуючі в мережі прикладні компоненти, які можуть бути викликані шляхом відправлення повідомлення. При цьому сервіс виконує свою задачу та (в деяких випадках) виробляє відповідь тому, хто його викликав. Таким чином, відсутня безперервна взаємодія між запитувачем сервісу та самим сервісом. Опис таких Веб-сервісів фокусується на їх функціональності в термінах імені сервіса, імен операцій, імен повідомлень (що відомі також як вхідні та вихідні повідомлення/параметри), імені інтерфейсу.

Сервіси процесного рівня містять декілька операцій, які слідують загальній поведінці сервіса. Такі сервіси потребують розширеної взаємодії між запитувачем сервісу та множиною операцій, забезпечуючи конкретну функціональність. Таким чином, це, як правило, композитні сервіси, тобто взаємодія з ними складається не лише з окремого крока запит-відповідь, для досягнення потрібного результату вони повинні слідувати складному протоколу. Ці кроки можуть складатися з довільних (умовних та ітеративних) комбінацій з

умовними виходами. На цьому рівні необхідний деталізований опис внутрішньої поведінки сервісів, наприклад, за допомогою (STS). Очевидно, що на процесному рівні Веб-сервіси також мають функціональний опис своїх операцій та сервісів. Але сервіси на функціональному рівні не мають поведінкових взаємодій. Це є головною відмінністю цих двох моделей.

Предметом розгляду данної статті є вирішення задачі виявлення для онтологічно анотованих Веб-сервісів, що представлені функціональною моделлю, із застосуванням апарату дескриптивної логіки (ДЛ).

Визначення семантичного Веб-сервісу

ДЛ – формальна мова, яка підтримує концепцію відкритого світу та власні механізми міркувань. Все це робить її бажаною та ефективною як для представлення функціональної частини опису Веб-сервісу, так і для представлення семантичних елементів (анотацій) у процедурному описі Веб-сервісу. А така властивість ДЛ, як підтримка можливостей логічного виведення, забезпечує ефективність вирішення багатьох задач Веб-сервісів, як елементів загальної складної задачі композиції. Так, наприклад, при анотуванні сервісу може виникнути потреба у розширенні загальної онтології домена чи онтології домена конкретного сервісу. Резонери ДЛ дозволять перевірити коректність доданих аксіом. Це зводиться до стандартної задачі виконуваності, що вирішується для цих аксіом.

Онтологія домена

Відповідно до понять ДЛ [5], в межах загальноприйнятої онтології існує термінологічний компонент (T-BOX) [6] і стверджувальний (assertional) компонент (A-BOX). Вважаємо, що T-BOX є єдиним для всіх сервісів, які необхідно анотувати в домені. Він містить всі концепти, які необхідно представляти в домені застосування. Якщо анотації винести до окремого файлу (із посиланнями на файл процесу), то це дозволить глобальні твердження, які дійсні для всіх станів процесу, а саме твердження T-BOX, не повторювати в описі кожного стану процесу, а визначити однократно у відповідному розділі файлу анотацій. A-BOX містить визначення твердження двох різних типів: твердження концептів та твердження ролі. З кожним станом кожного сервісу пов'язується свій A-BOX. Він описує наслідки даної дії в термінах тверджень концепту і ролі. Тим не менш, є деякі твердження, які не залежать від будь-яких дій, але виконуються скрізь. Такі твердження завжди є правильними.

Розглянемо як приклад задачу бронювання квітків на літак. Призначення сервісу – визначити прийнятний до запиту користувача рейс. Входами сервісу є ім'я клієнта, дата відправлення, пункт відправлення та призначення. Виходом є квіток на літак. Онтологія домена буде описана засобами ДЛ. Анотації входів, виходів, передумов та ефектів сервісу є аксіомами визначення або аксіомами включення ДЛ. У WSDL-описі сервісу вони будуть представлені посиланнями на концепти онтології домену *POOntology*.

Спочатку продемонструємо використання апарату ДЛ для визначення онтології домену *POOntology*. Для цього необхідно визначити TBox та ABox ПО.

Відповідний узагальнений TBox може містити такі твердження: *Year, Month, Day, Date, Client, Status, Location, Country, FlightID*

$Date \sqsubseteq has.Year;$

$Date \sqsubseteq has.Month;$

$Date \sqsubseteq has.Day;$

$Location \sqsubseteq isLocatedIn.Country;\ Trip \sqsubseteq hasStartPoint.Location;$

$Trip \sqsubseteq hasEndPoint.Location;\ Trip \sqsubseteq hasDeparture.Date$

$Trip \sqsubseteq hasArrival.Date;\ Flight \sqsubseteq belonguesTo.AirLineRoute$

$Flight \sqsubseteq hasDepartur Date.Date;\ Flight \sqsubseteq hasDepartur Time.Time$

$Flight \sqsubseteq from.Location;\ Flight \sqsubseteq to.Location$

$Flight \sqsubseteq hasSeats.NUMBER;\ Flight \sqsubseteq hasStatus.Status;$

Status = {Available, NotAvailable, booked}

Роль *hasStatus,* що зв'язує два концепти *Flight* та *Status*, фіксує поточний стан запиту клієнта і може приймати наступні значення: *Available* – якщо поїздка доступна; *NotAvailable* – коли поїздка не доступна, *booked* – квиток заброньований.

Додамо передумову до нашого сервісу. Припустимо, що сервіс виконується лише при бронювання квитків до країн Європи. Ефектом є бронювання квитка на літак. Визначимо відповідні умови у TBox, щоб уможливити їх використання. Для цього додамо до TBox концепт *Europe*, а до ABox множину індивидів, які будуть визначати цей клас.

Europe

Ukraine:Europe, France:Europe,

Italy:Europe, Lituva:Europe,

Hungary:Europe, etc.

EuropeanTrip⊑ hasStartPoint Europe ⊓

hasEndPoint Europe

FlightBooked ≡ *hasStatus.{booked}*

EuropeanTrip ⊑Trip

Семантичне анотування сервісу на функціональному рівні

Для вирішення задач Веб-сервісів їх опис має спиратися на існуючі стандарти. Функціональна модель сервісу (його профілі), що зберігаються в UDDI, визначаються у WSDL, а семантизованих Веб-сервісів, опис яких розширений анотаціями, у розширені WSDL, як, наприклад WSDL-S або ASWSDL.

WSDL-S представлення сервісу, що описаний у прикладі, може мати наступний вигляд:

```
<?xml version="1.0" encoding="iso-
8859-1"?>
      <definitions name="
      flightTicket"
      targetNamespace=http://lsdis.cs
      .uga.edu/projects/meteor-
      s/wsdl-
      s/examples/flightTicket.wsdl
      xmlns="http://www.w3.org/2004/0
      8/wsdl"
      xmlns:tns=http://lsdis.cs.uga.e
      du/projects/meteor-s/wsdl-
      s/examples/flightTicket.wsdl
      xmlns:xs="http://www.w3.org/200
      1/XMLSchema"
      xmlns:xsd1="http://lsdis.cs.uga
      .edu/projects/wsdl-
      s/examples/flightTicket.wsdl"

      xmlns:wssem="http://lsdis.cs.ug
      a.edu/projects/wsdl-
      s/examples/flightTicket.wsdl"
      xmlns:Flight="http://lsdis.cs.u
      ga.edu/projects/wsdl-
      s/ontologies/POOntology">
      <--опис типів -->
      <types>
      <xs:import
      namespace=http://lsdis.cs.uga.e
      du/projects/wsdl-
      s/examples/flightTicket.wsdl
      schemaLocation="http://lsdis.cs
      .uga.edu/projects/wsdl-
      s/examples/WSSemantics.xsd"/>
      <xs:schema
      xmlns:xs="http://www.w3.org/200
      1/XMLSchema"
      targetNamespace=http://lsdis.cs
      .uga.edu/projects/wsdl-
      s/examples/flightTicket.wsdl
      xmlns="http://lsdis.cs.uga.edu/
      projects/wsdl-
      s/examples/flightTicket.wsdl">
      <!--Semantic annotation is
      added directly to leaf element
      -->
      <xs:complexType
      name="processflightTicketReques
      t">
                <xs:all>
```

```
            <xs:element
            name="locationToInf
            o" type="xs:string"
            wssem:modelReferenc
            e="POOntology#Locat
            ion />
            <xs:element
name="locationFromInfo"
type="xsd1:string"/>
            <xs:element
            name="dateitem"
            type="xs:date"
            wssem:modelReferenc
            e="POOntology#Date
            />
            <xs:element
            name="ClientInfo"
            type="xs:string"
            wssem:modelReferenc
            e="POOntology#Clien
            t />
          </xs:all>
  </xs:complexType>
<!--Semantic annotation is
added directly to leaf element
-->
        <xs:element
name="processflightTicketRespon
se" type="xs:string"

wssem:modelReference="POOntolog
y#Flight"/>
        </xs:schema>
</types>
<interface
name="PurchaseOrder">
<operation
name="processflightTicket"
pattern="wsdl:in-out">
             <input
messageLabel
="processflightTicketRequest"
element="tns:processflightTicke
tRequest"/>
             <output
messageLabel
="processflightTicketResponse"
             element="
processflightTicketResponse"/>
           <!--Precondition and
effect are added as extensible
elements on an operation-->
           <wssem:precondition
name="ExistingLocationsPrecond"

wssem:modelReference="POOntolog
y#LocationsExist"/>
           <wssem:effect
name="FlightBookedEffect"

wssem:modelReference="POOntolog
y#flightBooked"/>
           </operation>
   </interface>
</definitions>
```

У WSDL-S, як і у більшості формальних мов, зв'язок з онтологією домена *POOntology* здійснюється за допомогою використання простору імен. Використання конкретних концептів онтологій у семантичних описах вхідних та вихідних параметрів, перед- та пост-умов досягається використанням префіксу з ім'ям онтології перед назвою концепта. Наприклад, конструкція *<xs:element name="locationToInfo" type="xs:string" wssem:modelReference= "POOntology#Location />* вказує, що елемент *locationToInfo* є екземпляром концепта *Location* онтології *POOntology*. Зв'язок онтологій між собою, наприклад, онтології конкретного сервісу та онтології сервісу верхнього рівня, або (та) онтології домену прикладної області, досягається шляхом інтеграції онтологій. Побудова інтегрованої онтології може здійснюватися за допомогою застосування різноманітних операцій маніпуляції онтологіями, таких як відображення, узгодження, уточнення, пого-

дження онтології та інше, а також шляхом застосування до онтологій операцій алгебри онтологій. Ці питання були досліджені та викладені у [7].

Вирішення задачі автоматизованого виявлення семантичних Веб-сервісів

Загальні вимоги до алгоритму співставлення. Щоб знайти Веб-сервіси для виконання конкретних задач в бізнес-процесі, запитувач має, перш за все, визначити умови, які повинен задовільняти цей Веб-сервіс. Такий підхід називається співставлення на основі цілі (goal based matchmaking) [8, 9] та має справу з визначенням умов на оголошення Веб-сервісу та перевіркою, чи може Веб-сервіс задовільняти ці умови. Цілі, як правило, слідують з бізнес-цілей, та можуть виводитись з них автоматично або не автоматично.

Опис чи оголошення сервісу відповідає запиту, якщо запит є досить близьким до сервісу, що запитується. Тут необхідно специфікувати, що означає «досить близький». У найсуворішій інтерпретації, оголошення та запит є «досить близькими», якщо вони описують точно один і той самий сервіс. Але це визначення є занадто обмеженим. Потрібно більш гнучке визначення поняття «достатньої близькості». Тобто, необхідні механізми співставлення, які розпізнають ступінь подібності між оголошеннями сервісів та запитами. А в того, хто запитує сервіс, має бути можливість визначати ступінь гнучкості, яку вони надають системі. Чим менша гнучкість, тим менша вірогідність знаходження сервісів, що задовільняють вимогам.

Таким чином:

- механізм співставлення має підтримувати гнучке семантичне співставлення між оголошеннями сервісів та запитами на основі онтологій, що доступні сервісам та механізму співставлення;
- запитувач має володіти деяким контролем ступеня гнучкості співставлення, що дозволяється системі;
- механізм співставлення має заохочувати обидві сторони бути чесними у своїх описах у питаннях вартості;
- процес співставлення має бути ефективним: він не повинен навантажувати запитувача надмірними затримками, які перешкоджатимуть його ефективності.

Алгоритми співставлення взагалі є ключовим питанням в області досліджень Веб-сервісів, та є основою вирішення не лише задачі виявлення сервісів, але й їх автоматизованої композиції. У загальному випадку, співставлення (matchmaking) Веб-сервісів містить у собі співставлення IOPE описів (співставлення за входами, виходами, передумовами, та ефектами). Дійсно, входи, виходи, передумови та ефекти утворюють опис функціональних можливостей сервісу. Наочно, IO надають синтаксичну інформацію про Веб-сервіси, тоді як PE – відображають їх семантику. Метод, який використовується у співставленні входів-виходів відрізняється від того, що використовується для передумов та ефектів. Та семантики, які відображаються входами-виходами й передумовами та ефектами також різні. На сьогодні досягнуто значних результатів у співставленні Веб-сервісів за входами-виходами, але відсутній ефективний підхід щодо співставлення перед-умов та ефектів. У роботі [10] наводиться алгоритм PE спіставлення, на основі формалізма ДЛ SHOIN+(D).

Безперечна доцільність використання апарату ДЛ для вирішення багатьох задач Веб-сервісів обумовлена тим, що системи ДЛ забезпечують користувачів різними механізмами виводу, які виводять неявні знання з тих, що явно представлені, та більше того, на сьогодні існує вже чимало реалізованих механізмів міркувань (резонерів) ДЛ, що готові до використання. Але, варто пам’ятати, що окремою складною задачею залишається вибір такої ДЛ, що є компромисним рішенням між її виразністю та розв’язуваністю.

Далі пропонується алгоритм виявлення Веб-сервісів на основі ДЛ, який адоптує множину стратегій. Задача виявлення полягає у знаходженні всіх Веб-сервісів, що задовільняють запиту. Іншими словами, це можна назвати фільтрацією множини сервісів у репозиторії за запитом. Тобто оголошення кожного сервісу співставляють із запитом. При чому запит та-

кож є своєрідним оголошенням сервісу, а саме, того сервісу, який реалізує потрібну користувачеві функціональність. Таким чином, відношення між запитом та оголошенням сервісу ідентичні відношенням між оголошеннями двох сервісів.

Критерій та ступені відповідності

Спочатку розглянемо задачу виявлення для сервісів, що задаються лише входами-виходами. Основу вирішення такої задачі становить встановлення IO-відповідності (Input-Output).

Вважаємо, що оголошення сервісу відповідає запиту, якщо всі виходи запиту співпадають з виходами оголошення, а всі входи оголошення співпадають зі входами запиту. Точне співпадання знайти досить важко, тому визначають ступені відповідності і обирають сервіси з найбільшим ступенем відповідності. Базовий критерій виявлення полягає у тому, що сервіс має задовільняти потреби запитувача, а запитувач має забезпечувати сервісу всі входи, які необхідні для його коректного функціонування.

Критерій відповідності. Якщо $\boldsymbol{\mathcal{T}}$ – ациклічний Tbox ДЛ $\mathcal{L}$, $S = (In_s, Out_s)$ – сервіс та $Q = (In_q, Out_q)$ – запит, де:

- In_q – кінцева множина входів запиту,
- In_s – кінцева множина входів оголошення сервісу,
- Out_q – кінцева множина виходів запиту,
- Out_s – кінцева множина виходів оголошення сервісу, що виражені твердженнями $\boldsymbol{\mathcal{L}}$, то відповідність сервісу запиту гарантується включеннями $In_s \subseteq In_q$ та $Out_q \subseteq Out_s$.

Таким чином, сервіс відповідає запиту, якщо запит містить всі вхідні твердження сервісу та, можливо, ще додаткові, а множина вихідних тверджень запиту, навпаки, має бути вужча за множину вихідних тверджень сервісу, тобто сервіс може повертати більше ефектів ніж того потребує запит.

Запити співставляються за цим критерієм з усіма оголошеннями сервісів, які є у реєстрі. Як тільки знайдена відповідність запиту та оголошення, вона записується, щоб знайти відповідність більш високого ступеня. Ступінь успіху залежить від його виявленого співпадання. Ступінь відповідності між двома входами або двома виходами залежить від відношення між концептами, які пов'язані з цими входами або виходами, а саме, мінімальною відстаню між цими поняттями у дереві таксономії. Розрізняють такі ступені відповідності [11, 12]:

- *Exact* – точне співпадання:
  - $Out_q = Out_s$ – еквівалентність, або
  - Out_q *subclassOf* Out_s – результат точний у припущенні, що оголошуючи Out_s провайдер сервісу повинен забезпечити виходи, що погоджені з кожним безпосереднім підтипом Out_s.
- *Plug In* – більш слабкий зв'язок $(Out_s \; subsumes \; Out_q)$. Якщо Out_s *subsumes* Out_q, то Out_s – це множина, яка включає виходи Out_q, або іншими словами, Out_s може бути підключений замість Out_q.
- *Subsumes* $(Out_q \; subsumes \; Out_s)$ – якщо Out_q *subsumes* Out_s, то провайдер не повністю задовільняє запит. Це означає, що запитувач може використовувати провайдера для досягнення своєї мети, але йомовірно йому прийдеться модифікувати свій план або виконати інші запити, щоб завершити свою задачу.
- *Fail* – невдача, не знайдено будь-якого *subsumption* між Out_q та Out_s.

Ступінь відповідності відображається на дискретній шкалі, де найбільш переважним є точна відповідність (*Exact*), наступний рівень – *Plug In*, так як вихід, що повертається, може бути використаний замість того, що очікує запитувач. *Subsumes* – третій рівень, так як вимоги запитувача виконуються лише частково: оголошений сервіс може забезпечити лише деякі конкретні варіанти того, що бажає запитувач.

Самий низький рівень *Fail* – не прийнятний результат. Ці значення дозволяють ранжувати знайдені сервіси за ступенем відповідності запиту. Обираються сервіси з найбільшим ступенем відповідності виходів. Співставлення входів використовується лише як допоміжна (вторинна) оцінка, щоб розмежувати еквівалентно оцінені виходи.

Функціональна (IOPE) модель сервісу у загальному випадку буде мати вигляд:

$$S = \{s | s = (CI;\ I;\ O;\ CO)\}, \qquad (1)$$

де CI – передумови сервісу, I – список вхідних параметрів, O – список вихідних параметрів та CO – після-умови.

Запит сервісу [13], в цьому випадку, розширюється та визначається як

$$Q = (CI'; I'; O'; CO'), \qquad (2)$$

де CI' – передумови, I' – список вхідних параметрів, O' – список вихідних параметрів та CO' – після-умови. Всі ці елементи є параметрами сервісу, що запитується.

Валідні рішення щодо запиту, у даному випадку, мають задовільняти наступним умовам:

(i) вони мають виробляти хоча б один вихідний параметр запиту;

(ii) вони мають використовувати вхідні параметри тільки з наданого списку вхідних параметрів та задовільняти передумови запиту;

(iii) вони мають виробляти ефекти запиту.

Деякі рішення можуть бути занадто обмеженими, але вони розглядаються як дійсні, якщо задовільняють вимогам вхідних та вихідних параметрів, перед/після умов та ефектів.

Окрім IOPE атрибутів, співставлятися можуть і категорії сервісів, і додаткові параметри сервісів, як наприклад, не функціональні характеристики. В такому випадку, мова йде про багаторівневе співставлення. Значення ступеней відповідності на різних рівнях мають різну вагу. Результуюче значення є агрегацією багаторівневих зважених оцінок.

Запитувач може вирішити будь-які невідповідності шляхом вирішення додаткової задачі або шляхом запиту до реєстру, щоб знайти додаткових провайдерів.

Розглянемо задачу виявлення на прикладі бронювання авіаквітків. Тобто треба знайти сервіс бронювання квитків на літак, директорія сервісів містить сервіси S_1 та S_2 (таблиця далі).

Таблиця

Сервіс	Входи	Передумови	Виходи	Після-умови
S_1	*StartL*, *DestinationL*, *DateFrom*, *DateTo*, *CCNumber*	*Location Exist*	*status, Offer*	-
S_2	*StartL*, *Dt*, *DestinationL*, *CCNumber*	*CorrectCard*	*status, Offer, TrNumber*	-
Запит				-
Q	*Start*, *Dt*, *Destination*, *CCNumber*	*CorrectCard*		-

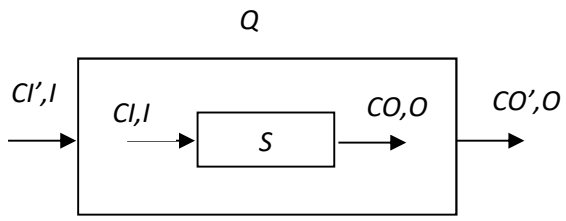
Сервіс S_1 має вхідні параметри *StartL*, *DestinationL*, *DateFrom*, *DateTo*, *CCNumber* та вихідні параметри *Status*, *Flight*. Для його виконання задана передумова *LocationExist*. Сервіс S_2 має вхідні параметри *StartL*, *DestinationL*, *Dt*, *CCNumber*, та вихідні параметри *Status*, *Flight*, *TrNumber* та для його виконання задана передумова *CorrectCard*. Для пошуку сервісу бронювання квитків заданий запит зі вхідними параметрами *Start*, *Destination*, *Date*, *CCNumber*, вихідними

параметрами *Status*, *Flight* та передумовою на номер кредитної картки *CorrectCard*.

Семантичні описи вхідних та вихідних параметрів сервісу мають бути такі самі як для параметрів запиту або мати відношення *subsumption*. Механізм виявлення повинен мати можливість вивести той факт, що параметри запиту *Start*, *Destination* та вхідні параметри *StartL*, *DestinationL* сервісів семантично одні й ті самі концепти. Це може бути виведено, використовуючи семантики онтології домена. (При цьому припускаємо, що сервіси та запит спираються на єдину інтегровану онтологію домену.) Запит також має передумову на *CCNumber,* яке повинно мати числове значення, що логічно має означати передумову сервісу, який шукається.

Визначення задачі виявлення. Заданий репозиторій $\mathcal{R}$ (3) та запит $\mathcal{Q}$ (4), задача виявлення може бути визначена як автоматичне знаходження множини сервісів S з репозиторію $\mathcal{R}$ таких, $s \in R$, та $I \sqsubseteq I'$, $A = A'$, $AO = AO'$, $O \sqsupseteq O'\}$, та існує така інтерпретація $\mathcal{I}$, що $CI'^{\mathcal{I}} \subseteq CI^{\mathcal{I}}$, $CO^{\mathcal{I}} \subseteq CO'^{\mathcal{I}}$, де $\sqsubseteq$ – означає відношення включення (subsumes). Слід зазначити, що в даному випадку CI', CI, I, I', A, A', AO, AO', CO, CO', O, O' – це кінцеві множини вхідних/вихідних параметрів, перед- та пост-умов. Операції включення, що наведені в цьому визначенні, це операції на цих множинах.

Задача виявлення графічно пояснюється на рисунку.



Де $CI'^{\mathcal{I}} \subseteq CI^{\mathcal{I}}$, $CO^{\mathcal{I}} \subseteq CO'^{\mathcal{I}}$, $I' \supseteq I$, $O \supseteq O'$

Рисунок. Задача виявлення

Визначення сервісу та запиту за допомогою ДЛ-онтологій. ДЛ [14] дозволяють представити домен, що нас цікавить, у термінах концептів або описів (унарні предикати), що характеризують підмножини об'єктів (екземплярів) в домені, та ролей (бінарні предикати) на такому домені. Концепти визначаються виразами, які формуються за допомогою спеціальних конструкторів [14]: верхній концепт ($\top$), нижній концепт ($\bot$), кон'юнкція концептів ($\sqcap$), універсальний кваліфікатор ($\forall$R.C), числові обмеження ($\geq$ nR) та ($\leq$ nR) (для різних ДЛ цей набір конструкторів різниться).

Онтологія домену для наведеного прикладу була предствлена вище. Для опису наших сервісів, доповнимо TBox онтології *POOntology* декількома концептами та твердженнями. Додамо до онтології концепти: *Operation, TrackingNumber,* та визначимо наступні твердження, які дозволять нам описати параметри запиту та сервісів:

CreditCardNumber $\sqsubseteq$ *hasType.{String}*

Flight $\sqsubseteq$ *hasCreditCard.CreditCardNumber*

Operation $\sqsubseteq$ *hasTrackingNumber.Number*

Number $\sqsubseteq$ *hasType.Type*

Type $=$ *{String, Numeric, Boolean}*

Для вирішення задач Веб-сервісів визначимо онтологію сервісу верхнього рівня *ServiceOntology*.

Service, InputParameter, OutputParameter, PreCondition, PostCondition, Parameter, Name, Value, Type

Service $\sqsubseteq$ *has.InputParameter; Service* $\sqsubseteq$ *has.OutputParameter*

Service $\sqsubseteq$ *has.PreCondition; PreCondition* $\sqsubseteq$ *Condition*

PostCondition $\sqsubseteq$ *Condition; Condition* $\sqsubseteq$ *hasValue.Boolean*

Service $\sqsubseteq$ *has.PostCondition; InputParameter* $\sqsubseteq$ *Parameter*

OutputParameter $\sqsubseteq$ *Parameter; Parameter* $\sqsubseteq$ *has.Name*

Type $=$ *{String, Numeric, Boolean}*

$I_S \sqsubseteq InputParameter$

$O_S \sqsubseteq OutputParameter;$

$CI_S \sqsubseteq PreCondition;$

$CO_S \sqsubseteq PostCondition;$

$Parameter \sqsubseteq has.Value;$ $Name \sqsubseteq hasType.\{String\};$ $Value \sqsubseteq hasType.Type$

Запит є також сервісом, але абстрактним. Точніше, абстрактним описом сервісу, що реалізує поставлену бізнес-задачу. Таким чином, онтологію запиту Q можна визначити як:

TBox

$Q \sqsubseteq Service$ зв'язок з онтологією сервісу верхнього рівня

$Q \sqsubseteq =4has.InputParameter;$

$Q \sqsubseteq =2has.OutputParameter$

$Q \sqsubseteq =1has.PreCondition;$

$I_Q \sqsubseteq InputParameter$

$O_Q \sqsubseteq OutputParameter;$

$CI_Q \sqsubseteq PreCondition$

ABox

$Start: I_Q;$ $Destination : I_Q;$ $Dt : I_Q;$

$CCNumber : I_Q$

$Offer: O_Q,$ $status: O_Q;$ $CorrectCard : CI$

$Start: Location;$ $Destination: Location;$

$Dt : Date;$

$Offer \sqsubseteq hasCreditCard.CCNumber;$

$Offer: Flight;$

$status: Status$

Сервіс S_1 можна визначити як:

TBox

$S_1 \sqsubseteq Service$ зв'язок з онтологією сервісу верхнього рівня

$S_1 \sqsubseteq = 5has.InputParameter;$

$S_1 \sqsubseteq =2has.OutputParameter;$

$S_1 \sqsubseteq = 1has.PreCondition;$

$I_{S1} \sqsubseteq InputParameter$

$O_{S1} \sqsubseteq OutputParameter;$

$CI_{S1} \sqsubseteq PreCondition$

ABox

$StartL: I_{S1},$

$DestinationL : I_{S1},$

$DateFrom : I_{S1},$

$DateTo: I_{S1},$

$CCNumber : I_{S1}$

$Status: O_{S1},$

$Offer: O_{S1};$

$EuropeanTrip: CI_{S1}$

$StartL: Location$ зв'язок з онтологією *POOntology*

$DestinationL: Location;$

$DateTo: Date;$

$DateFrom: Date$

$Offer \sqsubseteq hasCreditCard.CCNumber;$

$Offer: Flight;$

$status: Status$

Сервіс S_2 можна визначити як:

TBox

$S_2 \sqsubseteq Service$ зв'язок з онтологією сервісу верхнього рівня

$S_2 \sqsubseteq = 4has.InputParameter;$

$S_2 \sqsubseteq =3has.OutputParameter;$

$S_2 \sqsubseteq = 1has.PreCondition;$

$I_{S2} \sqsubseteq InputParameter;$

$O_{S2} \sqsubseteq OutputParameter;$

$CI_{S2} \sqsubseteq PreCondition;$

$StartL: I_{S2},$

DestinationL: $I_{S2,}$

Dt: $I_{S2,}$

CCNumber: $I_{S2,}$

Offer: $O_{S2,}$

Status: $O_{S2.}$

TrNumber: $O_{S2\ ;}$

CorrectCard: CI_{S2}

StartL: Location зв'язок з онтологією *POOntology*

DestinationL: Location;

DateTo: Date;

DateFrom: Date;

Offer $\sqsubseteq$ *hasCreditCard.CCNumber;*

Offer:Flight;

status:Status;

TrNumber:TrackingNumber

Визначимо онтологію задачі виявлення *DiscoveryOntology*:

$Q \sqsubseteq Service;$

$S \sqsubseteq Service$

$ExactInput \equiv (I_Q \sqsubseteq I_S) \sqcap (I_S \sqsubseteq I_Q)$

$SubsumesInput \equiv (I_S \sqsubseteq I_Q)$

$SubsumedInput \equiv (I_Q \sqsubseteq I_S)$

$ExactOutput \equiv (O_Q \sqsubseteq O_S) \sqcap (O_S \sqsubseteq O_Q)$

$SubsumesOutput \equiv (O_Q \sqsubseteq O_S)$

$SubsumedOutput \equiv (O_S \sqsubseteq O_Q)$

$FailOutput \equiv \neg((OQ \sqsubseteq OS) \sqcap (OS \sqsubseteq OQ))$

$FailInput \equiv \neg((IQ \sqsubseteq IS) \sqcap (IS \sqsubseteq IQ))$

$ExactPreCondition \equiv (CI_Q \sqsubseteq CI_S) \sqcap (CI_s \sqsubseteq CI_Q)$

$SubsumesPreC \equiv (CI_s \sqsubseteq CI_Q)$

$ExactPostCondition \equiv (CO_Q \sqsubseteq CO_S) \sqcap (CO_s \sqsubseteq CO_Q)$

$SubsumesPostC \equiv (CO_Q \sqsubseteq CO_S)$

Можемо встановити наступні співвідношення між елементами кортежей описів сервісів та запиту.

Для сервісу S_2:

$I_{S2} \sqsubseteq I_Q$ – для входів,

$O_{S2} \sqsupseteq O_Q$ – для виходів,

$CI_Q \Rightarrow CI_{S2}$, в силу їх еквівалентності – для передумов.

Для сервісу S_1:

$I_{S1} \not\sqsubseteq I_Q$ – для входів,

$O_{S1} \sqsupseteq O_Q$ та $O_{S1} \sqsubseteq O_Q$ – для виходів.

Таким чином, сервіс S_2 задовільняє запиту, а сервіс S_1 – ні, тому що він вимагає як вхідний параметр *DateTo*, що не забезпечується запитом. Запит вимагає *Flight* та Status як вихідний параметр та S_2 виробляє *Flight*, *Status* та *TrNumber*. Додатковий вихідний параметр може просто ігноруватися. При такому підході до опису запиту та сервісів (за допомогою ДЛ), пошук, співставлення та висновок про відповідність сервісу та запиту може здійснюватися автоматично, за допомогою існуючих резонерів ДЛ.

Але слід зазначити, що при вирішенні задачі включення для пар концептів, що є елементами кортежу опису сервісу та запиту, наприклад, I_S та I_Q виконується не лише співставлення екземплярів, що задані іменами, аналізується їх семантика – зв'язок з онтолгією домена. Вирішення задачі знаходження найбільш специфічного класу для індивіда є стандартним алгоритмом ДЛ, як і задача включення на концептах.

Таким чином, якщо є два параметри x і y такі, що *x:C1* та *y:C2,* та $C1 \sqsubseteq C2$, то *x:C2*, та можна казати про відповідність вхідних параметрів.

Формально задачу виявлення можна визначити наступним чином:

$Q \sqsubseteq Service$

$S \sqsubseteq Service$

$$MatchingQueryXY(Q,S) \sqsubseteq$$

$$\exists x.Iq, \exists y.Is: (C1(x) \sqcap C2(y) \sqcap$$

$$((C1 \sqsubseteq C2) \sqcap (C2 \sqsubseteq C1)) \sqcup (C1 \sqsubseteq C2)).$$

Висновки

Наведений підхід базується на використанні ДЛ як для представлення предметної області задачі (онтологія домена), так і для вирішення безпосередньо задачі виявлення Веб-сервісів, шляхом представлення сервісів та запиту за допомогою онтологій.

Семантичні описи Веб-сервісу являють собою твердження або аксіоми ДЛ. Кожен сервіс представляється ДЛ-онтологією на основі загальної онтології сервісу верхнього рівня. Кожна онтологія конкретного сервісу базується на базі знань інтегрованої ДЛ-онтології домену.

Враховуючі, що опис профілей семантичних Веб-сервісів, що зберігаються в реєстрі, є WSDL документом, який доповнений семантичними анотаціями, тобто є звичайним XML, можливий автоматичний розбір цього XML-визначення та побудова онтології конкретного сервісу на основі онтології сервісу верхнього рівня.

Існуючі резонери ДЛ дозволять здійснювати автоматичне співставлення запиту та сервісів, сервісів один з одним за входами виходами. Це стандартні задачі виводу ДЛ. Відношення між концептами та індивідами, що представлятимуть вхідну та вихідну інформацію запиту, визначають ступінь відповідності сервісів, що співставляються та ранжувати їх за цим показником. Запропонований Алгоритм співставлення на основі ДЛ може бути застосований для співставлення сервісів при композиції.

Але, слід зазначити, що наведений підхід розглядає задачу встановлення відповідності окремого сервісу з репозиторію та запиту (пряме співставлення), але в реальності малоймовірно знайти один існуючий сервіс, що буде задовільняти запиту для вирішення задачі. Як правило, нам треба знайти множину сервісів, що разом дозволять нам визначити новий сервіс, який буде відповідати запиту, або іншими словами, «покривати» цей запит. Ця множина сервісів формує композитний сервіс, що покриває запит, та виконує поставлену бізнес-задачу (не пряме співставлення). Для формування такого сервісу може бути використана задача знаходження найкращого покриття запиту [15, 16], що з технічної точки зору належить до загальної структури для рерайтингу (перезапису) [17, 18], використовуючи термінологію з [19].

1. http://www.w3.org/2002/ws/
2. http://www.informit.com/articles/article.aspx?p=336265
3. Formal Description of Web Services for Expressive Matchmaking. Dipl.-Inform. Sudhir Agarwal, 2007 Karlsruhe
4. Web Service composition: Semantic Links based approach. Freddy L´ecu´, Doctor of Philosophy, 2008
5. Ortiz M., Calvanese D., and Eiter T. Characterizing Data Complexity for Conjunctive Query Answering in Expressive Description Logics./AAAI, 2006.
6. Staab S., Studer R. Handbook on Ontologies. Second edition
7. Захарова О.В. Основні принципи побудови онтологічного граф-орієнтованого опису прикладної області. *Проблеми програмування.* 2010. № 4. С. 51–59.
8. Ruben Lara. Definition of semantics for web service discovery and composi tion. In Knowledge Web Deliverable D2.4.2, 2004.
9. D2.4.6 A Theoretical Integration of Web Service Discovery and Composition. Roberti Pierluigi (ITC-IRST) Marco Pistore (University of Trento) with contributions from: Walter Binder (EPFL), Ion Constantinescu (EPFL) Axel Polleres (UIBK), Holger Lausen (UIBK), Paolo Traverso (ITC-IRST), Michal Zaremba (NUIG). 2005. KWEB/2005/D2.4.6A/v1.0.
10. Hai Wang, Zengzhi Li. A Semantic Matchmaking Method of Web Services Based On SHOIN+(D). /Institute of Computer System Structure and Networks School of Electronics & Information Engineering, Xi'an Jiaotong University, Xi'an Shaanxi 710049,

PR China hwang@mailst.xjtu.edu.cn, lzz@mail.xjtu.edu.cn.
11. Integrating Description Logics and Action Formalisms for Reasoning about Web-services. Franz Baader, Carsten Lutz, Maja Milieie, Ulrike Sattler, Frank Wolter. LTCS-Report 05-02.
12. Akkiraju R., et al. (2005, December 6). Web Service Semantics. WSDL-S. Available: http://www.w3.org/Submission/WSDL-S/.
13. http://life-prog.ru/view_zam2.php?id= 204&cat=5&page=13
14. Baader F. and Nutt W.; In Baader F., Calvanese D., McGuinness D., Nardi D., and Patel-Schneider P. Basic Description Logics./ The Description Logic Handbook. P. 43–95. Cambridge University Press, 2003.
15. Baader F., Ku¨sters R., and Molitor R. Computing Least Common Subsumer in Description Logics with Existential Restrictions. In T. Dean, editor, Proc. of the 16th Int. Joint Conf. on AI. P. 96–101. M.K, 1999.
16. Franz Baader, Ralf Kiisters, and Ralf Molitor LuFg Theoretische Informatik, RWTH Aachen. Computing Least Common Subsumers in Description Logics with Existential.
17. Beeri C., Levy A.Y., and Rousset M-C. Rewriting Queries Using Views in Description Logics. In L. Yuan, editor, Proc. of the ACM PODS , New York, USA. P. 99–108, Apr. 1997.
18. Alon Y. Halevy. Answering queries using views: A survey. *VLDB Journal*, 10(4):270–294, 2001.
19. Franz Baader, Ralf Ku¨sters, and Ralf Molitor. Rewriting Concepts Using Terminologies. In Proc. of the Int. Conf.KRColorado, USA. P. 297–308, Apr. 2000.

References

1. http://www.w3.org/2002/ws/
2. http://www.informit.com/articles/article.aspx?p=336265.
3. Formal Description of Web Services for Expressive Matchmaking. Dipl.-Inform. Sudhir Agarwal, 2007 Karlsruhe.
4. Web Service composition: Semantic Links based approach. Freddy L´ecu´, Doctor of Philosophy, 2008.
5. M. Ortiz, D. Calvanese, and T. Eiter. Characterizing Data Complexity for Conjunctive Query Answering in Expressive Description Logics./AAAI, 2006.
6. S. Staab, R. Studer. Handbook on Ontologies. Second edition.
7. Zakharova O. General Principes for building the ontological graph – oriented description of application area. /Problems of programming. - №4, 2010. pp.51-59.(Ukrainian).
8. Ruben Lara. Definition of semantics for web service discovery and composi tion. In Knowledge Web Deliverable D2.42, 2004.
9. D2.4.6 A Theoretical Integration of Web Service Discovery and Composition. Roberti Pierluigi (ITC-IRST) Marco Pistore (University of Trento) with contributions from: Walter Binder (EPFL), Ion Constantinescu (EPFL) Axel Polleres (UIBK), Holger Lausen (UIBK), Paolo Traverso (ITC-IRST), Michal Zaremba (NUIG). 2005. KWEB/2005/D2.4.6A/v1.0.
10. Hai Wang, Zengzhi Li. A Semantic Matchmaking Method of Web Services Based On SHOIN+(D). /Institute of Computer System Structure and Networks School of Electronics & Information Engineering, Xi'an Jiaotong University, Xi'an Shaanxi 710049, PR China hwang@mailst.xjtu.edu.cn, lzz@mail.xjtu.edu.cn.
11. Integrating Description Logics and Action Formalisms for Reasoning about Web-services. Franz Baader, Carsten Lutz, Maja Milieie, Ulrike Sattler, Frank Wolter. LTCS-Report 05-02.
12. R. Akkiraju, et al. (2005, December 6). Web Service Semantics. WSDL-S. Available: http://www.w3.org/Submission/WSDL-S/
13. http://life-prog.ru/view_zam2.php?id=204&cat=5&page=13
14. F. Baader and W. Nutt; In F. Baader, D. Calvanese, D. McGuinness, D. Nardi, and P. Patel-Schneider. Basic Description Logics./ The Description Logic Handbook, pages 43–95. Cambridge University Press, 2003.
15. F. Baader, R. Ku¨sters, and R. Molitor. Computing Least Common Subsumer in Description Logics with Existential Restrictions. In T. Dean, editor, Proc. of the 16th Int. Joint Conf. on AI, pages 96–101. M.K, 1999.
16. Franz Baader, Ralf Kiisters, and Ralf Molitor LuFg Theoretische Informatik, RWTH Aachen. Computing Least Common Subsumers in Description Logics with Existential.

17. C. Beeri, A.Y. Levy, and M-C. Rousset. Rewriting Queries Using Views in Description Logics. In L. Yuan, editor, Proc. of the ACM PODS , New York, USA, pages 99–108, Apr. 1997.
18. Alon Y. Halevy. Answering queries using views: A survey. VLDB Journal, 10(4):270–294, 2001.
19. Franz Baader, Ralf Ku¨sters, and Ralf Molitor. Rewriting Concepts Using Terminologies. In Proc. of the Int. Conf.KRColorado, USA, pages 297–308, Apr. 2000.

Одержано 29.08.2017

Про автора:

Захарова Ольга Вікторівна,
кандидат технічних наук,
старший науковий співробітник.
Кількість наукових публікацій в українських виданнях – 25.
http://orcid.org/0000-0002-9579-2973.

Місце роботи автора:

Інститут програмних систем
НАН України,
проспект Академіка Глушкова, 40.
Тел.: 526 5139.
E-mail: ozakharova68@gmail.com.

УДК 004.822,681.3,519.81

Е.П. Ильина

ЭКСПЕРТНО-АНАЛИТИЧЕСКИЙ ПРОЦЕСС ВЫБОРА УПРАВЛЯЮЩИХ ОРГАНИЗАЦИОННЫХ ВОЗДЕЙСТВИЙ С ИСПОЛЬЗОВАНИЕМ КОРПОРАТИВНОГО ЗНАНИЯ. ЧАСТЬ 1. ФОРМАЛЬНЫЙ АНАЛИЗ СИСТЕМЫ ЦЕЛЕЙ

Работа посвящена моделям и методам поддержки экспертного выбора варианта управляющего воздействия, реализуемого организационными мероприятиями. В первой части представлена модель системы целей организации и методы анализа отношений эквивалентности, противоположности и достижимости целей. Такой анализ служит поддержке аналитической функции формирования контекста и рекомендаций для экспертно-аналитического процесса оценивания и выбора вариантов воздействия, который будет рассмотрен в следующей части статьи.

Ключевые слова: поддержка принятия решений, организационное решение, модель знаний об организации, система организационных целей, экспертный выбор, оценка достижимости цели, перспективность организационного воздействия.

Постановка задачи

Принятие организационного решения представляет собой процесс выбора воздействия на объект управления, которое наилучшим образом обеспечивает достижение заданной цели организации.

В зависимости от того, является решение регламентным, типовым, стратегическим, антикризисным или инновационным, модель процесса принятия решения и необходимые средства его поддержки могут существенно различаться [1]. Примером служат парадигмы, полярные с позиций уровня предполагаемой формализованности процесса. Один полюс составляет поддержка так называемых деловых решений [2], базирующаяся на процедуре автоматического выбора варианта по заданным правилам на основании заданных информационных источников (стандарт *DMN* [3]. Другой полюс представлен моделью [4], рассматривающей многоэтапный процесс принятия уникального решения как процесс управления проектом для максимального обеспечения интересов стейкхолдеров.

В работе [5] показано, что современные парадигмы и условия осуществления менеджмента организаций определяют специфику процесса принятия решений, которую составляют: множественность моделей процессов принятия для решений заданного типа; ведущая роль эффектов взаимовлияния в поле решений организации; необходимость использования экспертного опыта, включая интеграцию мнений различных деловых групп по поводу проблемной ситуации и предпочтений.

Данная работа посвящена построению методов поддержки процесса принятия решения, занимающего промежуточную позицию по сравнению с описанным ранее [6] ситуативно обусловленным решением общего вида и деловым решением [2]. Такая позиция характерна для нерегламентных решений, имеющих дело с новой проблемной ситуацией, но реализующих выбор между известными мероприятиями или их композициями. Ей соответствуют также регламентные решения, находящиеся на этапе установления или пересмотра процедур их принятия [1].

Общая модель процесса решения имеет вид последовательности этапов

$$MPD = <PS, TS, AP, CA, NT>, \quad (1)$$

где *PS* – анализ проблемной ситуации,

TS – постановка задачи влияния на *PS*,

AP – формирование пакета альтернативных воздействий,

CA – выбор воздействия,

NT – анализ возможных негативных последствий и путей их устранения.

В работе [5] рассмотрены общий случай процедур реализации этапов (1) и формируемые ими структуры знания. В качестве объекта исследования данной работы рассматривается экспертно-аналитический процесс реализации этапа *CA* из (1), ориентированный на решения охарактеризованного выше типа. Он объединяет в своем составе как функции получения экспертных мнений, так и автоматизированные аналитические функции формирования контекстов экспертного оценивания, а также интеграцию и обоснование экспертных мнений [7] с использованием структур корпоративного знания.

Модель экспертно-аналитического процесса имеет вид

$$MEA = (KPS, KTS, KAP, FMV, \{IMV_i, FI_i\}_{i=1,N}, OO, PF, GF, EG, CON, KNT, CH, AR, FAR, FC, FFI, FFMI, FFMG), \quad (2)$$

где *KPS* – знания о проблемной ситуации, сформированные на первом этапе (1), *KPS* = <{*CC*}, *L*>, где *CC* – элемент, определяющий конфликт, *L* – требуемый уровень вмешательства в проблемную ситуацию;

KTS – знание о требуемом влиянии на проблемную ситуацию, имеющее формат цели, который будет рассмотрен в следующем разделе;

KAP – множество вариантов воздействия (автоматически либо экспертно сформированных);

FMV – рамочная модель перспективности воздействия;

IMV_i – индивидуальная модель ценности, осуществляемая *i*-м экспертом для детализации рамочной;

FI_i – оценка параметров IMV_i;

OO – онтология организации [6], используемая для формализованной идентификации и анализа объектов, субъектов и их отношений;

PF – поле решений, текущее состояние которого содержит информацию о всех принимавшихся решениях организации;

GF – поле целей, включающее цели, входящие в состав элементов *PF*, а также нормативные и плановые для организации и отражающие интересы стейкхолдеров;

EG – состав экспертной группы, основанный на представительстве бизнес-ролей, точки зрения которых должны быть учтены;

CON – контекст, автоматически формируемый для оценки элементов IMV_i на основании *OO*, *PF* и *PG*;

KNT – знания о возможных негативных последствиях, экспертно формируемые в данном процессе <*OBN*, *DN*>, где *OBN* – объект негативного влияния, *DN* – решение по преодолению последствий (в модели вида (2) для $DN\ CC \in KTS$ включает влияющее решение, а *KTS* задает цель компенсирующего воздействия);

CH – выбранный вариант воздействия;

AR – рекомендации, автоматически формируемые посредством функции *FAR*;

FC – функция формирования контекстов;

FFI – функция интеграции частных критериев IMV_i;

FFMI – функция интеграции оценок критериев индивидуальных моделей в оценки критериев;

FFMG – функция получения обобщенного экспертного мнения.

В первой части статьи представлена формализация структур знания, служащих представлению целей в модели (2), и аналитических функций, необходимых для работы с ними при организации процесса поддержки принятия решений рассмотренного класса.

Роль и представление структур целей

Необходимость выработки рассмотренных решений как эффективных по отношению к поставленным целям управле-

ния требует от структур знаний, используемых процессом (2):

- формализации и отображения целей организации и интересов стейкхолдеров;
- отображения в (2) целей, входящих в состав решения, определенного моделью (1) (воздействия $G_1 \in KTS$, сохранения свойств объекта влияния $G_2 \in KNT$, цели компенсирующего воздействия $G_3 \in \in KNT$);
- формализации модели полноты достижения цели выбираемого воздействия при текущем состоянии поля решений и поля целей *PF* и *GF* (в качестве специальной подмодели в составе *FMV* и IMV_i);
- формального представления концепта Дерево целедостижения и его использования – в составе *FMV*, *IMV* и *GF*.

Для последующей формализации отношений, необходимых для использования аналитическими функциями *FC* и *FAR* из (2), предлагаемая формальная структура цели основывалась на сочетании черт таких подходов как реализованный в проекте *KAOS* [8] и предложенный в работе [9]. Такое сочетание представляет собой введение в определение онтологической семантики цели как описания ее объекта и требуемого для него результирующего состояния, так и описания имеющихся вхождений цели в деревья целедостижения, включенные в онтологию. При этом отношения между целями в дереве взяты из модели [8]. Концепция соотношения жестких и мягких целей и направлений воздействия на целевые объекты является развитием подхода [9]. Возможность распространения на мягкие цели тех же принципов онтологического анализа, которые используются для жестких целей, обеспечивается использованием, в качестве их объекта, организационной структуры в целом и ее крупных фрагментов (как структурных, так и функциональных). Кроме того, в качестве целевых свойств используются при этом оценочные характеристики с вербальными шкалами и отношения, означиваемые для целевого объекта индикатором его принадлежности. Направления целевого воздействия предполагают при этом установление, сохранение, недопущение или разрыв связи.

Модель Цели имеет вид

$$MG = (OB, PR, T_1, T_2, T_3, ST(TT)), \qquad (3)$$

где OB – целевой объект;

PR – его свойства, используемые для определения целевого состояния;

T_1 – характер неопределенности цели: $T_1 \in \{ZT_{11}, ZT_{12}\}$, ZT_{11} = жесткая, ZT_{12} = мягкая;

$T_2 \in \{ZT_{21}, ZT_{22}, ZT_{23}, ZT_{24}\}$ – тип используемых целевых свойств, где ZT_{21} – Параметры, ZT_{22} – Идентификаторы, ZT_{23} – Оценочные характеристики, ZT_{24} – Отношения;

T_3 – целевое условие для PR, $T_3 =< C, ET >$, C – тип условия, ET – эталон сравнения.

$$C \in \{T_3.ZC_i\}_{i=1}^{14},$$

$T_3.ZC_1$ = повышение до ET, $T_3.ZC_i$ = повышение, $T_3.ZC_3$ = понижение до ET, $T_3.ZC_4$ = понижение, $T_3.ZC_5$ = минимизация, $T_3.ZC_6$ = максимизация, $T_3.ZC_7$ = приближение к оптимальному уровню, $T_3.ZC_8$ = фиксация имеющегося состояния, $T_3.ZC_9$ = недопущение состояния ET, $T_3.ZC_{10}$ = снижение риска достижения состояния ET, $T_3.ZC_{11}$ = формирование, $T_3.ZC_{12}$ = разрушение, $T_3.ZC_{13}$ = способствование, $T_3.ZC_{14}$ = противодействие.

$ST(TT)$ – уровень актуальности действия цели в момент TT,

$ST(TT) \in \{ZST_i\}_{i=1}^{4}$, ZST_1 = достигнутая, ZST_2 = находящаяся в реализации, ZST_3 = запланированная на будущее, ZST_4 = вырабатываемая или предварительно уточняемая.

Для цели G значение $< ZT_1.ZT_2 >$ определяет допустимое для такой пары подмножество MT_3 значений T_3. Пусть

$$T_2K =< ZT_{21}, ZT_{23} >, T_2L =< ZT_{22}, ZT_{24} >, (4)$$

Тогда для $i = 1,2$

$$< ZT_{11}, T_2K_i > \rightarrow (MT_3 \in \{T_3.ZC_1, T_3.ZC_3,$$

$$T_3.ZC_5, T_3.ZC_6, T_3.ZC_8, T_3.ZC_9\})$$

$$< ZT_{11}, T_2L_i > \rightarrow (MT_3 \in \{T_3.ZC_{11}, T_3.ZC_{12}\})$$

$$< ZT_{12}, T_2K_i > \rightarrow (MT_3 \in \{T_3.ZC_2, T_3.ZC_4,$$

$$T_3.ZC_7, T_3.ZC_{10}\})$$

$$< ZT_{12}, T_2L_i > \rightarrow (MT_3 \in \{T_3.ZC_{13}, T_3.ZC_{14}\}).$$

В качестве конгломерата целей рассматривается концепт Дерево целедостижения, который представляет иерархию целей, однонаправленно влияющих на достижение друг друга. Модель этого концепта имеет следующий вид:

$$MTR = (GG, \{G_i, \{G_j, B(G_i, G_j)\}_{j=1}^{mi}\}_{i=1}^{nt}), (5)$$

где GG – корневая вершина, соответствующая итоговой цели;

G_i – i-я цель, оказывающая влияние на достижимость GG;

$G_j, j > i$ – цель, влияющая на достижимость G_i;

$B(G_i, G_j)$ – тип связи между целями,

$$B(G_i, G_j) = < S, U >,$$

где S – направление влияния (позитивное pos или негативное neg), U – характер влияния (определяющее d или способствующее h).

Внутренне определенные отношения между целями

Рассмотрим класс отношений между целями, предикаты которых используют элементы модели цели (3), не адресуясь к положению в деревьях целедостижения.

Будем обозначать $X_i(G)$ онтологическую интерпретацию элемента X модели (3) для цели G_i, $ISA_o(X_1, X_2)$ – отношение класс-подкласс между X_1, X_2 в онтологии O, $PARTOF_o(X_1, X_2)$ – отношение целое-часть между X_1, X_2 в онтологии O предметной области деятельности организации [6].

Определим как объектно-эквивалентные $(OEQ(G_1, G_2))$ цели G_1, G_2, для которых выполнено условие

$$(OB(G_1) = OB(G_2)) \vee \underline{ISA_o}\,(OB(G_1),$$

$$(OB(G_2)) \wedge PR(G_1) = PR(G_2)).$$

Далее, определим отношение соответствия $COR(G_1, G_2)$ значений $T_3(G_1)$, $T_3(G_2)$ между жесткой целью G_1 и мягкой G_2, основанное на значениях элементов T_3 из их модели вида (3).

$$COR(G_1, G_2) \rightarrow ((T_3.ZC_1, T_3.ZC_2) \vee$$

$$(T_3.ZC_3, T_3.ZC_4) \vee (T_3.ZC_5, T_3.ZC_7) \vee$$

$$(T_3.ZC_6, T_3.ZC_7) \vee (T_3.ZC_9, T_3.ZC_{10}) \vee$$

$$(T_3.ZC_{11}, T_3.ZC_{13}) \vee (T_3.ZC_{12}, T_3.ZC_{14})).$$

Тогда G_1, G_2 будем называть вполне эквивалентными $(PEQ(G_1, G_2))$, если:

$$OEQ(G_1, G_2) \wedge (((T_1(G_1) = T_2(G_1)) \wedge$$

$$(T_3.ZC(G_1) = T_3.ZC(G_2))) \vee$$

$$(T_1(G_1) \neq T_1(G_2) \wedge COR(G_1, G_2))).$$

G_1 будем называть локально сниженно эквивалентной $(LEQ(G_1, G_2))$, относительно G_2, если

$$PARTOF_o(OB(G_2), OB(G_1)) \wedge ((PR(G_2) =$$

$$PR(G_1)) \wedge ((T_1(G_1) = T_1(G_2)) \vee (COR(G_2, G_1)).$$

Определим также отношение противоположности $OPPOS(G_1, G_2)$, имеющее место при выполненности одной из шести систем условий.

1. Если $OEQ(G_1, G_2) \wedge (T_1(G_1) = T_1(G_2) = ZT_{11}) \wedge (T_2(G_1) = T_2(G_2)) \vee \vee T_2(G_1) \in T_2K$ (см. (4))

и выполнено одно из подусловий:

а) $(T_3(G_1) = T_3.ZC_1) \wedge (T_3(G_2) = T_3.ZC_3) \wedge T_3.ET(G_2) < T_3.ET(G_1))$;

б) $(T_3(G_1) = T_3.ZC_5) \wedge ((T_3(G_2) = T_3.ZC_6) \vee (T_3(G_2) = T_3.ZC_8))$;

в) $(T_3(G_1) = T_3.ZC_8) \wedge (T_3(G_2) = T_3.ZC_9) \wedge (T_3.ET(G_2) = A)$,

где A –имеющееся на момент постановки цели состояние целевого объекта.

2. Если $OEQ(G_1, G_2) \wedge (T_1(G_1) = T_1(G_2) = ZT_{11}) \wedge (T_2(G_1) = (T_2(G_2) = ZT_{24})$

и

$T_3(G_1) = T_3.ZC_{11} \wedge T_3(G_2) = T_3.ZC_{12}$.

3. Если $OEQ(G_1, G_2) \wedge T_1(G_1) = ZT_{11} \wedge T_1(G_2) = ZT_{12} \wedge T_2(G_1) = T_2(G_2) = ZT_{20}$

и выполнено одно из подусловий:

а) $(T_3(G_1) = T_3.ZC_3) \wedge T_3(G_2) = T_3.ZC_2$;

б) $T_3(G_1) = T_3.ZC_1 \wedge T_3(G_2) = T_3.ZC_4$;

в) $T_3(G_1) = T_3.ZC_1 \wedge T_3(G_2) = T_3.ZC_{10} \wedge \wedge (T_3\ ET(G_1) \geq T_3.ET(G_2))$;

г) $T_3(G_1) = T_3.ZC_3 \wedge T_3(G_2) = T_3.ZC_{10} \wedge (T_3\ ET(G_1) \leq T_3.ET(G_2)$.

4. Если $OEQ(G_1, G_2) \wedge T_1(G_1) = ZT_{11} \wedge (T_1(G_2) = ZT_{12} \wedge T_2(G_1) = T_2(G_2) = ZT_{24}$: $(T_3(G_1) = T_3.ZC_{11} \wedge \wedge (T_3(G_2) = T_3.ZC_{14}) \vee (T_3(G_1) = T_3.ZC_{12} \wedge T_3(G_2) = T_3.ZC_{13})$.

5. Если $OEQ(G_1, G_2) \wedge (T_1(G_1) = T_1(G_2) = ZT_{12}) \wedge (T_2(G_1) = (T_2(G_2) = ZT_4)$

и выполнено одно из подусловий а), б):

а) $T_3(G_1) = T_3.ZC_2 \wedge ((T_3(G_2) = T_3.ZC_4) \vee (T_3(G_2) = T_3.ZC_7 \wedge \wedge ET_{opt} < ET_{act}) \vee (T_3G_2) = T_3.ZC_7 \wedge \wedge ET_{opt} < ET_{act}) \vee (T_3(G_2) = T_3.ZC_{10}) \wedge E_r > ET_{act}))$,

где ET_{opt} – оптимальное значение свойства,

E_r – пороговое значение свойства, определяющее неприемлемый риск достижения ET,

ET_{act} – текущее значение свойства;

б) $T_3(G_1) = T_3.ZC_4 \wedge ((T_3(G_2) = T_3.ZC_7 \wedge ET_{opt} > ET_{act}) \vee (T_3(G_2) = T_3.ZC_9 \wedge E_r < ET_{act}))$.

6. Если $OEQ(G_1, G_2) \wedge (T_1(G_1) = T_1(G_2) = ZT_{12}) \wedge (T_2(G_1) = T_2(G_2) = ZT_{24}) \wedge (T_3(G_1) = T_3.ZC_{13}) \wedge (T_3(G_2) = T_3.ZC_{14})$.

Внешние интерактивные отношения между целями

Каждое из отношений рассматриваемого класса определяется для целей G_1, G_2, для которых

$\exists (TR \in (PF \cup GF)) \mid INFL(G_1, G_2, TR)$,

где $INFL$ – отношение влияния с предикатом $B(G_1, G_2) \in MTR(TR)$ (см. (5)).

Класс составляют следующие отношения.

Отношение обеспечения

$$POSB\,(G_1, G_2, TR) \rightarrow$$

$$\rightarrow (B(G_1, G_2) = <\text{pos}, d>). \qquad (6)$$

Отношение способствования

$$POSH\,(G_1, G_2, TR) \rightarrow$$

$$\rightarrow (B(G_1, G_2) = <\text{pos}, h>). \qquad (7)$$

Отношение запрета

$$NEGB\,(G_1, G_2, TR) \rightarrow$$

$$\rightarrow (B(G_1, G_2) = <\text{neg}, d>). \qquad (8)$$

Отношение помехи

$$NEGH\,(G_1, G_2, TR) \rightarrow$$

$$\rightarrow (B(G_1, G_2) = <\text{neg}, h>). \qquad (9)$$

Свойство актуализированности цели

Рассмотрим класс метризованных отношений, позволяющих формализовать свойство актуализированности цели G_0 в рассматриваемом на момент TT состоянии PF и GF. Формализуемое свойство характеризует потенциальную возможность заданной цели G_0 проявлять свои влияния в связи с имеющимися при текущем положении дел предпосылками ее выполнения.

В онтологии целей организации будем различать цели по их генезису: нормативные, плановые, фигурирующие в моделях решений (2) (как цель влияния, как цель сохранения состояния объекта негативного влияния и как цель действий по его нейтрализации), а также отражающие интересы стейкхолдеров. Таким образом, возникают три возможных для цели онтологических позиции $\{P_j\}_{j=1,\ldots 3}$. Введем показатель OP_i онтологического веса цели G_i, которому приписано значение в интервале (0, 1), априори поставленное в соответствие ее онтологической позиции P_j.

Кроме того, будем полагать заданным значение показателя OW_i веса актуального статуса цели G_i, аналогично приписанное тому из четырех значений элемента $ST(TT)$ из модели G_i вида (3), которое он имеет в момент TT рассмотрения этой цели. Таким образом, цель G_i характеризуется кортежем

$$<OP_i,\ OW_i>. \qquad (10)$$

Для некоторой цели G_0 обозначим как NE мощность множества целей $\{G_{0k}\}$, представленных на момент TT в PF и GF, которые эквивалентны G_0 в смысле отношения $PEQ(G_0, G_{0k})$. Аналогично, определим NC как мощность множества

$$\{G_{0l} \mid OPPOS\ (G_0,\ G_{0l})\}.$$

Уровнем онтологической поддержанности G_0 в момент TT будем полагать

$$HO_1(G_0) = \sum_{k=1}^{NE} OP_k \cdot OW_k / NE,$$

а уровнем онтологических помех для G_0

$$HO_2(G_0) = \sum_{l=1}^{NC} OP_l \cdot OW_l / NC.$$

При $NE = 0$ или $NC = 0$ соответствующий уровень определяется как нулевой.

Определим свойство онтологической актуализированности G_0 $OA(G_0)$ с метрикой

$$MOA(G_0) = \frac{1 + HO_1(G_0) - HO_2(G_0)}{2}. \qquad (11)$$

Перейдем к определению свойства интерактивной актуализированности G_0, обусловленной ее положением в деревьях целедостижения (5), представленных на момент TT в PF и GF, и позицией тех целей, которые составляют ее поддержку либо препятствие для нее в составе этих деревьев.

Определим для цели G_0 множество GG интерактивно релевантных ей целей

$$GG = \{ G_i \mid \exists TR \mid (TR \in (PF \cup GF) \wedge B(G_0, G_i) \in TR) \}_{i=1}^{NG}, \quad (12)$$

где TR – дерево целедостижения,

$B(G_0, G_i)$ – непосредственная связь G_0 с G_i как с детализирующей целью,

NG – мощность множества GG.

Для каждой G_i определим множество $TTR(G_0, G_i)$ деревьев целедостижения, которые удовлетворяют подусловию для них из (12), а также его разбиение

$$TTR(G_0, G_i) = TTR_1(G_0, G_i) \cup \cup TTR_2(G_0, G_i) \cup TTR_3(G_0, G_i) \cup \cup TTR_4(G_0, G_i), \quad (13)$$

в котором k-й элемент из (13) ($k = 1,\ldots,4$) имеет предикат, постулирующий отношение между G_0 и G_i для всех входящих в него деревьев, соответствующее k-му элементу кортежа отношений

$$< POSB,\ POSH,\ NEGB,\ NEGH > \quad (14)$$

и мощность $NO_k = |TTR_k|$.

Учитывая тот факт, что деревья целедостижения, рассмотренные в (12)–(13), имеют те же типы онтологических позиций, что и цели (принадлежность нормам, плановым структурам либо решениям), можно ввести для них показатель онтологического веса TOP (по аналогии с показателем OP для целей (см. (10))).

Цели, рассматриваемые в составе деревьев в (12)–(13), могут быть охарактеризованы весом своего актуального статуса OW, в полном соответствии с (10).

Определим степень влияния k-го типа (согласно (14)) цели G_i на цель G_0

$$IH_k(G_0, G_i, TT) = \sum_{j=1}^{NO_k} TOP_j / NO_k$$

(со значением 0 при $NO_k = 0$).

Посредством этого показателя можно построить оценки обусловленной G_i поддержки G_0

$$LP(G_0, G_i, TT) = (IH_1 + 0.5IH_2)/2,$$

а также обусловленного препятствования

$$LN(G_0, G_i, TT) = (IH_3 + 0.5IH_4)/2.$$

Определим для G_0 соответствующие показатели, интегрированные по всем G_i из (12) (при $NG \neq 0$).

$$LP(G_0, TT) = \sum_{i=1}^{NG} LP(G_0, G_i, TT) \cdot OW_i / NG,$$

$$LN(G_0, TT) = \sum_{i=1}^{NG} LN(G_0, G_i, TT) \cdot OW_i / NG.$$

Далее определим свойство интерактивной актуализированности $IA(G_0, TT)$ с метрикой

$$MIA(G_0, TT) = = (1 + LP(G_0, TT) - LN(G_0, TT)) / 2.$$

Впрочем, используя (11), можно определить свойство результирующей актуализированности RA (G_0, TT) с метрикой

$$MRA(G_0, TT) = (OA(G_0, TT) + + IA(G_0, TT)) / 2. \quad (15)$$

Предложенная формализация использует ряд упрощающих предположений:

- об отсутствии вклада в онтологическую актуализированность целей, соотнесенных с G_0 отношением локально сниженной эквивалентности;
- о равноценности всех трех онтологических позиций цели в составе модели решения;
- об одинаковом влиянии онтологической и интерактивной актуализированности.

Все они могут быть упразднены при введении дополнительных весовых коэффициентов, специфичных для модели деятельности конкретной организационной системы.

Свойство гипотетической достижимости цели в составе дерева целедостижения

На основе произведенной формализации свойства результирующей актуализированности цели при фиксированном состоянии поля целей и поля решений организации определим соответствующее свойство цели G_0 в составе дерева целедостижения.

Будем рассматривать конструктивную модель G_0 в составе дерева TR следующим образом:

$$KM(G_0) = < A_1, A_2 >,$$

где A_1 – аспект достаточности поддержки;

A_2 – аспект приемлемости помех.

Определим

$$A_1 = \{GP_i, W_i\}_{i=1}^{NP}, \; A_2 = \{GN_j, W_j\}_{j=1}^{NN},$$

где GP_i – цель, поддерживающая G_0,

W_i – сила поддержки (1 или 0.5 в зависимости от вида отношения из (14) между GP_i и G_0),

NP – число целей поддержки,

GN_j – цель препятствования,

W_j – сила препятствования,

NN – число целей препятствования.

Тогда, сопоставив каждой цели GP_i либо GN_j оценку ее результирующей актуализированности MRA из (15), можно предложить метрики MA_1 и MA_2

$$MA_1 = \sum_{i=1}^{NP} W_i \cdot MRA(GP_i) / NP,$$

$$MA_2 = \sum_{j=1}^{NN} W_j \cdot (1 - MRA(GP_j) / NN,$$

равные 0 при $NP = 0$ или $NN = 0$.

Такое представление формальных оценок позволяет оценить достижимость $MR(G_0)$ для G_0 посредством линейной свертки соответствующего фрагмента дерева

$$MR(G_0) = (K_1 \cdot MA_1 + K_2 \cdot MA_2)/((K_1 + K_2),$$

где коэффициенты K_1, K_2 определяют соотношение важности критериев поддержанности и невоспрещенности, характеризующее прерогативу рисков в конкретной задаче целедостижения.

Предложенный подход реализует многокритериальное представление достижимости цели. Он не позволяет выйти за рамки компенсаторной стратегии [10]. Однако он позволяет осуществлять гипотетическое сравнение разных вариантов управляющих воздействий, предоставляя экспертам контекст для их оценивания с позиций достижимости целей посредством характеризации положения дел и его возможного влияния. При этом представленные выше формальные оценки свойств могут быть применены вместе с экспертными для использования в правилах вывода диагностических и рекомендационных утверждений в рамках инструментария [11].

Модель экспертно-аналитического процесса, осуществляющего реализацию этих возможностей при поддержке выбора воздействия в широком контексте рассмотрения принимаемого решения, будет описана во второй части данной работы.

Выводы

Предложенный формальный аппарат служит основой для оценивания достижимости целей выбираемого управляющего воздействия при имеющемся состоянии системы целей организации. Использование данных о таком состоянии опирается на ведение онтологически систематизируемого корпоративного знания о планах организации, интересах ее стейкхолде-

ров и принимаемых решениях. Это соответствует принципам и задачам, постулируемым направлением Semantic Business Process Management.

Объединение в используемом представлении целей как характеристик объекта целевого воздействия, так и отношений взаимовлияния целей позволяет рассматривать широкий спектр влияний на достижимость и систему разнотипных целей (по уровню управления, характеру влияния на объект, четкости критериев достижения).

Анализ уровня достижимости цели воздействия, рассматриваемого и оцениваемого в предложенной трактовке, обеспечивает возможность его использования в системе критериев оценки перспективности такого воздействия. При этом оценка осуществляется в связи с проблемной ситуацией решения и его возможными побочными последствиями.

Модели и процессы экспертно-аналитической реализации необходимого оценивания будут рассмотрены в следующей части статьи.

1. Ильина Е.П., Синицын И.П. Модели и методы поддержки аналитического сопровождения поля решений организации. *Проблеми програмування*. 2017. № 3. С. 113–127.
2. Taylor J. An Introduction to Decision Modeling with DMN. Decision Management Solutions, 2016. 12 p. Available at www.omg.org/news/whitepapers/An_Introduction_to_Decision_Modeling_with_ DMN.pdf.
3. Decision Model and Notation (DMN). Version 1.1. Object Management Group, Inc, 2016. 182 p. [Electronic resourse]. Mode of access: http://www.omg.org/spec/DMN/1.1.
4. Skinner D.C. Introduction to Decision Analysis: A Practitioner's Guide to Improving Decision Quality. 3rd ed. Sugar Land, TX: Probabalistic Publishing, 2009. 350 p.
5. Ильина Е.П. Управление качеством организационных решений на основе формализованного корпоративного знания. Ч 1. Онтология организационных решений. *Математические машины и системы.* 2014. № 1. С. 129–142.
6. Ильина Е.П. Методы и модели использования экспертно-аналитического знания для поддержки принятия решений в организации. Часть 1. Модели знания о решениях. *Проблеми програмування*. 2016. № 1. С. 89–101.
7. Ильина Е.П. Экспертно-аналитическое сопровождение системы решений организации. МОДС2017. Доклады XII Международной научно-практической конференции Математическое и имитационное моделирование систем. 26–29 июня 2017 года, Чернигов. 2017. С. 319–323.
8. Giorgini P., Myloupoulos J., Nicchiatelli E., Sebastiani R. Formal Reasoning Techniques for Goal Models. In: S. Spaccapietra et al. (Eds), Conceptual Modeling. – ER2002. Proc. of the 21-st Conference on Conceptual Modeling. LNCS2503. Springer, 2002.
9. Popova V., Sharpanskykch A. Formal modelling of organizational goals based on performance indicators. *In: Data & Knowledge Engineering*. Vol. 70. ISS.4. Apr. 2011. P. 335–364.
10. Ногин В.Д. Линейная свертка критериев в многокритериальной оптимизации. *Искусственный интеллект и принятие решений.* 2014. № 4. С. 73–82.
11. Ильина Е.П., Слабоспицкая О.А., Синицын И.П., Яблокова Т.Л. Автоматизированная поддержка принятия решений по управлению программами фундаментальных научных исследований с использованием экспертной методологии. Препр. Киев: Институт программных систем НАН Украины, 2010. 94 с.

References

1. Ilina E.P., Sinitsyn I.P. Models and Methods for Automated Analytic Support of the Organization Decision Field [In Russian] In: Problems in Programming. 2017. N 3. P. 113–127
2. Taylor J. An Introduction to Decision Modeling with DMN. Decision Management Solutions, 2016. 12 p. Available at www.omg.org/news/whitepapers/An_Introduction_to_Decision_Modeling_with_DMN.pdf.
3. Decision Model and Notation (DMN). Version 1.1. – Object Management Group, Inc, 2016. – 182 p. [Electronic resourse]. – Mode of access: http://www.omg.org/spec/DMN/1.1.

4. Skinner D.C. Introduction to Decision Analysis: A Practitioner's Guide to Improving Decision Quality. 3rd ed. / D.C.Skinner – Sugar Land, TX: Probabalistic Publishing, 2009. – 350 p.
5. Ilina E.P. "Management of quality of organization decisions grounded on formalized corporation knowledge. P1. Ontology of organization decisions" [In Russian] In: Mathematic machines and Systems. 2014. N 1. P. 129–142.
6. Ilina E.P. Methods and models of the expert analytic knowledge using for the decision support in organization. P1. Decisions models [In Russian] In: Problems in Programming. 2016. N 1. P. 89–101.
7. Ilina E.P. Expert the Organization Decisions System [In Russian] // MODC2017. Reports of XII International scientific and practical conference/ "Mathematical and Simulating Modeling of Systems".– 26-29 June 2017. – Chernigov / 2017. P. 319–323.
8. Giorgini P., Myloupoulos J., Nicchiatelli E., Sebastiani R. Formal Reasoning Techniques for Goal Models // In: S. Spaccapietra et al. (Eds), Conceptual Modeling. – ER2002 // Proc. of the 21-st Conference on Conceptual Modeling. – LNCS2503. – Springer, 2002.
9. Popova V., Sharpanskykch A. Formal modelling of organizational goals based on performance indicators // In: Data & Knowledge Engineering. – V. 70. – ISS.4. – Apr.2011. P. 335–364.
10. Nogin V.D. Linear Convolution of Criteria in Multicriterial Optimization [In Russian] In: Artificial Intelligence and Decision Making. – 2014. – N 4. P. 73–82.
11. Ilina E.P., Slabospitskaya O.A., Sinitsyn I.P., Yablokova T.L. Computer Support of decision making in the fundamental scientific research programs management using the expert methodology [In Russian] Kiev, 2010. – 94P. (Preprint. – Kiev. – Software Systems Institute of NAS of Ukraine, 2010).

Получено 30.08.2017

Об авторе:

Ильина Елена Павловна,
кандидат физико-математических наук,
старший научный сотрудник,
ведущий научный сотрудник
Количество научных публикаций в украинских изданиях – 60
http://orcid.org/0000-0002-1528-366X

Место работы автора:

Институт программных систем
НАН Украины,
03187, Киев-187,
проспект Академика Глушкова, 40.
E-mail: Ilyina elena1@ukr.net

УДК 004.02+004.05+005.93+ 510.3

Є.С. Родін

ЗАДАЧІ З КЕРУВАННЯ РИЗИКАМИ ІНФОРМАЦІЙНОЇ БЕЗПЕКИ АПАРАТА ПРИЙНЯТТЯ РІШЕНЬ

Запропоновано перелік завдань, вирішення яких веде до побудови моделі залежності рівня ризику інформаційної безпеки ресурсу від наявності зв'язків та ступеню впливу вразливостей, загроз і наслідків на даний інформаційний ресурс та організацію в цілому. Наведено приклади побудови для окремого інформаційного ресурсу дерев зв'язків між вразливостями, загрозами, наслідками подій інформаційної безпеки, а також приклади математичної формалізації залежності рівня ризику реалізації певної загрози пошкодження або втрати інформаційного ресурсу від стану вразливостей та їх послідовного впливу на даний ресурс.

Ключові слова: ризик, інформація, безпека, загроза, вразливість, нечіткість, логіка, модель.

Вступ

Актуальна проблема сьогодення – це прийняття рішень у багатофакторному середовищі. Аналіз багатофакторності ускладнюється відсутністю достовірної фактичної інформації про суб'єкти, події, процеси. Залежність реалізації одних подій від інших, представлення реалізації різних сценаріїв ґрунтуються переважно на експертних оцінках. Багато сучасних моделей процесів прийняття рішень намагаються врахувати якомога більше факторів впливу та водночас нівелювати суб'єктивізм експертних оцінок. Саме такі умови притаманні сфері керування ризиками інформаційної безпеки апарата прийняття рішень.

Стаття пропонує до уваги читача приклади моделей і методів, які б у доступному вигляді допомогли непрофесіоналу в галузі оцінювання ризиків інформаційної безпеки (ІБ) побудувати її модель керування в сегменті своєї відповідальності.

Проблематика

Ключова модель, використовувана в сфері керування ризиками інформаційної безпеки (КРІБ), є процесна модель, що знайшла відображення в усіх стандартних підходах до КРІБ та являє собою основу стандартів ISO/IEC 27005 і BS 7799-3 [1]. Процесна модель дає перелік, послідовність та розкриває сутність таких необхідних для керування ризиками ІБ процесів, як планування, реалізація, перевірка, дія [2−5].

Базою для визначення рівня ризику майже в усіх методиках є ймовірність виникнення тієї чи іншої події, яка впливає на ймовірність реалізації загрози. У більшості методик визначення ймовірності здійснюється експертним методом або за основу береться статистика минулих періодів щодо таких самих подій.

Чи відповідає така методика реаліям, наскільки вона точна? По-перше, необхідно внести поправку на помилку експертів, по-друге, статистика минулих періодів не буде відповідати реальності, особливо у випадках швидкої зміни програмного та технічного забезпечення (вразливості якого ще невідомі), по-третє, існує більше факторів впливу на визначення ризику, ніж ймовірність реалізації загрози та сума збитків.

Пропоноване рішення полягає у застосуванні інструментів нечіткої логіки для розв'язання задач керування ризиком інформаційної безпеки [6−8].

Ціль керування ризиком – це визначення пріоритетів бюджетування напрямків зменшення ризику ІБ апарата прийняття рішень, а саме – зменшення або ліквідація вразливостей, загроз, можливих наслідків. Керування ризиком являє собою безперервне циклічне виконання певного переліку завдань. Окреслимо коло задач, вирішення яких дозволить досягнути поставленої цілі.

Задачі, що стоять перед менеджментом інформаційної безпеки

Перелік завдань може змінюватись залежно від того, на якому етапі та якісному рівні знаходиться процес керування ризиком ІБ в організації. Але навіть якщо деякі завдання вже виконані на якісному рівні, спеціаліст з безпеки має повернутися до них в наступному циклі керування ризиком ІБ.

1. Визначити профіль інформаційних ресурсів апарата прийняття рішень (АПР).

2. Визначити ролі суб'єктів АПР (у тому числі порушників).

3. Визначити шляхи пересування інформаційних ресурсів (ІР).

4. Ідентифікувати вразливості по кожному ІР.

5. Ідентифікувати загрози по кожному ІР.

6. Визначити фактори впливу (вразливості, загрози, наслідки) на величини ризику (вагомості) реалізації кожної загрози.

7. Визначити рівень ризику по кожній загрозі.

8. Розподілити річний бюджет витрат на ІБ та пріоритетність дій плану впровадження політики ІБ.

Розглянемо сутність цих задач і деякі способи їх розв'язання.

1. Визначити профіль ІР АПР.

Задача передбачає визначення переліку ІР АПР, опис кожного за стандартним профілем. Даний профіль може бути змінено відповідно до специфіки роботи організації.

Опис профілю ІР може бути проведений за такими ознаками:

- форма представлення:
  - дані в електронному вигляді,
  - дані в паперовому вигляді;
- статичність:
  - ресурс не переміщується (архіви),
  - ресурс переміщується;
- оригінальність ресурсу:
  - оригінал,
  - копія;
- місця появи ресурсу:
  - персональні пристрої,
  - сервери замкнутої внутрішньої мережі,
  - сервери зовнішньої мережі,
  - аналогові сховища;
- шляхи пересування:
  - всередині організації,
  - зовні організації;
- варіанти доступу до ресурсу:
  - повний внутрішній та зовнішній доступ,
  - обмежений внутрішній та зовнішній доступ,
  - доступ тільки внутрішньому персоналу,
  - обмежений доступ для внутрішнього персоналу;
- методи оцінювання:
  - критеріальний,
  - вартісний.

2. Визначити ролі суб'єктів АПР (у тому числі порушників).

Задача передбачає визначення переліку ролей суб'єктів АПР, побудову їх профілів щодо кожного ІР на базі рівнів доступу до ІР, побудову моделі порушника (мотиви, кваліфікація, рівень доступу).

На рис. 1 пропонується приклад моделі порушника.

3. Визначити шляхи пересування ІР.

Задача передбачає графічну побудову пересування ресурсів з урахуванням суб'єктів та технічних засобів АПР.

4. Ідентифікувати вразливості по кожному ІР.

При вирішенні задач 1−3, здійснюючи аналіз середовища ІБ, спеціаліст з

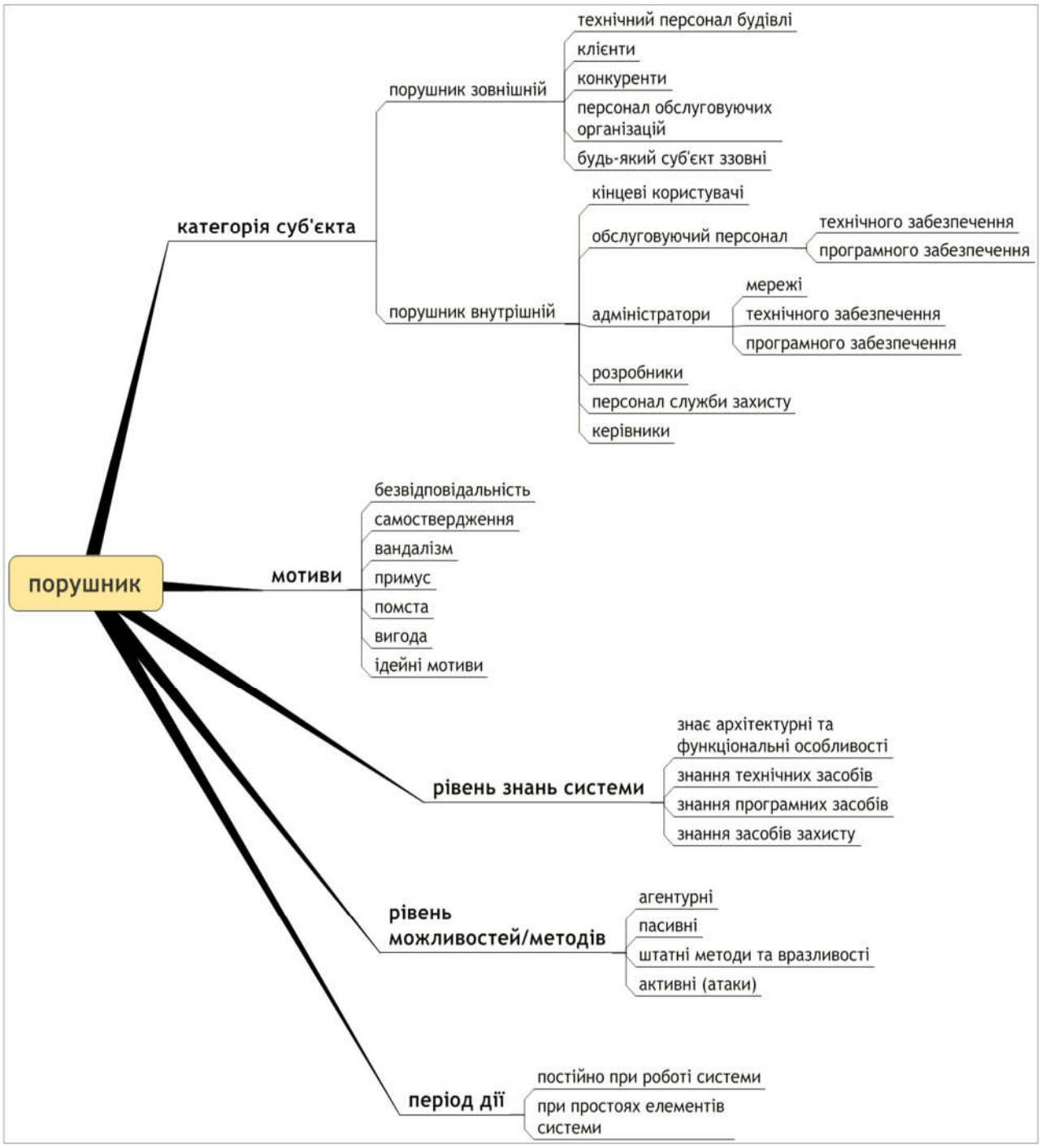


Рис. 1. Модель порушника

безпеки вже може виділяти слабкі місця організаційної структури, рівня обізнаності користувачів, мережевого обладнання, технічного забезпечення, програмного забезпечення. При виконанні завдання 4 визначені слабкі місця необхідно згрупувати у перелік вразливостей та співвіднести їх з кожним інформаційним ресурсом.

5. Ідентифікувати загрози по кожному ІР.

Виходячи з профілю ІР, ролей суб'єктів АПР, технічних засобів, необхідно визначити вразливості та загрози за правилом *вразливість - > загроза - >наслідок*.

Залежність величини ризику від рівня небезпеки вразливості, загрози та можливого наслідку можна переглянути на рис. 2 [8].

Наприклад:

Вразливість – відсутність внутрішнього документа «Політика інформаційної безпеки організації», що веде до виникнення іншої вразливості – недбале керування паролем. Загроза – втрата пароля. Наслідок – розкриття доступу до конфіденційної інформації.

Слід зазначити, що декілька вразливостей можуть впливати на рівень декількох загроз, або рівень небезпеки однієї вразливості може залежати від рівня безпеки декількох інших вразливостей.

6. Визначити фактори впливу на величини ризику реалізації кожної загрози.

Задача передбачає визначення факторів впливу на рівень ризику реалізації певної загрози та представлення факторів у контексті апарата нечіткої логіки.

Приклад визначення вразливостей та наслідків, що впливають на рівень ризику втрати доступу до робочих файлів, показано на рис. 3. Таке представлення можна також назвати деревом подій, що ведуть до виникнення загрози втрати інформаційного ресурсу.

Приклад переліку лінгвістичних змінних (βі), що впливають на величину ризику реалізації загрози втрати пароля до приватного ключа:

β1 – рівень кваліфікації персоналу, Х – процент співробітників з досвідом більше 5 років;

β2 – рівень ймовірності реалізації загрози, Х – ймовірність (або кількість інцидентів за останні 5 років) – може складатися з ймовірностей реалізації декількох подій;

β3 – рівень ймовірності реалізації найгіршого сценарію, Х – ймовірність (або – кількість інцидентів за останні 5 років);

β4 – рівень вартості контрзаходів, Х – вартість;

β5 – рівень критичності ресурсу, Х – можливий час роботи системи без ресурсу;

β6 – рівень втрати репутації, доступності, конфіденційності, цілісності;

β7 – час дії загрози, Х – шкала часу;

β8 – рівень наслідків дії загрози у вартісному представленні;

β9 – наявність та якість політики інформаційної безпеки організації.

Приклад представлення атрибутів лінгвістичної змінної, що описує рівень кваліфікації персоналу:

β1– рівень кваліфікації персоналу – лінгвістична змінна,

Х – універсальна множина – процент співробітників з досвідом більше 5 років,

Т – терми (значення лінгвістичної змінної): персонал досить кваліфікований, середньо, слабо кваліфікований,

α − нечітка змінна – <найменування нечіткої змінної> – персонал досить кваліфікований, область визначення – від 0 до 100 %, нечітка множина по даній змінній – від 70 % до 100 %.

Приклад представлення нечіткої множини по кваліфікації персоналу:

$S=\{x \mid x \in X \ \& \ M(x)>0\}$ [6], X – процент співробітників з досвідом роботи більше 5 років, x – значення кількості кваліфікованого персоналу в %, що описують нечітку змінну «персонал досить кваліфікований», $M(x)$ – ступінь належності x до нечіткого визначення «персонал досить кваліфікований».

У такий спосіб можна описати й інші вразливості.

Наприклад, розглянемо вразливість, що стосується низької якості політики інформаційної безпеки. Універсальна множина – рівень якості документа, що описує політику інформаційної безпеки, приймає значення на числовій множині від 0 до 10. Приклад термів: документ політики відсутній (нечітка множина від 0 до 2), документ є в наявності, але не досконалий (нечітка множина від 3 до 4),

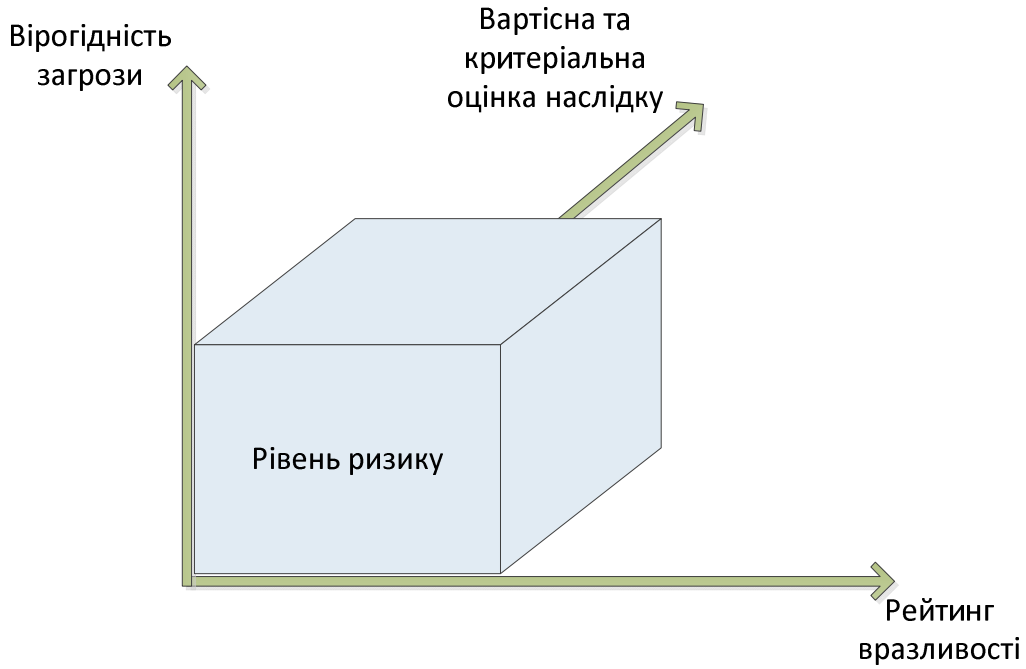


Рис. 2. Функція рівня ризику

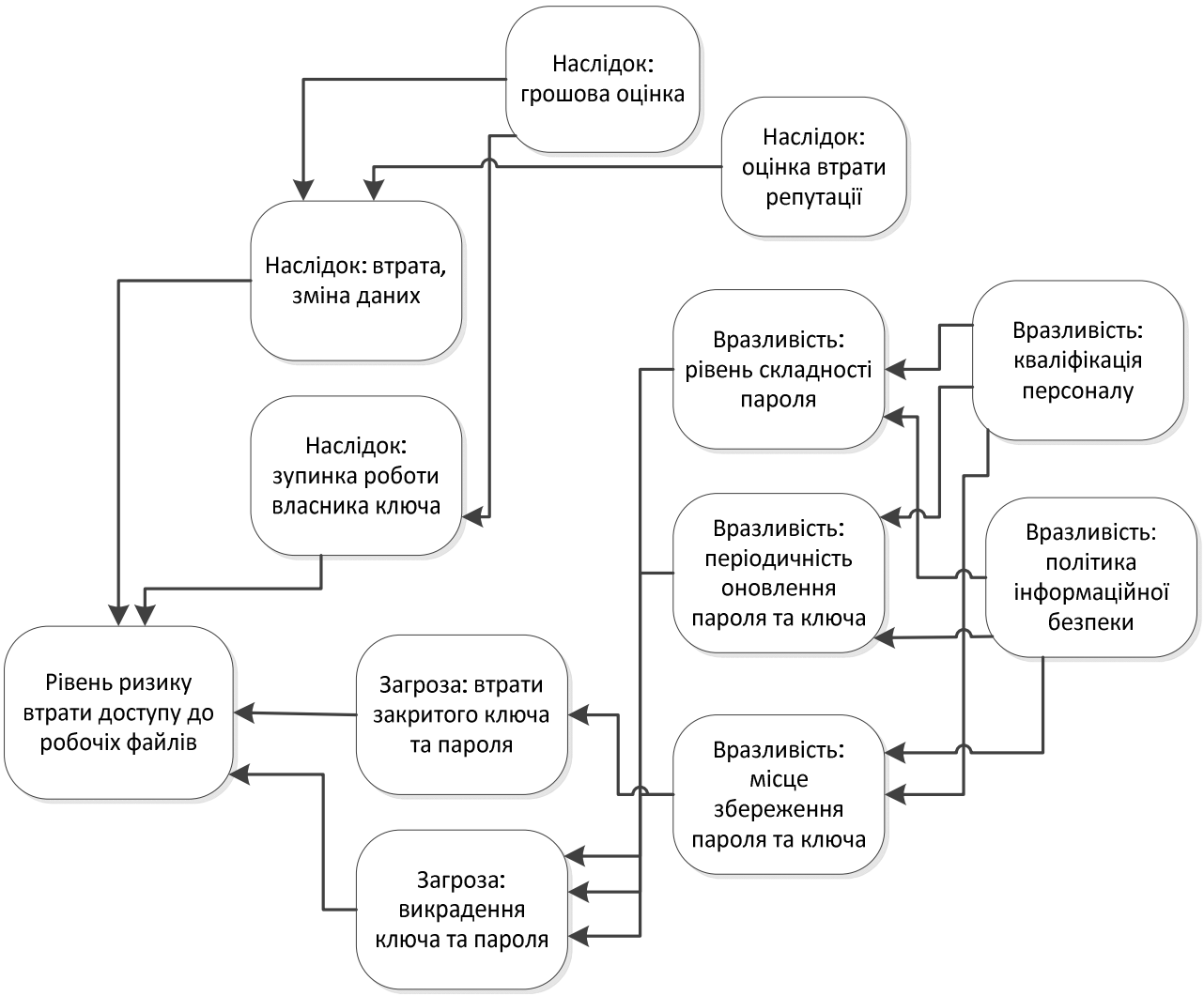


Рис. 3. Чинники виникнення ризику втрати доступу

документ досконалий, але не оновлюється (нечітка множина від 5 до 8), документ повний та оновлюється щороку (нечітка множина від 9 до 10).

7. Визначити рівень ризику по кожній загрозі.

Задача передбачає визначення бази нечітких правил, на вході яких йдуть умови впливу на рівень ризику (вразливості, загрози, наслідки), на виході – значення лінгвістичної змінної рівня ризику із заданої множини нечітких змінних.

Визначення рівня ризику втрати закритого ключа доступу до хмарного сервера показано рис. 4–6 (для даного прикладу використано інструменти MATLAB).

Обрані фактори впливу: вартість контрзаходів, кількість інцидентів минулого періоду, рівень кваліфікації персоналу.

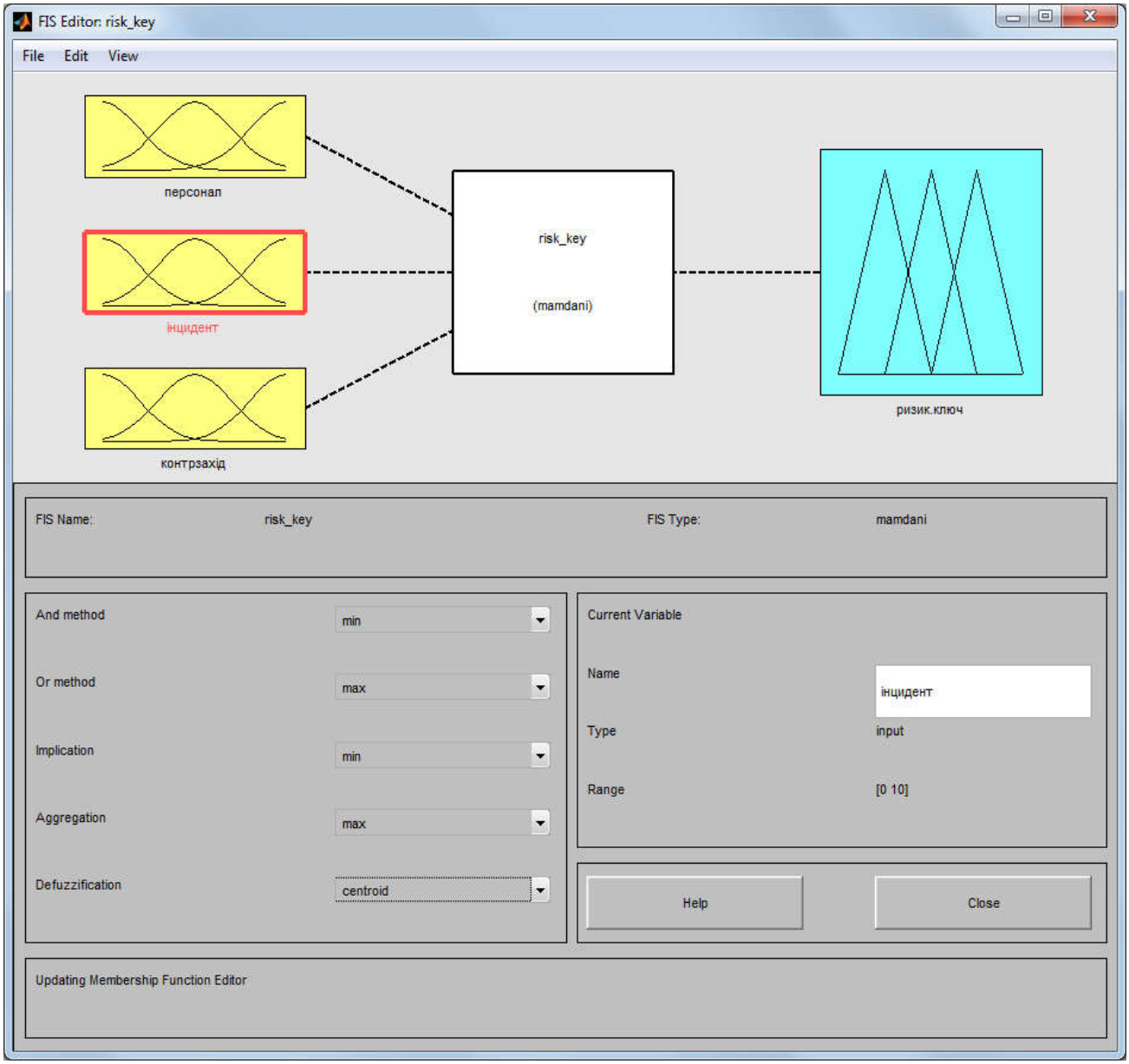


Рис. 4. Параметри вводу даних та виводу оцінки ризику втрати закритого ключа

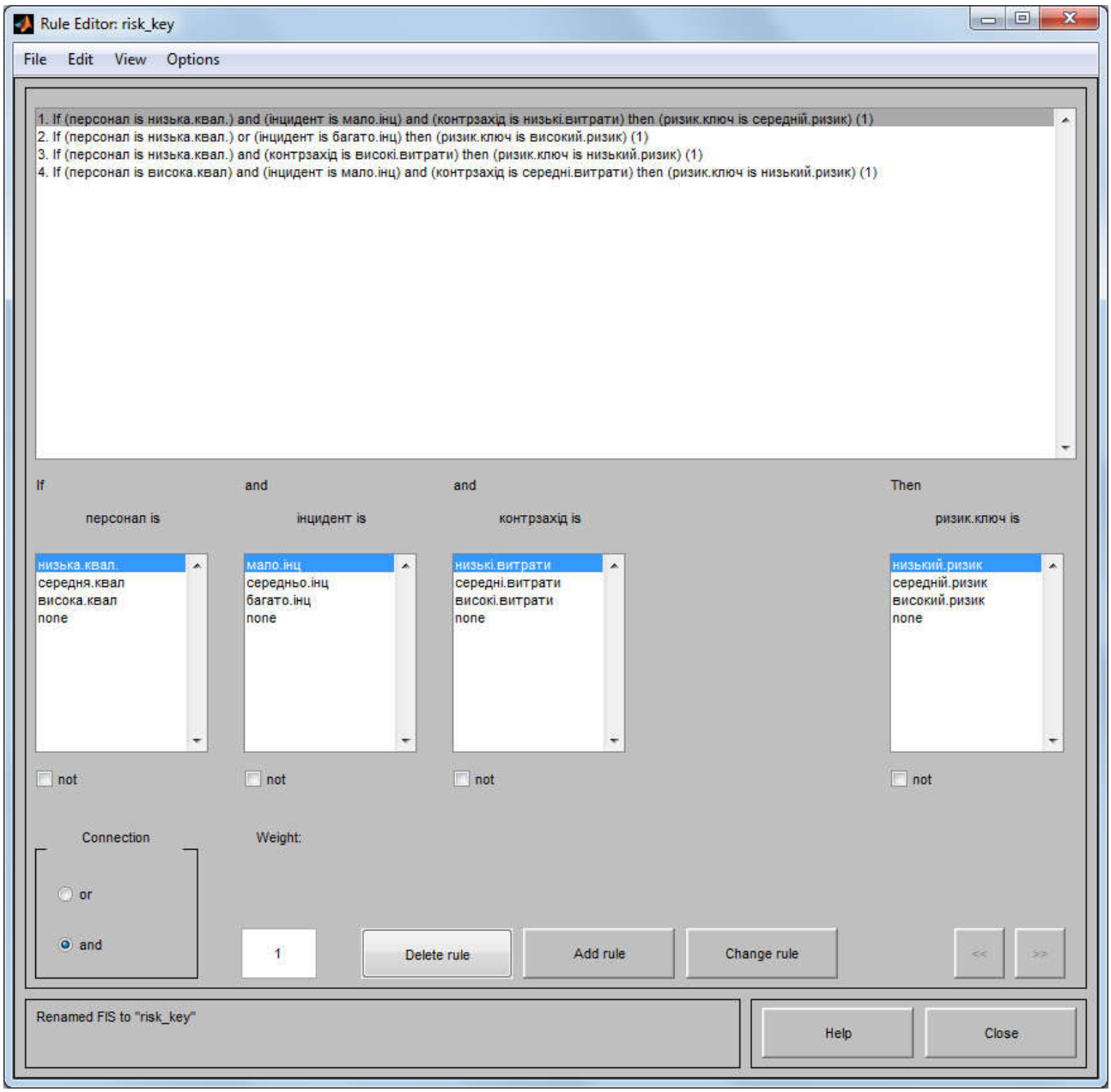


Рис. 5. Формування нечітких правил

8. Розподілити річний бюджет витрат на ІБ та пріоритетність дій плану впровадження політики ІБ.

У даному випадку статті бюджету можуть бути представленими за двома категоріями:

а) як резерв витрат на ліквідацію наслідків реалізації загрози;

б) як витрати на технічні засоби, програмні засоби, організаційні заходи, навчальні програми за напрямками ліквідації або мінімізації вразливостей та відповідних загроз.

Представимо бюджет:

$$B = \sum_{i=1}^{n} Bi,$$

B – загальний бюджет, Bi – стаття бюджету, що відповідає вразливості, загрозі,

$$Bi = B * \frac{Wi}{\sum_{i=1}^{n} Wi},$$

Wi – рівень ризику, пріоритет статті витрат на ІБ.

При розрахунку необхідно враховувати базову вартість статті, у випадку повного фінансування статті із залишком – решту слід переформувати на інші статті.

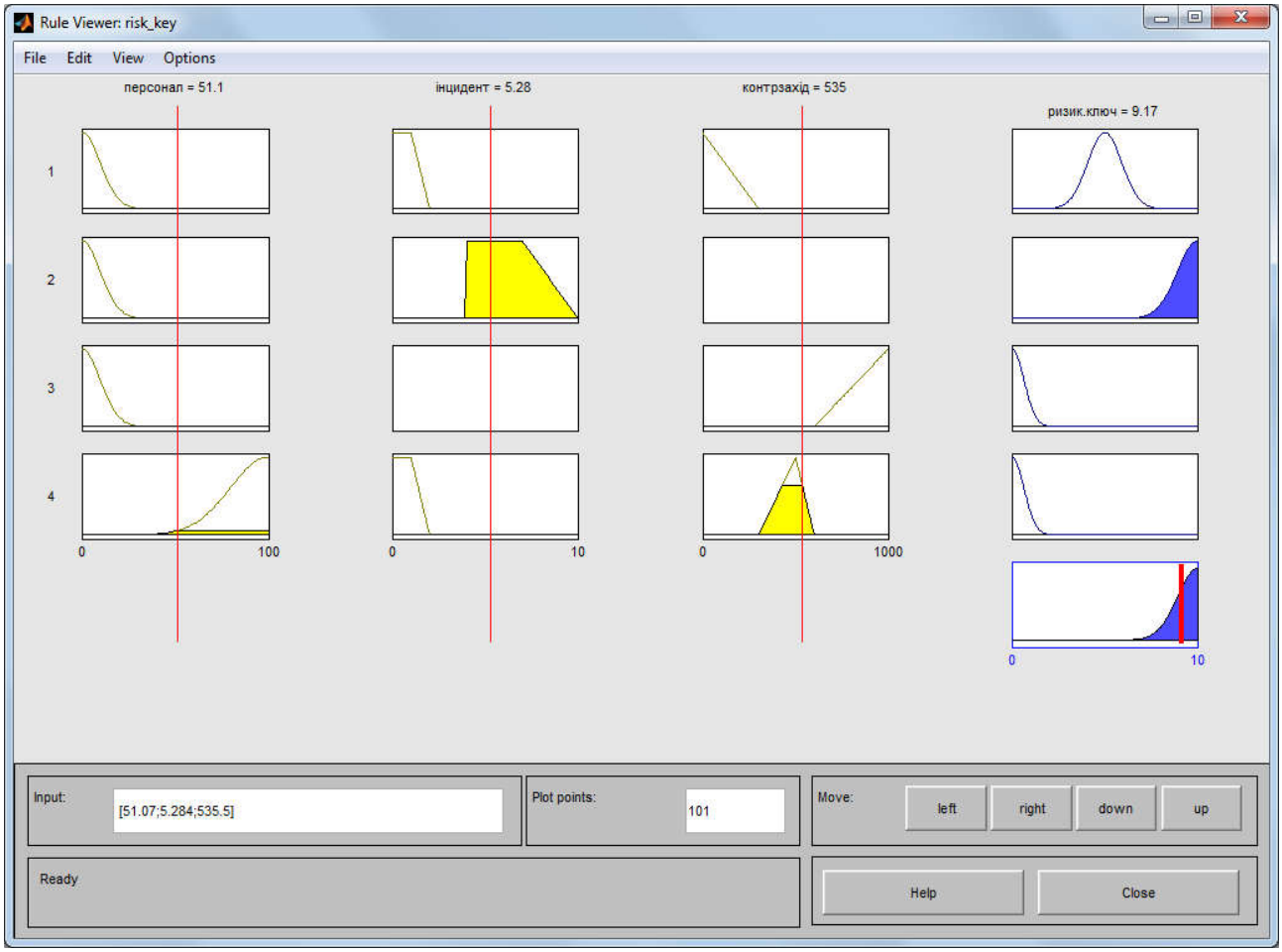


Рис. 6. Графічне зображення функцій належності та дефазифікація висновку

Висновки

Розв'язання запропонованих задач керування ризиком інформаційної безпеки передбачає використання комбінації процесних, експертних та математичних підходів. Задачі аналізу середовища, визначення вразливостей та загроз не потребують високої технічної кваліфікації. Виконання задач 1−3 дозволить майже паралельно створити документ політики інформаційної безпеки. Складнішим буде визначити дерево подій, що є наслідком, а що – причиною. Для збирання цієї інформації та подальшої побудови бази нечітких знань необхідно заохочувати експертів технічних відділів та аналізувати минулі періоди подій інформаційної безпеки. Апарат нечіткої логіки дозволяє перетворити будь-які зв'язки *причина* (*декілька причин*) – *наслідок* (*декілька наслідків*) у зручну математичну модель. Важливим питанням при використанні апарата нечіткої логіки є побудова функції належності. Правильно побудовану модель можна в майбутній життєдіяльності організації вдосконалювати та розширювати можливості її навчання.

1. International standard BS ISO/IEC 27005:2008, 2008-06-15.
2. Загородній А.Г., Боровська О.М., Свістунов С.Я., Сініцин І.П., Родін Є.С. Створення комплексної системи захисту інформаційних ресурсів у національній грід-інфраструктурі України. К.: Сталь, 2014. 373 с.
3. Родін Є.С. Процесні підходи до моделювання у сфері управління ризиками інформаційної безпеки. *Математичні машини і системи*. 2012. № 4. С. 142−148.
4. Боровська О.М., Сініцин І.П., Родін Є.С. Порівняння національного та міжнародного підходів побудови системи захисту інформації в грід. *Проблеми програмування*. 2011. № 5. С. 99−109.

5. Боровська О.М., Свістунов С.Я., Сініцин І.П., Шилін В.П., Родін Є.С. Підходи до створення комплексної системи захисту інформації в Національній грід-інфраструктурі. К., 2010. 51 с. (Препр. / НАНУ. Ін-т теоретичної фізики ім. Боголюбова М.М.).
6. Zadeh L.A. The concept of linguistic variable and its application to approximate reasoning. *Information sciences.* 1975. N 8. P. 199−249.
7. Малышев, Л.С. Берштейн, А.В. Боженюк. Нечеткие модели для экспертных систем в САПР. М.: Энергоатомиздат, 1991. 136 с.
8. Integrated Site Security for Grids. https://isseg-training.web.cern.ch/ISSeG-training/

References

1. International standard BS ISO/IEC 27005:2008, 2008-06-15.
2. Borovska, O., Sinitsyn, I., Svistunov, S., Rodin, Y. and Zagorodniy, A. (2014). Development of information resources security system in the national grid infrastructure of Ukraine. Kyiv: Stal, p.373. (In Ukrainian)
3. Rodin, Y. (2012). Processing approaches in the field of information security risk management modeling. Mathematical machines and systems, 4, P. 142-148. (In Ukrainian)
4. Borovska, O., Sinitsyn, I., and Rodin, Y. Comparing national and worldwide approaches in developing grid information security system. Programming Problems, 5, P. 99-109. (In Ukrainian)
5. Borovska, O., Sinitsyn, I., Svistunov, S., Rodin, Y. and Shilin, V. (2010). Approaches in developing information security system in the national grid infrastructure. Kyiv: Bogolyubov Institute for Theoretical Physics, p. 51. (In Ukrainian)
6. Zadeh, L. (1975). The concept of linguistic variable and its application to approximate reasoning. Information sciences, 8, pp. 199−249.
7. Bershtein, L., Bozhenyuk, A, Malyshev, L. (1991). Fuzzy modeling for experts systems in SAPR. Moscow: Energoatomizdat, p. 136. (In Russian)
8. Integrated Site Security for Grids. https://isseg-training.web.cern.ch/ISSeG-training/

Одержано 27.06.2017

***Про автора**:*

Родін Євген Сергійович,
молодший науковий співробітник.
Кількість наукових публікацій в українських виданнях – 5.
http://orcid.org/0000-0003-2416-8572.

Місце роботи автора:

Інститут програмних систем
НАН України,
м. Київ, 03187,
проспект Академіка Глушкова, 40, корп.5.
Тел.: 044 526 5507.
E-mail: yevheniy.s.rodin@gmail.com.

УДК 612.51.001.57+519.6

Р.Д. Григорян, А.Г. Дегода, Е.А. Джуринский, В.С. Харсун

СИМУЛЯТОР ПУЛЬСИРУЮЩЕГО СЕРДЦА

Для совместимых с IBM персональных компьютеров создан программный симулятор (ПС), основанный на математической модели (ММ) гемодинамики человека. ММ представляет сердечно-сосудистую систему (ССС) в сосредоточенных параметрах и при фиксированном нейрогуморальном фоне, соответствующем состоянию покоя. Назначение ПС – имитация пользователем сценариев перемен в исходном состоянии ССС для вычисления реакций гемодинамики. Есть два способа задания сценария: либо из списка, представленного в интерфейсе пользователя, либо формированием из комбинаций элементарных процедур. Планируется на базе ММ создать комплексную модель с учетом нейрогуморальной регуляции сердечного насоса, тонуса сосудов и объема крови. В данной версии ПС предсердия отсутствуют, сердечная деятельность моделируется описанием пульсаций желудочков. ПС демонстрирует основные закономерностей физиологии ССС в малом и большом кругах кровообращения. ПС позволяет визуализировать трансформации гемодинамики при дефектах сердечных клапанов, гипофункции правого/левого желудочков (из-за ухудшения коронарного кровотока, патологии кардиомиоцитов, гипертрофии миокарда). Исследовано влияние скорости и мощности сокращения правого или левого желудочков сердца на гемодинамику. После разработки комплексной модели на базе ПС будет создана информационная технология, ориентированная на кардиолога. Язык программирования – Java.

Ключевые слова: математическая модель, желудочки сердца, клапаны, кардиология, учебный процесс, информационная технология.

Введение

В списке причин смерти кардиологические патологии занимают одно из первых мест. Главными предшественниками дисфункции сердца и недостаточности кровообращения являются инфаркт миокарда (ИМ), фибрилляции желудочков, низкое сократительное усилие миокарда, аритмии, а также гипертрофия [1–3]. За исключением возникшего из-за внезапной закупорки венечных сосудов ИМ, перечисленные патологии развиваются как следствие постепенных изменений в биофизических характеристиках ряда узлов сердца. Стремясь компенсировать сердечную недостаточность, адаптационные механизмы перестраивают функционирование всей ССС. Поэтому успешная борьба с каждой из перечисленных патологий предполагает знания механизмов функционирования ССС. Хотя в этом деле существенную помощь оказали математические модели [4–8], они слишком схематично описывают пульсирующую деятельность сердца и гемодинамику в целом.

В частности, значительное число пациентов с проблемами нарушенной функции пейсмекеров синусного узла имеют имплантированный электростимулятор сердца. Если человек ведет активный образ жизни, желательно чтобы ритм электростимуляции управлялся автоматически. Прежде чем разработать адекватные алгоритмы автоматического управления сердечным ритмом в разных состояниях человека, требуются более фундаментальные физиологические знания о функционировании ССС [9]. Такие знания появились лишь недавно [10, 11]: стало понятно, что ССС – это часть физиологических суперсистем, совместно оптимизирующих жизнедеятельность клеток на фоне экзогенных (эндогенных) возмущений. Новое понимание организма требует переосмысления подходов к моделированию физиологических систем человека [12]. Поэтому после первой попытки моделирования пульсирующего сердца [13], мы вновь вернулись к данной проблеме с учетом концептуальных сдвигов [9–10].

Хотя пульсации моделируются посредством переменной эластичности камер сердца [4–8, 14–16], описания процесса электромеханического сопряжения и последующего развития сократительной деятельности миоцитов соответствующих камер сердца далеки от реальной кардиодинамики [17, 18]. Нередко авторы ограничивались лишь схематичным представ-

лением пульсаций с помощью функции синуса [7]. Между тем, изменения эластичности органа представляет собой популяционный эффект [10], т. е. складывается из темпоральной динамики состояния клеток разной пространственной локализации. Известно, что на ультраструктурном уровне клетки миокарда одной и той же популяции (например, левого желудочка) слегка различаются [11]. Различия порогов реагирования на пришедший возбуждающий электрический сигнал привязаны к пространственной локализации кардиомиоцита. Ранее нами была показана, что подобная неоднородность элементов популяции порождает асинхронность их реагирования на сигнал и детерминирует динамику популяционного эффекта [10, 13]. Таким образом, применение этих знаний позволит создать максимально адекватную модель пульсаций сердца. Ее интегрирование в модель гемодинамики большого и малого кругов кровообращения с учетом нервных и гуморальных механизмов регуляции функции сердца, тонуса сосудов, а также общего объема крови станет основой для разработки качественно нового компьютерного симулятора, ориентированного на актуальные задачи кардиологии.

Цель статьи – описать базовую математическую модель гемодинамики человека с учетом пульсаций сердца.

Основные требования к модели: допущения и ограничения

Описываемая в статье модель ориентирована на симуляции широкого круга сценариев развития кардиологических проблем и является модификацией опубликованной ранее модели [13]. Являясь первым шагом на пути создания специализированного моделирующего комплекса, новая разработка имеет два назначения. Во-первых, моделируя и визуализируя пульсации сердца и гемодинамики в обоих кругах кровообращения, симулятор должен способствовать повышению эффективности усвоения физиологических и патофизиологических закономерностей студентами-медиками. Во-вторых, специальная версия симулятора, ориентированная на кардиолога-исследователя, должна способствовать внедрению современных информационных технологий в процесс постановки диагноза и выбора терапии кардиологом.

На первом шаге было необходимо создать базовую модель объекта регулирования (МОР) для ее дальнейшей интеграции в модель физиологической регуляции (МФР) ССС посредством комплекса нейрогуморальных механизмов. В МОР должны быть включены модели сердца, малого и большого кругов кровообращения.

Опыт моделирования показал, что при имитации пульсирующей гемодинамики шаг интегрирования должен быть маленьким, что приводит к большому количеству вычислений. Мы ориентируемся на IBM-совместимый персональный компьютер с ограниченными вычислительными ресурсами. Это не позволяет симулировать длительные процессы в ССС в режиме пульсирующей гемодинамики. Наша цель заключается в разработке такой модели, которая позволит исследовать и срочные рефлекторные реакции ССС на внешние перемены, и долговременные адаптивные перенастройки нервных и гормональных ее регуляторов. Исходя из этих требований, планируется создать три версии модели сердца. Одна модель для имитации желудочковых пульсаций. Вторая – желудочковых и предсердных пульсаций. Третья – для описания медленной динамики средних значений давлений, объемов и потоков крови в сердце и сосудах. Во всех моделях ССС представляется в сосредоточенных параметрах.

При исследованиях длительных процессов в ССС последняя модель будет использоваться как основная. Шаг интегрирования в этой модели на два порядка больше того шага, который требуется для детального воспроизведения быстрых изменений внутрисердечной гемодинамики. Для контроля сердечной гемодинамики эпизодически (при необходимости) будут активизироваться модели пульсирующего сердца. Такая стратегия позволит существенно экономить время без потери качества симуляций.

Модель сердца должна имитировать не только физиологическую норму, но и основные гемодинамические эффекты, возникающие при: а) дефектах клапанов; б) нарушениях ритма (экстрасистолы, фибрилляции предсердий или желудочков); в) изменениях характера распределения пространственно-временной не-

однородности кардиомиоцитов; г) ишемического ослабления инотропного состояния желудочков; д) сочетанных изменениях конечно-диастолической эластичности миокарда (правого и/или левого желудочков) и их конечно-диастолического объема (имитация сердечной гипертрофии). Планируется также исследование причинно-следственных отношений между гемодинамикой и энергетикой кардиомиоцитов.

Основные допущения модели пульсирующего сердца сводятся к следующим:

- длительность сердечного цикла T_{hc} ($T_{hc} = 1/F$, где F – частота сокращений сердца) принимаем постоянной;
- пульсации предсердий не моделируем, а их объем включаем в объем полых (для правого предсердия) или легочных вен (для правого предсердия);
- принимаем длительности фаз диастолы (T_d) и систолы (T_s) константами;
- общий объем крови в ССС (V_T) постоянен.

Основные ограничения моделирования обусловлены отсутствием региональных механизмов саморегуляции. Но в контексте цели и задач данной разработки, эти ограничения несущественны.

Структура ССС в базовой версии МОР показана на рис. 1.

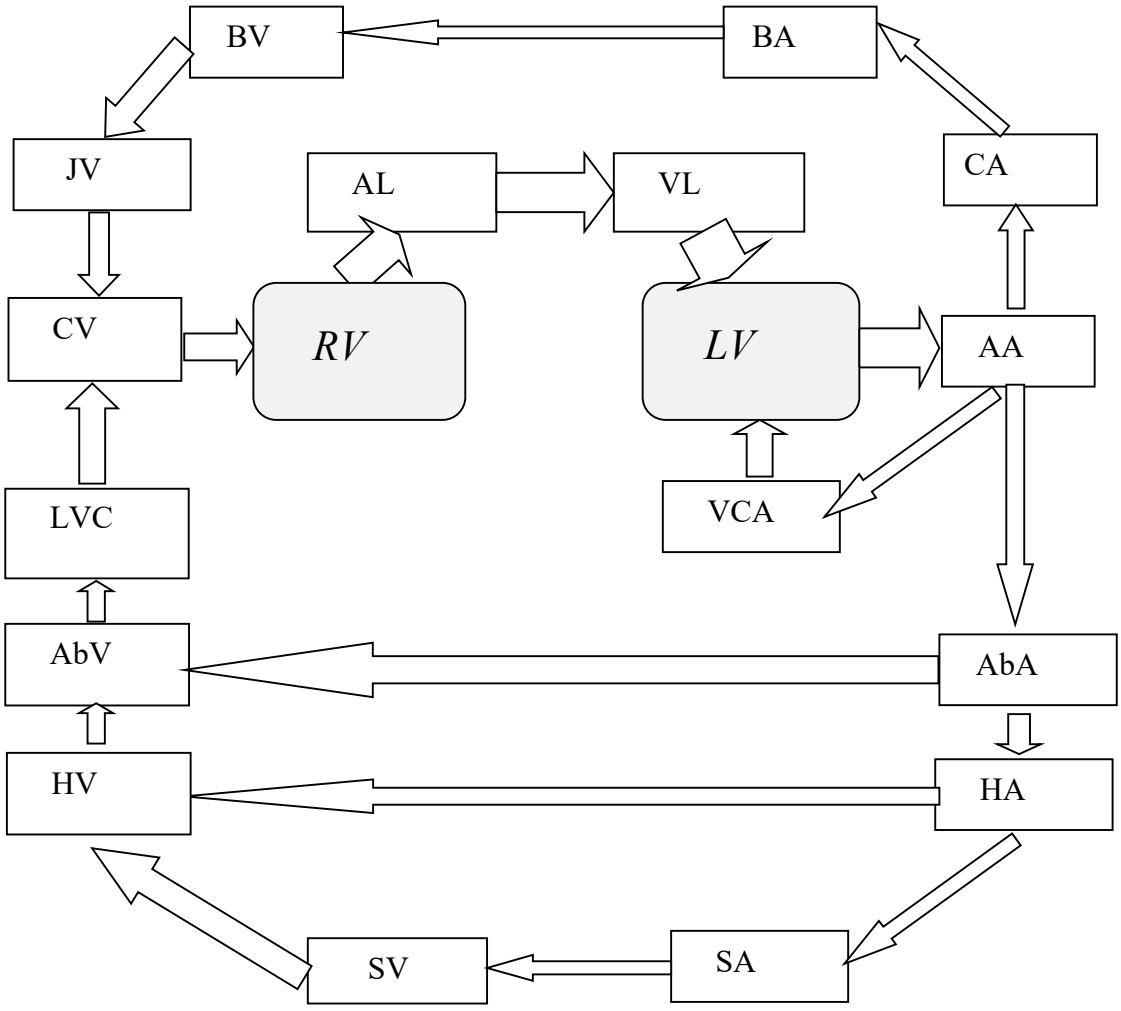


Рис. 1. Схема представления ССС в модели

RV – правый желудочек, *LV* – левый желудочек, AL – легочная артерия, VL – легочная вена, AA – дуга аорты, CA – каротидная артерия, BA – артерии головы, BV – вены головы, JV – яремная вена, CV – центральная вена; LVC – нижняя полая вена, AbA – абдоминальная артерия, AbV – абдоминальная вена, HV – бедренная вена, HA – бедренная артерия, SA – артерия голеней, SV – вена голеней, VCA – виртуальный участок коронарных артерий

Математическое описание гемодинамики в сосудах

Во всех участках сосудов для каждого момента времени (t) давление $(P_i(t))$ рассчитывается на основе жесткости $(D_i(t))$, объема $(V_i(t))$ и ненапряженного объема $(U_i(t))$ как:

$$P_i(t) = (V_i(t) - U_i(t)) \cdot D_i(t).$$

Поток крови $q_{ij}(t)$ из участка i в участок j рассчитывается как:

$$q_{ij}(t) = (P_i(t) - P_j(t)) / r_{ij}(t),$$

где r_{ij} – гидравлическое сопротивление соединения этих участков. Упрощенно зависимость r_{ij} от поперечного сечения сосуда i–го участка вычисляем по формуле:

$$r_{ij}(t) = \begin{cases} r0_{ij}, V_i(t) \geq V0_i \\ r0_{ij} \cdot (V0_i / V_i(t))^2, V_i(t) < V0_i \end{cases}.$$

где $V0_i = V_i(0)$, а $r0_{ij}$ и $V0_i$ – представляют сопротивление и кровенаполнение i-го участка в состоянии покоя (физиологической нормы).

Изменения кровенаполнения i-го участка за время h вычисляем как:

$$V_i(t) = V_i(t-h) + \int_{t-h}^{t} (q_{ij}(t) - q_{ij}(t-h))dt.$$

Математическое описание пульсирующего сердца

Выше приведенные уравнения модифицированы для расчета давлений, потоков крови и кровенаполнений в камерах (желудочках) сердца. Модификация касается трех переменных. Во-первых, внутри сердечного цикла величины жесткости, сопротивлений входного и выходного клапанов перемены. Во-вторых, жесткий перикард ограничивает рост объема желудочков в фазе диастолы. В-третьих, уравнение для описания объемной скорости крови из желудочков переписано в дифференциальной форме с учетом инерционности потока.

Для формального описания циклических изменений характеристик камер сердца введено понятие «время внутри сердечного цикла» – t_h. Независимо от длительности (T_{hc}) сердечного цикла, t_h равномерно (с заданным шагом h) увеличивается до достижения T_{hc}, после чего ход t_h вновь начинается с нуля:

$$t_h = \begin{cases} t_h + h, & t_h \leq T_{hc} \\ 0, & t_h > T_{hc} \end{cases}.$$

T_{hc} разделена на длительности систолы (T_s) и диастолы $T_d = T_{hc} - T_s$. Жесткость желудочков $D_k(t_h)$ периодически изменяется от минимального значения в конце диастолы до максимального – в конце систолы:

$$D_k(t_h) =$$

$$= \begin{cases} D_{edk} + A_k \cdot (1 - e^{-\alpha_k \cdot t_h})^{m_k}, 0 < t_h \leq T_s \\ D_{edk} + C_k \cdot e^{\beta_k \cdot (T_s - t_h)}), T_s < t_h \leq T_{hc} \end{cases},$$

$$C_k = A_k \cdot (1 - e^{-\alpha_k \cdot t_h})^{m_k}, \; k = \overline{1,2}. \quad (1)$$

В формуле (1) D_{edk} – жесткость k-го желудочка, константы α_k и β_k определяют скорости систолического увеличения и диастолического снижения жесткости миокарда правого и левого желудочков, а изменением константы m – имитируется специфический эффект, обусловленный асинхронностью сокращения кардиомиоцитов данного желудочка. Константа A_k позволяет имитировать различия максимальных величин D_k правого $(k = 1)$ и левого $(k = 2)$ желудочков.

Формула для расчета потока крови из правого (левого) желудочка учитывает инерционность потока, т. е.:

$$T_k \frac{dq_k(t)}{dt} = \Delta P_{Ok}(t) / r_{Ok}(t) - q_k(t),$$

где

$$\Delta P_{Ok}(t) = \begin{cases} P(t)_k - P_{AL}(t), k = 1 \\ P(t)_k - P_{AA}(t), k = 2 \end{cases}$$

представляет разницу (градиент) давлений внутри желудочка и в сосуде (легочная артерия или дуга аорты), T_k – постоянная времени переходного процесса, а r_{Ok} – сопротивление выходного клапана правого или левого желудочка рассчитывается в соответствии с уравнениями:

$$r_{Ok}(t) =$$

$$= \begin{cases} r_{Ok}^{\max} \cdot e^{-\gamma_{Ok} \cdot \Delta P_{Ok}} + r_{Ok}^{\min}, & \Delta P_{Ok}(t) > 0 \\ r_{Ok}^{\max} \cdot (1 - e^{-\eta_{Ok} \cdot \Delta P_{Ok}}) + r_{Ok}^{\min}, & \Delta P_{Ok}(t) \le 0 \end{cases},$$

Сопротивления же входного клапана ($r_{Ik}(t)$) каждого желудочка определяется в зависимости от значения градиента давлений $\Delta P_{Ik}(t)$ на обе стороны клапана:

$$\Delta P_{Ik}(t) = \begin{cases} P_{CV}(t) - P_k(t), k = 1 \\ P_{VL}(t) - P_k(t), k = 2 \end{cases},$$

$$r_{Ik}(t) =$$

$$= \begin{cases} r_{Ik}^{\max} \cdot e^{-\chi_{Ik} \cdot \Delta P_{Ik}(t)} + r_{Ik}^{\min}, & \Delta P_{Ik}(t) > 0 \\ r_{Ik}^{\max} \cdot (1 - e^{-\delta_{Ik} \cdot \Delta P_{Ik}(t)}) + r_{Ik}^{\min}, & \Delta P_{Ik}(t) \le 0 \end{cases}.$$

Константы (χ_{Ik}, δ_{Ik}) задают инерцию процесса открывания входных клапанов, а константы (γ_{Ok}, η_{Ok}) – инерцию закрытия выходных клапанов желудочков сердца.

Ограничительная роль перикарда на объем желудочков учтена посредством введения нелинейного члена в формуле определения давления $P_{Ik}(t)$:

$$P_k(t) = \begin{cases} (V_k(t) - U_k(t)) \cdot D_k(t), V_k(t) \le V1_k \\ (V_k(t) - U_k(t)) \cdot D_k(t) + \\ + \lambda \cdot (V_k(t) - V1_k)^2, V_k(t) > V1_k \end{cases},$$

где λ и $V1_k$ – константы аппроксимации.

Для симуляции типичных гемодинамических эффектов, связанных с ухудшением коронарного кровотока, в модели введен виртуальный коронарный участок. Виртуальность состоит в том, что за динамикой объема крови в этом участке не следим, а контролируем лишь средний за n-й кардиоцикл кровоток ($q_{kc}(n)$). Он вычисляется с помощью средних за этот период значений давления в дуге аорты ($P_{AA}(n)$) и сопротивления коронарных сосудов ($q_{kc}(n)$):

$$P_{AA}(n) = \frac{1}{T_{hc}(n)} \int_0^{T_{hc}(n)} q_2(t) dt,$$

$$q_{kc}(n) = P_{AA}(n) / r_{kc}(n),$$

$$r_{kc}(n) = \frac{1}{T_{hc}(n)} \int_0^{T_{hc}(n)} r_{O2}(t) dt.$$

Значение $q_{kc}(n)$ используется для перерасчета A_k в формуле (*):

$$A_k = A0_k \cdot \Delta q_{kc}(n) \cdot e^{-\mu \cdot n},$$

$$\Delta q_{kc}(n) = \sum_n q0_{kc}(n) - q_{kc}(n),$$

где $q0_{kc}(n)$ представляет нормальное значение $q_{kc}(T_{hc})$, $A0_k$ – нормальное значение A_k, а μ – константа аппроксимации скорости ухудшения сократительной способности миокарда при скачкообразном падении коронарного кровотока.

Итак, при неизменных величинах нейрогуморальных воздействий на ССС,

система выше приведенных уравнений представляет собой математическую модель гемодинамики человека с учетом пульсаций желудочков сердца. В дальнейшем эта система будет представлять МОР. Для его превращения в компьютерный симулятор была написана специальная программа на языке Java (openjdk-8).

Сведения о программе

Программа использует ряд сторонних модулей, среди них: JFreeChart (для отображения вычислений на график), JDBI (для упрощенной работы с базой данных) и другие, менее значительные модули.

Архитектурной моделью программы является так называемая трёхуровневая архитектура, которая дифференцирует программу на 3 слоя: пользовательский интерфейс, бизнес логика (вычисление и модель ССС человека) и слой доступа к данным.

Данные вычислений и модель ССС хранятся во встроенной базе данных. Доступ к данным осуществляется с помощью программного драйвера SQLite, который обеспечивает максимально эффективную работу с данными.

ССС человека расположена в бизнес слое программы, состоит из участков и соединений между ними. Участки представляют собой контрольные точки в системе, которые обладают некоторыми свойствами, например, давление, объем, жёсткость и т. д. Примерами участков являются: лёгочная артерия, дуга аорты, брюшная вена и т. д. Среди всех участков особо системы выделены левый и правый желудочки сердца. Они отличаются тем, что логика вычислений их характеристик может меняться в зависимости от различных факторов, например, времени внутри кардиоцикла.

Уравнения модели решаются численным методом с помощью метода Эйлера. Шаг интегрирования уравнений и начальные значения физиологических данных для участков и соединений можно установить через пользовательский интерфейс. Его вид, ориентированный на физиолога, показан на рис. 2.

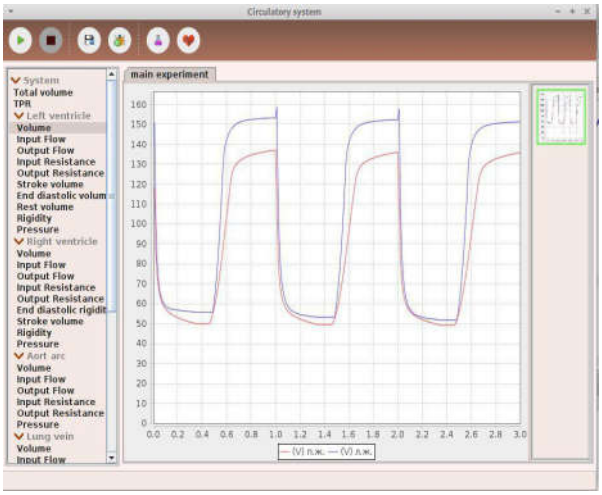


Рис. 2. Интерфейс пользователя

Тестовые результаты симуляций

Прежде чем рассмотреть результаты тестовых симуляций отметим, что общий объем крови ($V_s = 5300\ cm^3$) и численные значения параметров модели ССС соответствуют здоровому мужчине массой тела 70 кг. В данной версии модели частота сокращений сердца равна $F = 60\ \text{min}^{-1}$ (длительность сердечного цикла $T_{hc} = 1\ s$, $T_s = 0.35\ s$, $T_d = 0.65\ s$).

Целью тестовых исследований была настройка значений констант МОР так, чтобы: а) был достигнут установившийся режим гемодинамики; б) колебания центрального венозного давления (P_{CV}), легочного артериального (P_{AL}) и венозного (P_{VL}) давлений, давления в дуге аорты P_{AA}, объемов правого (V_{RV}) и левого (V_{LV}) желудочков сердца находились в пределах нормы. Эти характеристики центральной гемодинамики в модели показаны на рис. 3–5.

Исходное распределение общего объема крови по участкам сосудов и сердца задавалось приблизительным образом. Так как модель ССС гемодинамически замкнута, после старта вычислений с

любых случайных начальных условий через определенное время, зависящее от разброса стартовых данных объемов крови в участках, должен наступать установившийся режим гемодинамики. Как видно из рис. 3, давления в аорте и в левом желудочке достигают установившихся циклических колебаний примерно к 15-й секунде реального времени.

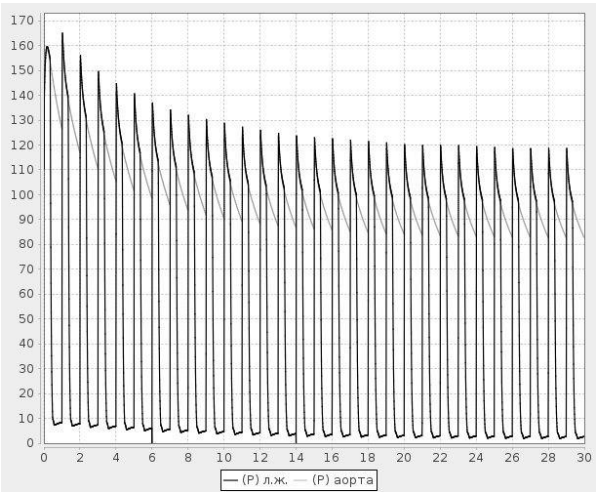
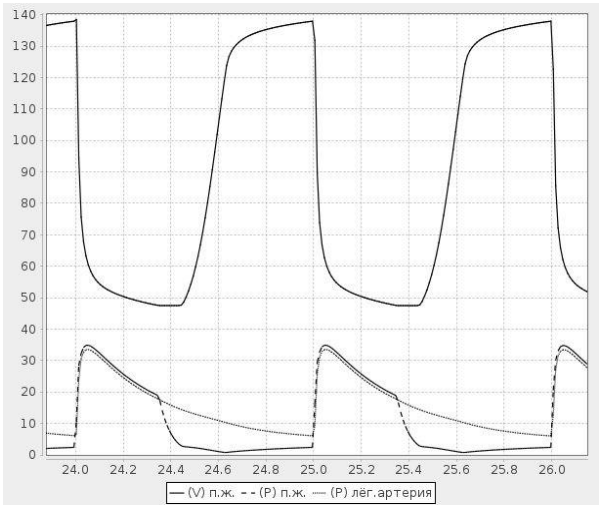


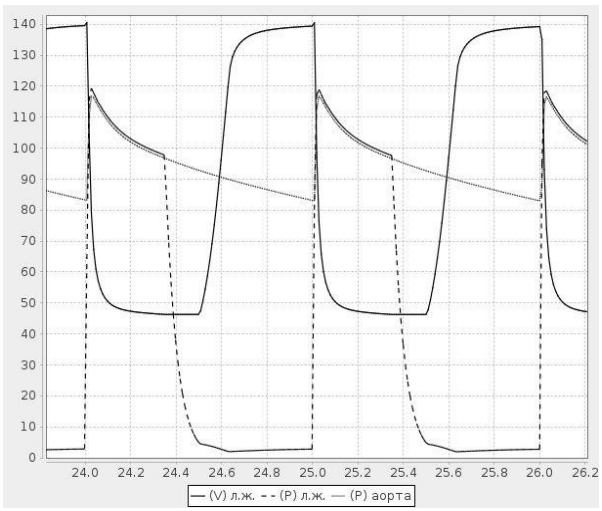
Рис. 3. Симуляция циклических колебаний давлений в левом желудочке ($P_{л.ж.}$) и аорте ($P_{аорта}$)

После того, как был достигнут режим физиологической нормы ($P_{аорта}$=118/82 мм рт.ст.), были симулированы отдельные динамические характеристики внутри кардиоцикла. Примеры показаны на рис. 4.

Мы не будем иллюстрировать весь спектр результатов симуляций. Пожалуй, на один аспект исследований, отличающий нашу модель от всех известных, следует специально остановиться. Речь идет об исследовании гемодинамических эффектов, вызываемых изменениями численного значения параметра m в формуле (1). Напомним, что этот параметр определяет форму кривой роста объемной жесткости миокарда в фазу систолы. Также заметим, что единственным способом количественной оценки этого свойства желудочка пока остается определение максимального значения скорости нарастания систолического давления dP/dt в желудочке. В нашей



а



б

Рис. 4. Симуляция кардиоциклических колебаний центральной гемодинамики:

а – давление в легочной артерии, объем и давление в правом желудочке;

б – давление в аорте, объем и давление в левом желудочке

модели параметр m имитирует степень синхронизации электромеханического сопряжения миоцитов данного желудочка. Иначе говоря, изменение численного значения m ведет к изменению скорости нарастания жесткости миокарда. Рис. 5 ниже иллюстрирует характер связи между m и динамикой систолического нарастания жесткости миокарда $D(t)$ левого желудочка сердца при трех значениях $m = 1.25; 5.0; u\, 10.0$.

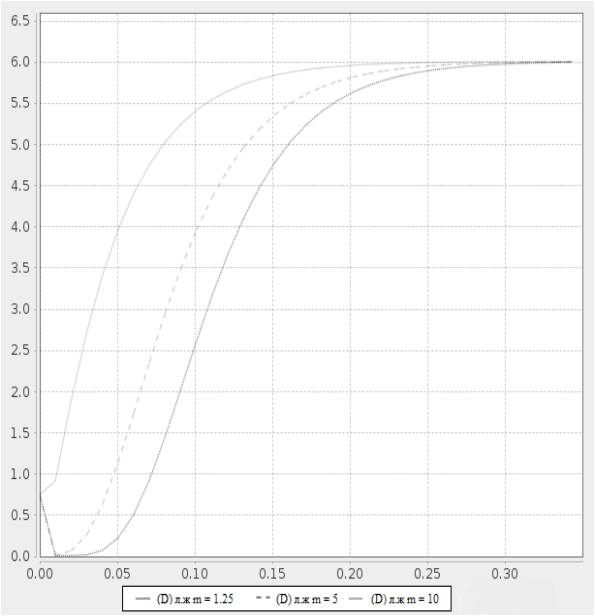


Рис. 5. Симуляция влияния параметра m в формуле (1) на динамику систолического роста жесткости миокарда левого желудочка сердца

Обсуждение результатов моделирования

Диагностические возможности современной скорой помощи ограничены фрагментарными данными, извлекаемыми из приборов, которыми оснащена мобильная группа врачей. При экспресс-диагностике ишемической болезни сердца и инфаркта миокарда трансформации электрокардиограммы косвенно указывают на метаболические неурядицы, вызванные нарушениями гемодинамики локальных зон миокарда. Но без специальных инвазивных исследований в условиях кардиологической клиники доктору очень трудно понять реальную картину системной и кардиальной гемодинамики. В причинно-следственных отношениях между гемодинамикой, метаболизмом и энергетикой кардиомиоцитов, с одной стороны, и состоянием сердца, его клапанов и гетерогенностями кардиомиоцитов, с другой стороны есть неочевидные закономерности. Проблема в том, что методы эмпирической физиологии ССС позволяют исследовать лишь часть этих закономерностей. В помощь эмпирике предлагались математические модели, но лишь единицы из них описывают естественные пульсации сердца [15, 18, 20].

Наш симулятор пока охватывает лишь часть патологического спектра кардиодинамики. К тому же в статье иллюстрирована лишь незначительная часть реальных возможностей симулятора. Но дополнительное тестирование базовой версии модели и симулятора убедило нас в том, что она обладает достаточной новизной и потенциалом для существенного расширения и углубления исследовательских возможностей кардиолога. Наиболее существенным считаем установление причинной связи между dP/dt желудочка и параметрами (m_k, D_k, α_k и β_k). Фактически, модель впервые используется для диагностики ослабления сократительной функции миокарда из-за локальных нарушений его кровоснабжения. Возможно, дальнейшее исследование этой проблемы с учетом ее энергетического аспекта позволит рекомендовать специфические методы профилактики и ранней диагностики доклинических проявлений дисфункции миокарда.

Симулятор предоставляет новые возможности для исследования гемодинамических эффектов клапанных дефектов, аритмий, а также межжелудочковых дефектов (при определенных дополнениях в математическом описании модели). Результаты симуляций, не вошедших в настоящую статью, станут предметом отдельной публикации для специалистов (физиологов и кардиологов).

Заключение

Создана базовая математическая модель гемодинамики здорового человека при неизменном нейрогуморальном фоне. Модель учитывает пульсации желудочков сердца. На основе модели создана компьютерная программа на языке Java, представляющая специализированный симулятор. Он показал адекватные реакции основных характеристик кардиальной гемодинамики при некоторых вариациях значений принятых констант. Это позволяет считать модель как базовую для

включения в разрабатываемую МФР в качестве модели объекта регулирования – МОР. В МФР T_{hc}, T_s и T_d станут изменяться от цикла к циклу. Переменными станут также другие характеристики сердца ($m_k, D_k, D_{edk}, \alpha_k$ и β_k) и сосудов (D_i, U_i). Такая модель и основанная на ней компьютерная программа станут удобной информационной технологией, ориентированной на повышение эффективности кардиологических исследований, а также средством визуализации причинно-следственных зависимостей в процессе подготовки будущих кардиологов.

1. Lopshire J.C., Zipes D.P. Device therapy to modulate the autonomic nervous system to treat heart failure. Curr Cardiol Rep. 2012. 14(5). P. 593–600.
2. Chatterjee N.A., Singh J.P. Novel Interventional Therapies to Modulate the Autonomic Tone in Heart Failure. JACC Heart Fail. 2015. 3(10). P. 786–802.
3. Schwartz P.J., La Rovere M.T., De Ferrari G.M., Mann D.L. Circ Heart Fail. Autonomic modulation for the management of patients with chronic heart failure. 2015; 8(3). P. 619–628.
4. Shen M.J., Zipes D.P. Interventional and device-based autonomic modulation in heart failure. Heart Fail Clin. 2015. 11(2). P. 337–348.
5. Fresiello L., Ferrari G., Di Molfetta A., Zieliński K., Tzallas A., Jacobs S., et al. A cardiovascular simulator tailored for training and clinical uses. J Biomed Inform 2015. 57. P. 100–112.
6. Burkhoff D., Tyberg J.V. Why does pulmonary venous pressure rise after onset of LV dysfunction: a theoretical analysis. Am J Physiol 1993. 265. P. 1819–1828.
7. Smith B.W., Chase J.G., Shaw G.M., Nokes R.I. Experimentally verified minimal cardiovascular system model for rapid diagnostic assistance. Control Engineering Practice. 2005. 13. P. 1183–1193.
8. Paeme K., Moorhead K.T., Chase J.G., Lambermont B., Kolh P., D'orio V., Pierard L., Moonen M., Lancellotti P., Dauby P. C., Desaive T. Mathematical multi-scale model of the cardiovascular system including mitral valve dynamics. Application to ischemic mitral insufficiency. BioMedical Engineering OnLine 2011, 10: 86.
9. Acton A. Advances in Heart Research and Application: 2013 Scolarly Edition, Atlanta (Georgia), USA: 424 p.
10. Grygoryan R.D. The Energy Basis of Reversible Adaptation. New York, USA: Nova Science, 2012. 253 p.
11. Grygoryan R.D. The Optimal Circulation: Cells' Contribution to Arterial Pressure. New York, USA: Nova Science, 2017. 287 p.
12. Grygoryan R.D., Sagach V.F. The concept of physiological supersystems: a new stage of integrative physiology. Fiziol. Zh. 2017. 63(3): P. 58–67.
13. Grygoryan R.D., Lissov P.N. A software simulator of human cardiovascular system based on its mathematical model. Problems in programming. 2004. N 4. P. 100–111.
14. Bers D.M. Excitation-Contraction Coupling and Cardiac Contractile Force. Kluwer Academic Publishers, 2002. 452 p.
15. Mynard J.P., Davidson M.R., Penny D.J., Smolich J.J. A simple, versatile valve model for use in lumped parameter and one-dimensional cardiovascular models. Int J Number Methods Biomed Eng. 2012. 28. P. 626–641.
16. Avolio A.P. Multi-branched model of the human arterial system. Med Biol Eng Comput. 1980. 18. P. 709–718.
17. Heldt T., Mukkamala R., Moody G.B., Mark R.G. CVSim: an open-source cardiovascular simulator for teaching and research. Open Pacing Electrophysiol Ther J. 2010. 3. P. 45–54.
18. van Meurs W. Modeling and simulation in biomedical engineering: applications in cardiorespiratory physiology. 1st ed. McGraw-Hill Education; 2011.
19. Mateják M, Kulhánek T, Šilar J, Privitzer P, Ježek F, Kofránek J. Physiolibrary – Modelica library for physiology. 10th International Modelica conference; 2014.
20. Hann C.E., Chase J.G., Shaw G.M. Efficient implementation of non-linear valve law and ventricular interaction dynamics in the minimal cardiac model. Comput Methods Programs Biomed. 2005. 80. P. 65–74.

References

1. Lopshire J.C., Zipes D.P. Device therapy to modulate the autonomic nervous system to treat heart failure. Curr Cardiol Rep. 2012. 14(5). P. 593–600.
2. Chatterjee N.A., Singh J.P. Novel Interventional Therapies to Modulate the Autonomic Tone in Heart Failure. JACC Heart Fail. 2015. 3(10). P. 786–802.
3. Schwartz P.J., La Rovere M.T., De Ferrari G.M., Mann D.L. Circ Heart Fail. Autonomic modulation for the management of patients with chronic heart failure. 2015; 8(3). P. 619–628.
4. Shen M.J., Zipes D.P. Interventional and device-based autonomic modulation in heart failure. Heart Fail Clin. 2015. 11(2). P. 337–348.
5. Fresiello L., Ferrari G., Di Molfetta A., Zieliński K., Tzallas A., Jacobs S., et al. A cardiovascular simulator tailored for training and clinical uses. J Biomed Inform 2015. 57. P. 100–112.
6. Burkhoff D., Tyberg J.V. Why does pulmonary venous pressure rise after onset of LV dysfunction: a theoretical analysis. Am J Physiol 1993. 265. P. 1819–1828.
7. Smith B.W., Chase J.G., Shaw G.M., Nokes R.I. Experimentally verified minimal cardiovascular system model for rapid diagnostic assistance. Control Engineering Practice .2005. 13. P. 1183–1193.
8. Paeme K., Moorhead K.T., Chase J.G., Lambermont B., Kolh P., D'orio V., Pierard L., Moonen M., Lancellotti P., Dauby P. C., Desaive T. Mathematical multi-scale model of the cardiovascular system including mitral valve dynamics. Application to ischemic mitral insufficiency. BioMedical Engineering OnLine 2011, 10: 86.
9. Acton A. Advances in Heart Research and Application: 2013 Scolarly Edition, Atlanta (Georgia), USA: 424 p.
10. Grygoryan R.D. The Energy Basis of Reversible Adaptation. New York, USA: Nova Science, 2012. 253 p.
11. Grygoryan R.D. The Optimal Circulation: Cells' Contribution to Arterial Pressure. New York, USA: Nova Science, 2017. 287 p.
12. Grygoryan R.D., Sagach V.F. The concept of physiological supersystems: a new stage of integrative physiology. Fiziol. Zh. 2017. 63(3): P. 58–67.
13. Grygoryan R.D., Lissov P.N. A software simulator of human cardiovascular system based on its mathematical model. Problems in programming. 2004. N 4. P. 100–111.
14. Bers D.M. Excitation-Contraction Coupling and Cardiac Contractile Force. Kluwer Academic Publishers, 2002. 452 p.
15. Mynard J.P., Davidson M.R., Penny D.J., Smolich J.J. A simple, versatile valve model for use in lumped parameter and one-dimensional cardiovascular models. Int J Number Methods Biomed Eng. 2012. 28. P. 626–641.
16. Avolio A.P. Multi-branched model of the human arterial system. Med Biol Eng Comput. 1980. 18. P. 709–718.
17. Heldt T., Mukkamala R., Moody G.B., Mark R.G. CVSim: an open-source cardiovascular simulator for teaching and research. Open Pacing Electrophysiol Ther J. 2010. 3. P. 45–54.
18. van Meurs W. Modeling and simulation in biomedical engineering: applications in cardiorespiratory physiology. 1st ed. McGraw-Hill Education; 2011.
19. Mateják M, Kulhánek T, Šilar J, Privitzer P, Ježek F, Kofránek J. Physiolibrary – Modelica library for physiology. 10th International Modelica conference; 2014.
20. Hann C.E., Chase J.G., Shaw G.M. Efficient implementation of non-linear valve law and ventricular interaction dynamics in the minimal cardiac model. Comput Methods Programs Biomed. 2005. 80. P. 65–74.

Получено 12.10.2017

Про авторов:

Григорян Рафик Давидович,
заведующий отдела,
доктор биологических наук.
Количество научных публикаций в украинских изданиях – 130.
Количество научных публикаций в зарубежных изданиях – 40.
Индекс Хирша – 7.
http://orcid.org/0000-0001-8762-733X,

Дегода Анна Григорьевна,
старший научный сотрудник,
кандидат физ.-мат. наук.
Количество научных публикаций в украинских изданиях – 10.
Количество научных публикаций в зарубежных изданиях – 1.
Индекс Хирша – 3.
http://orcid.org/0000-0001-6364-5568,

Джуринский Егор Антонович,
инженер-программист.
http://orcid.org/0000-0002-1636-1447,

Харсун Вадим Сергеевич,
инженер-программист,
http://orcid.org/0000-0001-5745-0932.

Место работы авторов:

Институт программных систем
НАН Украины,
03187, г. Киев,
проспект Академика Глушкова, 40.
Тел.: (044) 526 5169.
E-mail: rgrygoryan@gmail.com,
anna@silverlinecrm.com,
y.a.dzhurynskyi@gmail.com,
vakharsun@gmail.com

УДК 681.03

В.В. Туманов, А.Ю. Дорошенко

ВИКОРИСТАННЯ КОМП'ЮТЕРНОГО ЗОРУ В СИСТЕМІ ЦИФРОВОЇ НАРІЗКИ МАТЕРІАЛІВ

Запропонований підхід до реалізації системи комп'ютерного зору для розпізнавання та позиціонування об'єктів на різальній поверхні станків для цифрової нарізки матеріалів за допомогою фотознімку їх робочої поверхні разом з промаркованими об'єктами нарізки на ній. Розроблено алгоритми роботи модулів системи, які відповідають за калібрування камери і розпізнавання реєстраційних марок двома принципово різними способами, які доповнюють один одного. Розроблено також алгоритм ідентифікації об'єктів нарізки на основі застосування елементів теорії графів.
Ключові слова: система комп'ютерного зору, цифрова нарізка, OpenCV, калібрування камери, розпізнавання об'єктів.

Вступ

Роль систем комп'ютерного зору в деяких сферах промисловості дуже важко переоцінити. Робота з розпізнавання та ідентифікації об'єктів, що раніше покладалась виключно на людину, нині досить успішно виконується обчислювальною технікою [1]. При цьому, з роками зростає як швидкість виконання цієї роботи технікою, так і якість її результатів. Це пов'язано не тільки зі збільшенням потужності апаратного забезпечення, але й з розробкою нових досконаліших алгоритмів розпізнавання об'єктів на цифровому зображенні.

Деякі галузі промисловості раніше навіть не передбачали, або через недостатній рівень технологій просто не могли використовувати тогочасні системи комп'ютерного зору для своїх цілей. Це стосувалось в основному сфер виробництва з високими вимогами до точності класифікації або позиціонування об'єктів. Але в ході технічного прогресу постійно зростає деталізація та якість зображень, отримуваних за допомогою фото- та відеокамер, розробляються нові, більш досконалі алгоритми розпізнавання, позиціонування та 3D-реконструкції об'єктів за допомогою цифрових знімків. Так стає можливим впровадження систем комп'ютерного зору там, де раніше цієї можливості не було.

Подібна ситуація склалась і в сфері цифрової нарізки. Тут впровадження систем комп'ютерного зору відбувалось крок за кроком, відповідно до ступеню розвитку техніки. Першим і головним кроком стало встановлення *локальної камери* прямо біля ріжучого механізму. Її «погляд» охоплює лише невелику ділянку різальної поверхні безпосередньо під різцем. Дані, отримані від камери після ретельного дослідження всієї різальної поверхні, дозволяють скласти точну карту розміщення об'єкта(-ів) на ній і визначити програму слідування різака по матеріалу нарізки згідно з заздалегідь визначеним шаблоном. Близькість до об'єктів на різальній поверхні забезпечила високу точність розпізнавання і позиціонування об'єктів, і, як наслідок максимальну точність нарізки. Всі сучасні системи цифрової нарізки використовують дану технологію як базову.

Однак такий підхід має суттєвий недолік – швидкодію. Різальному механізму з камерою необхідно пройти вздовж усього контуру об'єкта нарізки щоб скласти повну і точну картину для подальшого нарізання. Частково вирішити цю проблему дозволяє реконструкція позиції всього об'єкта, базуючись на отриманих даних про положення його частини. Втім, економія часу за рахунок цього неминуче призводить до втрати точності, оскільки розрахунок ведеться на ідеально рівний, недеформований відносно шаблону об'єкт, чого інколи досить важко уникнути.

Інший підхід до рішення проблеми швидкодії полягає у нанесенні на нарізуваний матеріал опорних маркерів, так

званих реєстраційних марок. Реєстраційні марки – це чорні круги діаметром від 8 до 16 міліметрів. Набір реєстраційних марок і їх взаємне розташування ідентифікують об'єкт і визначають спосіб його нарізки. Такий спосіб є основним у сфері серійної нарізки, де на неперервному матеріалі (наприклад, на рулоні текстилю, паперу і т. п.) можна розмістити довільну кількість промаркованих об'єктів для вирізання.

Тим не менш, навіть використання реєстраційних марок не дозволяє повністю вирішити проблему швидкодії, так як камері перед виконанням нарізки так само треба виконати пошук марок на всій різальній поверхні. Ще одним серйозним недоліком такого підходу до пошуку об'єктів нарізки є подвійне зношування механізмів різака, оскільки для виконання однієї операції нарізки йому необхідно двічі пройти вздовж різальної поверхні: перший раз для пошуку і позиціонування об'єктів на ній, а другий – для їх нарізки.

В даній статті розглядатимемо наступний етап технології цифрової нарізки, котрий полягає у застосуванні аналізу повного зображення всієї різальної поверхні і практично позбавлений недоліків локального пошуку об'єктів.

Мета даної статті – це опис загальної концепції технології обробки і аналізу зображення різальної поверхні для ідентифікації і позиціонування об'єктів на ній без суттєвих втрат точності.

1. Постановка задачі та вихідні дані

Загальна задача описуваної системи комп'ютерного зору полягає в ідентифікації об'єктів на різальній поверхні станка для цифрової нарізки за допомогою растрового зображення цієї поверхні разом з об'єктами нарізки на ній. Цільова система призначена виключно для роботи з об'єктами нарізки що мають універсальний спосіб ідентифікації – реєстраційні марки. Крім фотографії різальної поверхні, системі також необхідна база даних шаблонів нарізки у вигляді файлів з координатами реєстраційних марок у межах самого об'єкта, для ідентифікації об'єктів знайдених на фотографії різальної поверхні.

В результаті, описувана система повинна за мінімальний час, проаналізувавши зображення робочої поверхні різального станка, ідентифікувати всі розташовані на ній об'єкти, а також визначити їх положення відносно точки відліку різального механізму. Далі отримані дані передаються керуючій програмі різального станку у вигляді структур даних, що містять ідентифікатор об'єкта, та координати всіх його реєстраційних марок щодо заданої точки відліку. Процес позиціонування буде детально розглянутий далі в даній статті.

Для забезпечення технічної бази описуваної системи комп'ютерного зору необхідне встановлення додаткової *глобальної фотокамери* над різальною поверхнею станка для цифрової нарізки. З метою отримання оптимального для аналізу зображення, камеру слід розташувати над центром різальної поверхні, так щоб оптична вісь була перпендикулярна до неї. Висоту треба обрати достатню для отримання зображення всієї робочої поверхні станка з мінімумом зайвого на фото. Чим вища роздільність та якість знімків камери, тим кращою буде точність позиціонування і ідентифікації об'єктів на них, що свідчить на користь використання якомога більш якісних фотокамер для даної цілі.

Оскільки цифрові різальні станки оснащені комп'ютером за замовчуванням, вимога наявності базової обчислювальної техніки для запуску цільової програми задовольняється заздалегідь. Необхідна лише наявність операційної системи сімейства Windows для запуску та коректної роботи програми.

Реалізація основних алгоритмів обробки та аналізу растрових зображень вже є у відкритій бібліотеці OpenCV (англ. Open Source Computer Vision Library), яку може вільно використовувати в академічних та комерційних цілях. Алгоритми цільової програми повністю базуються на використанні даної бібліотеки.

2. Калібрування камери і позиціонування об'єктів

Перед початком аналізу зображення різальної поверхні слід переконатись, що воно повністю відображає реальність. Отримані за допомогою фото-зйомки зображення часто «страждають» від різних оптичних спотворень (дисторсії) [2], вплив яких варіюється в залежності від якості оптики об'єктива камери. На противагу, існують доволі ефективні методи боротьби з оптичними спотвореннями на фотографіях.

В першу чергу необхідно виконати калібрування камери. Калібруванням камери називають розрахунок її внутрішніх і зовнішніх параметрів за отриманими з її допомогою фото та відео. В результаті отримується необхідна для усунення дисторсії інформація про камеру.

В розрахунках коефіцієнтів дисторсії бібліотека OpenCV враховує як тангенціальні та і радіальні складові [3]. Для радіальної складової використовується наступна формула:

$$x_{corrected} = x(1 + k_1 r^2 + k_2 r^4 + k_3 r^6),$$

$$y_{corrected} = y(1 + k_1 r^2 + k_2 r^4 + k_3 r^6).$$

Таким чином, піксельна точка на спотвореному зображенні маючі координати (x, y), отримує координати $(x_{corrected}\, y_{corrected})$ на виправленому зображенні. Присутність радіальної дисторсії виявляється в ефектах «бочки» та «риб'ячого ока» на фотографіях.

Тангенціальна дисторсія виникає через те, що лінзи об'єктива камери не можуть бути ідеально паралельні площині, яка фотографується. Її можна усунути за наступною формулою:

$$x_{corrected} = x + [2p_1 xy + p_2(r^2 + 2x^2)],$$

$$y_{corrected} = y + [p_1(r^2 + 2y^2) + 2p_2 xy].$$

В підсумку можна виділити 5 параметрів спотворення, які в OpenCV представлені у вигляді однорядкової матриці з п'ятьма стовпцями.

$$Distortion_{coefficients} = (k_1 \quad k_2 \quad p_1 \quad p_2 \quad k_3).$$

Для перетворення координат точок реального світу в координати пікселей на фотознімку, застосовується *матриця камери:*

$$\begin{bmatrix} x \\ y \\ w \end{bmatrix} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}.$$

Присутність складової w у складі матриці піксельних координат пояснюється необхідністю дотримання відповідності розмірностей матриць координатних систем при множенні, і не носить реального змісту. Невідомими параметрами є f_x і f_y – фокусні відстані камери, та (c_x, c_y), які представляють оптичні центри, виражені в піксельних координатах. Якщо для обох осей використовується спільна фокусна відстань зі співвідношенням a (зазвичай 1), тоді $f_y = f_x * a$ і рівняння вище буде мати єдину фокусну відстань f.

Процес знаходження матриць камери і коефіцієнтів дисторсії називається калібруванням. Розрахунки їх параметрів виконуються за допомогою геометричних рівнянь. Застосовувані рівняння залежать від типу об'єктів, що використовуються для калібрування. В даний момент OpenCV підтримує 3 типи об'єктів для калібрування:

- класична чорно-біла шахова дошка;
- симетричний круговий патерн;
- асиметричний круговий патерн.

Необхідно зробити знімок цих патернів і знайти їх на ньому за допомогою засобів OpenCV. Кожному знайденому патерну відповідає нове рівняння. Для рішення цього рівняння треба зробити деяку кількість знімків і проаналізувати їх, щоби скласти систему з рівнянь. Наприклад, теоретично, шаховий патерн передбачає мінімум 2 знімки. Але цього можна уникнути, зробивши один знімок кількох патернів одразу.

Для наших цілей краще за все підходить шаховий патерн, оскільки він, по суті, потребує лише «сітку» з точок. Справа в тому, що шахова дошка є лише обгорткою для полегшення розпізнавання. Насправді OpenCV цікавлять лише позиції вузлів шахового патерну (точок зіткнення 4 клітинок).

Таким чином, необхідно лише надрукувати і сфотографувати *калібрувальну сітку*, яка складається зі зручних для розпізнавання об'єктів, розташованих у вузлах уявної решітки з комірками у вигляді квадратів з попередньо заданою стороною. Найбільш зручними для розпізнавання об'єктами будуть вище згадані реєстраційні марки, які мають радіальну симетрію з огляду на їх круглу форму і можуть бути легко розпізнані як камерою на різаку станка, так і засобами OpenCV. Сторони квадрата особливого значення для розпізнавання не має, але мусить бути достатньо великою щоб, вузли сітки зливались між собою на фотографії, і водночас достатньо малою щоб забезпечити велику точність позиціонування. Останнє твердження витікає з того, що калібрувальна сітка в нашому випадку служить для двох цілей – надає патерн для калібрування камери і утворює основу системи позиціонування об'єктів на різальній поверхні.

Розроблювана система передбачає нерухоме закріплення камери над різальною поверхнею, з чого слідує, що один і той самий піксель на різних фотографіях камери буде представляти одну й ту саму точку в просторі (на різальній поверхні). Залишається лише задати точку відліку і встановити опорні точки на різальній поверхні з відомими координатами, відносно яких будуть розраховуватись координати всіх інших.

Позиціонування в межах самої калібрувальної сітки є найпростішою задачею, оскільки крок сітки (сторона квадрата комірки) відомий заздалегідь. Зробивши знімок калібрувальної сітки (*калібрувальний знімок*), достатньо тільки розрахувати відношення відстані в міліметрах до відстані в пікселях між вузлами сітки і користуватись його значенням для знаходження відстаней в межах сітки. Наприклад, між вузлами калібрувальної сітки реальна відстань складає 50 мм, а на знімку – 100 пікселей. Співвідношення мм на 1 піксель (mm/px) складає ½. Знаючи, що відстань на фотографії між об'єктами складає, скажімо 482 пікселі, достатньо помножити це значення на розраховане раніше відношення mm/px щоб взнати реальну відстань в міліметрах: 241 мм.

Для позиціонування щодо точки відліку різака необхідне разове застосування локальної камери, яка має знайти кожен вузол калібрувальної сітки і надати його координати, знайдені з високою точністю*, системі комп'ютерного зору, яка «прив'яже» реальні координати вузлів у міліметрах щодо точки відліку, до їх координат на фотографіях у пікселях. Це буде єдиний етап роботи розроблюваної системи, коли треба застосовувати камеру різака для пошуку марок. Заново виконувати таку процедуру потрібно буде лише у випадку необхідності зміни калібрувальної сітки (з іншим кроком, патерном тощо) і повторного виконання калібрування вже для нової сітки.

Подальша схема позиціонування нічим не відрізняється від поданої вище. Проте точність буде вищою, оскільки без даних від локальної камери ми можемо знати лише заданий при виготовленні крок між вузлами калібрувальної сітки, який у процесі її виготовлення (при друкуванні) може бути спотворений у силу різних технічних факторів.

Все вище сказане свідчить про важливість міцної фіксації камери щодо різальної поверхні. Найменше її зміщення призведе до невідповідності наступних знімків калібрувальному та необхідності виконувати процес калібрування заново, який, враховуючи час на пошук вузлів камерою різака і в залежності від розмірів стола та калібрувальної сітки може зайняти досить багато часу.

В контексті описуваної системи комп'ютерного зору, процес калібрування включає не тільки знаходження матриць властивостей камери. Це також процедура

* зазвичай локальні камери забезпечують точність +-0.5 мм

повної підготовки системи до подальшої коректної роботи з ідентифікації та позиціонуванню об'єктів на різальній поверхні на основі калібрувальної фотографії та даних від камери різака.

Після отримання даних від обох камер, інформації про розміри та крок калібрувальної сітки та команди запуску калібрування, програма починає пошук реєстраційних марок на калібрувальній фотографії. Процедуру пошуку марок буде детально описано в наступному розділі даної статті. Результати пошуку у вигляді списку координат пікселей центрів марок на фотографії проходять процедуру пошуку закономірностей розташування, на основі яких будуються об'єкти калібрувальних сіток. Основу цих об'єктів складають три двомірні масиви точок, розміри яких збігаються з розмірами сіток, а комірки призначені для збереження координат вузлів сітки, тобто центрів марок. Перший масив зберігає координати вузлів сітки в піксельних координатах фотографії, а другий – отримані від камери різака відповідні координати реального світу в міліметрах. Третій призначений для зберігання неспотворених координат вузлів на «вирівняній» фотографії.

Після побудови всіх об'єктів сіток відповідно до кількості реальних сіток на фотографії, та заповнення їх перших масивів у порядку реального розташування вузлів, відбувається пошук відповідності розташування комірок першого масиву та точок, отриманих від камери різака. Таким чином встановлюється однозначна відповідність реальних координат вузлів калібрувальних сіток у міліметрах до їх координат на фотографії в пікселях та заповнюються другі масиви об'єктів сіток. На основі отриманих даних визначається також позиція точки відліку (початку координат) на фотографії.

Далі слідує власне калібрування камери. Для отримання калібрувальних матриць OpenCV пропонує метод calibrateCamera() [4]. Він приймає координати точок патернів (калібрувальних сіток) в просторі, їх координати на фотографії (перші два масиви об'єктів сіток), а також пусті матриці для власних коефіцієнтів та коефіцієнтів спотворення камери, які заповнює розрахованими значеннями. Отримані матриці зберігаються і далі використовуються для розрахунку «правильних» координат вузлів калібрувальної сітки за допомогою функції OpenCV undistortPoints(). Перераховані неспотворені координати записуються в третій масив об'єктів сіток. В подальшому всі точки, знайдені на фотографіях, будуть проходити таку обробку для підвищення точності позиціонування.

На цьому процес калібрування закінчується. В результаті, програма комп'ютерного зору отримує всю необхідну інформацію для точної ідентифікації та позиціонування об'єктів на наступних фотографіях різальної поверхні.

3. Обробка зображення та знаходження реєстраційних марок

Найголовнішою задачею розроблюваної системи комп'ютерного зору є пошук реєстраційних марок на фотографії. Бібліотека OpenCV пропонує широкий набір інструментів для пошуку різноманітних геометричних об'єктів на растрових зображеннях. Трохи спрощує задачу сама форма марки – круг переважно чорного кольору. На бінарному зображенні це виглядатиме як скупчення темних пікселей з округлим контуром. Спеціально з ціллю пошуку таких об'єктів було розроблено клас SimpleBlobDetector [5]. Цей клас реалізує простий алгоритм для знаходження чорних «плям», до яких як раз відносяться реєстраційні марки на зображенні.

Спочатку виконується конвертація кольорового зображення в кілька чорно-білих з застосуванням різних порогів від мінімального до максимального заданих (або прийнятих за замовчуванням) з деяким кроком. Суть застосування порогів полягає у порівнянні значення пікселя зі значенням порогу. Попередньо зображення з кольорового переводиться у відтінки сірого. Далі, якщо значення пікселя більше за значення порогу, йому присвоюється один колір (наприклад, чорний), а якщо менше – інший (наприклад, білий). Далі

алгоритм проводить пошук контурів на отриманих чорно-білих зображеннях та знаходить їх центри. Потім, знайдені на різних зображеннях контури групуються по координатах. Близько розташовані центри формують єдину групу, яка відповідає окремій «плямі». Нарешті, аналізуючи кожну групу, алгоритм знаходить остаточні центри та радіуси «плям», та повертає їх у формі колекції структур, що представляють координати та властивості знайдених «плям».

Також клас перед поверненням результатів піддає фільтрації за заданими параметрами знайдені «плями». До цих параметрів відносяться.

1. Яскравість. Фільтр порівнює інтенсивність кольору в центрі «плями» з заданим параметром blobColor. Якщо вони сильно різняться, «пляма» відкидається.

2. Площа. Допускаються лише «плями», що мають площу від заданої параметром minArea (включно) до maxArea (не включно). Тобто встановлення minArea рівним 100, відкине всі контури з площею менше 100 пікселей.

3. Округлість. Тут оцінюється на скільки сильно «пляма» схожа на коло. Наприклад, шестикутник буде більш округлим аніж, скажімо, квадрат. Треба лише встановити відповідне значення параметрам minCircularity та maxCircularity (від 0 до 1). Округлість розраховується за формулою:

$$\frac{4\pi Area}{(perimeter)^2}.$$

З якої слідує, що ідеальне коло має округлість рівну 1, квадрат – 0,785 і так далі.

4. Опуклість. Ступінь опуклості визначається як відношення площі контуру до площі її опуклої оболонки. Опуклою оболонкою контуру є найменший опуклий контур, який повністю включає у себе заданий. Для включення цього фільтра необхідно встановити значення minConvexity та maxConvexity (обидва від 0 до 1).

5. Коефіцієнт інерції. Вимірюється ступінь видовженості контуру. Коло має цей коефіцієнт рівний 1, у еліпса від 0 до 1, а в лінії рівно 0. Для використання фільтрації за коефіцієнтом інерції необхідно встановити параметри minInertiaRatio та maxInertiaRatio відповідно.

Однак у класу SimpleBlobDetector є один суттєвий недолік – нечутливість до кольору. Тому він підходить лише для пошуку чорних реєстраційних марок. Що стосується марок інших кольорів, то тут варто звернути увагу на геометричні властивості марки. Як не повертай коло однорідного кольору навколо його центру – його вигляд завжди залишатиметься однаковим. Така його властивість дозволяє застосувати до нього метод *співставлення зразка*.

Співставлення зразка – це метод пошуку місця зразка на зображенні більшого розміру. Іншими словами, маючи шматочок зображення, можна знайти його положення на цілому зображенні. В OpenCV цей метод реалізує функція matchTemplate(). Вона просто переміщує зразок по цільовому зображенню та порівнює зразок з поточним шматочком під ним. Існує декілька реалізацій методу співставлення зразка в OpenCV. Вибрати конкретний можна встановленням значення одного з параметрів matchTemplate(). Повертає функція сіре зображення, де кожен піксель показує наскільки сусідня з ним область збігається із зразком [6].

Якщо основне зображення має розміри (WxH), а зразок (wxh), вихідне зображення матиме розміри (W–w+1, H–h+1). Як тільки результуюче зображення отримано, можна використовувати функцію minMaxLoc() для отримання координат мінімального (максимального – в залежності від використаного методу співставлення зразка), за шкалою сірого пікселя. Це буде верхній лівий кут області, що відповідає зразку.

Однак, такий метод підходить лише для випадку колу розшукується єдине збігання заданого зразка на цільовій фотографії. Якщо таких збігів очікується більше

одного, необхідно використовувати поріг відтінків сірого [7]. Якщо метод передбачує мінімальне значення пікселя (світлий) в області збігання, то необхідно встановити поріг від 0 до 1 ближче до 0 і циклічно викликати функцію minMaxLoc(), порівнювати найменше значення яке вона повертає з заданим порогом. Якщо воно менше, то можна зберігати його координати та зафарбовувати вже знайдену область в колір зворотній шуканому (для мінімуму – чорний) для запобігання повторного повернення тієї ж самої координати. Якщо ж minMaxLoc() повернула мінімальне значення більше порогового, значить можна переривати цикл – досить чіткі збігання закінчились.

Для методів, які повертають збігання з максимальним значенням відтінку сірого (темним) пікселя, загальний принцип зберігається, але поріг береться ближче до 1, а приймаються точки з більшим ніж порогове значенням відтінку.

Використання співставлення зразка в описуваній системі потребує лише створення кількох зразків марок (простим вирізанням області марки з фотографії) різного кольору, діаметру, освітленості та інших параметрів, що впливають на її вигляд. Далі можна застосовувати пошук відповідного зразка в залежності від того які марки представлені на фотографії.

Загальним недоліком як класу SimpleBlobDetector, так і методу співставлення зразка є знаходження на фотографії зайвих чорних точок окрім реєстраційних марок. Одним з можливих рішень може бути жорстке обмеження значень параметрів для SimpleBlobDetector і зменшення порога для співставлення зразка. Однак, так скоріш за все будуть відкинуті й потрібні, але з тих чи інших причин (освітлення, тіні й т. п.) невдало сфотографовані марки.

Найкращим варіантом використання цих методів разом за даних умов, буде їх комбінування з застосуванням великих допусків з наступним прийняттям тільки тих результатів, які знайдені обома алгоритмами. Коли стосується пошуку кольорових марок, то тут варто використовувати лише співставлення зразка.

4. Ідентифікація об’єктів нарізки

Ідентифікація об’єктів нарізки починається із створення знімку різальної поверхні глобальною камерою. Разом із знімком системі комп’ютерного зору також на вхід подається шаблон(и) об’єктів розпізнавання які присутні на знімку, та зразки марок для розпізнавання, якщо планується використовувати метод співставлення зразка. Після отримання всіх даних та відповідної команди система починає власне процес ідентифікації.

Спочатку відбувається пошук реєстраційних марок описаними попередньо способами. Результат пошуку у вигляді списку координат центрів знайдених реєстраційних марок подається на вхід процедури пошуку відповідностей шаблонам об’єктів нарізки. Запускається цикл, який за чергою перевіряє наявність конфігурації марок поточного шаблону серед знайдених на фотографії різальної поверхні.

Алгоритм пошуку відповідностей шаблонам нарізки полягає у використанні деяких елементів з теорії графів. Марки шаблону мають координати та утворюють вершини зваженого графа з ребрами, вага яких дорівнює відстані між марками шаблону. Залишається лише знайти такий самий граф серед усіх знайдених на знімку реєстраційних марок. Зрозуміло, що пошук марок допускає, що деякі марки на фото можуть бути не знайдені через недосконаліть алгоритмів пошуку, якість зображення, освітлення тощо. Це означає, що відсутність деяких вершин цільового графа серед знайдених на знімку реєстраційних марок є допустимою, якщо з урахуванням всіх інших наявних марок, можна встановити однозначну їх відповідність шуканому шаблону об’єкта нарізки.

Оскільки координати марок шаблону дані в міліметрах відносно невідомого нам початку координат матеріалу, на який ці марки нанесені, оперувати можна лише абсолютними значеннями відстаней між ними. Результатом процедури калібрування, яка має бути обов’язково виконана перед ідентифікацією об’єктів нарізки, є система позиціонування, яка дозволяє досить точно визначати відстані між

точками на фотографії. Так для кожної знайденої марки ми можемо знайти всі відстані від неї до інших (ребра графа) та порівняти їх з набором ребер між марками з шаблону, виявивши збігання. В цьому і полягає основна ідея алгоритму пошуку відповідностей набору знайдених реєстраційних марок заданим шаблонам об'єктів нарізки.

Процедура розпізнавання шаблону об'єкта нарізки серед марок виглядає наступним чином. На вхід приймається цільовий шаблон та набір піксельних координат знайдених на знімку марок. Спочатку перевіряється чи достатньо знайдено марок на фото. Якщо всього знайдених марок менше ніж половина від кількості марок у шаблоні об'єкта, то процедура переривається, вважаючи, що об'єктів з таким шаблоном на знімку різальної поверхні немає. Далі циклічно перебираються всі марки шаблону. Для кожної марки шаблону запускається ще один внутрішній цикл, який проходить по всім знайденим на фото маркам та перевіряє їх відповідність поточній марці шаблону за допомогою порівняння їх наборів ребер. Для цього розраховуються всі відстані в міліметрах (ребра) від поточної знайденої марки до інших, а ребра шаблонних марок розраховуються попередньо, ще при зчитуванні з файлу та містяться в спеціальній структурі, яка представляє шаблонну марку разом з її ребрами. Далі проходить процедура пошуку ребер шаблонної марки серед ребер марки з фото.

Якщо марка, знайдена на знімку, має більше 50 % ребер, що відповідають марці з шаблону, вона заноситься до *словника пар відповідностей* вигляду: «марка шаблону – марка на фотографії», та видаляється з набору знайдених марок. Перебір продовжується далі, оскільки на різальній поверхні може бути одночасно кілька однакових об'єктів нарізки, тож одній марці з шаблону може відповідати декілька марок із знімку.

Слід зазначити, що порівняння ребер відбувається з допуском в 5 % від довжини ребра шаблонної марки, адже повної рівності довжин відповідних ребер бути не може через присутність допустимої похибки в системі позиціонування (приблизно +-1 мм).

Після закінчення циклу пошуку марок шаблону серед знайдених на знімку методом порівняння наборів ребер, маємо заповнений словник їх відповідностей. Якщо кількість пар у словнику менша ніж половина від кількості марок в цільовому шаблоні, процедура закінчується, оскільки це означає, що об'єктів з таким шаблоном на різальній поверхні немає. Якщо ж пар в словнику достатньо, то починається пошук вже окремих екземплярів заданого шаблону об'єкта серед відібраних пар марок.

Береться довільна (перша) *опорна пара* зі словника «марка шаблону – марка на фотографії», та видаляється з нього. Далі, програма намагається знайти відносно неї всі інші марки, які разом сформують об'єкт нарізки. Одразу ж створюється новий програмний екземпляр об'єкта, який містить контейнер для його марок та поле ідентифікатора шаблона нарізки. До списку його марок заноситься позиція марки на фото з опорної пари. Також йому присвоюється ідентифікатор поточного шаблону. Після цього починається цикл перебору ребер шаблонної марки з опорної пари, всередині якого знаходиться вкладений цикл, який проходить по всім парам словника. У внутрішньому циклі для шаблонної марки поточної пари розраховується відстань від неї, до шаблонної марки опорної пари та порівнюється з поточним ребром зовнішнього циклу. Якщо вони рівні, то розраховується відстань від марки на фото з поточної пари до марки на фото з опорної пари. Якщо ця відстань приблизно (з урахуванням 5 % допуску) рівна значенню поточного ребра, то поточна пара є ще однією маркою об'єкта нарізки, і значення марки на фото додається до списку марок екземпляра цього об'єкта.

Після закінчення обох циклів, перевіряється кількість марок у списку екземпляра об'єкта. Якщо вона більша ніж половина кількості марок шаблона, то такий об'єкт приймається як розпізнаний та додається в колекцію для повернення процедурою. Якщо в словнику залишилось ще достатньо марок, то це означає

що потенційно є ще один екземпляр такого шаблону об'єкта. Обирається нова опорна пара і для неї так само виконуються описані вище дії.

Після того як в словнику залишиться замало пар для відбудови об'єкта шаблону, або не залишиться зовсім, процедура повертає колекцію розпізнаних об'єктів нарізки, що відповідають заданому шаблону. Ці об'єкти в свою чергу додаються до контейнера, що містить всі розпізнані об'єкти різних шаблонів, адже на різальній поверхні одночасно можуть бути розміщені представники різних шаблонів нарізки.

Після завершення процедури розпізнавання, програма повертає колекцію, де зібрані всі розпізнані об'єкти нарізки з їх унікальними ідентифікаторами, та координатами марок у міліметрах. Форма даних що повертає система комп'ютерного зору залежить від того, в якому вигляді їх приймає на вхід комп'ютер різального станка (файл, пакети даних тощо).

Отримані дані дозволяють програмі комп'ютера станка сформувати траєкторію руху різака по робочій поверхні та правильно вирізати розпізнані об'єкти на ній.

Висновки

Розроблена та описана концепція системи комп'ютерного зору для точної ідентифікації та позиціонування об'єктів цифрової нарізки з реєстраційними марками за фотознімком робочої поверхні станка для цифрової нарізки. Створено алгоритми роботи модулів системи, які відповідають за калібрування камери та утворення системи позиціонування на основі калібрувальної сітки та даних від локальної камери різака, розпізнавання реєстраційних марок двома принципово різними способами, які доповнюють один одного, а також алгоритм ідентифікації об'єктів нарізки на основі застосування елементів теорії графів.

1. Шаіпро Л., Стокман Дж. Комп'ютерний зір: переклад з англ. М.: БИНОМ. Лаборатория знаний, 2006. 752 с.
2. Іофис Є.А. Фотокінотехніка. М.,: «Советская энциклопедия», 1981. 447 с.
3. Шеліський Р. Комп'ютерний зір: Алгоритми та Застосунки. Нью-Йорк: Springer, 2011. 957 с.
4. Калібрування камери з OpenCV [Електронний ресурс]. Документація OpenCV 2.4.13.3. Режим доступу до ресурсу: http://docs.opencv.org/2.4/doc/tutorials/calib3d/camera_calibration/camera_calibration.html.
5. Маллік С. Знаходження точок за допомогою OpenCV [Електронний ресурс]. Вивчення OpenCV. 2015. Режим доступу до ресурсу: https://www.learnopencv.com/blob-detection-using-opencv-python-c/.
6. Співставлення зразка [Електронний ресурс]. Документація OpenCV 2.4.13.3. Режим доступу до ресурсу: http://docs.opencv.org/2.4/doc/tutorials/imgproc/histograms/template_matching/template_matching.html.
7. Брадський Г.Р., Келер А. Вивчаючи OpenCV. Себастополь: O'Reilly Media, 2008. 556 с.

References

1. Shapiro, L. & Stockman, G. (2006) Computer vision. Moscow: BINOM. Laboratoriya znaniy. (in Russian)
2. Iofis, E.(1981) Fotokinotehnika. Moscow: "Sovietskaya encyklopediya". (in Russian)
3. Szeliski, R. (2011) Computer Vision: Algorithms and Applications. New York: Springer.
4. OpenCV 2.4.13.3 documentation. Camera calibration With OpenCV. Available at: http://docs.opencv.org/2.4/doc/tutorials/calib3d/camera_calibration/camera_calibration.html.(accessed 1 October 2017).
5. Satya, M. (2015) Learn OpenCV. Blob Detection Using OpenCV. Available at: https://www.learnopencv.com/blob-detection-

using-opencv-python-c/. (accessed 29 September 2017).
6. OpenCV 2.4.13.3 documentation. Template Matching. Available at: http://docs.opencv.org/2.4/doc/tutorials/imgproc/histograms/template_matching/template_matching.html (accessed 1 October 2017).
7. Bradski, G. & Kaehler, A. (2008) Learning OpenCV. Sebastopol: O'Reilly Media.

Одержано 04.10.2017

***Про авторів*:**

Туманов Владислав Валерійович,
студент магістерської програми НТУУ «КПІ імені Ігоря Сікорського»
Кількість наукових публікацій в українських виданнях – 2.
http:// orcid.org/0000-0002-1813-5021,

Дорошенко Анатолій Юхимович,
доктор фізико-математичних наук, професор, завідувач відділом теорії комп'ютерних обчислень Інституту програмних систем НАН України професор кафедри автоматики та управління в технічних системах НТУУ "КПІ імені Ігоря Сікорського".
Кількість наукових публікацій в українських виданнях – більше 150.
Кількість наукових публікацій в закордонних виданнях – більше 40.
Індекс Хірша – 5.
http://orcid.org /0000-0002-8435-1451,

Місце роботи авторів:

НТУУ «КПІ імені Ігоря Сікорського».
03056, Київ, просп. Перемоги, 37,
Тел.: (095) 486 3307.
E-mail: tumanovlad@gmail.com.

Інститут програмних систем
НАН України.
03187, Київ,
проспект Академіка Глушкова, 40.
Тел.: (044) 526 3559.
E-mail: doroshenkoanatoliy2@gmail.com

УДК 681.3.06

Метод разработки параллельных систем нечеткого логического вывода на основе графических ускорителей / С.В. Ершов, Р.Н. Пономаренко. – С. 3–15.

Разработан и обоснован метод построения параллельных иерархических систем нечеткого логического вывода с использованием ярусно-параллельной формы алгоритмов на основе графических ускорителей Nvidia и технологии CUDA. Обоснована эффективность применения иерархических систем нечеткого логического вывода для построения диагностических интеллектуальных систем. Рассматриваются способы организации эффективных вычислений на графических ускорителях с целью распараллеливания нечетких иерархических систем. Разработана интеллектуальная система оценивания качества стартапов на основе иерархии блоков нечетких правил Такаги – Сугено. Получены оценки ускорения для разработанной интеллектуальной системы оценивания качества стартапов, а также для нечетких систем, зависимости между блоками правил которых сгенерированы случайным образом. Представлена сравнительная характеристика полученных оценок ускорения и оценок ускорения вариантов нечетких систем на основе технологии MPI. Обоснованы преимущества разработанного метода распараллеливания систем нечеткого логического вывода на основе графических ускорителей.

Ключевые слова: интеллектуальная система, ярусно-параллельная форма вычислений, графические ускорители, нечеткая логика, система Такаги – Сугено, CUDA.

UDC 681.3.06

Method of construction of parallel systems for fuzzy logical inference based on GPU accelerators / S.V. Yershov, R.N. Ponomarenko. – P. 3–15.

We examine a new method for constructing parallel hierarchical systems of fuzzy logic inference using multi-tier parallel form of algorithms based on Nvidia GPU accelerators and CUDA technology. The efficiency of application of hierarchical systems of fuzzy inference for development of diagnostic intelligent system is substantiated. We characterize the organization of efficient computations on graphic accelerators with the aim of achieving maximum degree of parallelism in fuzzy hierarchical systems. In particular, the above organization relies on parallelization of rules inside each block of fuzzy rules. An intelligent software system for assessing the quality of startups based on parallel inference architecture that contains Takagi – Sugeno blocks of fuzzy rules is developed. The experiment is conducted to demonstrate acceleration estimates for developed intellectual system for assessing the quality of startups as well as for more complex systems with randomly generated dependencies between blocks of rules. We compare characteristics of the obtained acceleration estimates with corresponding estimates for hierarchical fuzzy systems based on the distributed computing technology and MPI message exchange. The advantages of developed method for construction of parallel fuzzy inference based on GPU are substantiated.

Key words: intelligent system, multi-tier parallel form of computing, GPU accelerators, fuzzy logic, Takagi – Sugeno system, CUDA.

УДК 004.94; 004.4; 004.62

Управление ресурсами облачных центров обработки данных на основе эвристического поиска / Э.В. Жариков. – С. 16–27.

Проанализированы особенности облачного центра обработки данных с точки зрения управления ресурсами. Для решения задачи управления ресурсами облачного центра обработки данных предложено и исследовано двухэтапный метод консолидации виртуальных машин на основе использования локального лучевого поиска. В статье проанализирована работа эвристики первой и второй стадий предложенного метода, разработан алгоритм лучевого поиска для решения задачи управления ресурсами. Для анализа работы метода использованы данные о поступлении задач в кластер Google. Предложенный метод позволяет переключить в режим пониженного энергопотребления в среднем 56 процентов физических серверов, потенциально определенных для переключения в режим сна на основе верхней оценки необходимой емкости ресурсов. Перераспределение виртуальных машин выполняется с учетом ограничения допустимого количества миграций на один физический сервер.

Ключевые слова: виртуализация, управление ресурсами, облачные вычисления, эвристический поиск.

UDC 004.94; 004.4; 004.62

Managing data center resources using heuristic search / E.V. Zharikov. – P. 16–27.

The features of the cloud data center are analyzed from the point of view of resource management. The two-stage method for consolidating virtual machines based on the use of local beam search algorithm is proposed and investigated with aim to solve the problem of managing the resources of a cloud data center. In this paper, the work of heuristics of the first and second stages of the proposed method is analyzed. The beam search algorithm was developed for solving the data center resource management problem. The data about tasks and physical machines from the Google cluster-usage traces are used to evaluate the proposed method. The proposed method allows to switch to a low-power mode on average 56 percent of physical servers potentially identified for switching to sleep mode based on an upper estimate of the required capacity of resources. Virtual machine consolidation is performed taking into account the limitation of the permissible number of migrations per physical server.

Key words: virtualization, resource management, cloud computing, heuristic search.

УДК 004.4'24

Алгоритм автоматизированного распараллеливания циклических операторов для графических ускорителей / А.Е. Дорошенко, Е.А. Яценко, О.Г. Бекетов. – С. 28–36.

Распараллеливание циклических операторов является давно известной проблемой параллельного программи-

UDC 004.4'24

Algorithm for automatic loop parallelization for graphics processing units / A.Yu. Doroshenko, O.A. Yatsenko, O.G. Beketov. – P. 28–36.

Parallelization of loop operators is a long standing problem of parallel programming. The widespread use of graphics

рования. С широким использованием графических ускорителей для вычислительных задач возникла новая постановка данной проблемы для этого класса мультиядерных систем. Целью данной работы является усовершенствование механизма преобразования операторов цикла для его параллелизации для выполнения на графическом ускорителе. Разработано программное средство для оптимизации вычислений, которое позволяет в полуавтоматическом режиме осуществлять параллелизацию циклических операторов программы. Осуществлена буферизация данных, синхронизированная с выполнением основного цикла, и с помощью системы переписывающих правил TermWare построено средство, которое интегрировано с инструментарием проектирования и синтеза программ ИПС. Проведены испытания разработанной системы на гетерогенном мультиядерном кластере. Выполнено сравнение с известной системой параллелизации Par4All, в результате которого выявлены преимущества разработанной системы в плане быстродействия и возможности обработки объёмов данных, которые превышают объём памяти графического ускорителя, а также возможности использования нескольких ускорителей одновременно. Созданная система применена для распараллеливания последовательного цикла, входящего в состав программы численного прогнозирования погоды.

Ключевые слова: методы параллелизации, оптимизация циклов, вычисления общего назначения на графических процессорах, проектирование и синтез программ.

processing units for computational tasks has resulted in the new statement of the mentioned problem for this class of multicore systems. The purpose of this work is to improve the mechanism of transformation of cyclic operators for loop parallelization for execution on a graphics processing unit. Software tool for computation optimization that allows to parallelize cyclic operators semi-automatically was developed. Data bufferization synchronized with main loop execution was implemented, and the software tool using the rewriting rules system TermWare was built and integrated with the toolkit for design and synthesis of programs IDS. The developed system was tested using heterogeneous multicore cluster. The advantages of the developed system in comparison with well-known parallelization system Par4All consist in processing speed and the possibility of processing of data amounts exceeding the amount of memory of a graphics processing unit, and also the ability to use several graphics processing units simultaneously. The developed system was applied for parallelization of a serial loop, which is the part of a numerical weather forecasting program.

Key words: parallelization methods, loop optimization, general-purpose computing on graphics processing units, program design and synthesis.

УДК 004.8

UDC 004.8

Визначення формальних мов у метамові нормальних форм знань / О.П. Кургаєв, С.М. Григор'єв. – С. 37–50.

The definition of formal languages in the meta language of normal forms of knowledge / A.F. Kurgaev, S.M. Grygoryev. – P. 37–50.

Досліджено виразні можливості метамови нормальних форм знань стосовно формальних мов різного рівня. Дано

The expressive means of the metalanguage of normal forms of knowledge in relation to different level formal lan-

формальні описи множини лінгвістичних прикладів. Розроблено графічне метавизначення інтерпретатора універсальної машини Тьюринга, еквівалентне наведеному текстовому опису. Представлене визначення інтерпретатора універсальної машини Тьюринга може бути застосоване для розв'язку задач моделювання поведінки будь-якої Т-машини (яка вирішує задачу перетворення даних на її стрічці пам'яті). Дано формальне визначення транслятора детермінованої машини Тьюринга. Обґрунтовано й сформульовано можливість опису в метамові нормальних форм знань інтерпретуюче- транслюючого й транслююче-транслюючого процесів постановки й розв'язку довільної задачі, що має розв'язок.

Ключові слова: метамова, регулярні мови, контекстно-вільні мови, контекстні мови, 0-мови, машини Тьюринга, текстові й графічні описи мов.

guages are investigated. The formal descriptions of a multitude of linguistic examples are given. The graphic metadescription equivalent to the given text description of the interpreter of the universal Turing machine is developed. The description provided of the universal Turing machine interpreter can be applied to solving the task of modelling the behavior of any T-machine (which performs the task of data conversion on its memory tape). The formal definition of the translator of the determined Turing machine is given. The ability to describe in the metalanguage of normal forms of knowledge the interpreting-translating and the translating-translating processes of formulating and solving of a given task, for which a solution exists is substantiated and defined.

Key words: meta-language, regular languages, context-free languages, context languages, 0-languages, Turing machines, formal description of languages.

УДК 519.164:004.4

UDC 519.164:004.4

Рамочная модель адаптивного композитного сервиса в среде Семантического Веба / О.А. Слабоспицкая. – С. 51–65.

Обоснован новый подход к on-line композиции такого семантического Веб-сервиса как адаптивный семантический сервис (АКС), пригодный к изменению поведения для удовлетворения новых требований и приспособления к новым (не)предусмотренным ситуациям, и применимый третьими сторонами, для произвольного метода композирования. Представлена рамочная модель АКС как динамической линии изменяемых семантических сервисов для потребителей в целевой предметной области. Сформулированы основы построения АКС за счет управления динамической вариабельностью линии. Разработана диагностическая модель вариабельности АКС для выявления потребностей и стратегий его on-line адаптирования. Формализованы операции реализации

Reference model for Semantic Web adaptive composite service / O.A. Slabospitskaya. – P. 51–65.

An innovative Approach is substantiated for such a Semantic Web Service (ACS) on-line composing that is adaptive – able to change its behaviour to meet new requirements and to fit new (un)foreseen situations – and applicable by third parties for any composing method. Reference Model is presented for ACS being considered as Dynamic Line of changeable Semantic Services for customers in target domain. The Basics for ACS engineering through Dynamic Variability Management over the Line are stated. ACS Diagnostic Variability Model to clarify its on-line adaptation Needs and Strategies is elaborated. The operations are formalized to implement the Strategies – namely, (un)anticipated change of the Component Services set, their interim Compositions' structure and functions, ACS itself functions – with the isomorphism proposed between the Models of its func-

стратегий – (не)предвиденного изменения состава компонентных сервисов, структуры и функций их промежуточных композиций, функций самого АКС – за счет предложенного изоморфизма между моделями его функций и, соответственно, композитных сервисов.
Внедрение подхода способствует повышению эффективности применения и реинжиниринга деловых процессов с разнородными и изменчивыми контекстами.

Ключевые слова: адаптивный композитный сервис; семантический Веб-сервис, динамическая линия программных продуктов, динамическая вариабельность,, метод композиции семантических Веб-сервисов, поиск сервисов, операция адаптирования, изоморфизм, сервис-ориентированная программная система.

УДК 004.94

Определение и решение задачи поиска Веб-сервисов с использованием аппарата дескриптивных логик / О. Захарова. – С. 66–78.

В настоящее время Веб-сервисы позволяют решать конкретные бизнес-задачи, реализующие бизнес процессы в разных сферах жизнедеятельности человека. Но для того, чтоб получить выполняемый Веб-сервис, необходимо уметь эффективно решать целое множество задач самих Веб-сервисов на всех этапах их жизненного цикла. Аппарат дескриптивных логик, благодаря своим механизмам суждений и возможностям логического вывода и придания описаниям семантического содержания, является эффективным и мощным инструментом для решения задач Веб-сервисов. Цель данной работы заключается в определении цепочки задач Веб-сервисов на функциональном уровне и подходов к их решению с использованием аппарата дескриптивных логик.

Ключевые слова: семантический Веб-сервис, дескриптивная логика, задача

tions and, respectively, Composite Services.
Putting Approach into practice enables effectiveness and efficiency increasing of business processes with the heterogeneous and unstable contexts usage and re-engineering.

Key words: adaptive composite service, semantic Web service, dynamic software product line, dynamic variability, semantic Web services composition method, service discovery, adaptation operation, isomorphism, service-oriented software system.

UDC 004.94

Defining and resolving Web-services discovery problems using description logics formalism / O. Zakharova. – P. 66–78.

Now Web-services allow to solve the business-problems that realize business-processes in different areas of human activities. But it is necessary to solve a lot of problems of Web-services on all stages of their life cycle to obtain executed Web-service. Description logics formalism is the effective and powerful tool to solve problems of Web–services due to its reasoners and abilities of logical inference and giving semantics to descriptions. Goal of this research is to define Web-services tasks chain on functional level and to find approaches for their resolving with description logics formalisms.

Key words: semantic Web-service, description logics, discovery task, web-service search, web-services composition, semantic description, domain ontology, service ontology.

поиска, поиск веб-сервиса, композиция веб-сервисов, семантическое описание, онтология домена, онтология сервиса.

УДК 004.822,681.3,519.81

UDC 004.822,681.3,519.81

Експертно-аналітичній процес вибору управляючих організаційних дій з використанням корпоративного знання. Частина І. Формальний аналіз системи цілей / О.П. Ільїна. – С. 79–88.

The expert analytic process for the choice of the organization management actions using corporative knowledge. Part I. Formal analysis of the goals system / E.P. Ilina. – P. 79–88.

Робота присвячена моделям та методам підтримки експертного вибору варіанта керуючої дії, яка реалізується організаційними заходами. В першій частині подано модель системи цілей організації та методи аналізу відношень еквівалентності, протилежності та достатності цілей. Такий аналіз здійснює підтримку аналітичної функції формування контексту та рекомендацій для експертно-аналітичного процесу, який розглядатиметься у наступній частині статті.

Ключові слова: підтримка прийняття рішень, організаційне рішення, модель знань про організацію, система організаційних цілей, експертний вибір, оцінка досяжності цілі, перспективність організаційної дії.

Paper is devoted to the models and the methods of support for the special process of the alternative management action expert choice when the action has to be realized by organizational measures. Such the choice takes into account the influence of the action on the decidable problem situation and its probable negative aftereffects in organization activity field. In Part I the model of the organization goals is presented. This model includes such the types of goals as standard, planned, produced by stakeholders' interests and brought about the decisions made in organization ones. The goals included are both hard and soft ones. The properties of goals such as equivalence, opposition and accessibility are formalized by means the model proposed. The analysis of these properties that is realized under actual state of the decision field and goals field of organization is employed as important analytical support for the expert analytical choice process. It includes the functions of the context and recommendations forming for the action perspectivity expert evaluation that will be considered in the next part of this paper.

Key words: decision making support, organization decision, organization knowledge model, system of organization goals, expert choice, evaluation of goal accessibility, perspectivity of organization measure.

УДК 004.02+004.05+005.93+ 510.3

Задачи управления рисками информационной безопасности аппарата принятия решений / Е.С. Родин. – С. 89–97.

Предложен перечень задач, решение которых позволяет создать модель зависимости уровня риска информационной безопасности от наличия связей, степени влияния и последствий реализации уязвимостей и угроз на отдельный информационный ресурс и организацию в целом. Приведены примеры построения для отдельного информационного ресурса деревьев связи между уязвимостями, угрозами и последствиями событий информационной безопасности, а также примеры математической формализации зависимости уровня риска реализации отдельной угрозы повреждения или потери информационного ресурса от состояния уязвимостей и их последовательного влияния на данный ресурс.

Ключевые слова: риск, информация, безопасность, угроза, уязвимость, нечеткость, логика, модель.

UDC 004.02+004.05+005.93+ 510.3

Tasks for information security risks management in making-decision process / Y.S. Rodin. – P. 89–97.

The author has submitted and tried to perform the number of tasks helping in modeling multi-factors information security risk management system. The author has prioritized financing different vulnerabilities by combining two approaches: processing approach in building events tree and mathematical formalizing the connections and affections levels of different events (vulnerabilities and threats) on information resource.

Key words: risk, information, security, threats, vulnerability, fuzzy, logic, model.

УДК 612.51.001.57+519.6

Симулятор пульсуючого серця / Р.Д. Григорян, А.Г. Дегода, Е.А. Джуринский, Д.С. Харсун. – С. 98–108.

Для сумісних з IBM персональних комп'ютерів створено програмний симулятор (ПС), заснований на математичній моделі (ММ) гемодинаміки людини. ММ описує серцево-судинну систему (ССС) в зосереджених параметрах і на тлі фіксованого нейрогуморального фону, що відповідає стану спокою. Призначення ПС – імітація користувачем сценаріїв змін у стані ССС для обчислення реакцій гемодинаміки. Є два способи завдання симуляційного сценарію: або з списку, представленого в інтерфейсі користувача, або формуванням з комбінацій елементарних процедур. Планується

UDC 612.51.001.57+519.6

A simulator of a pulsatile heart / R.D. Grygoryan, A.G. Degoda, E.A. Dzhurinsky, D.S. Kharsun. – P. 98–108.

For IBM-compatible personal computers, a software simulator (SS) is created. SS is based on special mathematical model (MM) of human hemodynamics. MM represents the cardiovascular system (CVS) in lumped parameters and with a fixed neurohumoral background corresponding to the rest. By means of the SS the user can provide simulation scenarios including of changes in the initial state of the CVS for the calculation of the hemodynamic reactions. There are two ways to set the scenario: either from the list presented in the user interface, or from the combination of elementary procedures. In current version of the SS the

на базі ММ створити комплексну модель з урахуванням нейрогуморальної регуляції серцевого насоса, тонусу судин і об'єму крові. У даній версії ПС передсердя відсутні, серцева діяльність моделюється описом пульсацій шлуночків. ПС демонструє основні закономірностей фізіології ССС в малому і великому колах кровообігу. ПС позволяє візуалізувати трансформації гемодинаміки при дефектах серцевих клапанів, гіпофункції правого/лівого шлуночків (через погіршення коронарного кровотоку, патології кардіоміоцитів, гіпертрофії міокарда). Досліджено вплив швидкості і потужності скорочення правого або лівого шлуночків серця на гемодинаміку. Після розробки комплексної моделі на базі ПС буде створена інформаційна технологія, орієнтована на кардіолога. ПС комбінує програми на C ++ і Java.

Ключові слова: математична модель, шлуночки серця, клапани, кардіологія, навчальний процес, інформаційна технологія.

atriums are absent, the cardiac activity is modeled by the description of ventricular pulsations. SS demonstrates the main regularities of the physiology of CVS in small and large circles of the circulation. SS visualizes hemodynamic transformations caused by defects of the heart valves, right / left ventricular hypofunction (due to worsening of the coronary blood flow, pathology of cardiomyocytes, myocardial hypertrophy). The hemodynamic effects caused via variations of the rate and power of contraction of the right or left ventricles were studied. It is planned to create a complex model based on MM, but taking into account the neurohumoral regulation of the heart pump, the vascular tone and blood volume. Such a complex model will be a problem-oriented information technology for the cardiologist. SS is programmed in Java.

Key words: mathematical model, ventricles of the heart, valves, cardiology, educational process, information technology.

УДК 681.03

UDC 681.03

Использование компьютерного зрения в системе цифровой нарезки материалов / В.В. Туманов, А.Ю. Дорошенко. – С. 109–118.

Предложен подход к реализации системы компьютерного зрения для распознавания и позиционирования объектов на резательной поверхности станков для цифровой нарезки материалов с помощью фотоснимка их рабочей поверхности вместе с промаркированными объектами нарезки на ней. Разработаны алгоритмы работы модулей системы, которые отвечают по калибровке камеры и распознаванию регистрационных марок двумя принципиально различными способами, которые дополняют друг друга. Также разработан алгоритм идентификации объектов нарезки на основе применения элементов теории графов.

Ключевые слова: компьютерного зрения, цифровая нарезка, OpenCV, калибрование камеры, распознавание объектов.

The usage of computer vision in the system of digital cutting of materials / V.V. Tumanov, A.Yu. Doroshenko. – P. 109–118.

An approach to the implementation of the computer vision system for recognizing and positioning objects on the cutting surface of machines for digital cutting of materials with the help of a photograph of their working surface with the marked cutting objects on it is proposed. Algorithms of the work of the system modules, which are responsible for the calibration of the camera, the recognition of registration marks by two fundamentally different methods, complemented each other, are developed. Also, an algorithm of identification of cutting objects on the basis of the application of elements of the graph theory is proposed.

Key words: computer vision system, digital cutting, OpenCV, camera calibration, object recognition.

ДО УВАГИ АВТОРІВ!

У журналі "Проблеми програмування" публікуються наукові матеріали, які раніше не публікувалися в інших виданнях.

Мова статті: українська, російська, англійська. Обсяг статті — від 6 до 16 сторінок формату А4.

Документ зберігається у форматі doc або docx. Ім'я подається транслітерацією, як прізвище автора (авторів), наприклад, «Petrenko.doc».

Автори можуть користуватися електронною поштою і також телефаксом для ділової переписки та передачі до редакції тексту статті та правки при коректурі. E-mail редакції: tsok@isofts.kiev.ua. FAX: +380 (44) 526 6263, Телефон: 526 5065.

1. Оформлення файлу з текстом статті.

При підготовці файлу використовуються: стиль нормальний (звичайний) або normal; шрифт Times New Roman, розмір шрифта 12 пт.; міжрядковий інтервал – 1,0; абзацний відступ – 1,25 см; вирівнювання – по ширині. У тексті не допускається вирівнювання пропусками; розстановка переносів – автоматична. Формат паперу А4, розміри полів документа – 20 мм. Текст статті після анотації має бути оформлений у 2 колонки, ширина яких – 7,86 см, а пробіл між ними – 1,27 см.

2. Послідовність розміщення та оформлення матеріалу статті.

УДК: індекс за універсальною десятковою класифікацією.

Автори: ініціали та прізвища авторів, курсив (світлий).

Заголовок 1 **(назва статті):** не містить абревіатур та строго відповідає змісту статті. Шрифт 15 пт, напівжирний, регістр верхній.

Анотація **(мовою статті):** 50–100 слів, не містить абревіатур, зрозумілих із змісту статті. Шрифт 10 пт, звичайний.

Ключові слова **(мовою статті):** не більше10 слів, не містить абревіатур, зрозумілих із змісту статті, подаються в називному відмінку, розділені комами. Шрифт 10 пт, звичайний.

Заголовок 2 **(назва розділу):** шрифт 14 пт, напівжирний; абзац із центральним вирівнюванням, без переносів. Заголовки нижчого рівня (пункти і т. п.) у самостійний абзац не виділяються і проходять першим реченням текстового абзацу, шрифт 12 пт, напівжирний.

Основний текст статті, має такі необхідні елементи:

постановка проблеми в загальному вигляді і її зв'язок з важливими науковими або практичними завданнями;

аналіз останніх досліджень і публікацій, у яких розпочато рішення даної проблеми і на які спирається автор, виділення невирішених раніше частин загальної проблеми, яким присвячується дана стаття;

формулювання цілей статті (постановка задачі);

виклад основного матеріалу дослідження з повним обґрунтуванням отриманих наукових результатів;

висновки з даного дослідження і перспективи подальших розробок у даному напрямку;

подяка (за наявності такої).

Формули створюються в редакторі Microsoft Equation 3.0 або MathType. Формули, на які є посилання в тексті, повинні мати наскрізну нумерацію. Номер формули друкується в круглих дужках біля краю правого поля. Розмір основного шрифту редактора формул – 12 пт. Розміри символів у формулах: звичайний – 12 пт, великий індекс – 9 пт, дрібний індекс – 7 пт, великий символ – 18 пт, дрібний символ – 11 пт. Не допускається масштабування формульних об'єктів.

Рисунки мають бути створені вбудованим редактором Word Picture або експортовані з прикладних програм Windows у графічних форматах (bmp, pcx, gif, jpg або tif). Рисунки розташовуються по центру. Нумерація рисунків здійснюється відповідно до порядку

згадування у тексті. Нумеровані підписи розміщуються під рисунком з позначенням «Рис. », далі вказується номер рисунка і текст підпису.

Таблиці мають бути підготовлені стандартним вбудованим в Word інструментарієм "Таблиця". Таблиці нумеруються за порядком згадування. На номер таблиці повинно бути посилання в тексті. Номер таблиці вказується в окремому рядку з вирівнюванням по правій стороні (наприклад, «Таблиця 1»). Назви таблиць розміщуються над таблицею з вирівнюванням по центру. Мінімальний розмір шрифту в таблицях – 11 пт.

Література**:** нумерований список джерел згідно ДСТУ 8302:2015 від 01.07.2016 р., шрифт 11 пт, відступ: спеціальний, навислий, 0,63 см.

Література англійською мовою (References): список використовуваних джерел згідно ***Harvard Style***. Джерела з заголовками на латиниці наводяться без перекладу. Для літератури джерел на мовах, що не використовують латинський алфавіт, необхідно забезпечити переведення назв джерел і вказати після них у дужках мову оригіналу. Прізвища та ініціали авторів, слід транслітерувати за правилами як для закордонного паспорта. Приклади оформлення бібліографічних посилань згідно з вимогами ***Harvard Style*** наведені в багатьох публікаціях, наприклад, за електронною адресою http://www.staffs.ac.uk/assets/harvard_referencing_examples_tcm44-39847.pdf

Дані про авторів**:** мають починатися рядком "Про авторів:", напівжирний курсив. Далі вказуються для кожного з авторів ПІБ повністю, наукове звання, посада, адреса, кількість публікацій в українських виданнях (приблизна), кількість публікацій в зарубіжних індексованих виданнях (приблизна), індекс Хірша (за наявності), обов'язково номер ORCID (сайт ORCID http://orcid.org/).

Дані про місце роботи авторів**:** починаються рядком "Місце роботи авторів:", напівжирний курсив. Далі вказуються місце роботи, адреса, телефон, факс, електронна пошта, контактний телефон.

3. Оформлення файлу з анотаціями.

Файл з анотаціями містить інформацію двома мовами (наприклад, якщо стаття написана на українській мові, то анотації та ключові слова – на російській та англійській мовах) та має бути оформлений у дві колонки: УДК (шрифт – 8 пт); назва статті (шрифт – 12 пт, напівжирний); прізвища та ініціали авторів (шрифт – 12 пт); текст анотації, ключові слова (шрифт – 10 пт).

Вимоги до анотації англійською мовою: обсяг від 100 до 250 слів, інформативність, оригінальність (не є калькою української або російськомовної анотації), змістовність (відображає основний зміст статті і результати досліджень), структурованість (дотримується логіки опису результатів у статті).

Документ зберігається у форматі doc або docx. Ім'я подається транслітерацією, як прізвище автора (авторів), наприклад, «Petrenko_Annot.doc».

Примітка: Підписний індекс журналу "Проблеми програмування" – **90853**.