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

№ 1

Теми випуску:

- ***Теоретичні та методологічні основи програмування***
- ***Моделі та засоби систем баз даних і знань***
- ***Моделі та засоби паралельних і розподілених програм***
- ***Експертні та інтелектуальні інформаційні системи***
- ***Архітектура програмного забезпечення***
- ***Математичне моделювання об'єктів та процесів***
- ***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 27.02.2017. Формат 60x84/8. Папір офс. Ум. друк. арк. 15,11. Обл.-вид. арк. 12,41. Тираж 120 прим. Ціна договірна. Замовл.

Друкарня Видавничого дому «Академперіодика» НАН України
01004 Київ-4, вул. Терещенківська, 4

Свідоцтво про внесення до Державного реєстру суб'єкта видавничої справи
серії ДК № 544 від 27.07.2001 р.

НАЦІОНАЛЬНА АКАДЕМІЯ НАУК УКРАЇНИ
ІНСТИТУТ ПРОГРАМНИХ СИСТЕМ

ПРОБЛЕМИ ПРОГРАМУВАННЯ

науковий журнал

№ 1 січень-березень 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

№ 1 January – March 2017

Founded in March, 1999

CONTENTS

УДК 004.424, 004.415

В.Г. Колесник

DS-ТЕОРИЯ. НАУЧНЫЕ АСПЕКТЫ И ПЕРСПЕКТИВЫ РАЗВИТИЯ

В работе представлена теория схем декомпозиции как научная теория. Описаны ее атрибуты – парадигма, поле исследований, цель исследований, основная задача, метод решения, основная теоретическая модель – схема декомпозиции. Утилитарная, практическая цель теории – предложить механизм генерации прикладных алгоритмов (не машинного кода). Показано, что схема декомпозиции как описание заменяет описание алгоритма и при этом остается декларативным в противоположность описанию алгоритма, как императивному. При этом схема декомпозиции есть описание, исходное для генерации алгоритмов. Описаны виды схем декомпозиции и операции над ними. Описаны алгоритмически релевантные факторы, которые следует учитывать при генерации алгоритмов для того, чтобы алгоритмы становились реальными. В работе описана работа по формализации и математическому описанию явлений и объектов теории схем декомпозиции. Предложен механизм контроля выводов и результатов теории. В работе также описано направление развития теории схем декомпозиции. Приоритетное направление соотносится с созданием и развитием системы емких понятий и абстракций. С точки зрения практического применения теории схем декомпозиции, предложен подход подобный тому, что есть в машиностроении – развитие и продвижение передовых технологий по запросу, по потребности – как результат целенаправленных исследований.

Ключевые слова: алгоритм, анализ, генерация, данные, декомпозиция, наука, программная инженерия, программирование, процесс, структура, синтез.

Введение

Несмотря на то, что в сфере программной инженерии накоплено множество фактов, разработаны методы, предложены обобщения и абстракции, – теории общего плана не существует. *DS*-теория, описанная в [1–5] – это попытка построить систематическую теорию для программной инженерии (для технологии программирования). В настоящей работе обсуждаются различные аспекты *DS*-теории как научной теории, развивающейся на основе единой парадигмы. Описаны факторы, являющиеся препятствием к применению *DS*-теории в практике. Принимая их во внимание, теоретические выводы можно применять в практике реального практического программирования. Проводится сравнение программирования с применением *DS*-теории с одной стороны и машиностроительным производством с другой. На основании этого сравнения обсуждается возможный эффект в сокращении издержек – именно то, что может дать индустрии программирования *DS*-теория как научная теория.

Атрибуты научной теории

Поле исследований. *DS*-теория объектом своего внимания имеет алгоритмы программ (или программных комплексов) во всех прикладных областях. Прежде всего, исследуется управление – конструкции из циклов, условий, подпрограмм. *DS*-теория занимается исследованием тех факторов, что порождают управление, структуру алгоритмических конструкций. Исследуются причины, из-за которых растет сложность алгоритмических конструкций.

Парадигма. Явно или неявно, в самом общем виде в разработке программного обеспечения существует одна теоретическая модель – некое подобие машины Тьюринга. С точки зрения этой модели, обрабатываемые данные являются абстрактными и безликими. А это значит, что основные элементарные операции над данными являются абстрактными, примитивными и бедными. Эта модель с точки зрения программной инженерии не формализована, но на ее основе построены текстовые процессоры, СУБД, компиляторы и

интерпретаторы, на ее принципах построено объектно-ориентированное программирование [6] и посредством него эта модель массово внедряется в разрабатываемые программы. В настоящее время эта модель превратилась в методологическое препятствие в развитии программной инженерии.

В *DS*-теории вместо модели "Машина Тьюринга" используется модель "Формирование знаний". Принципиальная ключевая исходная идея *DS*-теории (парадигма) звучит так: "В компьютере производятся новые знания". Эта парадигма принимается в *DS*-теории как аксиома, как исходное положение и противопоставляется другой парадигме: "В компьютере происходит обработка информации". Развитие основной идеи в следующем: "Знания имеют свою внутреннюю структуру и свой механизм их формирования – процедуру декомпозиции". И далее: "Алгоритм и программа в компьютере являются отражением процедуры декомпозиции".

Процедура декомпозиции известна и интуитивно понятна практически любому исследователю, так как это один из мыслительных инструментов исследований. Кроме того, схема декомпозиции – это то, что очень похоже на структурный анализ. Но в *DS*-теории акценты устанавливаются на процедуре формирования знаний, а не на наборе декомпонуемых конструкций и механизме управления ими.

Цель исследований. Главной целью *DS*-теории есть создание средств эффективного управления сложными программными проектами – борьба со сложностью. Понимая, что для многих проектов сложность – это естественное свойство, – это природа проекта и уменьшить ее не представляется возможным, то *DS*-теория нацелена на то, чтобы разработать средства для управления сложными проектами.

Производными целями *DS*-теории, которые могут быть достигнуты, есть те же цели, что определились как основные в индустрии программного обеспечения:

- повысить качество программ. Более конкретно – уменьшить количество ошибок на порядок;
- обеспечить сокращение времени программирования. Для этого уйти от программирования с использованием алгоритмических языков строчного типа к языкам графического типа;
- обеспечить сокращение издержек на разработку и сопровождение программ в разы.

Основная задача. Поставленные выше цели *DS*-теория предполагает с помощью решения основной задачи. Такая основная задача – это проектирование схемы получения новых знаний об объекте познавательной схемы (ПС). Построение этой схемы подразумевает существование трех обстоятельств. Первое то, что ПС будет записана строго формально. Второе – как правило, ПС будет сложной и структурированной, и будет состоять из более простых схем как компонент. И третье то, что ПС будет содержать повторяющиеся расчетные действия.

В рамках *DS*-теории дополнительно предпринимаются шаги для решения еще некоторых задач. Разрабатываются средства компактного описания алгоритмов и программ. А для этого ведется поиск емких понятий – основное продуктивное средство *DS*-теории. Компактное средство описания программ подразумевает последующую разработку графических визуальных средств описания, – это основная предпосылка для создания эффективной графики.

Метод решения. Для того, чтобы представить ПС в формальном виде как универсальную модель, используется формальная схема декомпозиции. Процедура построения включает эмпирические исследования ПС в прикладной области. В простейшем случае проектирование будет заключаться в том, чтобы только записать формально ПС (или, то же самое – решение прикладной задачи), которая реально существует в ручном расчете, либо в автоматизированном расчете, но с использование менее продуктивных технических средств. Возможно, что ПС уже записана и реализована в виде программы, но ее необходимо расширить функционально. Самый сложный вариант

– ПС не существует и ее необходимо спроектировать сначала при отсутствии прототипов.

Завершением решения будет преобразование ПС в реальный алгоритм, представленный на алгоритмическом языке или как исполняемая программа в машинном коде. Собственно, это есть генерация алгоритма на основе формального описания схемы декомпозиции.

Основная теоретическая модель. Основной инструмент теории или основная теоретическая модель – схема декомпозиции[1]. Простейшее описание схемы декомпозиции [1] включает дерево полной схемы декомпозиции (DPS), где для каждого узла K_i должен быть определен кортеж описания узла $KR_i = (T_i, A^a_i, A^s_i, DM_i)$, который включает:

T_i – перечень типов свойств.

A^a_i – аналитические алгоритмические зависимости.

A^s_i – синтетические алгоритмические зависимости (САЗ).

DM_i – частную схему декомпозиции (ЧСД).

K_i – идентификатор узла, где индекс отражает номер узла при обходе дерева сверху вниз слева направо. Аналитические и синтетические зависимости имеют общее название – алгоритмическая зависимость (А-зависимость). ЧСД предписывает способ разделения объекта или его части только на непосредственно составляющие их компоненты. ЧСД есть описание одного шага декомпозиции и отличается от полной схемы, тем, что последняя описывает полную декомпозицию. Полная схема декомпозиции описывает части объекта и его свойства сколь угодно глубокого деления или, с точки зрения дерева декомпозиции, может содержать произвольное количество уровней. Символьное описание схемы декомпозиции имеет вид:

$$DPS=\{K_0[KR_0]\ (K_1[KR_1]\ (\ldots), K_r[KR_r]\ (\ldots), \ldots, K_z[KR_z]\ (\ldots))\}.$$

Кортежи листьев не содержат ЧСД и САЗ. Схема декомпозиции изображается древовидным графом, в котором нагружены и ребра, и узлы. Ребра нагружены в том смысле, что каждая ЧСД и САЗ соотносится с одним конкретным ребром. А-зависимости представляются символьными строками, которые реализуют аналитические формулы, что может повлечь дополнительные грамматические конструкции строчного типа. Следует заметить, что А-зависимость более широкое понятие, чем формула и предполагает любой способ описания зависимости – аналитический, словесный, табличный и т. п.

Описание схемы декомпозиции – это описание статического объекта. DPS – это дерево, а А-зависимости и ЧСД – это описание отношений. Кроме этого при более детальном описании узлов схемы декомпозиции могут быть использованы опции и параметры. Схема декомпозиции есть симбиоз всех этих видов описаний. При описании схемы декомпозиции отсутствуют любые императивные утверждения.

Более детальное описание всего, что связано с узлом, это не просто кортеж как перечисление, а некоторая конструкция – алгоритмическая конструкция узла (АКУ). Ее компоненты – А-зависимости и ЧСД сочленены в единое целое. Структура АКУ зависит от вида и количества компонент и способа их сочленения. Тот порядок, в котором сочленены компоненты АКУ и вид сочленений определяют предварительный порядок расчета – реализации А-зависимостей. Точный порядок расчета будет зависеть от многих факторов, которые задаются опциями и параметрами. АКУ более развитое понятие чем узел с кортежем.

Для реализации схемы декомпозиции (СД) в [1] предложен канонический алгоритм (КА) – это универсальный алгоритм, не зависящий от структуры DPS. Он может содержать произвольное количество узлов, ветвей, уровней, а у любого узла

[1] В научной среде есть понимание того, что для развития программной инженерии как научной дисциплины необходима теоретическая модель как некая обобщающая родовая конструкция для как можно большей совокупности алгоритмов и алгоритмических конструкций [7, 8].

может быть произвольное количество исходящих ветвей. Универсальность его в следующем:

- с каждым не конечным узлом соотносится набор определенных видов компонент (с листьями соотносится меньший набор видов компонент);
- задан определенный порядок реализации А-зависимостей – обход дерева сверху вниз слева направо.

КА как порядок выполнения некоторых действий не зависит от прикладной области. Кроме того, каким бы ни были DPS и А-зависимости, которые соотносятся с узлами DPS, в результате выполнения должен быть замкнут контур синтеза (КС) [1]. Это является критерием правильности КА.

АКУ, несмотря на то, что набор видов компонент определен и ограничен, не универсален. Структура АКУ зависит от вида, количества и взаимодействия А-зависимостей, от того как сочленены компоненты АКУ. Все перечисленные факторы порождают многообразие АКУ.

ЧСД вместе с понятиями “КА”, “Р-данное”, “А-зависимость” и “АКУ” представляет кластер понятий. В таком смысле ЧСД с концептуальной точки зрения есть базовое понятие подобно тому, как базовым есть понятия “класс” и “объект” в объектно-ориентированном программировании. ЧСД как базовое понятие имеет методическую ценность, поскольку на общем уровне указывает направление проектных действий. В плане методического эффекта ЧСД – это предпосылка для создания системы программирования на основе *DS*-теории существенно превосходящей ООП в плане производительности.

Практическое использование СД. Любая ПС или любая задача в прикладной области рассматривается с точки зрения того, является ли она схемой декомпозиции. То есть, СД рассматривается как универсальный шаблон.

Если ПС уже существует в каком-либо виде, в виде программы или ручного расчета, то анализ заключается в том, чтобы увидеть, может ли быть представлена эта программа или расчет в виде СД. Если это возможно то, первый этап выполнен. Далее предстоит превратить СД в работающую программу. Если ПС не существует ни в каком виде и ее следует проектировать, то она проектируется и представляется в виде СД. В процессе анализа может оказаться, что ПС может быть представлена в виде СД только частично – только как фрагмент СД и ее необходимо дополнять ручным или автоматизированным расчетом. В практике это обычное явление.

Как правило, и существующие ПС, и такие, что проектируются сначала, в качестве результата содержат более одного данного. Каждое из искомых данных имеет уникальную схему расчета. В таком случае в СД совмещаются, насколько возможно, схемы расчета для всех искомых данных. То есть, конечная СД является результатом синтеза более одной СД.

Далее формальное описание СД используется как заказ на генерацию. Возможность генерации алгоритмов и программ есть неотъемлемой частью *DS*-теории.

Объект исследований

СД – это универсальное средство в руках человека и применима к исследованиям и решению различных задач практически во всех сферах деятельности человека. Тем не менее, между СД в различных прикладных областях есть различия. Эти различия порождают различия в прикладных программах и алгоритмах. Дополнительный источник многообразия в алгоритмах исполняемых программ – это вычислительная среда, в которой данные программы работают.

Виды СД и ее компоненты. Декомпозиция – это один из основных методов, применяемых человеком для исследования и преобразования окружающей действительности. Человек, выполняя декомпозицию, преследует следующие цели:

- исследование существующего объекта. Расчленяя (разбирая, раскладывая) объект или вычленяя компоненты объекта, человек намерен узнать новые неизвестные ему ранее, свойства, новые знания об объекте. Это может быть его конечной целью;

- создание нового объекта. Проектирование объекта мысленно или с помощью каких-либо средств, предполагает создание проекта. Последовательно могут создаваться эскизный проект, технический, рабочий. Но этот процесс также есть и декомпозиция. Создание проекта есть мысленная композиция;
- преобразование существующего объекта. Преобразование может быть вызвано необходимостью ремонта или модернизации объекта. В результате преобразования предполагается, что объект получит новые свойства. Преобразование предполагает разборку или расчленение объекта, а затем удаление или замена некоторых старых компонент или (и) добавление новых. В результате этого преобразования объект получает новые свойства или новые функции, что есть то же самое. В этом процессе есть и декомпозиция, и композиция.

Эти три цели из-за сочетания различных ЧСД при их достижении порождают алгоритмическое многообразие. Поэтому одной из функций *DS*-теории есть исследование и типизация как полных схем декомпозиции, так и ЧСД.

Декомпозиции может быть подвергнуто все, куда устремляется осмысленный познавательный взгляд человека. Декомпозиция предшествует всему, что намерен произвести человек, для последующего использования. Поэтому объектов декомпозиции в реальном мире весьма много. Много также различных видов и аспектов декомпозиции. Ей могут подвергаться:

- объекты живой или неживой природы, или продукт человеческого труда (артефакт);
- процесс, как динамический объект или предмет (организм), как статический объект;
- объекты начиная от микромира до объектов макромира;
- объекты, которые подвергаются декомпозиции в соответствии с заранее составленным планом или объекты, схема декомпозиции которых, может меняться в процессе исследования;
- объекты техники и объекты искусства;
- объекты, на которые воздействуют или объекты, за которыми наблюдают без воздействия;
- объекты, которые существуют реально или только в сознании исследователя;
- процесс реальный или материальный и процесс, полностью моделируемый в компьютере (различного вида тренажерные программы или динамические игры).

Декомпозиция может выполняться с помощью технических средств полностью или частичным их использованием. Также декомпозиция может учитывать результаты предшествующих шагов или изменение свойств объекта или не учитывать.

Объектами декомпозиции могут быть информационные объекты, полностью размещаемые в компьютере. Схемы декомпозиции и композиции таких объектов – это прототипы таких программ как браузеры, функционально-развитые редакторы текстов или графических изображений, издательские системы, системы САПР, игровые программы, различные модели. Из-за их сложности их специфика такова, что порой они до конца не бывают отлажены в принципе.

Все это многообразие объектов и видов декомпозиции порождает многообразие ЧСД и САЗ. Однако *DS*-теория акцентирует внимание на алгоритмических аспектах ЧСД и САЗ. Важным есть то, что в процессе представления в виде алгоритмов порождает циклы, условия – то, что порождает управление в алгоритме.

Кроме того, использование компьютеров при выполнении декомпозиции порождает дополнительные вариации в алгоритмах процедур декомпозиции. С абстрактной точки зрения основной работой, которая выполняется в процессе декомпозиции, есть работа с данными (значения свойств) – это сбор, фиксация, генерация, вывод данных. Более детально это может выглядеть так:

- данные расположены в файлах или в базах данных. В процессе декомпо-

зиции их только вводят в компьютер для обработки;

- данные снимают с датчиков и вводят в компьютер в реальном режиме времени. Данные вводятся либо по мере поступления, либо в соответствии с некоторым регламентом;
- декомпозиция моделируется в компьютере. Данные при этом формируются в соответствии с некоторой функциональной или более того, в соответствии с алгоритмической зависимостью и после выводятся на управляющие устройства, либо в базы данных.

Возможны сочетания этих вариантов. Эти три варианта работы алгоритмов схем декомпозиции и их возможные сочетания тоже являются основаниями для исследования и типизации ЧСД.

Разнообразие СД есть еще и в том, что различается степень участия человека (и компьютера) в реализации расчетов СД. Диапазон взаимного участия в комплексе человек-компьютер распространяется от полностью автоматического расчета, до полностью человеческого расчета с использованием калькулятора на экране монитора компьютера или просто чтение текста на планшете. Полностью автоматизированный расчет – это значит, то вся СД полностью вычисляется в компьютере. Или что, то же самое, КС полностью вычисляется в компьютере. В этом отношении есть вариации двух видов:

- СД, начиная с корневого узла и до какого-то уровня, вычисляется в компьютере. Вне компьютера вычисляется часть СД, начиная с узлов определенного уровня и ниже. По разным ветвям уровни, где завершается автоматизированный расчет, могут быть различными. Для такого типа СД необходимо обеспечить вывод данных для последующего ручного расчета;
- СД, начиная с корневого узла и до какого-то уровня, вычисляется вне компьютера. В компьютере вычисляется часть СД, начиная с узла следующего уровня и ниже. Для такого типа СД необходимо обеспечить ввод данных для последующего ручного расчета.

Процесс расчета СД может быть разделенным во времени. Данные частично сформированы ранее и хранятся в сети, в базах данных или в Интернете – это первая часть расчета, а потом эти данные используются для расчетов – это вторая часть схемы.

Процесс расчета может инициироваться человеком или компьютер может быть в режиме ожидания, а расчет в соответствии с СД инициируется неким процессом. СД с подобными вариантами запуска расчета могут быть синтезированы в одно целое.

Таким образом, СД и подвергаемые ей объекты есть объектами исследований в *DS*-теории. Более точно, объектом исследований *DS*-теории есть алгоритмическое представление СД.

Операции над СД. Существует несколько операций над СД[2]: дополнение, усечение, сцепление и три вида синтеза. Синтез как дополнение А-зависимостей; синтез как совмещение СД и синтез схем декомпозиции и композиции.

1. Дополнение. Данная операция означает, что в процессе исследования объекта есть необходимость сделать более глубокую декомпозицию. К существующим компонентам объекта применяется более глубокая декомпозиция. С точки зрения дерева, с помощью которого представлена СД, это значит, что появилась еще одна ветвь – дерево дополнено еще одной ветвью. Хотя дерево исходной схемы может быть дополнено схемой, у которой ветви более чем на одном уровне.

2. Усечение. Эта операция означает, что в процессе исследования объекта исчезла необходимость делать глубокую декомпозицию. Дерево СД стало короче на одну или более ветвей.

3. Сцепление. Данная операция значит, что в процессе исследования объекта необходимо выполнить дополнительный расчет (один или более). С точки зрения дерева схемы декомпозиции это значит, что добавляются дополнительные ветви на любом узле.

[2] Первые три из перечисленных операций и один из видов синтеза описаны в [1].

4. Синтез как добавление А-зависимостей. Эта операция означает, что к объекту необходимо применить еще одну СД с механизмом декомпозиции, который был применен ранее и выполнить расчет в соответствии с дополнительной А-зависимостью. Дерево исходной СД в этом случае не меняется.

5. Синтез как совмещение СД. Данная операция означает, что к объекту необходимо применить более одной СД с различными механизмами декомпозиции. Механизмы декомпозиции различаются тем, что делят объект на части, размеры которых не совпадают. То есть, для каждой схемы уникальный размер отделяемой части. Хотя размеры частей могут различаться и для одной схемы.

6. Синтез схем декомпозиции и схем композиции. Как правило, работающая программа одновременно управляет процессом декомпозиции и процессом синтеза или композиции нового объекта. Так как любой процесс (и декомпозиции и композиции) может быть представлен СД (и, соответственно, деревом), то конечный работающий в компьютере процесс есть результат синтеза СД (и, соответственно, синтеза деревьев). Если быть предельно точным, то простейший алгоритм – чтение файла по записям с одновременным выводом записей в другой файл тоже есть результат синтеза двух схем. Одна из этих схем изображает чтение файла, а другая – вывод. Но более важным является синтез СД, которые, будучи результатами синтеза, уже являются весьма сложными многоуровневыми деревьями. Алгоритмы, строящиеся традиционными методами для выполнения подобных функций, которые могут быть выполнены подобными СД, порой с трудом поддаются осмыслению и контролю[3].

[3] По мнению многих программистов, суть программирования – построение сложных конструкций из условных операторов, операторов цикла, – построение конструкций управления и их завершенная отладка. Но ни одна из предложенных ранее систем программирования или методологий не пыталась системно решить задачу генерации управления. DS-теория предпринимает такую попытку.

Эффективное использование операций над СД предполагает их типизацию и изучение с точки зрения их алгоритмической природы. Возможны ситуации, когда в каких-то случаях ЧСД или СД необходимо совмещать (или синтезировать) с САЗ. Это есть дополнительная причина исследования и типизации ЧСД и САЗ.

Синтез алгоритмов. Операции над СД уже есть синтез алгоритмов и, соответственно, программ. После того как СД синтезирована, *DS*-теория обеспечивает ее преобразование в алгоритм и программу на алгоритмическом языке. Дальше нет препятствий для того, чтобы преобразовать эту программу в исполняемую программу.

СД имеет алгоритмическую природу и именно поэтому один из видов ее описания, в котором представляется СД есть КА. Одновременно КА – это заказ на основании, которого производится синтез конечного алгоритма и программы. Синтаксические структуры КА и программы, и взаимосвязь их синтаксических компонент показаны на рис. 1.

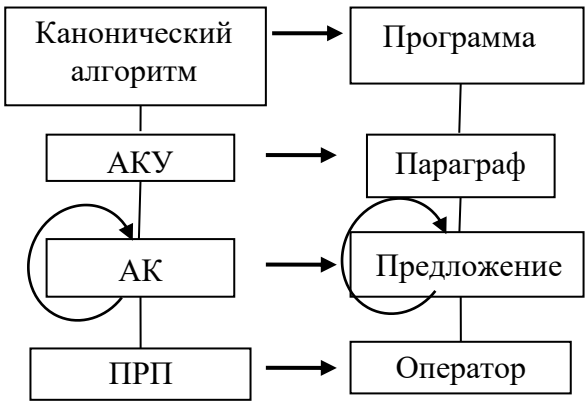


Рис. 1. Взаимодействие синтаксических конструкций СД и конструкций алгоритма

Простейшим элементом из которого составляется КА – это простейшая расчетная процедура (ПРП) [2]. Из ПРП составляются алгоритмические конструкции (АК) [2]. АК может иметь произвольно большой размер, так как структура ее рекурсивна. Компонентами АК могут быть как ПРП, так и АК, последние тоже могут

быть составными и т. д. Из АК составляется АКУ. Последние сочленяются иерархическим сочленением и формируют канонический алгоритм.

ПРП и АК могут быть условными или безусловными. Условные и безусловные АК и ПРП могут быть зависимыми или независимыми. АК и ПРП могут сочленяться последовательным, условным или альтернативным сочленениями.

В процессе генерации ПРП порождают операторы, АК – предложения. Операторы и предложения тоже сочленяются условным, альтернативным и последовательным сочленением. Операторы и предложения тоже могут быть условными и безусловными, а также зависимыми или независимыми.

Структурно из операторов и предложений может быть создано АТ-предложение [2]. Структура его определяется структурой А-фрагментов [2] на А-ленте [1]. Именно из АТ-предложений формируется параграф [2]. Параграфы с помощью иерархического сочленения составляют прикладной алгоритм и программу.

Состав АКУ и то, какой порядок размещения главного А-данного на входной и выходной А-лентах, – это информация достаточная для того, чтобы сгенерировать один параграф.

С точки зрения графического представления, СД – это дерево, с каждым узлом которого соотносится АКУ. СД используется как каркас или как скелет канонического алгоритма, а далее и как каркас прикладного алгоритма. Каждая АКУ порождает параграф. Алгоритм прикладной программы – это тоже дерево, узлами которого являются параграфы.

С точки зрения генерации прикладного алгоритма объектами исследования есть ПРП и их виды, способы сочленения ПРП и АК – различные вариации этих сочленений.

Описание СД в отличие от текста программы декларативно и является симбиозом текста и графики. При этом объем описания относительно меньше, чем объем текста программы.

Термин “синтез” в работе используется в сочетании с различными терминами, но обсуждается одно и то же явление. Термины зависят от контекста. Если обсуждается функциональный аспект создаваемой системы, то речь идет о синтезе процессов, иначе, если обсуждается структурный аспект, то объектом синтеза является СД. Если обсуждается внешний, изобразительный аспект – графическое изображение СД – деревья, то объектом анализа является синтез деревьев. Если обсуждается алгоритм – внутреннее представление создаваемой системы, то речь идет о синтезе алгоритмических конструкций – синтезе алгоритмов.

Алгоритмически релевантные факторы (АРФ)

В работе [1] перечислены причины, из-за которых от канонического алгоритма до реально работающей прикладной программы предстоит сделать еще ряд шагов. Эти шаги должны адаптировать алгоритм к реальным условиям обработки данных. Причины адаптации, следующие:

- алгоритмическая модель может быть сложной из-за того, что входных А-лент может быть более одной. Также и выходных А-лент может быть более одной. Источниками Р-данных могут быть базы данных, Интернет и т. п.;
- размер реальных данных, как правило, не совпадает с размерами ячеек на входных и на выходных А-лентах. Из-за этого необходимо включать в алгоритм блоки форматирования Р-данных;
- модель последовательного доступа, рассмотренная в *DS*-теории, к Р-данным самая простая. В практике электронной обработки данных есть более сложные методы доступа к данным;
- Р-данные могут поступать в режиме ONLINE от различных датчиков или в режиме диалога от оператора;
- Р-данные могут быть выведены на монитор, на различные управляющие устройства;
- Р-данные при размещении могут кодироваться, упаковываться и т. п.

В настоящее время *DS*-теория различает следующие направления работы с Р-данными порождающие группы факторов, которые усложняют канонический алгоритм на пути его преобразовании в реальный прикладной алгоритм:

- размещение Р-данных на А-ленте;
- механизм доступа к Р-данным;
- деление Р-данных;
- форматирование Р-данных;
- организация расчета;
- отражение Р-данных.

Размещение Р-данных на А-ленте. В *DS*-теории в качестве носителя Р-данных, рассматривается абстрактная А-лента. Р-данные размещаются в записях. Из записей формируются подобласти, которые могут быть вложены как компоненты в объемлющие их подобласти (рис. 2). Кроме того записи и подобласти могут объединяться и составлять А-фрагменты [2].

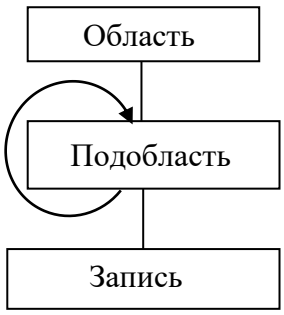


Рис. 2. Структура А-ленты

Записи и подобласти одного типа внутри объемлющей их подобласти могут быть упорядочены или нет. Записи, подобласти разных типов могут объединяться и составлять А-фрагменты. Внутри А-фрагмента записи или подобласти разных типов могут иметь порядок относительно его границ, относительно друг друга, а также иметь произвольное размещение и сочетание этих видов взаимного размещения. Такие же виды взаимного размещения возможны и между А-фрагментами. А-фрагменты могут быть как условными, так и обязательными.

В практике электронной обработки данных есть много различных видов структур хранения данных: базы данных, списки, потоки данных из локальной сети или из Интернета и т. п. Объектом исследования *DS*-теории есть виды структур хранения данных, и порядок взаимного размещения данных внутри подобных структур.

Соответствующая группа факторов определяет то, как размещаются Р-данных в подобластях на носителях-источниках.

Деление Р-данных. А-данные и С-данные, являющиеся операндами при реализации А-зависимостей, перед началом расчета могут быть разделены и размещены в различных записях или в различных подобластях. В практике электронной обработки данных это значит, что группы и совокупности данных, могут быть размещены в различных файлах. Файлы могут быть расположены как на одном носителе, так и на нескольких носителях. Соответственно, Р-данные, как абстракция реальных данных, тоже могут быть расположены на различных носителях.

Разделены могут быть А-данные и С-данные всех видов – простые, расширенные и сложные. Ситуации размещения фрагментов А-данных в областях на А-лентах возможны такие же, как и для А-данных, которые не разделены. А именно: все виды порядка размещения и все виды совместного размещения [2]. Аналогично ситуации размещения фрагментов С-данных в областях на А-лентах возможны такие же как и для неразделенных С-данных. В областях размещения фрагментов А-данных и С-данных наряду с последними могут быть Р-данные, которые являются непродуктивными для исходной СД.

Соответствующая группа факторов определяет то, сколько входных и выходных А-лент используется для хранения обрабатываемых Р-данных и то, как размещаются подобласти на этих А-лентах.

Форматирование Р-данных. В *DS*-теории внешним носителем есть А-лента. Порция хранения Р-данных, которая записывается и считывается с А-ленты, есть

запись. Размер записи не оговаривается. Предполагается, что запись по размеру совпадает с размером, сохраняемого в ней Эл-данного или простого А-данного. Из записей на А-ленте составляются подобласти. Но “запись” и “подобласть” – это не технические, а концептуальные понятия *DS*-теории. В практике электронной обработки данных обмен между компьютером и внешними устройствами выполняется блоками. Размер этих блоков или порций данных, как правило, не совпадает ни с размерами записей и подобластей, ни с размерами их содержимого – Эл-данных или простых А-данных. Блоки для хранения обмениваемой информации могут быть компонентами объемлющих их структур хранения. То есть, с технической стороны возможна иерархия порций или структур хранения информации на внешних носителях.

При выводе Р-данных на реальные носители информации записи абстрактной длины и подобласти должны быть вложены в тот формат, в котором они будут храниться на этих носителях, – должны быть форматированы. При вводе с реальных носителей информации должны быть восстановлены[4] абстрактные записи с Р-данными предназначенными для обработки.

Соответствующая группа факторов устанавливает размерные отношения между деревом типов свойств и описанием структуры подобластей с одной стороны и структурой хранения информации на реальном внешнем носителе с другой.

DS-теория учитывает то, что Р-данные могут сохраняться не только на устройствах с последовательным доступом, для измерения емкости которых, требуется одна координатная ось. Это могут быть двухкоординатные плоские экраны или различные трехкоординатные структуры.

Механизм доступа к Р-данным. При описании концептуального содержания *DS*-теории для размещения и доступа к Р-данным была использована абстрактная А-лента с записями и подобластями. Записи и подобласти – это прообраз линейных файлов с записями различных типов. Доступ к Р-данным на А-ленте простейший – последовательный. Но в практике электронной обработки данных большое разнообразие видов носителей. Соответственно, существует разнообразие способов организации данных и методов доступа к ним.

Р-данные могут размещаться в очередях различных видов. Если на устройстве доступен прямой доступ, то Р-данные, которые соотносятся с полной схемой декомпозиции, могут быть размещены в списках, что еще добавляет разнообразие в методах доступа. С помощью списков может моделироваться декомпозиция объектов принадлежащих пространствам с двумя, тремя и более размерностями.

Доступ к Р-данным может быть совмещен с одновременным кодированием или декодированием, с одновременным и архивированием или извлечением из архива. Доступ может предполагать передачу данных на большие расстояния.

К вопросам доступа с точки зрения *DS*-теории следует соотнести потребность в повторных проходах по коллекциям данных с последовательной организацией.

Факторы подобного рода составляют группу, подлежащую исследованию.

Управление расчетом. Еще одна группа факторов порождается тем, что процессами расчета необходимо управлять. Существуют различные причины и способы управления расчетами в компьютере, а именно:

- дублирование расчета на различных компьютерах;
- дублирование результирующей информации на различных носителях;
- отслеживание транзакций к базе данных выполняемых другими програм-

[4] Не существует термина, которым можно было бы обозначить процесс восстановления Р-данного – снятие или упразднение формы, в которой оно сохранялось. Наиболее близкое по смыслу понятие – деконструкция, но это из другой сферы деятельности. Понятия “распаковка”, “разархивирование”, “реформатирование” или “деформатирование” (unformat) – ассоциируются с этим процессом, но не совпадают с его сутью. В *DS*-теории используется понятие “реформатирование”.

мами. Сохранение протокола в обращении к сайту;

- прерывание и возобновление расчетов. Приостановка расчета с активацией (или без) расчета по другой программе;
- повторение расчетов с контрольных точек;
- распараллеливание расчетов;
- управление (манипулирование) окнами (мониторами);
- управление расчетами на больших расстояниях и т. п.

С точки зрения *DS*-теории в подобных ситуациях объектом декомпозиции есть сам процесс расчета. Возможность управления процессом расчета – это факторы, которые усложняют канонический алгоритм, в соответствии с которым протекает расчет. Помимо признаков, которые описывают ситуацию управления, существуют Р-данные, которые необходимы для управления процессом расчета.

Отражение на мониторе. А-лента – это носитель, измерение которого производится с помощью одной координатой оси, но для размещения информации используются различные табло, экраны и т. п. – для измерения поверхности изображения, которых, используются две координатные оси. В связи с выводом Р-данных на такие носители (и в связи с вводом с подобных носителей) определяется еще одна группа факторов. Они описывают размещение Р-данных в окнах, порядок вывода Р-данных в окно. Учитывая то, что на экран монитора могут быть выведены текст, графика – векторная и растровая, – видео, группа этих факторов обширна и влияние ее на канонический алгоритм весьма существенное.

В случае отражения на мониторах выводимые (и вводимые) Р-данные сами по себе могут быть объектами декомпозиции и поэтому надо исследовать не просто влияние факторов отражения Р-данных на канонический алгоритм, а синтез схем декомпозиции Р-данных.

Влияние АРФ на канонический алгоритм. В процессе адаптации канонического алгоритма может возникнуть необходимость учитывать факторы любой из вышеперечисленных групп. Группы, факторы, влияющие на канонический алгоритм, могут влиять каждая отдельно или в любом сочетании – даже все вместе. Перечень групп АРФ не завершен и вполне может быть дополнен новыми группами.

Особенные свойства *DS*-теории

Практически общепризнано, что программная инженерия нуждается в научной теории [7 – 12]. Далее приведены некоторые особенности *DS*-теории как научной теории.

Нововведения. В *DS*-теории вводится ряд понятий, позволяющих глубже понимать природу алгоритма и программы. Абстракция – одно из важных понятийных средств в программировании. Но исключительно важны *информативно емкие* абстракции, емкие понятия [13 – 16]. В *DS*-теории такими понятиями являются совокупное свойство, алгоритмическая зависимость, квалификационное предложение [1], СД и КС.

Понятие СД позволило разделить функциональность и управление алгоритма и программы. Введены понятия функционального ядра, алгоритмического фрейма, функционального содержания и алгоритмической матрицы [2].

Вместо понятия “данное” введены и используются понятия “Р-свойство” [1], “размещение”, “Р-данное” и “Д-данное” [2].

Введено понятие порядка при размещении Р-данных на внешних носителях. Р-данные могут быть перемешаны в областях и подобластях на носителях. То, каким образом они перемешаны – характер смешения, – определяется и описывается формально.

Разделены DPS и дерево алгоритма. Каждое из этих понятий может быть изображено деревом, но назначение этих описаний различное[5].

В *DS*-теории введено понятие “Квалификационного предложения”. Эта идея создает предпосылки манипулирова-

[5] Это был вполне ожидаемый шаг в развитии R-технологии [17] и JSP [18].

ния квалификационными предложениями. Подобно тому, как есть операции над схемами декомпозиции, так возможны операции над квалификационными предложениями. С этой точки зрения *DS*-теория есть альтернативой реляционной модели данных [19].

В *DS*-теории созданы условия для частичной классификации алгоритмов[6]. Группы АРФ определяют группы алгоритмов реализующие их. В *DS*-теории проводится исследование групп алгоритмов вместо проектирования универсальных классов или универсальных алгоритмов. Вместо того, чтобы исследовать многообразие алгоритмов, исследуются факторы[7], которые влияют на алгоритмы.

Разделение поля алгоритмов на группы позволяет оценить уровень издержек в каждой из групп и после этого уделить особое внимание исследованию групп критичных с точки зрения издержек[8]. Исследование каждой из групп АРФ сужает поле алгоритмов с точки зрения неопределенности и неуправляемости. Результат исследования внутри группы – это рекомендации по генерации алгоритмов внутри этой группы, это общие принципы генерации.

В *DS*-теории важным есть прозрачный механизм синтеза алгоритмов и содержимое библиотеки алгоритмических примитивов. Создание подобной библиотеки имеет аналог в машиностроении – это унификация узлов и деталей. Как для машиностроения унификация – это средство сокращения издержек, так для программирования создание библиотеки унифицированных примитивов – это средство сокращения издержек в построении алгоритмов. Хотя *DS*-теория создает предпосылки автоматической генерации алгоритмов, а не ручного проектирования. Механизм синтеза, кроме того, что он дополняемый и расширяемый, также открытый для тестирования подобно тому, как открыты для анализа доказательства теорем в математике.

Дополнительные ожидаемые результаты. Как упоминалось в [2], несмотря на то, что алгоритм как результат генерации, представляется с помощью алгоритмического языка императивного типа, СД, как описание исходное для генерации, является описанием декларативного типа. Сгенерированный алгоритм по определению отражает порядок вычислений, а описание СД явно этот порядок не показывает. В процессе генерации алгоритма учитывается порядок обхода DPS сверху вниз и слева направо и благодаря этому он появляется в сгенерированном алгоритме. По сути, СД является объектом декларативного программирования. *DS*-теория благодаря системе емких понятий позволяет создать систему умолчаний в программировании. Можно говорить об инкапсуляции алгоритма, об умолчании описания алгоритма. Но пока можно говорить только о последовательном приближении к такому стилю программирования по мере исследования видов СД и групп АРФ.

Декларативный стиль программирования обеспечивает описание статического объекта, при этом алгоритм – это описание динамического объекта. Последнее сложнее по сравнению с описанием статического объекта. Но и в этом отношении можно говорить только как о постепенном и последовательном приближении от одного вида описания к другому. Удельный вес статического описания будет увеличиваться, а динамического описания уменьшаться.

---

[6] В науке существует прецеденты подобной классификации. Например, идея “атомного веса” создала предпосылки к классификации химических элементов.

[7] Нечто подобное имело место в истории математики. Математики Абель и Галуа не стали искать универсальное решение алгебраических уравнений пятой, шестой и больших степеней, а исследовали свойства коэффициентов, от которых зависит решение. Это дало возможность ответить на вопрос о возможности нахождения корней в принципе. Такой же подход реализован при исследовании алгоритмов.

[8] Подобно тому, как для создания вакуума в замкнутом пространстве различными химическими реактивами связывают различные газы – компоненты воздуха, пока не достигают необходимого уровня, так исследуя фактор за фактором, от поля алгоритмов постепенно отсекают группы алгоритмов. Как один реактив связывает один газ, подобно этому один фактор выделяет одну группу из поля алгоритмов.

DS-теория ориентирована на создание (генерацию) уникального специализированного кода и объективно необходимого количества подпрограмм. А специализированный код – это более плотный и продуктивный код.

Математические аспекты *DS*-теории. В *DS*-теории предпринимаются попытки к тому, чтобы рассматриваемым явлениям, объектам и зависимостям давать в плане точности определения максимально приближенные к математическим определениям. Здесь есть все предпосылки к тому, чтобы *DS*-теория как теория прикладных алгоритмов стала частью математической науки подобно классической теории алгоритмов[9] [20].

В *DS*-теории прилагаются усилия к тому, чтобы любую изучаемую или порождаемую конструкцию понимать (или рассматривать) как результат операций над некоторыми примитивами. Таких видов конструкций есть несколько:

- любая СД – это либо примитив (ЧСД), либо результат операций над ЧСД;
- АК или АКУ – это, как правило, результат операций над ПСД [2]. Операциями есть последовательное, условное, альтернативное сочленения;
- АКУ являются операндом, к которому применяется иерархическое сочленение при построении дерева полной схемы декомпозиции;
- параграфы – это результат операций над операторами и предложениями. Операции – последовательные, условные, альтернативные сочленения.

Эти усилия формализации направлены на то, чтобы: а) конструкции, составляемые человеком, могли быть подвергнуты семантическому контролю – как предварительному шагу по устранению ошибок; б) обеспечить логический вывод алгоритмов.

Алгоритмические конструкции в реальных алгоритмах могут быть весьма сложные. Но любая подобная конструкция – это только очередной шаг в растущей сложности алгоритмов. В предшествующих шагах конструкции были проще. Последующие шаги в проектировании алгоритмов вполне могут быть отмечены более сложными алгоритмами. Усилия в *DS*-теории прилагаются с целью найти причины растущей сложности. А для этого требуется формализация и математизация процесса анализа алгоритмов.

Механизм контроля результатов. Контроль результатов, которые обеспечивает *DS*-теория, имеет два аспекта – результаты, получаемые в процессе исследований и результаты, получаемые в процессе генерации конкретной прикладной программы.

Результат исследований – это система алгоритмических примитивов и механизм синтеза алгоритмов. Так как теория имеет свою внутреннюю логику – развивается от единой парадигмы, выше объявленной, то контролироваться должен процесс построения системы примитивов и сама система, а также механизм синтеза. Это разовая процедура и она может быть выполнена только умозрительно (с карандашом в руках). Но эти объекты обозримы (в отличии от текстов многих реальных программ) и вполне могут быть доступны многим исследователям подобно тому как доступны для изучения и контроля доказательства теорем в математике.

Конкретная программа генерируется на основании описания конкретной СД и набора связанных с ней АРФ. На этом этапе возможен произвол того, кто составляет эти исходные данные и, соответственно, возможны ошибки. Здесь предусматривается проверка КС. Если КС замкнут, то это значит, что алгоритм программы корректен. Как упоминалось выше, для проверки КС исходные конструкции подвергаются семантическому контролю. Но следует понимать, если формулы или отношения между исходными данными – функциональность СД – некорректны по сути, то средствами *DS*-теории это определить невозможно.

[9] Исследование алгоритмов и групп алгоритмов в *DS*-теории проводится так же, как проводились (и проводятся) исследования алгоритмов в классической теории алгоритмов. Но при этом цели исследований различные. Для *DS*-теории приоритетная цель – синтез алгоритмов и работа по достижению этой цели выполняется системно.

И, самое главное, основной критерий корректности программы – это корректный результат расчета. *DS*-теория ориентирована на последовательное сужение поля ошибок, но средствами для полного их устранения не располагает.

Возможность развития. Исходя из текущего состояния *DS*-теории, рассматриваются следующие направления развития.

В настоящее время в различных сферах человеческой деятельности исследуются и проектируются объекты многомерные, динамичные, многофункциональные и многофакторные. Объекты могут находиться в микро или в макромире, в труднодоступных средах. Все это может быть источником разнообразия СД с точки зрения КА. Поэтому исследуются виды и компоненты СД и операции над ними. Особенно важны операции синтеза СД.

Выше в статье приведен перечень АРФ. В процессе поисковых исследований этот перечень может быть дополнен. *DS*-теория благодаря идее АРФ позволила разделить поле алгоритмов на секторы. Внутри каждого сектора очерчены (выделены) группы простых АК и механизмы синтеза реальных прикладных алгоритмов. Эти обстоятельства определяют одно из направлений исследований – поиск новых групп АРФ и анализ алгоритмов связанных с этими группами.

Объекты, подлежащие декомпозиции – это, прежде всего, процессы. Притом, это процессы, которые создаются, проектируются. Синтез процессов (проектирование процессов) – это наиболее важное и наименее развитое направление *DS*-теории.

С точки зрения конкретных прикладных программных комплексов *DS*-теория в настоящее время – это теория достаточно общего плана, хотя она уже применима на более конкретном практическом уровне. Существенный эффект может быть получен от *DS*-теории, когда будут разработаны специализированные схемы декомпозиции. Это в области АСУ [21], распределенной обработки данных – в тех областях, где большие масштабы программирования электронной обработки данных.

Еще одно из направлений исследования – это целенаправленный поиск изобразительных средств – двумерная графика. Но эффективная графика может быть получена только в том случае, если будет система эффективных емких понятий.

Сравнение программирования с машиностроением

DS-теория создает предпосылки для создания некоторого другого вида производства, по сравнению с тем, что реально существует в производстве программ в настоящее время. Речь идет о предпосылках создания индустриального производства программ. Можно проследить некоторые аналогии.

Разделение труда. В проектировании архитектуры программных систем, в проектировании алгоритмов и программировании нет глубокого объективного разделения труда. В машиностроении разделяется описание конструкции изделия и технологии производства. Соответственно, есть разделение на конструирование, технологическую подготовку производства и само производство. Сейчас программирование – это конструирование и изготовление одновременно. Должно быть:

- конструирование;
- технологическая подготовка производства;
- инструментальная подготовка производства;
- изготовление.

В машиностроении есть более глубокая детализация, но в программной инженерии желательно дойти хотя бы до этого уровня. Желаемого уровня разделения труда нет по той причине, что текст программы – это уже конечный продукт, а описания изделия не существует. Имеется в виду компактное и желательно двумерное описание изделия и притом детальное. Описание схемы декомпозиции – это описание конструкции, текст программы – конечное изделие.

Проблема программной инженерии заключается в том, что текст программы,

написанный с использованием традиционных строчных языках очень громоздкий и труднообозримый. В *DS*-теории предпринимается попытка решить эту проблему. То описание СД, которое приведено в этой и предыдущих статьях еще далеко от желаемого. Такое положение обусловлено тем, что в описании надо совместить сжатую компактную запись и детализацию[10]. Но предпосылки для создания компактного описания – это система емких понятий, предлагающиеся *DS*-теорией.

DS-теория предлагает и использует несколько схем деления алгоритмов. Разделяются дерево полной схемы декомпозиции и дерево алгоритма. Полная схема декомпозиции разделяется на АКУ. Разделяются дерево типов свойств и описание структуры областей размещения. Слои АРФ это еще одно направление деления алгоритмов [1]. Разделяются алгоритмическая матрица и функциональное наполнение. Следует также упомянуть тот факт, что операции над СД выполняются задолго до того, как начинается описание операционной среды, в которой будет реализована схема декомпозиции. Эти схемы деления создают объективные предпосылки к разделению труда в программировании.

Разработка и доработка изобразительных средств и генераторов алгоритмов должны быть подобны работе инструментального производства в машиностроении. В машиностроении постоянно разрабатывается оснастка, внедряются новые технологии с целыми техническими комплексами, что реализуют новые технологии. Подобно этому в программной индустрии постоянными должны быть доработка генераторов алгоритмов и изобразительных средств.

Целенаправленное и системное сокращение издержек. *DS*-теория создает условия для того, чтобы проводить исследования и разрабатывать (или дорабатывать) инструмент и среду проектирования, и не ждать когда появится очередной метод или технология [10, 11]. В машиностроении действует другой принцип – здесь не ожидается появление новых методов, но для трудоемких операций или переделов целенаправленно разрабатываются технологии, инструмент, оснастка. Если технологическая операция, технологический передел или технология в целом влекут большие издержки, то инженеры и исследователи планомерно ищут новые технологические средства, чтобы сократить эти издержки.

Именно для этого в *DS*-теории производится классификация. Предполагается, что сокращение издержек эффективно внутри групп алгоритмов, порождаемых АРФ, а также внутри групп вариаций канонического алгоритма порождаемых видам систем.

Конвейер или поточная линия [22] – это наиболее мощное средство экономии времени и издержек изобретенных в машиностроении и таким остается и в настоящее время. В программировании поточная линия в традиционном понимании невозможна. Но это не значит, что в программировании не может быть тотальной системной экономии издержек. Емкие понятия, система умолчаний и генерация алгоритмов – вот те ключевые факторы, что позволят целенаправленно и планомерно сокращать издержки в программировании.

Гносеологический аспект *DS*-теории. В промышленности протекают, активно взаимодействуя, два процесса: производство (изготовление продукции) – практика, и научные исследования – теория. Из производства наука пополняется фактами, задачами и проблемами. Теория обобщает, индуцирует факты, явления, а

[10] Первый и необходимый шаг в повышении производительности благодаря разделению труда в промышленности был сделан в арсеналах вооружений Англии, Германии, Франции, Швеции в XIX веке. Именно там были детальнейшим образом описаны и изображены на чертежах комплектующие компоненты изготавливаемого оружия. Именно полный и точный чертеж детали позволил записать технологию ее изготовления и тем самым создать условия к разделению труда. Чертеж детали обеспечил отчуждение производственной операции от субъективного мастерства человека и обеспечил дальнейшую специализацию производственных операций. Чертеж детали позволил моделировать как конструкцию, так и технологию, что позволило существенно сократить издержки изготовления.

затем дедуктивно поставляет в производство решения задач и проблем, это, как правило, новые технологии. Благодаря такой деятельности поддерживает активность науки. Взаимодействие этих двух процессов – источник постоянного развития промышленности. В индустрии программного обеспечения есть только один из упомянутых процессов – производство. Науки нет и, следовательно, практически нет развития.

Взаимодействие науки и производства далеко не безоблачно – возможны и сбои. Если новую задачу в науке не получилось разрешить, то поиск вполне продолжается и дальше. Теория может развиваться самостоятельно – не обязательно по запросу практики. Кроме того, могут быть ложные выводы, результаты могут получаться слабые или бесполезные (как может оказаться впоследствии). Это вынужденные издержки в науке. Но они не останавливают ее развитие. Несмотря на то, что негативные явления возможны и в отношении *DS*-теории, она должна развиваться.

Разработка *DS*-теории – это попытка создать параллельно текущий процесс научного развития для процесса производства программ. Сомнения может вызывать обобщенная модель *DS*-теории – СД. Можно сомневаться в том, насколько адекватны модели и алгоритмы прикладных программ, выведенных из обобщенной СД, какой уровень автоматизации или генерации алгоритмов, насколько алгоритмы адекватны спецификациям. Можно сомневаться в том, насколько *DS*-теория логически целостна и самодостаточна, но то, что научный процесс должен дополнить программную инженерию не должно вызвать сомнения.

Набор новых емких понятий, которые предложены *DS*-теорией, не предполагается завершенным. Наряду с тем как выше предложено организовать целенаправленный процесс сокращения издержек, так предлагается организовать целенаправленный и сознательный поиск и разработку емких понятий. Предпосылки для этого существуют.

Выводы

DS-теория представлена как научная теория. Описаны ее атрибуты – парадигма, поле исследований, цель исследований, основная задача, метод решения, основная теоретическая модель – СД. Описаны виды СД и операции на СД. Показано что СД есть описание, исходное для генерации алгоритмов. Также показано, что СД как описание заменяет описание алгоритма и при этом остается декларативным в противоположность описанию алгоритма, как императивному. Описаны АРФ, которые следует учитывать при генерации алгоритмов для того, чтобы алгоритмы становились реальными. В работе акцентируется внимание на формализации и математическому описанию явлений и объектов *DS*-теории. Предложен механизм контроля выводов и результатов теории.

В работе описано направление развития *DS*-теории. Приоритетное направление соотносится с созданием и развитием системы емких понятий и абстракций. С точки зрения практического применения *DS*-теории предложен подход подобный тому, что есть в машиностроении – развитие и продвижение передовых технологий по запросу, по потребности – как результат целенаправленных исследований. Новые методы программирования и синтеза алгоритмов должны быть результатом целенаправленных исследований.

1. *Колесник В.Г. DS*-теория как прототип теории прикладных алгоритмов. Проблеми програмування. 2012. № 1. С. 17–33.
2. *Колесник В.Г. DS*-теория. Представление канонического алгоритма с помощью алгоритмического языка. Проблеми програмування. 2015. № 1. С. 3–18.
3. *Колесник В.Г. DS*-теория. Исследование факторов деления Р-данных с целью генерации прикладных алгоритмов. Первая часть. Проблеми програмування. 2015. № 3. С. 3–12.
4. *Колесник В.Г. DS*-теория. Исследование факторов деления Р-данных с целью генерации прикладных алгоритмов. Вторая часть. Проблеми програмування. 2015. № 4. С. 3–13.

5. *Колесник В.Г. DS*-теория. Исследование факторов форматирования Р-данных. Проблеми програмування. 2016. № 4. С. 14–26.
6. *Бертран Мейер.* Объектно-ориентированное конструирование программных систем, 2-е издание. Русская редакция. 2005. 1204 с.
7. *Vincent Rosener and Denis Avrilionis.* Elements for the Definition of a Model of Software Engineering. OneTree Technologies. [Electronic resource]. URL: http://www.irisa.fr/lande/lande/icse-proceedings/ gamma/p29.pdf
8. More comments on SEMAT. [Electronic resource] // URL: http://blog.hexacta.com/more-comments-on-semat/
9. *Dewayne E. Perry and Don Batory*. On the Structure of General Theories of Software Engineering. – The University of Texas at Austin. [Electronic resource]. URL: http://users.ece.utexas.edu/~perry/work/papers/gtse-structure.pdf
10. *Ivar Jacobson and Ed Seidewitz.* What happened to the promise of rigorous, disciplined, professional practices for software development? Ivar Jacobson International. Communications of the ACM, Vol. 57, N 12. P. 49–5410.
11. *Victor R. Basili.* The role of experimentation in software engineering: past, current, and future. Proceedings of the Eighteenth International Conference on Software Engineering (ICSE), Berlin, Germany, March 1996.
12. *National* Science Foundation. Computing and Communication Foundations (CCF). The Algorithmic Foundations (AF) program. [Electronic resource] // URL: http://www.nsf.gov/cise/ccf/af_pgm15.jsp http://www.nsf.gov/funding/pgm_summ.jsp?pims_id=503299&org=CCF&from=home
13. *Dijkstra E.W.* On the role of scientific thought (EWD447). [Electronic resource] // URL: http://www.cs.utexas.edu/users/EWD/transcriptions/EWD04xx/EWD447.html
14. *Victor R. Basili.* Learning through Application. The Maturing of the QIP in the SEL – [Electronic resource] // URL: https://www.cs.umd.edu/~basili/publications/chapters/C29.pdf.
15. *Brooks F.P.* The Mythical Man Month – Silver Anniversary Edition, Addison-Wesley, 1995.
16. *Edmund M. Clarke, Jeannette M. Wing, Et Al.* Formal Methods: State of the Art and Future Directions. ACM Computing Surveys. 1996. Vol. 28, N 28. P. 626–643.
17. *Вельбицкий И.В.* Технология программирования. Киев: Техніка, 1984. 250 с.
18. *Jackson M.A.* Principles of Program Design. New York: Academic Press, 1975.
19. *Дж. Дейт.* Введение в системы баз данных. Восьмое издание. К.; 1328 с. Вильямс, 2008, 3 кв.
20. *Алферова З.А*. Теория алгоритмов. М.: Статистика. 1973. 164 с.
21. *Колесник В.Г., Семченко В.А.* Концептуальное проектирование сети ЭВМ для цеховой АСУ. Донецк: ИЭП АН УССР, 1989. 16 с.
22. *Генри Форд.* Моя жизнь, мои достижения. 5-е изд. Попурри, 2014. 352 с.

References

1. Kolesnyk V.G. DS-theory as a prototype of the theory of applied algorithms. Problems in programming. 2012. N 1. P. 17–33. (In Russian).
2. Kolesnyk V.G. DS-theory. Presentation of canonical algorithm by means of algorithmic language. Problems in programming. 2015. N 1. P. 3–18. (In Russian).
3. Kolesnyk V.G. DS-theory. Research of R-data division factors in order to generate applied algorithms. Part 1. Problems in programming. 2015. N 3. P. 3–12. (In Russian).
4. Kolesnyk V.G. DS-theory. Research of R-data division factors in order to generate applied algorithms. Part 2. Problems in programming. 2015. N 4. P. 3–13. (In Russian).
5. Kolesnyk V.G. DS-theory. The research of R-data factors formatting. Problems in programming. 2016. N 4. P. 14–26. (In Russian).
6. Bertrand Meyer. Object-Oriented Software Construction, 2nd Edition. Russian Edition, 2005. 1204 p.
7. Vincent Rosener and Denis Avrilionis. Elements for the Definition of a Model of Software Engineering. OneTree Technologies. [Electronic resource]. URL: http://www.irisa.fr/lande/lande/icse-proceedings/ gamma/p29.pdf
8. More comments on SEMAT. [Electronic resource] URL: http://blog.hexacta. com/more-comments-on-semat/

9. Dewayne E. Perry and Don Batory. On the Structure of General Theories of Software Engineering. – The University of Texas at Austin. [Electronic resource] URL: http://users.ece.utexas.edu/~perry/work/papers/gtse-structure.pdf
10. Ivar Jacobson and Ed Seidewitz. What happened to the promise of rigorous, disciplined, professional practices for software development? Ivar Jacobson International. Communications of the ACM, Vol. 57, N 12. P. 49–5410.1145/2677034.
11. Victor R. Basili. The role of experimentation in software engineering: past, current, and future. Proceedings of the Eighteenth International Conference on Software Engineering (ICSE), Berlin, Germany, March 1996.
12. National Science Foundation. Computing and Communication Foundations (CCF). The Algorithmic Foundations (AF) program. [Electronic resource] URL:
http://www.nsf.gov/cise/ccf/af_pgm15.jsp
http://www.nsf.gov/funding/pgm_summ.jsp?pims_id=503299&org=CCF&from=home
13. Dijkstra E.W. On the role of scientific thought (EWD447). [Electronic resource] URL: http://www.cs.utexas.edu/users/EWD/transcriptions/EWD04xx/EWD447.html
14. Victor R. Basili. Learning through Application. The Maturing of the QIP in the SEL – [Electronic resource] URL:
https://www.cs.umd.edu/~basili/publications/chapters/C29.pdf
15. Brooks F.P. The Mythical Man Month – Silver Anniversary Edition, Addison-Wesley, 1995.
16. Edmund M. Clarke, Jeannette M. Wing, Et Al. Formal Methods: State of the Art and Future Directions. ACM Computing Surveys. 1996. V. 28, N 28. P. 626–643.
17. Velbitsky I.V. "Technology of programming." Tekhnika, Kiev, 1984. (In Russian).
18. Jackson M.A. Principles of Program Design. New York: Academic Press, 1975.
19. C.J. Date. An Introduction to Database Systems, 8th Edition ruction. Williams; 2008. К.; 1328 p.
20. Alferova Z.A. The theory of algorithms. Moscow: Statistics, 1973. 164 p. (In Russian).
21. Kolesnyk V.G. Semchenko V.A. Conceptual design of computer network for shop floor control system. Doneck, 1989. 16 p. (In Russian).
22. Henry Ford. My Life and Work, Doubleday, Page & company; 1922. 140 p.

Получено 04.01.2017

Об авторе:

Колесник Валерий Георгиевич,
старший научный сотрудник
кафедры АПП.
Количество научных публикаций в
украинских изданиях – 25.
http://orcid.org/0000-0002-2313-9852.

Место работы автора:

Донбасская государственная
машиностроительная академия.
г. Краматорск, ул. Шкадинова, 72,
п/я 13.

УДК 004.42:510.69

О.С. Шкільняк

ВІДНОШЕННЯ ЛОГІЧНОГО НАСЛІДКУ В ЛОГІКАХ МОНОТОННИХ ПРЕДИКАТІВ ТА ЛОГІКАХ АНТИТОННИХ ПРЕДИКАТІВ

Досліджено відношення логічного наслідку в чистих першопорядкових композиційно-номінативних логіках часткових однозначних та часткових неоднозначних квазіарних предикатів. Описано відношення неспростовнісного, істиннісного, хибнісного та сильного логічного наслідку в логіках монотонних предикатів і логіках антитонних предикатів. Наведено приклади, які засвідчують відмінності одних відношень від інших, та встановлено співвідношення між різними відношеннями логічного наслідку.
Ключові слова: логіка, предикат, семантика, логічний наслідок.

Вступ

Апарат математичної логіки лежить в основі сучасних інформаційних і програмних систем (див., напр., [1]). Для цього зазвичай використовується класична логіка предикатів та базовані на її основі спеціальні логіки. Проте класична логіка має [2] низку обмежень, що ускладнює її використання. Тому набуває актуальності проблема побудови нових, програмно-орієнтованих логічних формалізмів. Такими є композиційно-номінативні логіки часткових квазіарних предикатів.

Центральним для логіки є поняття логічного наслідку. Широке використання в програмуванні часткових відображень, які можуть бути неоднозначними, виводить на перший план проблему вивчення відношень логічного наслідку для логік із нетрадиційними семантиками. Для першопорядкових композиційно-номінативних логік такі відношення описано в [2–5]. Вивченню відношень логічного наслідку для логік монотонних предикатів і логік антитонних предикатів присвячена дана робота, вона є безпосереднім продовженням роботи [4].

Метою даної статті є дослідження відношень логічного наслідку в чистих першопорядкових композиційно-номінативних логік (ЧКНЛ) однозначних та неоднозначних квазіарних предикатів. Розглядаються відношення неспростовнісного, істиннісного, хибнісного, сильного логічного наслідку. Основний акцент зроблено на вивченні цих відношень в різних семантиках логік монотонних предикатів і логік антитонних предикатів. Виділено 20 таких відношень, з'ясовано, що із цих відношень лише 7 попарно різних. Наведено приклади, які засвідчують відмінності одних відношень від інших. Встановлено співвідношення між різними відношеннями логічного наслідку.

Невизначені в цій статті поняття тлумачимо в сенсі робіт [2, 5].

Для полегшення читання опишемо основні поняття і визначення.

1. Композиційні алгебри квазіарних предикатів

В першопорядкових логіках предикати задаються на множинах пар, перша компонента яких – ім'я, а друга – його значення. Такі множини пар названо іменними множинами (ІМ). Предикати, задані на ІМ, називають квазіарними.

V-A-іменна множина (*V-A*-ІМ) – це часткова однозначна функція вигляду $d : V \to A$. Трактуємо V і A як множини предметних імен (змінних) і предметних (базових) значень. Клас всіх *V-A*-іменних множин будемо позначати ${}^{V}A$.

V-A-квазіарний предикат – це часткова (неоднозначна, взагалі кажучи) функція вигляду $P : {}^{V}A \to \{T, F\}$. Тут $\{T, F\}$ – множина істиннісних значень.

Неоднозначні *V-A*-квазіарні предикати трактуємо як відношення між ${}^{V}A$ та $\{T, F\}$. Такі предикати названо [2] предикатами реляційного типу, або *R*-предикатами. Множину значень, які *R*-предикат *P*

може прийняти на $d \in {}^{V}A$, позначимо $P(d)$. Маємо $P(d) \subseteq \{T, F\}$, тому $P(d)$ може набувати одне із значень $\{\varnothing\}$, $\{T\}$, $\{F\}$, $\{T, F\}$.

Клас V-A-квазіарних R-предикатів позначають PrR_A^V.

Кожний R-предикат $P : {}^{V}A \to \{T, F\}$ задається двома множинами – областю істинності $T(P)$ та областю хибності $F(P)$:

$$T(P) = \{d \in {}^{V}A \mid T \in P(d)\};$$

$$F(P) = \{d \in {}^{V}A \mid F \in P(d)\}.$$

V-A-квазіарний предикат P:

– однозначний, якщо $T(P) \cap F(P) = \varnothing$;

– тотальний, якщо $T(P) \cup F(P) = {}^{V}A$;

– неспростовний, якщо $F(P) = \varnothing$;

– виконуваний, якщо $T(P) \neq \varnothing$;

– тотожно істинний (позначаємо T), якщо $T(P) = {}^{V}A$ та $F(P) = \varnothing$;

– тотожно хибний (позначаємо F), якщо $T(P) = \varnothing$ та $F(P) = {}^{V}A$;

– всюди невизначений (позначаємо $\bot$), якщо $T(P) = \varnothing$ та $F(P) = \varnothing$;

– тотально насичений (позначаємо $\top$), якщо $T(P) = {}^{V}A$ та $F(P) = {}^{V}A$.

Часткові однозначні предикати названо [4, 5] P-предикатами, тотальні названо T-предикатами, тотальні однозначні – TS-предикатами. Класи таких V-A-квазіарних предикатів будемо відповідно позначати PrP_A^V, PrT_A^V, $PrTS_A^V$.

Предикат $P : {}^{V}A \to \{T, F\}$ монотонний, якщо: $d \subseteq d' \Rightarrow P(d) \subseteq P(d')$.

Предикат $P : {}^{V}A \to \{T, F\}$ антитонний, якщо: $d \subseteq d' \Rightarrow P(d) \supseteq P(d')$.

Для однозначних предикатів монотонність стає еквітонністю. Однозначний $P : {}^{V}A \to \{T, F\}$ еквітонний, якщо:

$$(P(d)\downarrow \text{ та } d \subseteq d') \Rightarrow P(d')\downarrow = P(d).$$

Монотонні R-предикати, антитонні R-предикати, еквітонні P-предикати, антитонні T-предикати відповідно називають (див. [5]) RM-предикатами, RA-предикатами, PE-предикатами, TA-предикатами. Класи цих предикатів будемо позначати $PrRM_A^V, PrRA_A^V, PrPE_A^V, PrTA_A^V$.

Константні предикати $\bot, \top, \mathrm{T}, \mathrm{F}$ монотонні (еквітонні) й антитонні, при цьому предикати $\bot, \mathrm{T}, \mathrm{F}$ – однозначні.

Предикати-індикатори Ex наявності у вхідних даних компоненти з іменем $x \in V$ задаємо [5] так:

$$T(Ex) = \{d \mid d(x)\downarrow\}, \; F(Ex) = \{d \mid d(x)\uparrow\}.$$

Предикати-індикатори Ex тотальні, однозначні, немонотонні, неантитонні.

Предикат $\tilde{P}$ називають [5] дуальним до предиката P, якщо:

$$T(\tilde{P}) = \overline{F(P)}; \; F(\tilde{P}) = \overline{T(P)}.$$

Безпосередньо із визначень маємо:

Q – P-предикат $\Leftrightarrow$ $\tilde{Q}$ – T-предикат;

Q – T-предикат $\Leftrightarrow$ $\tilde{Q}$ – P-предикат;

Q – PE-предикат $\Leftrightarrow$ $\tilde{Q}$ – TA-предикат;

Q – TA-предикат $\Leftrightarrow$ $\tilde{Q}$ – PE-предикат;

якщо Q – TS-предикат, то $\tilde{Q} = Q$.

Спеціальне відображення дуалізації $\delta : PrR_A^V \to PrR_A^V$ задають [5] так: $\delta(P) = \tilde{P}$.

Маємо [5] такі властивості:

$$\delta(\mathrm{T}) = \mathrm{T}, \; \delta(\mathrm{F}) = \mathrm{F}, \; \delta(\bot) = \top, \; \delta(\top) = \bot;$$

$$\delta(PrR_A^V) = PrR_A^V, \; \delta(PrTS_A^V) = PrTS_A^V,$$

$$\delta(PrP_A^V) = PrT_A^V, \; \delta(PrT_A^V) = PrP_A^V;$$

$$\delta(PrPE_A^V) = PrTA_A^V, \; \delta(PrTA_A^V) = PrPE_A^V,$$

$$\delta(PrRM_A^V) = PrRA_A^V, \; \delta(\mathrm{Pr}\,RA_A^V) = \mathrm{Pr}\,RM_A^V.$$

Базовими композиціями ЧКНЛ є [2] логічні зв'язки $\neg$ і $\vee$, композиції реноменації $\mathrm{R}_{\bar{x}}^{\bar{v}}$ та квантифікації $\exists x$.

Для $\bot, \top$, T та F маємо (див. [5]):

$$\neg\bot = \bot, \; \bot \vee \bot = \bot, \; \mathrm{R}_{\bar{x}}^{\bar{v}}(\bot) = \bot, \; \exists x(\bot) = \bot;$$

$$\neg\top = \top, \; \top \vee \top = \top, \; \mathrm{R}_{\bar{x}}^{\bar{v}}(\top) = \top, \; \exists x(\top) = \top;$$

$$\neg\mathrm{T} = \mathrm{F}, \neg\mathrm{F} = \mathrm{T}, \; \mathrm{F} \vee \mathrm{F} = \mathrm{F};$$

$$\mathrm{R}_{\bar{x}}^{\bar{v}}(\mathrm{T}) = \mathrm{T}, \; \exists x(\mathrm{T}) = \mathrm{T}, \; \mathrm{R}_{\bar{x}}^{\bar{v}}(\mathrm{F}) = \mathrm{F}, \; \exists x(\mathrm{F}) = \mathrm{F};$$

$\mathrm{T} \vee \mathrm{T} = \mathrm{T} \vee \bot = \bot \vee \mathrm{T} = \mathrm{T} \vee \mathrm{F} = \mathrm{F} \vee \mathrm{T} = \mathrm{T}$;

$\mathrm{F} \vee \bot = \bot \vee \mathrm{F} = \bot$; $\bot \vee \top = \top \vee \bot = \mathrm{T}$;

$\mathrm{T} \vee \top = \top \vee \mathrm{T} = \mathrm{F} \vee \top = \top \vee \mathrm{F} = \top$.

Позначимо $CQ = \{\neg, \vee, \mathrm{R}_{\bar{x}}^{\bar{v}}, \exists x\}$.

Чиста першопорядкова алгебра квазіарних предикатів – це композиційна алгебра $QR_A^V = (PrR_A^V, CQ)$.

Композиції $\neg$, $\vee$, $\mathrm{R}_{\bar{x}}^{\bar{v}}, \exists x$ зберігають [2] однозначність, тотальність, монотонність, антитонність предикатів. Тому щодо $\neg$, $\vee$, $\mathrm{R}_{\bar{x}}^{\bar{v}}$, $\exists x$ замкнені такі класи: *P*-предикатів, *T*-предикатів, *TS*-предикатів; монотонних (еквітонних) предикатів, антитонних предикатів.

Таким чином, можна виділити такі підалгебри алгебри QR_A^V:

$$QP_A^V = (PrP_A^V, CQ),$$

$$QT_A^V = (PrT_A^V, CQ),$$

$$QTS_A^V = (PrTS_A^V, CQ),$$

$$QRM_A^V = (PrRM_A^V, CQ),$$

$$QRA_A^V = (PrRA_A^V, CQ),$$

$$QPE_A^V = (PrPE_A^V, CQ),$$

$$QTA_A^V = (PrTA_A^V, CQ).$$

Те, що $\aleph$ є підалгеброю алгебри $\Re$, позначаємо $\aleph \prec \Re$. Тоді [5] маємо:

$QTS_A^V \prec QP_A^V$ та $QTS_A^V \prec QT_A^V$;

$QPE_A^V \prec QP_A^V$ та $QPE_A^V \prec QRM_A^V$;

$QTA_A^V \prec QT_A^V$ та $QTA_A^V \prec QRA_A^V$.

Нехай δ – відображення дуалізації.

Алгебри (Pr_1, CQ) та (Pr_2, CQ) дуальні, якщо $\delta(\mathrm{Pr}_1) = \mathrm{Pr}_2$ та $\delta(\mathrm{Pr}_2) = \mathrm{Pr}_1$.

Маємо такі дуальні пари алгебр:

QP_A^V та QT_A^V,

QPE_A^V та QTA_A^V,

QRM_A^V та QRA_A^V.

Алгебри QR_A^V та QTS_A^V автодуальні.

2. Мови та їх інтерпретації

Семантичними моделями ЧКНЛ є [2] чисті першопорядкові композиційні системи квазіарних предикатів вигляду (*A*, Pr, *CQ*). Така система задає алгебру даних (*A*, Pr) і композиційну алгебру предикатів (Pr, *CQ*). Терми композиційної алгебри трактуємо як формули мови ЧКНЛ.

Алфавіт мови: множини *V* предметних імен (змінних) та $U \subseteq V$ тотально неістотних [2] імен; множина символів базових композицій $Cs = \{\neg, \vee, R_{\bar{x}}^{\bar{v}}, \exists x\}$, множина *Ps* предикатних символів (сигнатура). Розширена сигнатура мови – це $\Sigma = (V, U, Cs, Ps)$.

Індуктивне визначення множини *Fr* формул таке:

– $Ps \subseteq Fr$; формули $p \in Ps$ – атомарні;

– $\Phi, \Psi \in Fr \Rightarrow \neg\Phi, \vee\Phi\Psi, R_{\bar{x}}^{\bar{v}}\Phi, \exists x\Phi \in Fr$.

Інтерпретуємо мову сигнатури Σ на композиційній системі (*A*, Pr, *CQ*) за допомогою тотального однозначного відображення $I : Ps \to \mathrm{Pr}$. Розширимо *I* до відображення $I : Fr \to \mathrm{Pr}$ згідно побудови складних формул за допомогою символів *Cs*:

– $I(\neg\Phi) = \neg(I(\Phi))$,

– $I(\vee\Phi\Psi) = \vee(I(\Phi), I(\Psi))$,

– $I(R_{\bar{x}}^{\bar{v}}(\Phi)) = \mathrm{R}_{\bar{x}}^{\bar{v}}(I(\Phi))$,

– $I(\exists x\Phi) = \exists x(I(\Phi))$.

Інтерпретація мови ЧКНЛ сигнатури Σ – це $\boldsymbol{J} = (\mathrm{CS}, \Sigma, I)$. Скорочено інтерпретації будемо позначати (A, I).

Предикат $\boldsymbol{J}(\Phi)$ – значення формули Φ при інтерпретації $\boldsymbol{J}$ – позначимо Φ_J.

Виділення підалгебр квазіарних предикатів виділяє відповідні класи інтерпретацій. Такі класи інтерпретацій називають *семантиками*.

Маємо загальний клас *R*-інтерпретацій та підкласи *P*-інтерпретацій, *T*-інтерпретацій, *TS*-інтерпретацій, *PE*-інтерпретацій, *TA*-інтерпретацій, *RM*-інтерпретацій, *RA*-інтерпретацій. Ці класи інтерпретацій, або семантики, відповідно позначають так: ***R***, ***P***, ***T***, ***TS***, ***PE***, ***TA***, ***RM***, ***RA***.

Для семантик маємо (див. [5]):

$$\boldsymbol{PE} \subset \boldsymbol{P},\ \boldsymbol{TA} \subset \boldsymbol{T};$$

$$\boldsymbol{TS} \subset \boldsymbol{P} \subset \boldsymbol{R},\ \boldsymbol{TS} \subset \boldsymbol{T} \subset \boldsymbol{R};$$

$$\boldsymbol{PE} \subset \boldsymbol{RM} \subset \boldsymbol{R},\ \boldsymbol{TA} \subset \boldsymbol{RA} \subset \boldsymbol{R}.$$

Дуальна до інтерпретації $\boldsymbol{J} = (A, I)$ інтерпретація $\delta(\boldsymbol{J}) = (A, I_\delta)$ задається так: для всіх $p \in Ps$ маємо $T(p_{\delta(J)}) = \overline{F(p_J)}$ та $F(p_{\delta(J)}) = \overline{T(p_J)}$.

Тоді $\boldsymbol{J}$ дуальна до $\delta(\boldsymbol{J})$.

Якщо інтерпретації $\boldsymbol{J}$ та $\mathfrak{I}$ дуальні, то (див. [2]): для всіх $\Phi \in Fr$ маємо

$$T(\Phi_J) = \overline{F(\Phi_{\mathfrak{I}})} \text{ та } F(\Phi_{\mathfrak{I}}) = \overline{T(\Phi_J)}.$$

Якщо $\boldsymbol{J}$ та $\boldsymbol{G}$ дуальні, то:

– Φ_J монотонний $\Leftrightarrow$ Φ_G антитонний;

– Φ_G антитонний $\Leftrightarrow$ Φ_J монотонний.

Виділення дуальних пар предикатних алгебр індукує виділення дуальних пар семантик ***P*** та ***T***, ***PE*** та ***TA***, ***RM*** та ***RA***.

Семантики ***R*** та ***TS*** автодуальні.

Для формул мови ЧКНЛ введено [2] поняття виконуваної, неспростовної, тотожно істинної, тотожно хибної формули.

Семантичні властивості ЧКНЛ досліджено, зокрема, в [2–5].

3. Відношення логічного наслідку

На множині формул можна ввести [2, 4, 5] низку відношень, які формалізують центральне для логіки поняття логічного наслідку. Спочатку вводимо відношення наслідку для двох формул при фіксованій інтерпретації $\boldsymbol{J}$.

1. Істиннісний, або *T*-наслідок $\models_T$:

$$\Phi \models_T \Psi \Leftrightarrow T(\Phi_J) \subseteq T(\Psi_J).$$

2. Хибнісний, або *F*-наслідок $\models_F$:

$$\Phi \models_F \Psi \Leftrightarrow F(\Psi_J) \subseteq F(\Phi_J).$$

3. Сильний, або *TF*-наслідок $\models_{TF}$:

$$\Phi \models_{TF} \Psi \Leftrightarrow \Phi \models_T \Psi \text{ та } \Phi \models_F \Psi.$$

4. Неспростовнісний, або *IR*-наслідок $\models_{IR}$:

$$\Phi \models_{IR} \Psi \Leftrightarrow T(\Phi_J) \cap F(\Psi_J) = \varnothing.$$

5. Дуальний до *IR*, або *DI*-наслідок $\models_{DI}$:

$$\Phi \models_{DI} \Psi \Leftrightarrow F(\Phi_J) \cup T(\Psi_J) = {}^V A.$$

Відповідні відношення логічного наслідку в семантиці α визначаємо за такою схемою:

$\Phi \models_\alpha \Psi$, якщо $\Phi \models_* \Psi$ для кожної $\boldsymbol{J} \in \alpha$.

Зазначені відношення описано в [2].

Нехай інтерпретації $\boldsymbol{A}$, $\boldsymbol{B}$ дуальні. Тоді [2, 5] маємо:

$$\Phi_A \models_T \Psi \Leftrightarrow \Phi_B \models_F \Psi,$$

$$\Phi_A \models_F \Psi \Leftrightarrow \Phi_B \models_T \Psi,$$

$$\Phi_A \models_{IR} \Psi \Leftrightarrow \Phi_B \models_{DI} \Psi,$$

$$\Phi_A \models_{DI} \Psi \Leftrightarrow \Phi_B \models_{IR} \Psi,$$

$$\Phi_A \models_{TF} \Psi \Leftrightarrow \Phi_B \models_{TF} \Psi.$$

При розширенні семантики відношення логічного наслідку звужується (тут $*$ – це T, F, TF, IR, DI):

Теорема 1. Нехай для семантик α та β маємо $\alpha \subseteq \beta$. Тоді $\models_\beta \subseteq \models_\alpha$.

Справді, нехай $\Phi \models_\beta \Psi$, тоді $\Phi_J \models_* \Psi$ для кожної $\boldsymbol{J} \in \beta$. Але $\alpha \subseteq \beta$, тому $\Phi_J \models_* \Psi$ для кожної $\boldsymbol{J} \in \alpha$. Звідси $\Phi \models_\alpha \Psi$. Отже, $\models_\beta \subseteq \models_\alpha$.

Для наведених відношень логічного наслідку маємо (див. [2, 4]):

$$\models_P = \models_{DI} = \models_T = \models_{IR} = \models_R = \models_{IR} = \models_R = \models_{DI} = \varnothing;$$

$$\models_P = \models_T = \models_T = \models_F;\ \models_P = \models_F = \models_T = \models_T;\ \models_P = \models_{IR} = \models_T = \models_{DI};$$

$$^{P}|=_{TF} = {}^{T}|=_{TF};\quad {}^{R}|=_{T} = {}^{R}|=_{F} = {}^{R}|=_{TF}.$$

Серед цих відношень виявилося [2, 4] лише 5 попарно різних:

$$^{P}|=_{IR},\ {}^{P}|=_{T},\ {}^{P}|=_{F},\ {}^{P}|=_{TF},\ {}^{R}|=_{TF}.$$

Для логіки *TS*-предикатів і класичної логіки ці 5 відношень втрачають відмінності та стають єдиним відношенням:

$$^{TS}|=_{TF} = {}^{TS}|=_{T} = {}^{TS}|=_{F} = {}^{TS}|=_{IR} = {}^{TS}|=_{DI}.$$

Маємо [4, 5] такі співвідношення:

$$^{P}|=_{TF} \subset {}^{P}|=_{T},\ {}^{P}|=_{TF} \subset {}^{P}|=_{F};$$

$$^{R}|=_{TF} \subset {}^{P}|=_{TF};$$

$$^{P}|=_{T} \cup {}^{P}|=_{F} \subset {}^{P}|=_{IR};$$

$$^{P}|=_{T} \not\subset {}^{P}|=_{F},\ {}^{P}|=_{F} \not\subset {}^{P}|=_{T}.$$

Обмежуючи розглянуті відношення на семантики монотонних (еквітонних) предикатів і антитонних предикатів, отримуємо такі відношення:

$$^{PE}|=_{IR},\ {}^{TA}|=_{DI},\ {}^{PE}|=_{DI},\ {}^{TA}|=_{IR},$$

$$^{PE}|=_{T},\ {}^{TA}|=_{F},\ {}^{PE}|=_{F},\ {}^{TA}|=_{T},$$

$$^{PE}|=_{TF},\ {}^{TA}|=_{TF};$$

$$^{RM}|=_{IR},\ {}^{RA}|=_{IR},\ {}^{RM}|=_{DI},\ {}^{RA}|=_{DI},$$

$$^{RM}|=_{T},\ {}^{RA}|=_{F},\ {}^{RA}|=_{T},\ {}^{RM}|=_{F},$$

$$^{RM}|=_{TF},\ {}^{RA}|=_{TF}.$$

Беручи до уваги дуальність пар ***PE*** і ***TA*** та ***RM*** і ***RA***, отримуємо:

Теорема 2.

1) $^{PE}|=_{IR} = {}^{TA}|=_{DI}$, $^{PE}|=_{T} = {}^{TA}|=_{F}$,

$^{PE}|=_{F} = {}^{TA}|=_{T}$, $^{PE}|=_{TF} = {}^{TA}|=_{TF}$;

2) $^{RM}|=_{T} = {}^{RA}|=_{F}$, $^{RM}|=_{F} = {}^{RA}|=_{T}$,

$^{RM}|=_{TF} = {}^{RA}|=_{TF}$;

3) $^{PE}|=_{DI} = {}^{TA}|=_{IR} = \varnothing,$

$^{RM}|=_{IR} = {}^{RA}|=_{IR} = {}^{RM}|=_{DI} = {}^{RA}|=_{DI} = \varnothing.$

Таким чином, із роглянутих відношень логічного наслідку для монотонних (еквітонних) предикатів і антитонних предикатів залишається не більше 7 різних:

$$^{PE}|=_{IR},\ {}^{PE}|=_{T},\ {}^{PE}|=_{F},\ {}^{PE}|=_{TF},$$

$$^{RM}|=_{T},\ {}^{RM}|=_{F},\ {}^{RM}|=_{TF}.$$

Розглянемо співвідношення між відношеннями логічного наслідку.

Беручи до уваги теорему 1, отримуємо:

Теорема 3.

$$^{RM}|=_{TF} \subseteq {}^{RM}|=_{T},\ {}^{RM}|=_{TF} \subseteq {}^{RM}|=_{F},$$

$$^{RM}|=_{TF} \subseteq {}^{PE}|=_{TF},\ {}^{RM}|=_{T} \subseteq {}^{PE}|=_{T},$$

$$^{RM}|=_{F} \subseteq {}^{PE}|=_{F};$$

$$^{PE}|=_{TF} \subseteq {}^{PE}|=_{T} \subseteq {}^{PE}|=_{IR},$$

$$^{PE}|=_{TF} \subseteq {}^{PE}|=_{F} \subseteq {}^{PE}|=_{IR};$$

$$^{R}|=_{TF} \subseteq {}^{RM}|=_{TF},\ {}^{P}|=_{TF} \subseteq {}^{PE}|=_{TF},$$

$$^{P}|=_{T} \subseteq {}^{PE}|=_{T},\ {}^{P}|=_{F} \subseteq {}^{PE}|=_{F},$$

$$^{P}|=_{IR} \subseteq {}^{PE}|=_{IR}.$$

Відомо [2], що:

$$\neg\Phi \,\&\, \Phi^{P}|=_{T} \vartheta,\ \vartheta\,{}^{P}|=_{F} \Psi \vee \neg\Psi,$$

$$\neg\Phi \,\&\, \Phi^{P}|=_{TF} \Psi \vee \neg\Psi.$$

За теоремою 3 тоді отримуємо:

Твердження 1. 1) $\neg\Phi \,\&\, \Phi^{PE}|=_{T} \vartheta;$

2) $\vartheta\,{}^{PE}|=_{F} \Psi \vee \neg\Psi;$

3) $\neg\Phi \,\&\, \Phi^{PE}|=_{TF} \Psi \vee \neg\Psi.$

Водночас [2] маємо:

$$\neg\Phi \,\&\, \Phi^{P}|\neq_{F} \vartheta,\ \vartheta\,{}^{P}|\neq_{T} \Psi \vee \neg\Psi,$$

$$\neg\Phi \,\&\, \Phi^{R}|\neq_{TF} \Psi \vee \neg\Psi.$$

Останні твердження можна посилити, формулюючи їх для семантик *PE* і *RM*. Для цього розглянемо декілька прикладів. При формулюванні прикладів для зручнос-

ті використовуємо символи розширеної сигнатури: Ex для предикатів-індикаторів та $\bot, \top, \mathrm{T}, \mathrm{F}$ для константних предикатів.

Зафіксуємо довільну $\boldsymbol{J} = (A, I)$.

Приклад 1. $\neg\bot \,\&\, \bot\ _J \not\models_F \mathrm{F}$.

Маємо $F(\mathrm{F}) = {}^V\!A$, проте $F(\bot \,\&\, \neg\bot) = F(\bot) \cup F(\neg\bot) = F(\bot) \cup T(\bot) = \varnothing \cup \varnothing = \varnothing$.

Приклад 2. $\mathrm{T}_J \not\models_T \bot \vee \neg\bot$.

Маємо $T(\mathrm{T}) = {}^V\!A$, проте $T(\bot \vee \neg\bot) = T(\bot) \cup T(\neg\bot) = T(\bot) \cup F(\bot) = \varnothing \cup \varnothing = \varnothing$.

Приклад 3. $\top \,\&\, \neg\top_J \not\models_T \bot \vee \neg\bot$ та $\bot \,\&\, \neg\bot_J \not\models_F \top \vee \neg\top$.

Справді, маємо:

$T(\top \,\&\, \neg\top) = T(\top) \cap F(\top) = {}^V\!A \cap {}^V\!A = {}^V\!A$;

$T(\bot \vee \neg\bot) = T(\bot) \cup T(\neg\bot) = \varnothing \cup \varnothing = \varnothing$;

$F(\top \vee \neg\top) = F(\top) \cap T(\top) = {}^V\!A \cap {}^V\!A = {}^V\!A$;

$F(\bot \,\&\, \neg\bot) = F(\bot) \cup F(\neg\bot) = \varnothing \cup \varnothing = \varnothing$.

Таким чином, отримуємо:

Твердження 2. 1) $\neg\Phi \,\&\, \Phi^{PE} \not\models_F \vartheta$;

2) $\vartheta^{PE} \not\models_T \Psi \vee \neg\Psi$;

3) $\Phi \,\&\, \neg\Phi^{RM} \not\models_T \Psi \vee \neg\Psi$,

$\Phi \,\&\, \neg\Phi^{RM} \not\models_F \Psi \vee \neg\Psi$.

Згідно теореми 3 тоді отримуємо

Наслідок 1.

1) $\neg\Phi\&\,\Phi^{PE} \not\models_{TF} \vartheta$;

2) $\vartheta^{PE} \not\models_{TF} \Psi \vee \neg\Psi$;

3) $\Phi \,\&\, \neg\Phi^{RM} \not\models_{TF} \Psi \vee \neg\Psi$.

Маємо $\Phi^P \not\models_{IR} \exists x\Phi$ та $\forall x\,\Psi^P \not\models_{IR} \Psi$ (див. [2]). Зазначені твердження можна посилити до $\Phi\&\forall x\,\Psi^P \not\models_{IR} \exists x\Phi \vee \Psi$. Наведемо відповідні приклади.

Приклад 4. Маємо

$\neg Ex_J \not\models_{IR} \exists x\neg Ex$, $\forall x Ex_J \not\models_{IR} Ex$,

$\neg Ex \,\&\, \forall x Ex_J \not\models_{IR} \exists x\neg Ex \vee Ex$.

Маємо $(Ex)_J(d) = F$ і $(\neg Ex)_J(d) = T$ для всіх $d \in {}^V\!A$, тому $(\exists x\neg Ex)_J = \mathrm{F}$ і $(\forall x Ex)_J = \mathrm{T}$. Для $d \in {}^V\!A$ такого, що $d(x)\uparrow$, маємо $d \in T(\neg Ex_J)$, $d \in F(Ex_J)$, $d \in T((\neg Ex \,\&\, \forall x Ex)_J)$, $d \in F((\exists x\neg Ex \vee Ex)_J)$. Тому $\neg Ex_J \not\models_{IR} \exists x\neg Ex$, $\forall x Ex_J \not\models_{IR} Ex$, $\neg Ex \,\&\, \forall x Ex_J \not\models_{IR} \exists x\neg Ex \vee Ex$.

Таким чином.

Твердження 3.

1) $\Phi^P \not\models_{IR} \exists x\Phi$;

2) $\forall x\,\Psi^P \not\models_{IR} \Psi$;

3) $\Phi\&\forall x\,\Psi^P \not\models_{IR} \exists x\Phi \vee \Psi$.

Беручи до уваги теорему 3, отримуємо (тут $\not\models$ – це ${}^P\!\not\models_{IR}$, ${}^P\!\not\models_T$, ${}^P\!\not\models_F$, ${}^P\!\not\models_{TF}$, ${}^R\!\not\models_{TF}$):

Теорема 4.

1) $\Phi \not\models \exists x\,\Phi$;

2) $\forall x\,\Psi \not\models \Psi$;

3) $\Phi\&\forall x\,\Psi \not\models \exists x\Phi \vee \Psi$.

Для монотонних (еквітонних) предикатів маємо (див. [2]):

$T(Q) \subseteq T(\exists xQ)$ та $F(Q) \subseteq F(\forall xQ)$,

$F(\exists xQ) \not\subseteq F(Q)$ та $T(\forall xQ) \not\subseteq T(Q)$.

Звідси отримуємо:

Твердження 4.

$\Phi^{RM} \models_T \exists x\Phi$ та $\forall x\Psi^{RM} \models_F \Psi$,

$\Phi^{PE} \not\models_F \exists x\Phi$ та $\forall x\Psi^{PE} \not\models_T \Psi$.

Відмінність ${}^{PE}\!\models_T$ та ${}^{PE}\!\models_F$ засвідчує

Теорема 5.

1) $\neg\varphi\&\varphi\vee\Phi^{PE} \models_T \exists x\Phi$,

$\forall x\Psi^{PE} \models_F \Psi\&(\neg\varphi\vee\varphi)$;

2) $\neg\varphi\&\varphi\vee\Phi^{PE} \not\models_F \exists x\Phi$,

$$\forall x\Psi \;{}^{PE}\!\not\models_T \Psi\&(\neg\varphi\vee\varphi).$$

Для P-предикатів маємо $T(\neg Q\&Q) = F(\neg Q\vee Q) = \varnothing$, звідки $T(\neg Q\&Q\vee S) = T(S)$ та $F(S\&(\neg Q\vee Q)) = F(S)$; також маємо $F(\neg Q\&Q\vee S)\subseteq F(S)$ та $T(S\&(\neg Q\vee Q))\subseteq T(S)$. Враховуючи $T(S)\subseteq T(\exists xS)$ та $F(S)\subseteq F(\forall xS)$, маємо п. 1, а враховуючи $F(\exists xS)\not\subseteq F(S)$ та $T(\forall xS)\not\subseteq T(S)$, отримуємо п. 2.

Теорема 6.

1) $\neg\varphi\&\varphi\vee\Phi \;{}^{RM}\!\not\models_T \exists x\Phi,$

$\forall x\Psi \;{}^{RM}\!\not\models_F \Psi\&(\neg\varphi\vee\varphi);$

2) $\neg\varphi\&\varphi\vee\Phi \;{}^{RM}\!\not\models_F \exists x\Phi,$

$\forall x\Psi \;{}^{RM}\!\not\models_T \Psi\&(\neg\varphi\vee\varphi).$

Для доведення розглянемо приклад.

Приклад 5. $\neg\top\,\&\,\top\vee\bot_J \not\models_T \exists x\bot$ та $\forall x\bot_J \not\models_F \bot\&(\neg\top\vee\top)$.

Маємо $T(\top) = F(\top) = {}^V\!A$, звідки $T(\neg\top\,\&\,\top) = F(\neg\top\vee\top) = {}^V\!A$. Проте $T(\bot) = F(\bot) = \varnothing$, тому $T(\neg\top\,\&\,\top\vee\bot) = {}^V\!A$ та $F(\bot\,\&(\neg\top\vee\top)) = {}^V\!A$. Водночас маємо $T(\exists x\bot) = F(\exists x\bot) = T(\forall x\bot) = F(\forall x\bot) = \varnothing$, звідки отримуємо:

$$T(\neg\top\,\&\,\top\vee\bot)\not\subseteq T(\exists x\bot),$$

$$F(\bot\,\&(\neg\top\vee\top))\not\subseteq F(\forall x\bot).$$

Звідси випливає п. 1 теореми 6.

П. 2 теореми 6 випливає з теорем 3 та 5.

Твердження 5.

$\Phi\&\forall x\Psi \;{}^{RM}\!\models_{TF} \exists x\Phi\vee\Psi.$

Із $\Phi \;{}^{RM}\!\models_T \exists x\Phi$ та $\forall x\Psi \;{}^{RM}\!\models_F \Psi$ за монотонністю відношень ${}^{RM}\!\models_T$ та ${}^{RM}\!\models_F$ маємо $\Phi\&\forall x\Psi \;{}^{RM}\!\models_T \exists x\Phi\vee\Psi$ і $\Phi\&\forall x\Psi \;{}^{RM}\!\models_F \exists x\Phi\vee\Psi$, звідки отримуємо $\Phi\&\forall x\Psi \;{}^{RM}\!\models_{TF} \exists x\Phi\vee\Psi$.

Беручи до уваги теорему 3, отримуємо (тут $\models$ – це ${}^{PE}\!\models_{IR}$, ${}^{PE}\!\models_T$, ${}^{PE}\!\models_F$, ${}^{PE}\!\models_{TF}$, ${}^{RM}\!\models_T$, ${}^{RM}\!\models_F$, ${}^{RM}\!\models_{TF}$):

Теорема 7. $\Phi\&\forall x\Psi \models \exists x\Phi\vee\Psi$.

Теорема 8. Маємо

$$(\neg\varphi\&\varphi\vee\Phi)\&\forall x\Psi \;{}^{PE}\!\models_{TF} \exists x\Phi\vee\Psi\&(\neg\varphi\vee\varphi),$$

$$(\neg\varphi\&\varphi\vee\Phi)\&\forall x\Psi \;{}^{RM}\!\not\models_T \exists x\Phi\vee\Psi\&(\neg\varphi\vee\varphi),$$

$$(\neg\varphi\&\varphi\vee\Phi)\&\forall x\Psi \;{}^{RM}\!\not\models_F \exists x\Phi\vee\Psi\&(\neg\varphi\vee\varphi).$$

Перше твердження теореми випливає з п. 1 теореми 5 і монотонності відношень ${}^{PE}\!\models_T$ та ${}^{PE}\!\models_F$.

Маємо $\exists x\bot = \bot$, $\forall xEx = \mathrm{T}$, $\exists x\neg Ex = \mathrm{F}$; ${}^V\!A = T(\neg\top\,\&\,\top) = T(\neg\top\vee\top) = F(\neg\top\,\&\,\top) = T(\neg\top\vee\top)$. Нехай $(\neg\top\,\&\,\top\vee\bot)\&\forall xEx$, $\exists x\bot\vee Ex\&(\neg\top\vee\top)$, $(\neg\top\,\&\,\top\vee\neg Ex)\&\bot$, $\exists x\neg Ex\vee\bot\&(\neg\top\vee\top)$ – це α, β, γ, δ. Тоді $T(\alpha) = {}^V\!A$, $T(\beta)\subset {}^V\!A$, $F(\gamma)\subset {}^V\!A$, $F(\delta) = {}^V\!A$. Тому $(\neg\top\,\&\,\top\vee\bot)\&\forall xEx_J \not\models_T \exists x\bot\vee Ex\&(\neg\top\vee\top)$, $(\neg\top\,\&\,\top\vee\neg Ex)\&\forall x\bot_J \not\models_F \exists x\neg Ex\vee\bot\&(\neg\top\vee\top)$. Звідси друге та третє твердження теореми.

Для P-предикатів завжди маємо $T(\neg Q\&Q\vee S) = T(S)$ та $F(S\&(\neg Q\vee Q)) = F(S)$, тому, згідно теореми 4, отримуємо:

Твердження 6.

$$\neg\varphi\&\varphi\vee\Phi \;{}^{P}\!\not\models_{IR} \exists x\Phi,\ \forall x\Psi \;{}^{P}\!\not\models_{IR} \Psi\&(\neg\varphi\vee\varphi),$$

$$\neg\varphi\&\varphi\vee\Phi\&\forall x\Psi \;{}^{P}\!\not\models_{IR} (\exists x\Phi\vee\Psi)\&(\neg\varphi\vee\varphi).$$

Враховуючи теорему 3, остаточно отримуємо:

Теорема 9. Маємо

$$\neg\varphi\&\varphi\vee\Phi \not\models \exists x\Phi,\ \forall x\Psi \not\models \Psi\&(\neg\varphi\vee\varphi),$$

$$(\neg\varphi\&\varphi\vee\Phi)\&\forall x\Psi \not\models \exists x\Phi\vee\Psi\&(\neg\varphi\vee\varphi).$$

Тут $\not\models$ – це ${}^{P}\!\not\models_{IR}$, ${}^{P}\!\not\models_T$, ${}^{P}\!\not\models_F$, ${}^{P}\!\not\models_{TF}$, ${}^{R}\!\not\models_{TF}$.

Зведемо отримані результати щодо наявності чи відсутності того чи іншого логічного наслідку для відповідних пар формул в таблицю (таблиця).

В цій таблиці використано такі скорочення (тут $|=$ позначає одне з описаних відношень):

ρ_1 – це $\neg\Phi\,\&\Phi \mid= \vartheta$,

ρ_2 – це $\vartheta \mid= \Psi\vee\neg\Psi$,

ρ_3 – це $\neg\Phi\,\&\Phi \mid= \Psi\vee\neg\Psi$,

ρ_4 – це $\Phi \mid= \exists x\,\Phi$,

ρ_5 – це $\forall x\Psi \mid= \Psi$,

ρ_6 – це $\Phi\,\&\forall x\Psi \mid= \exists x\Phi\vee\Psi$,

ρ_7 – це $\neg\varphi\&\varphi\vee\Phi \mid= \exists x\,\Phi$,

ρ_8 – це $\forall x\Psi \mid= \Psi\&(\neg\varphi\vee\varphi)$,

ρ_9 – це

$(\neg\varphi\&\varphi\vee\Phi)\&\forall x\Psi \mid= \exists x\,\Phi\vee\Psi\&(\neg\varphi\vee\varphi)$.

Таблиця. Наявність логічного наслідку для певних пар формул

	ρ_1	ρ_2	ρ_3	ρ_4	ρ_5	ρ_6	ρ_7	ρ_8	ρ_9
$\overset{P}{\mid=}_{IR}$	+	+	+	–	–	–	–	–	–
$\overset{P}{\mid=}_{T}$	+	–	+	–	–	–	–	–	–
$\overset{P}{\mid=}_{F}$	–	+	+	–	–	–	–	–	–
$\overset{P}{\mid=}_{TF}$	–	–	+	–	–	–	–	–	–
$\overset{R}{\mid=}_{TF}$	–	–	–	–	–	–	–	–	–
$\overset{PE}{\mid=}_{IR}$	+	+	+	+	+	+	+	+	+
$\overset{PE}{\mid=}_{T}$	+	–	+	+	–	+	+	–	+
$\overset{PE}{\mid=}_{F}$	–	+	+	–	+	+	–	+	+
$\overset{PE}{\mid=}_{TF}$	–	–	+	–	–	+	–	–	+
$\overset{RM}{\mid=}_{T}$	–	–	–	+	–	+	–	–	–
$\overset{RM}{\mid=}_{F}$	–	–	–	–	+	+	–	–	–
$\overset{RM}{\mid=}_{TF}$	–	–	–	–	–	+	–	–	–

Усі відношення логічного наслідку, які фігурують в таблиці, виявилися різними.

Беручи до уваги вищенаведені результати, маємо такі співвідношення між відношеннями логічного наслідку:

Теорема 10.

$$\overset{RM}{\mid=}_{TF} \subset \overset{RM}{\mid=}_{T} \subset \overset{PE}{\mid=}_{T},$$

$$\overset{RM}{\mid=}_{TF} \subset \overset{RM}{\mid=}_{F} \subset \overset{PE}{\mid=}_{F},$$

$$\overset{RM}{\mid=}_{TF} \subset \overset{PE}{\mid=}_{TF};$$

$$\overset{PE}{\mid=}_{TF} \subset \overset{PE}{\mid=}_{T}, \quad \overset{PE}{\mid=}_{TF} \subset \overset{PE}{\mid=}_{F},$$

$$\overset{PE}{\mid=}_{T} \subset \overset{PE}{\mid=}_{IR}, \quad \overset{PE}{\mid=}_{F} \subset \overset{PE}{\mid=}_{IR};$$

$$\overset{R}{\mid=}_{TF} \subset \overset{RM}{\mid=}_{TF}, \quad \overset{P}{\mid=}_{TF} \subset \overset{PE}{\mid=}_{TF},$$

$$\overset{P}{\mid=}_{T} \subset \overset{PE}{\mid=}_{T}, \quad \overset{P}{\mid=}_{F} \subset \overset{PE}{\mid=}_{F},$$

$$\overset{P}{\mid=}_{IR} \subset \overset{PE}{\mid=}_{IR};$$

$$\overset{R}{\mid=}_{TF} \subset \overset{P}{\mid=}_{TF}, \quad \overset{P}{\mid=}_{TF} \subset \overset{P}{\mid=}_{T},$$

$$\overset{P}{\mid=}_{TF} \subset \overset{P}{\mid=}_{F},$$

$$\overset{P}{\mid=}_{T} \subset \overset{P}{\mid=}_{IR}, \quad \overset{P}{\mid=}_{F} \subset \overset{P}{\mid=}_{IR};$$

$$\overset{P}{\mid=}_{T} \not\subset \overset{P}{\mid=}_{F} \text{ та } \overset{P}{\mid=}_{F} \not\subset \overset{P}{\mid=}_{T};$$

$$\overset{PE}{\mid=}_{T} \not\subset \overset{PE}{\mid=}_{F} \text{ та } \overset{PE}{\mid=}_{F} \not\subset \overset{PE}{\mid=}_{T};$$

відношення $\overset{RM}{\mid=}_{T}$, $\overset{RM}{\mid=}_{F}$, $\overset{PE}{\mid=}_{TF}$ не включаються одне в інше.

Відношення логічного наслідку індукують відповідні відношення логічної еквівалентності, вони поширюються на множини формул.

Відношення логічного наслідку для множин формул в загальному випадку логік квазіарних предикатів описано, зокрема, в [2–5]. Дослідження таких відношень в логіках монотонних і логіках антитонних предикатів планується продовжити в наступних роботах.

Висновки

Досліджено відношення логічного наслідку для чистих першопорядкових композиційно-номінативних логік квазіа-

рних предикатів. Описано композиційні предикатні алгебри, мови і класи інтерпретацій (семантики) цих логік, виділено низку відношень логічного наслідку. Основну увагу зосереджено на вивченні таких відношень в логіках монотонних предикатів і логіках антитонних предикатів, для них визначено 20 відношень логічного наслідку, із яких попарно різними є $\overset{PE}{\models}_{IR}$, $\overset{PE}{\models}_{T}$, $\overset{PE}{\models}_{F}$, $\overset{PE}{\models}_{TF}$, $\overset{RM}{\models}_{T}$, $\overset{RM}{\models}_{F}$, $\overset{RM}{\models}_{TF}$. Наведено приклади, які засвідчують відмінності одних відношень від інших, встановлено співвідношення між різними відношеннями. Результати щодо наявності чи відсутності того чи іншого відношення логічного наслідку для певних пар формул зведено в таблицю.

1. *Handbook* of Logic in Computer Science / Edited by S. Abramsky, Dov M. Gabbay and T. S. E. Maibaum. – Oxford University Press, Vol. 1–5, 1993–2000.
2. *Нікітченко М.С., Шкільняк С.С.* Прикладна логіка. Київ: ВПЦ Київський університет, 2013. 278 с.
3. *Nikitchenko M., Shkilniak S.* Semantic Properties of Logics of Quasiary Predicates. Workshop on Foundations of Informatics: Proceedings FOI-2015. Chisinau, Moldova. P. 180–197.
4. *Шкільняк О.С.* Відношення логічного наслідку в логіках квазіарних предикатів. Проблеми програмування. 2016. № 1. С. 29–43.
5. *Нікітченко М.С., Шкільняк О.С., Шкільняк С.С.* Чисті першопорядкові логіки квазіарних предикатів. Проблеми програмування. 2016. № 2–3. С. 73–86.

References

1. Abramsky S., Gabbay D. and Maibaum T. (editors). (1993–2000). Handbook of Logic in Computer Science. Oxford University Press.
2. Nikitchenko M. and Shkilniak S. (2013). Applied logic. Kyiv: VPC Kyivskyi Universytet. (in Ukrainian).
3. Nikitchenko M. and Shkilniak S. (2015). Semantic Properties of Logics of Quasiary Predicates. In Workshop on Foundations of Informatics: Proceedings FOI-2015. Chisinau, Moldova. P. 180–197.
4. Shkilniak O. (2016). Logical consequence relations in logics of quasiary predicates. In Problems in Programming. N 1, P. 29–43. (in Ukrainian).
5. Nikitchenko M., Shkilniak O. and Shkilniak S. (2016). Pure first-order logics of quasiary predicates. In Problems in Programming. N 2–3, P. 73–86. (in Ukrainian).

Одержано 22.12.2016

Про автора:

Шкільняк Оксана Степанівна,
кандидат фізико-математичних наук,
доцент, доцент кафедри інформаційних систем.
Кількість наукових публікацій в українських виданнях – 81, у тому числі у фахових виданнях – 30.
Кількість наукових публікацій в зарубіжних виданнях – 9.
http://orcid.org/0000-0003-4139-2525.

***Місце роботи автора*:**

Київський національний університет імені Тараса Шевченка,
01601, Київ, вул. Володимирська, 60.
Тел.: (044) 259 0511, (050) 356 4875.
E-mail: me.oksana@gmail.com

УДК 004.853, 004.55

Ю.В. Рогушина

КЛАСИФІКАЦІЯ ЗАСОБІВ ТА МЕТОДІВ СЕМАНТИЧНОГО ПОШУКУ В WEB

Розглянуто проблеми, пов'язані з удосконаленням пошуку інформації у відкритому середовищі, обґрунтована потреба в його семантизації. Проаналізовано сучасний стан та перспективи розвитку систем семантичного пошуку, орієнтованих на обробку інформаційних ресурсів Web, розглянуто критерії класифікації таких систем. В цьому аналізі значна увага приділяється використанню у семантичному пошуку онтологій, що містять знання щодо предметної області пошуку та користувача, для якого виконується пошук.

На основі аналізу властивостей існуючих систем семантичного пошуку з точки зору цих критеріїв виділені області подальшого вдосконалення цих систем, запропоновано їх реалізацію у системі семантичного пошуку "МАІПС".

Ключові слова: семантичний пошук, онтологія, Semantic Web, персоніфікація пошуку.

Вступ

Сьогодні Web забезпечує доступ до значної частки інформаційних ресурсів людства. Обсяг такої інформації постійно збільшується, а її структура стає все складнішою та більш гетерогенною.

Пошук у Web – це ключова технологія, тому що використання *інформаційно-пошукових систем* (ІПС) є основним способом отримати доступ до вмісту його ресурсів і забезпечити їх ефективне використання. Це зумовлює потребу в розвитку засобів інформаційного пошуку, що враховують специфіку відкритого інформаційного середовища та здатні забезпечити специфічні інформаційні потреби окремих користувачів, позбавивши їх від рутинного аналізу тих *інформаційних ресурсів* (ІР), які не стосуються їх поточних інтересів.

Традиційно *інформаційні потреби* (ІП) користувачів формалізуються за допомогою *пошукового запиту* (ПЗ), на основі якого будується *пошуковий образ запиту* (ПОЗ). Такий ПОЗ співставляється *з пошуковими образами документів* (ПОД), доступ до контенту яких має система, що здійснює інформаційний пошук. Першими ПОД були дані бібліотечних, що містили назви книг та імена їх авторів. Класичними моделями інформаційного пошуку вважають булеві, імовірнісні, векторні та дескрипторні моделі, на основі яких виконується співставлення ПОД та ПОЗ. Яку саме інформацію про документи відображають ПОЗ та ПОД і як саме вони співставляються, залежить від конкретної реалізації пошукової системи.

Зазвичай пошуковий запит складається лише з кількох слів, але нині цього стає недостатньо, тому що кожному такому запиту релевантна надто велика кількість ІР, і виникає потреба в застосуванні додаткової інформації щодо інформаційної потреби користувача як для самого пошуку, так і для ранжування його результатів.

Інформаційні потреби залежать від персональних властивостей користувача, тобто за тим самим запитом різні користувачі прагнули б отримати різну інформацію (приміром, запит "прогноз погоди" є релевантним різноманітним синоптичним сайтам, але у більшості випадків користувача цікавить погода у тому місті, де він знаходиться). Можна ідентифікувати певні ІП як одноразові (у більшості випадків – довідкова інформація або однозначно ідентифікований інформаційний об'єкт) або постійні (приміром, пов'язані з професійною діяльністю або хобі, коли користувач зацікавлений у постійному отриманні нових відомостей з певної тематики), але більшість з них мають певну скінчену довжину у часі і перестають бути актуальними через певний час – користувач знаходить відомості, що йому потрібні, або змушений діяти без цих відомостей. Крім того, ІП безпосередньо пов'язані з тією поточною задачею, яку вирішує користувач, тобто пертинентність відповіді може

залежати від часу доби, дня тижня та інших додаткових ознак (приміром, у робочі дні та у вихідні або свята користувача цікавлять різні проблеми). Такі відомості щодо своїх ІП може безпосередньо вводити сам користувач, але це потребує багато зусиль та часу. Таким чином, виникає питання щодо того, яку саме інформацію доцільно враховувати в інформаційному пошуку, звідки її отримувати та як саме формалізовувати.

Основна тенденція розвитку семантичних пошукових засобів, які використовуються в найрізноманітніших сферах, пов'язана з переходом від виявлення документів, що містять певні ключові слова, до пошуку знань, необхідних для виконання поставленого перед користувачем завдання. Використання пошуку в Web, який в основному базується на комбінації пошуку тексту за ключовими словами з обчисленням ранжування важливості документів у залежності від структури посилань у мережі, має багато обмежень, і тому існує велика кількість науково-дослідних проектів, спрямованих у бік більш інтелектуальних форм отримання інформації, тобто *семантичного пошуку* [1, 2], під яким надалі будемо розуміти такий пошук інформації, що задовольняє ІП користувача в процесі розв'язання певної проблеми, в процесі якого застосовуються (наочно або приховано від користувача) знання щодо різних суб'єктів і об'єктів пошукової процедури й методи аналізу цих знань [3]. Ці знання можуть стосуватися як користувача та його інформаційних потреб (персоніфікація пошуку), так і ІР, серед яких здійснюється пошукова процедура.

Такий пошук, як правило, також ставить ціль забезпечити виконання більш складних пошукових запитів, оцінки яких потребують міркувань через Web. Інше поширене використання поняття семантичного пошуку в Web є пошук у великих наборах даних у Semantic Web, який у майбутньому замінить нинішній Web. Це використання тісно пов'язане з першим, тому що семантичне анотування Web-ресурсів та здобуття знань з Web-ресурсів, пов'язані з побудовою бази знань, яка може бути реалізована з використанням технології Semantic Web. Таким чином, семантичний пошук у Web можна аналізувати як розвиток існуючих пошукових засобів на основі нових семантичних технологій, орієнтованих на Web-застосування.

Ще один напрямок досліджень, тісно пов'язаний з семантичним пошуком, – це пошук у Web, де ІП формулюються як *природномовні вирази*. Існує багато підходів до перетворення таких запитів у формалізовані структуровані запити, які можуть виконуватися в різноманітних системах семантичного пошуку. Відповідями на ці запити можуть бути як звичайні Web-ресурси, так і структурована або природномовна інформація (приміром, фрагменти природномовних ІР або структурована інформація з Вікіпедії).

Часто поняття семантичного пошуку пов'язують також з деякими іншими семантичними концепціями. Наприклад, *фасетний* пошуку дозволяє вивчати результати відповідно з колекцією попередньо визначених категорій – фасет. У тісному зв'язку з семантичним пошуком знаходиться й кластерний пошук, де такі фасети можуть бути не тільки заздалегідь визначеними. Ще одним прикладом є надання рекомендацій щодо пов'язаних пошукових запитів, таких як продовження і корекції пошукових запитів, які добре відомі також у не семантичних ІПС. Також до проблеми семантичного пошуку стосується визначення ступеню повнотекстової подоби, де як запит використовуються не окремі ключові слова, а блоки тексту, починаючи від окремих фрази і до цілого документу. Досить близькою задачею є *онтологічний пошук подібності* (наприклад, [4]), що базується на подобі між онтологічними сутностями.

Постановка задачі

Ефективне застосування інформаційних технологій, орієнтованих на роботу з інформаційними ресурсами відкритого середовища Web, потребує аналізу вимог до засобів пошуку інформації та розробки критеріїв їх класифікації та оцінки. Це дозволить користувачам обирати такі систе-

ми семантичного пошуку, що більш ефективно задовольняють їх специфічні інформаційні потреби та вирішують різні типи проблем.

Специфіка пошуку в Web

Пошукові механізми, орієнтовані на пошук у Web, мають враховувати специфіку цього середовища:

- Web є *розподіленим*. Однією з рушійних чинників поширення в Web є відсутність централізації. Однак, оскільки Web є продуктом діяльності багатьох людей, відсутність централізованого контролю створює багато проблем для дослідження цієї інформації. По-перше, різні спільноти використовують різні словники, що призводить до проблем синонімії (коли два різних слова мають однакове значення) і багатозначності (коли те ж слово використовується в різних значеннях). По-друге, відсутність редагування або контролю якості означає, що надійність кожної сторінки потрібно перевіряти. Інтелектуальний Web-агент не може припускати, що вся інформація, яку він збирає, є коректною та не суперечною. Через те, що не може бути жодних глобальних обмежень цілісності Web, інформація з різних джерел може конфліктувати. Деякі з цих конфліктів можуть бути викликані тим, що різні спільноти – політичні, національні, релігійні, професійні тощо – можуть мати фундаментальні розходження в думках з певних питань.

- Web є *динамічним*. Web змінюється в неймовірно швидкому темпі, набагато швидше, ніж користувач або навіть спеціалізований програмний агент можуть відслідковувати усі зміни. Додаються нові сторінки, а зміст існуючих сторінок змінюється. Деякі сторінки досить статичні або міняються на регулярній основі, а інші змінюють контент з непередбачуваними інтервалами. Крім того, значущість таких змін може бути різною: від корекції орфографічних та пунктуаційних помилок, що практично не впливає на зміст документа, до таких, які можуть повністю змінити зміст документа або видалити великі обсяги інформації. Тому потрібно враховувати, що будь-яка знайдена в Web інформація може бути застарілою. Динамічність інформації у Web створює додаткову проблему для формалізації її семантики.

- *Web має великий обсяг* і постійно збільшується. Для масштабування за розміром постійно зростаючої мережі виникає необхідність в обмеженні виразності мови запитів або використанні спрощених алгоритмів виведення.

- Web – це *відкрите середовище*. Жоден Web-агент не може вважати, що він зібрав всі доступні знання (навіть найбільші ІПС індексують близько 25 % наявних сторінок).

Запити до Web-ресурсів можна поділити НА наступних класи відповідно до намірів користувачів [5]: *навігаційні* – побачити певний сайт; *інформаційні* – отримати певну інформацію; *транзакційні* – виконати якісь дії у Web. В роботі [6] проаналізовано різні визначення пошуку в Web, в яких ІПС, що орієновані на обробку ресурсів Web, розглядаються як комп'ютерні програми, що призначені для пошуку даних у мережі, та розглянуто ІПС різних поколінь. ІПС *першого* покоління (AltaVista, Excite, Webcrawler тощо) знаходило інформацію, що містилася безпосередньо на Web-сторінках, а ранжування результатів враховувало тільки контент знайдених сторінок. ІПС *другого* покоління для визначення релевантності IP використовують відомості про структуру самої мережі Web, аналізуючи посилання на відповідну сторінку й дані, що передаються за http-запитом, а також індекс популярності й репутацію IP. Вперше аналіз посилань між сторінками як один з основних факторів ранжування був використаний у Google (механізм PageRank), що й стало визначальним чинником популярності цієї ІПС. Для ІПС *третього* покоління характерна інтеграція відомостей, здобутих з різних джерел, на основі семантичного аналізу, що дає змогу враховувати в процесі пошуку та фільтрації як персональні інформаційні потреби користувачів, так і знання щодо ПрО пошуку [7]. Прикладами таких ІПС є Hakia, Google Squared, SenseBot та Wolfram Alpha. Ці ІПС спеціалізуються на пошуку в середовищі семантично струк-

турованої інформації та її метаописах у форматах OWL і RDF. З розвитком інфраструктури Semantic Web семантичні метадані стають більш доступними. Розробка стандартів семантичної розмітки, мов опису IP (RDF), онтологічних мов, сервісів, онтологічних баз, систем пошуку в семантичних даних (Swoogle, SWSE, WatsOn), точок SPARQL-доступу, систем логічного виведення, обробки правил тощо сприяє подальшому розвитку сфери інформаційного пошуку в напрямі використання семантики [8].

Семантизація пошукових систем

Різні підходи до пошуку інформації у Web, включаючи частково структуровані бази даних, машинне навчання та обробку природної мови, застосовувалися до проблеми аналізу та розуміння Web-сторінок у форматі HTML та XML. Проте відсутність семантики в цих засобах представлення інформації і в інструментах їх обробки значно обмежували якість методів. Тому виникає потреба у представленні знань, що пов'язані з тематикою пошуку.

Для того, щоб забезпечити значення для даних, знання мають бути представлені в певній формалізованій формі. На проблемі подання знань спеціалізується така галузь, як штучний інтелект. Ціллю подання знань є надання структур, які дозволяють ефективно зберігати інформацію, модифікувати її, здійснювати над нею логічне виведення. В процесі досліджень у цій сфері розвивалися численні мови подання знання з різними властивостями (від ранніх мов, таких як KL-ONE [9] і KRL [10] до сучасних мов опису онтологій OWL [11].

Для подання знань можна використовувати семантичні мережі, які представляють знання у вигляді набору вузлів, поєднаних поміченими зв'язками та фрейми. Розширені семантичні мережі та фреймові системи, як правило, включають у себе поняття абстракції, яка представлена за допомогою відношень *is-a* та *instance-of*. Відношення *is-a* вказує, що один клас є підкласом іншого, а відношення *instance-of* вказує, що поняття є членом класу. Ці відношення мають зв'язки з теорією множин: *is-a* – як відношення підмножини, *instance-of* – як відношення екземпляра множини. Множина відношень *is-a* визначає частковий порядок на класах, який часто називають таксономією або ієрархічною класифікацією. Таксономія може бути використана для узагальнення поняття до більш абстрактного класу або спеціалізації класу для своїх більш специфічних понять. Таксономії допомагають користувачам у пошуку необхідної інформації в Web (приміром, як це робиться в Yahoo і Open Directory).

Багато дослідників надають перевагу онтологіям для подання знань для інформаційного пошуку [12]. Якщо система представлення знань визначає, як представляти поняття, то онтологія визначає, які саме поняття визначені в такій системі й як саме вони взаємопов'язані. Такі формалізовані визначення дозволяють автоматизувати набагато глибші міркування, але такі визначення набагато важче побудувати.

Безпосередньо пов'язано з представлення знань у Web використання Resource Description Framework (RDF). Модель даних RDF [13] – це семантична мережа без успадкування: воно складається з вузлів, які пов'язані поміченими дугами, де вузли представляють собою Web-ресурсів, а дуги – атрибути цих ресурсів. RDF може бути вбудований у довільні Web-документів з використанням XML-синтаксису.

Щоб забезпечити створення контрольованих, сумісно доступних, розширюваних словників (наприклад, онтологій), створено специфікацію RDF Schema, яка визначає ряд властивостей зі специфічною семантикою. RDF Schema визначає властивості, які еквівалентні *instance-of* та *is-a*, які часто використовуються в поданні знань, а також засоби для опису властивостей, домену властивостей і діапазону тощо.

Хоча RDF є поліпшенням у порівнянні HTML і XML, цього недостатньо для відображення семантики, тому що він забезпечує дуже невеликий набір семантичних примітивів і має відносно слабкі меха-

нізми керування еволюцією схеми. Тому виникає потреба в мовах опису онтологій, таких як OWL.

Семантичний пошук – це метод інформаційного пошуку, в якому релевантність документа запиту визначається семантично (за близькістю змісту), а не синтаксично (приміром, за частотою використання ключових слів у документі) [14]. Його можна розглядати як надбудову над традиційним інформаційним пошуком, необхідну для підвищення пертинентності пошуку. Під *пертинентністю* пошуку будемо розуміти те, наскільки знайдена інформація задовольняє ІП користувача (на відміну від релевантності, яка визначає відповідність результатів пошуку наданому користувачем запиту).

Система семантичного пошуку (ССП) – це програмне забезпечення, що виконує семантичний пошук або незалежно, або як семантична надбудова над іншими ІПС. При цьому взаємодія ССП з іншим програмним забезпеченням і базами знань є наочною або прихованою від кінцевого користувача. Результатом роботи ССП може бути як здобуття інформації, яка приховано наявна в певному ІР (як текстовому, так і мультимедійному), так і надання користувачеві відомостей про наявні ІР у певному порядку та певній формі, що відповідають персональним потребам саме цього користувача [15].

Для семантичного пошуку у відкритому середовищі характерним є те, що потрібні для пошуку знання також динамічно здобуваються з цього відкритого середовища, а не закладаються у систему в процесі її створення.

Сучасні підходи до семантичного пошуку

У семантичному пошуку в Web виділяють:

1) підходи, що базуються на структурованих мовах запитів [16–22];

2) підходи, орієнтовані на некваліфікованих користувачів, які не потребують знайомства зі спеціалізованими мовами запитів.

У свою чергу, у другій групі можна виділити: підходи, де запити складаються із списків ключових слів [23–24]; підходи, де користувачі висловлюють запити природною мовою [25–29]. Більш детальний аналіз систем семантичного пошуку, класифікованих за цими параметрами, наведено в [30]. Розглянемо кілька прикладів цих підходів.

Одна з перших спроб створення семантичних запитів у Web – *SHOE* [31] – базується на структурованих мовах, та надає користувачеві:

1) інструмент для анотування Web-сторінок, що дозволяє користувачам додавати розмітку сторінки SHOE, обравши онтології, класи та властивості з списку;

2) Web-краулер, який шукає Web-сторінки з розміткою SHOE і зберігає інформацію щодо них в індексній базі знань;

3) машину виведення, яка забезпечує розмітку нових сторінок за допомогою правил виведення (в основному, правил логіки Хорна);

4) інструменти для запитів, які дозволяють користувачам задавати структуровані запити з використанням онтології.

Один з інструментів запитів дозволяє користувачам будувати граф, на якому вузли представляють постійні або змінні екземпляри, а дуги – відношення між ними. Щоб відповісти на запит, система знаходить підграфи, що відповідні до графа користувача. Пошуковий інструмент SHOE дозволяє користувачеві задавати запити, вибравши спочатку онтологію із списку, потім обрати класи та властивості з іншого списку. Після цього система будує запит, що з'єднує обрані елементи, здійснює цей запит до БЗ, і представляє результати його виконання у вигляді таблиці.

Більшість інших підходів цієї групи базуються на RDF. Наприклад, ІПС *Swoogle* [32] використовує краулер для виявлення, індексації та запитування документів у форматі RDF. Swoogle в основному забезпечує пошук документів та термінів Semantic Web (тобто, URI, класів і властивостей). Це дозволяє користувачам будувати запити, що містять умови щодо ме-

таданих на рівні документа (тобто, запити обробляють документи, що мають RDF як розширення файлу), а також дозволяє користувачам шукати документи Semantic Web, використовуючи RDF / XML як мову синтаксису. Знайдені документи ранжуються відповідно до алгоритму впорядкування, який враховує важливість документів у Semantic Web.

Система *Corese* [33] – це орієнтована на Semantic Web ІПС на онтологічній основі, яка знаходить Web-ресурси, що анотовані в RDF(S), за допомогою мови запитів, що базується теж на RDF(S). Corese дозволяє виконувати приблизний пошук у Semantic Web. Для приблизного пошуку використовуються правила виведення і обчислення семантичної відстані між класами та властивостями в онтологічних ієрархіях. За допомогою цього апарату Corese знаходить Web-ресурси, анотації яких є уточненням (спеціалізацією) запиту, а також ресурси, анотації яких відносяться до понять і відношень, які ієрархічно досить близькі до запиту.

Інший підхід до пошуку за структурованими запитами наведено в [34]. Метою цього підходу є апроксимація запитів до наборів даних RDF за допомогою SPARQL [35]. Для цього запит SPARQL кодується як набір трійок обмежень зі змінними, а приблизна відповідь – це заміна змінних з даними, які можуть не задовольняти всім обмеженням. Запропонована стратегія поступово покращує достовірність відповідей й спиняється, коли отриманий результат задовольняє користувача.

Подальший розвиток підходів на основі структурованих мов розглядається далі. *ONTOSEARCH2* [36] надає механізм для пошуку онтологій у Semantic Web. Він зберігає копії онтологій у легкій для обробки дескриптивній логіці і дозволяє виконувати запити SPARQL як до структури, так і до екземплярів онтології.

Система *Coraal* [37] – це ІПС на основі знань, що спеціалізується у пошуку літератури з біомедицини, яка використовує евристики на обробки природної мови. Це дозволяє аналізувати тексти і будувати з них трійки RDF, які інтегруються з наявними знаннями щодо ПрО пошуку, і користувач може надавати запити до всієї зібраної інформації за допомогою відповідної мови запитів.

NAGA – це семантична ІПС [38], яка пропонує мову запитів на основі графів для пошуку в базі знань (БЗ), яка також представлена у вигляді графу. БЗ створюється автоматично за допомогою інструменту для здобуття знань з Web-джерел на основі підходу, запропонованого в [39]. Вузли та ребра в графі знань представляють сутності й відношення між об’єктами, відповідно. Мова запитів NAGA розширює SPARQL таким чином, що дозволяє будувати складні запити у вигляді графів з регулярними виразами відносно відношень на мітках дуг. Відповідями на запит є підграфи графа знань, які відповідають графу запиту і впорядковуються за допомогою моделі зважених помічених графів.

Семантичний пошук за ключовими словами

Розглянемо кілька прикладів підходів до семантичного пошуку на основі ключових слів. ССП *OntoSelect* [40] фокусується на питаннях, пов’язаних з пошуком онтологій. Ця система дозволяє користувачам шукати онтологію, вказавши її назву або тему, яка цікавить користувача. Щоб описати область своєї зацікавленості, користувач може вказати URL Web-документа, який містить інформацію щодо цієї теми. Потім за допомогою лінгвістичних та статистичних методів автоматично будується набір ключових слів, які використовуються для пошуку.

В роботі [41] описано підхід, який фокусується на покращенні результатів традиційного пошуку за ключовими словами за допомогою даних, отриманих від Semantic Web. Коли користувач надає запит, то терміни цього запиту (ключові слова) відображаються у вузли Semantic Web. У випадку неоднозначного співставлення використовуються евристики (наприклад, з урахуванням профілю користувача) для зняття неоднозначності. Після того, як знайдені вузли, що відповідають пошуковим термінам, використовує евристики для

того, щоб визначити, яка саме частина графу Semantic Web навколо цих вузлів має бути надана користувачеві як результат пошуку (тобто, яким є поріг N для надання перших N трійок).

Пізніші підходи до семантичного пошуку за ключовими словами в основному орієнтовані на некваліфікованих користувачів. ССП *SemSearch* [42] забезпечує подібний до Google інтерфейс запитів, що не потребує від користувачів мати знання щодо онтологій або мов. Запити користувачів складаються з двох або більше ключових слів, семантичне значення яких враховується для переформулювання запиту відповідно до синтаксису формальної мови запитів. Ключовим словам присвоюються семантичні значення шляхом співставлення їх з набором класів, властивостей і екземплярів у репозиторії семантичних даних. Кожному ключовому слову може відповідати клас, властивість або екземпляр. Розглядаються різні комбінації семантичних співставлень ключових слів. Наприклад, кожному ключовому слову може відповідати клас, або першому ключовому слову відповідає клас, а другому – властивість і так далі. Всі комбінації співставлень враховуються в процесі переформулювання запиту, а за кожною комбінацією створюється вдосконалений формальний запит, який отримується із заздалегідь визначеного набору шаблонів запитів. Після переформулювання формальні запити точно оцінюються, і це забезпечує результати, які семантично пов’язані з усіма ключовими словами із запиту користувача.

У роботі [43], аналогічно до розглянутого вище підходу, запити за ключовими словами трансформуються в кон’юнктивні запити, які мають бути оцінені проти відносно базової БЗ. Але структура таких формальних запитів, які оцінюються, не відповідає якимось попередньо визначеним шаблонам. Формальні запити будуються шляхом використання техніки на основі графів, щоб знайти зв’язки між сутностями в запиті користувача. Трансформація запиту складається з трьох кроків:

- ключові слова запиту співставляються з елементами онтології;
- аналізуються відношення між цими елементами онтології, і на основі цього будуються підграфи БЗ, кожен з яких являє собою набір співвідношень, що пов’язують всі розглянуті елементи, а множина цих підграфів представляє всі можливі відношення між ключовими словами запиту користувача, які не можуть бути явно задані цим користувачем, тобто ці підграфи відповідають різним запитам, що можуть зацікавити користувача;
- ці підграфи трансформуються відповідно до правил мови подання запитів та генеруються формальні запити, які співставляються із вмістом БЗ.

Falcons [22] відноситься до ІПС на основі ключових слів для Semantic Web. Ця система підтримує пошук концептів та об'єктів. Пошук концептів здійснюється шляхом пошуку класів і властивостей в онтології, обраній користувачем, які відповідають умовам запиту. Крім того, інші онтології можуть бути рекомендовані користувачу на основі комбінації методу TF-IDF і рейтингу популярності онтологій. Пошук об’єкта виконується аналогічним чином: крім пошуку об’єктів, які відповідають умовам запиту, система також рекомендує інші типи об’єктів, які, ймовірно, теж зацікавлять користувача.

SWSE [23] і *Sig.Ma* [44] – це інструменти, що дозволяють користувачам знаходити об’єкти RDF за ключовими словами. Зокрема, результат пошуку за ключовим словом в SWSE – це список об’єктів, що співставлені з цим ключовим словом з невеликим описом і ім’ям концепту, приміром, “Person”, “Professor”. Якщо користувач натискає на "Person", то результати фільтруються і користувачу показують тільки список об’єктів класу "Person". Інформація про об’єкт інтегрується з кількох джерел і представлена в однорідному вигляді.

Ядром SWSE є розподілена архітектура YARS2 [45] для індексації та пошуку наборів даних RDF. YARS2 збирає фрагменти інформації і агрегує їх або за рахунок використання URI об’єктів (у тому випадку, якщо унікальний ідентифікатор використовується в різних джерелах), або шляхом використання інших методів кон-

солідації об'єкта. Крім того, SWSE дозволяє досвідченим користувачам будувати складні запити у SPARQL.

Подібно до SWSE, Sig.Ma об'єднує результати з кількох джерел, забезпечуючи користувачеві, крім самих ресурсів, узагальнений погляд на інформації. Фаза зняття неоднозначності аналогічна SWSE, але в цьому випадку дії користувача використовуються для усунення нерелевантних джерел. Sig.Ma також дозволяє користувачам вказувати список не тільки об'єктів, але й інших властивостей. Ключові слова користувача переводяться в набір запитань, деякі з них звертаються до Yahoo Boss [46] для отримання Web-сторінок, водночас як інші звертаються до індексу даних Semantic Web Sindice [47], щоб зібрати об'єкти і властивості RDF. Нарешті, вся отримана інформація інтегрується, використовуючи деякі евристики, що базуються на використанні URI і методів консолідації міток.

Новий підхід, спрямований на допомогу користувачеві в побудові семантичних запитів за запитами з ключових слів, представлено в системі *QUICK* [48]. Семантичний запит – це запит, який враховує онтологію відповідної ПрО. Коли користувач формулює запит з ключового слова, QUICK починає роботу з того, перетворює його в кілька семантичних запитів, кожен з яких отримується шляхом присвоєння якогось онтологічного поняття (властивість, об'єкт тощо з обраної онтології) для кожного з ключових слів. Потім користувач має обрати серед семантичних запитів, згенерованих системою, найбільш відповідний його інтересам. Якщо жоден семантичної запит, згенерований QUICK, не подобається користувачеві, то користувач самостійно може вести систему до генерації відповідного запиту, забезпечуючи додаткові характеристики (наприклад, вказуючи, що певне ключове слово має бути використане як властивість або як об'єкт).

Серед ІПС на основі ключових слів для Semantic Web важливо відмітити *YahooSearch-Monkey* [49], яка ставить ціллю поліпшення якості результатів пошуку ІПС Yahoo! Вона дозволяє власникам сторінки вказувати, яку інформацію по Web-сторінці вони хочуть запропонувати для відображення на сторінці результатів пошуку Yahoo! Видавці можуть надати ці характеристики у вигляді мікроформатів або метаданих RDF, які будуть автоматично отримані в процесі сканування сторінок і нададуть ІПС велику кількість інформації.

Семантичний пошук за природномовними запитами

Багато ССП виконують пошук на основі обробки природної мови. Приміром, у системі *ORAKEL* [27] запити спочатку переводяться в логічну форму, а потім переформулюються відповідно до цільової мови, тобто тієї мови, на якій базується БЗ. Переклад з логічної форми на цільову мовою описується декларативно за допомогою Prolog-програми. Загальний підхід не залежить від конкретної цільової мови, тому зміна мови онтології вимагає тільки декларативного опису трансформації в Prolog, але ніяких подальших змін в основній системі. Система орієнтована на користувача певного типу – інженера лексики, який визначає, як вирази природної мови відображаються на предикатів в БЗ, тобто, як дієслова, прикметники і відносні іменники тощо відображаються на відношення в онтології ПрО.

В роботі [29] представлена система семантичного пошуку, яка підтримує як пошук у Semantic Web через онтології, так і семантичний пошук у документах, що не відносяться до Semantic Web. Для першого виду пошуку, відповіді на запит природною мовою здобуваються шляхом застосування системи PowerAqua [50], яка працює таким чином: запит користувача перекладається з природної мови в структурований формат – "лінгвістичну трійку", а терміни цієї трійки відображаються на семантично релевантні об'єкти онтології. Потім обираються ті онтологічні сутності, які найкраще представляють запит користувача.

ССП *PowerAqua* розширює систему Aqua-Log, запропоновану в [51], що працює з використанням тільки однієї онтології, на випадок кількох онтологій, та вирі-

шує проблему неповноти знань шляхом переходу до традиційного пошуку за ключовими словами, якщо немає онтології, що задовольняє запиту. Другий вид пошуку в [29], а саме, семантичний пошук документів, що не відносяться до Semantic Web, здійснюється шляхом розширення системи, запропонованої в [52], за допомогою нового підходу до анотування документів, який містить такі етапи:

- здобуття текстового представлення семантичних сутностей;
- пошук цього текстового представлення у Web-документах;
- створення анотації, яка пов'язує семантичні сутності з кожним документом, що містить їх текстові представлення.

Сучасний підхід для побудови запитів SPARQL із запитів природною мовою представлено в [53]. Першим кроком у генерації запитів SPARQL є трансформація природномовних запитів у набір понять онтології (класів, екземплярів, властивостей і літералів), які базуються на призначенні правильного поняття онтології для кожного слова. Якщо система не може автоматично призначити правильне поняття онтології для слова, то викликається користувач для виконання цієї дії. Вибір користувачів використовується для навчання системи, щоб поліпшити її характеристики. Другим кроком є будівництво трійок онтологічних понять, які, нарешті, вставляються в розділи SELECT та WHERE для генерації запиту SPARQL. Результати оцінки отриманого SPARQL запиту показують користувачу в табличному та в графічному вигляді.

Використання онтологій у семантичному пошуку

Підхід до семантичного пошуку в Web, представлений в [54], базується на структурованій мові запитів, яка дозволяє сформулювати складні пошукові запити на основі онтологій. Онтологічно збагачений Web разом з комплексним пошуком на основі онтологій здійснюється на базі існуючого Web і з використанням існуючих орієнтованих на Web ІПС. Замість того, щоб інтерпретувати на синтаксису ключових слів чином, фрагменти даних існуючих Web-сторінках пов'язані з якоюсь онтологічною БЗ, а потім інтерпретуються відносно цієї БЗ. Web-контент пов'язаний з семантичним анотаціям, або, з іншої точки зору, Web співставляється з онтологічною БЗ, які потім забезпечує семантичний пошук у Web щодо цієї онтології. Вважається, що семантичні анотації та онтології, що лежать в їх основі, надаються явно. Обробка запитів у цьому підході до семантичного пошуку в Web поділяється на:

- офлайнове виведення для попередньої обробки онтологічних знань з використанням стандартних методів виведення на онтологіях, щоб перетворити семантичні анотації в так звані складні семантичні анотації, які публікуються як стандартні Web-сторінки, тобто вони можуть бути знайдені за допомогою стандартних пошукових систем для Web;
- онлайнове скорочення складних запитів на основі онтологій для пошуку в Web у послідовні або стандартні запити для пошуку в Web, відповіді на які можуть бути отримані за допомогою звичайного пошуку в Web, а потім використані для побудови відповіді на оригінальний пошуковий запит на основі онтології.

Цей спосіб обробки семантичних запитів є онтологічно коректним (і в багатьох випадках також повним). Ранжування результатів пошуку базується на ранжируванні об'єктів ObjectRank, яке узагальнює (і може бути зведено до) звичайне рейтингування PageRank для Web-сторінок. Таким чином, основні частини онтологічного пошуку в Web фактично зводяться до сучасних ІПС. Але важливою перевагою даного підходу є те, що цей підхід може бути негайно застосовуватися до всього існуючого Web, і це можна здійснити на основі існуючих Web-технологій пошуку. Такий напрям досліджень спрямований на додавання структури та семантики на основі онтологій до існуючих ІПС у Web шляхом об'єднання існуючих Web-сторінок і запитів з онтологічними знаннями.

Онтологічні знання і анотації, що лежать в основі такого семантичного по-

шуку, можна класифікувати відповідно до їх вмісту:

- загальні знання (наприклад, знання, які містяться у Вікіпедії) для загального пошуку в Web на основі онтології;
- спеціалізовані знання (наприклад, біомедичні знань) для ієрархічного пошуку в Web на основі онтологій.

Інтерфейс на основі загальної онтології для більш досвідчених користувачів базується на повній потужності структурованого мови запитів (для яких базова онтології може бути візуалізована для підтримки формулювання запитів). Для менш досвідчених користувачів використовується заздалегідь визначені прості форми інтерфейси (наприклад, схожі на ті, які використовуються в Google для розширеного пошуку).

У роботі [55] досліджується варіант описаного вище підходу, який використовує індуктивні методи міркування на додаток до дедуктивних. Це додає системі здатність обробляти суперечність, шум і неповноту у даних.

Хоча існує багато підходів до семантичного пошуку в Web, і вже створено різноманітні системи на основі цих підходів, дослідження в цій області ще знаходяться на самому початку, і багато проблем відкриті для подальших наукових досліджень. Виконання Web-пошуку у вигляді повернення простих природномовних відповідей на прості запитання природною мовою все ще залишається неможливим, не кажучи вже про виконання Web-пошуку для запитів, відповідь на які залежить від певної ПрО.

Деякі з найбільш нагальних проблем пов'язані з тим, як автоматично трансформувати природномовні запити в формальні на основі онтологій, як автоматично додати семантичні анотації до Web-контенту та як автоматично здобувати знання з Web-контенту.

Іншим важливим питанням у дослідженнях семантичного пошуку в Web є створення та підтримка базових онтологій. Це може здійснюватися:

- вручну відповідними фахівцями (наприклад, як це робиться в семантичній Вікіпедії, де різні спільноти можуть визначати свої власні онтології);
- автоматично, наприклад, шляхом знаходження в Web вже існуючих фрагментів онтологічних знань та анотацій (з існуючих онтологій або фрагментів онтологій та з існуючих анотацій Web-сторінок у мікроформатах або RDFa);
- напівавтоматично за допомогою комбінації перших двох підходів.

Очевидно, що чим більше ступінь автоматизації, тим більше і потенційний розмір онтологій, які можуть бути побудовані, і менші витрати і зусилля для їх створення і підтримки. Таким чином, для дуже великих масштабів Web є бажаним дуже високий ступінь автоматизації. З цим тісно пов'язаний такий важливий напрямок досліджень, як еволюція, оновлення і відображення онтологій, які є основою семантичного пошуку в Web.

Важливим є й питанням щодо того, як враховувати явну та неявну контекстну інформацію, щоб адаптувати результати пошуку до потреб користувачів. Наприклад, потреби і мотивація користувачів можуть бути визначені в термінах онтології на основі строгих або м'яких обмежень і умовних переваг.

Критерії класифікації систем семантичного пошуку

У роботі [56] проаналізовано підходи до пошуку в Web текстових документів, не враховуючи пошук структурованих документів (XML, RDF, OWL тощо). Також не аналізуються ті підходи, які вимагають, щоб користувач формулював запити формальною мовою, наприклад, SQL або SPARQL, та рішення, де користувач задає не тільки ключові слова, а ще й, приміром, використовує запит за зразком. В цьому огляді аналізуються ІПС, орієнтовані на пошук інформації у Web або на метапошук на основі інших ІПС. При цьому аналізуються тільки ті рішення, які використовують семантичний пошук. Якщо традиційний пошук документів залежить головним чином від наявності ключових слів у документах, то семантичний пошук відрізняється від нього

тим, що використовує ще й певні знання щодо домену пошук. Знання домену може бути формалізована за допомогою онтології ПрО, класи якої описують поняття даної ПрО, екземпляри характеризують факти, а властивості – відношення між поняттями та фактами. Ресурсами називають як класи, так і екземпляри.

Класифікувати різні підходи до семантичного пошуку можна за такими параметрами: архітектура, з'єднаність (coupling), прозорість, контекст користувача, модифікація запитів, структура онтології і онтологічна технологія. Ці критерії не є повністю незалежними один від одного, але вони відображають важливі характеристики ССП. Слід відмітити, що існують й інші критерії для класифікації ССП, наприклад, продуктивність, масштабованість, розподіленість, адаптованість і рейтинг результатів, але вони менше пов'язані зі специфікою саме семантичного пошуку.

Семантичні ІПС (так само, як і традиційні) поділяються за архітектурою на два типи:

1) автономні машини пошуку (з власними краулером та індексною БД);

2) метапошукові системи, яка використовує індексні БД інших ІПС та інтегрує отримані від них результати.

З'єднаність – це критерій класифікації ІПС, який стосується взаємодії між документами і онтологіями. Існують два типи зв'язку – тісний та слабкий.

Тісний зв'язок між документами і онтологіями існує тоді, коли метадані документів відносяться безпосередньо до понять конкретної онтології або навпаки (приміром, коли окремі документи розглядаються як екземпляри відповідної онтології). Такий підхід дозволяє легко вирішувати проблему омонімії через вибір відповідного поняття в онтології, але це підвищує вартість семантичного анотування документа. На практиці такий підхід застосовують не тільки для спеціалізованих інформаційних систем, які керують даними в обмеженій ПрО або для певної організації, але й у більш загальних областях застосування, таких як пошук у Web.

Слабкий зв'язок між документами і онтологіями встановлюється тоді, коли документи не прив'язані до певної онтології. У такому випадку існує проблема вибору відповідної онтології для даної ПрО. Таким чином, семантична потужність слабо пов'язаних систем обмежена: приміром, важко подолати проблему омонімії. Але цей підхід широко застосовують для пошуку в Web, якщо тільки дуже невелика частина документів має семантичні анотування. ІПС зі слабким зв'язком можуть бути легко реалізовані у вигляді метапошукових систем.

Цей критерій має велике значення для вибору ССП через те, що слабкий зв'язок між документами і онтологіями значно обмежує точність семантичного пошуку, водночас як тісний зв'язок між ними дозволяє знаходити тільки анотовані документи, тобто обмежує його повноту.

Прозорість ІПС відносно взаємодії користувача з особливостями семантичними систем можна поділити на наступні типи:

- *прозорі*: семантичні можливості системи невидимі для користувача, тобто семантична ІПС виглядає так саме, як і традиційна;
- *інтерактивні:* системи можуть запитувати користувача щодо пояснень до запиту або рекомендувати внести модифікації у запит;
- *гібридні:* системи поєднують властивості інтерактивних і прозорих систем, тобто у стандартному випадку вони виступають як прозорі системи, але для дуже специфічних завдань можуть використовувати інтерактивну взаємодію з користувачем.

Прозорість ІПС є одночасно і недоліком, і перевагою: з одного боку, користувач позбавлений тривалих діалогів з системою, що робить ІПС зручною у використанні, але з іншого боку, користувач не може впливати на семантичні рішення такої системи (наприклад, для вирішення омонімії), що потенційно призводить до зниження точності пошуку.

Контекст користувача дозволяє оцінити корисність документів для того,

щоб краще задовольняти інформаційні потреби конкретного користувача. Можна виділити наступні класи ІПС відносно того, як саме вони можуть враховувати цей контекст:

- *навчання*: контекст користувача динамічно здобувається в процесі взаємодії користувача з ІПС. На основі запиту та історії уточнень цього запиту система прогнозує, які саме результати прагне отримати цей користувач. Якщо терміни запиту завжди належать до того самого онтологічного контексту, то на основі цього система може вирішити омонімію, обравши термін з цієї ж онтології.

- *жорстке кодування* (hard-coded) – цей підхід поділяє запити на так звані категорії питань, які визначають інформаційну потребу користувача. Система забезпечує фіксовану кількість категорії питань, які використовуються для оцінки запиту. Типові категорії можна визначити, яку інформацію необхідно надати користувачеві, наприклад, "розташування" або "загальні ресурси для". Це дозволяє вирішити омонімію, відокремлюючи, приміром, назви географічних об'єктів від прізвищ осіб.

Зв'язок запиту користувача з категорією питань може здійснюватися явно самим користувачем, або неявно – на основі належності до групи користувачів, або шляхом аналізу самого запиту. Як правило, в рішеннях на основі онтологій класи інформаційних потреб мають відповідати певним онтологічним структурам, таких як типів властивостей. Контекст користувача забезпечує важливу інформацію про інформаційні потреби користувача. Ця інформація може використовуватися також для модифікації запиту.

Семантична *модифікація запитів* користувача – це добре відомий метод пошуку інформації. В області семантичного пошуку часто використовують інформацію з онтології. Він грає центральну роль у багатьох семантичних пошукових системах. Різні методи були розроблені для того, щоб збільшити повноту і точність пошуку. Наявність онтологій дозволяє відносно легко підвищити повноту запиту: онтологія стає для пошукової машини джерелом більш загальних, порівняно з ключовими словами запиту, термінів, – їх надкласів в онтології. Дещо важче поліпшити точність запиту. Це потребує вирішення проблеми омонімії та вибору певного підкласу для кожного з використаних у запиті термінів.

Модифікувати запит можна вручну, на основі онтологічного графу та шляхом переписування запиту. *Вручну* – це найпростіший спосіб змінити запит, залишивши модифікацію запиту користувачеві. Коли користувач вводить запит, система повертає йому не тільки знайдені ним документи, але й відповідну частину онтології. Використовуючи знання щодо ПрО, отримані з цього фрагмента онтології, користувач може сам переформулювати запит, додавши або видаливши певні терміни.

Щоб оптимізувати запити користувачів *на основі графу*, необхідно підтримувати тісний зв'язок між базою документів і онтологією. При цьому як онтологічні поняття, так і документи розглядаються як вузли графу. Терміни запиту використовуються для того, щоб знайти відповідні вузли графу. З цих вузлів алгоритм обходить граф, щоб визначити документи, семантично пов'язані з ними. Ця ціль може бути досягнута, наприклад, за допомогою алгоритму поширення збудження. Модифікація запиту на основі графу відрізняється від перезапису запитів тим, що не будується новий запит, який потім має оброблятися в пошуковій системі, а замість цього безпосередньо повертаються відповідні документи. Крім того, така обробка аналізує запит у цілому, а не розкладає його на окремі терміни.

Перезапис запиту базується на тому, що запит можна оптимізувати за допомогою семантичної ІПС. Основні способи перезапису запиту – розширення, обрізка і заміщення термінів. У випадку *розширення* запит доповнюється термінами, які отримуються від онтологічного контексту термінів оригінального запиту. Залежно від структури онтології, можуть бути використані різні семантики.

Обрізка запиту видаляє терміни з запиту і має протилежний ефект. Також

вона може бути реалізована шляхом порівняння результатів обрізаного запиту з результатами початкового запиту. Використання розширення і обрізки запитів у тому випадку, коли запит складається з кон'юнкції (AND) термінів, викликає те, що запит стає більш специфічним з кожним додатковим членом, а коли запит складається з диз'юнкції (OR) термінів, то він стає більш загальним. Іншими словами, відносно інформаційної потреби користувача, довгі кон'юнкції у запиті дають високу точність, а довгі диз'юнкції забезпечують високу повноту пошуку.

Отже, обидва методи, розширення диз'юнктивних запитів і обрізка кон'юнктивних запитів, збільшують повноту пошуку, а обрізка диз'юнктивних запитів і розширення кон'юнктивних запитів збільшують точність пошуку. Багато систем забезпечують лише кон'юнктивні або диз'юнктивні запити, тобто терміни пошуку неявно і незмінно пов'язані тільки оператором AND або тільки оператором OR. Тому ідея диз'юнктивного розширення або обрізки не застосовується до систем, які забезпечують тільки кон'юнктивні запити.

Заміщення термінів запиту – це заміна термінів запиту іншими онтологічно пов'язаними з ними термінами. В цілому, терміни можуть бути замінені на синоніми, підкласи або надкласи з онтології для того, щоб збільшити точність або повноту пошуку відповідно. Заміщення розглядається окремо від розширення та обрізки з наступних причин: у порівнянні з набором результатів оригінального запиту диз'юнктивне розширення та кон'юнктивна є надмножиною результатів, а диз'юнктивна обрізка і кон'юнктивне розширення є підмножиною результатів. На відміну від цього, заміна може дати набір результатів, який лише частково перекриває початковий набір результатів.

Якщо користувач ітеративно виконує кілька запитів, щоб задовольняють якусь інформаційну потребу, заміщення може бути більш ефективним методом для того, щоб допомогти користувачеві відповідно до його нових знань щодо ПрО.

ССП на основі онтологій використовують таку *онтологічну структуру,* яка містить поняття та відношення між ними. Серед відношень можна виділити наступні:

- *анонімні* відношення, коли ІПС ігнорує ім'я та семантику відношення, а враховує лише наявність взаємозв'язку між двома поняттями онтології, який означає тільки, що ці поняття мають той самий контекст;
- *стандартні* відношення, до яких можна віднести синонімію, меронімію, заперечення, "клас-підклас", "клас-екземпляр", використання яких збагачує семантичні можливості пошуку, але також вводить залежність від онтологічних структур;
- відношення, *специфічні для ПрО,* які визначають семантику зв'язку між термінами конкретної ПрО (приміром, для географічної ПрО відношенням "бути столицею" можуть бути пов'язані екземпляри класів "місто" та "країна").

В процесі пошуку в ССП можуть застосовуватися будь-які комбінації відношень всіх трьох типів.

Онтологічна технологія пов'язана з тим, яка мова опису онтологій використовується (різні рішення використовують для подання онтологій різні мови – F-Logic, RDF, DAML (+OIL) та OWL) та з технологічними питаннями повторного використання та взаємодії онтологій.

Аналіз існуючих ССП на основі цих критеріїв показує, що в різних системах реалізовані різні підмножини семантичних можливостей. Це дозволяє виділити області подальшого розвитку ССП:

- аналіз засобів модифікації запитів, який забезпечив би кількісне порівняння методів модифікації запиту;
- розвиток метапошукових семантичних ІПС: існуючі метапошукові системи змінюють запити користувачів і переспрямовують їх до традиційних ІПС, але не можуть переспрямовувати пошук до інших ССП;
- аналіз визнання користувачів: у більшості існуючих ССП семантика є прозорою для користувача, але чим більше

інтерактивність цих ІПС, тим потужніше може бути система. Проте важко оцінити, наскільки користувач готовий до певних зусиль з семантичної взаємодії, спрямованих на те, щоб поліпшити результати його пошуку;

- збільшення гнучкості пошуку – багато семантичних ІПС орієнтовані на певну структуру онтології, тобто накладають обмеження на клас онтологій, які вони можуть використовувати, водночас як інші системи (орієнтовані на обробку «анонімних» властивостей) можуть впоратися з довільними онтологій, але забезпечують більш слабкі семантичні можливості;
- впорядкування знайдених документів на основі обраної онтології;
- інтеграція з системами керування контентом та документами (CMS /DMS);
- продуктивність та масштабованість – семантичні пошукові системи мають бути здатні конкурувати зі стандартними пошуковими ІПС.

Система семантичного пошуку “МАІПС”

Вищенаведений аналіз показує, що при розробці засобів семантичного пошуку в Web необхідно звернути увагу на наступні питання:

- надати можливість використовувати зовнішні бази знань, які містять знання в сфері інформаційних потреб користувача;
- застосовувати інтероперабельні засоби подання знань, що підтримують розподілене та повторне використання відомостей щодо предметної області, що цікавить користувача, щодо інформаційних об’єктів, доступ до яких він прагне отримати, та щодо його особистих можливостей, пов’язаних із сприйняттям інформації;
- забезпечити користувачам можливість самостійно встановлювати, які саме знання використовуються в процесі пошуку (приміром, надати можливість обирати онтологію ПрО, а потім за цією онтологією модифікувати та вдосконалювати пошукові запити);
- враховувати історію взаємодії користувача з ІПС та контекст пошуку, щоб персоніфікувати результати пошуку та впорядковувати їх відповідно до інформаційних потреб користувача;
- забезпечити візуалізацію тих знань, що застосовуються для семантичного пошуку, щоб пояснити користувачеві шляхи отримання результатів та запобігти неправильного розуміння його інформаційних потреб;
- підтримувати збереження історії взаємодії користувача з ІПС, щоб враховувати його персональні потреби та зменшити час обробки запитів (приміром, зберігати постійні інформаційні запити та відомості про онтології, що використовувалися для їх побудови та виконання;
- підтримувати переадресацію семантично збагачених запитів до інших ІПС – як традиційних, так і семантичних;
- підтримувати колаборативний пошук та враховувати досвід взаємодії з іншими користувачами.

Всі ці питання були враховані в процесі проектування та реалізації ССП “МАІПС” [57], яка орієнтована на користувачів, що мають у мережі постійні інформаційні інтереси. Для цього “МАІПС” надає можливість зберігати й повторно виконувати запити, враховувати реакцію користувача на раніше запропоновані йому ІР (персональна фільтрація), відстежувати появу аналогічних запитів у інших користувачів (колаборативна фільтрація), зберігати формальний опис сфери інтересів користувача у вигляді онтології (семантична фільтрація) тощо.

Крім того, у ”МАІПС” у процесі профілювання користувачів використовується специфічний для природномовних ІР критерій оцінювання – складність тексту для розуміння. Особливістю системи є використання оригінального знання-орієнтованого алгоритму, що дає змогу визначити складність розуміння тексту для конкретного користувача (для того, щоб формалізувати рівень обізнаності користувача в

певних ПрО, використовуються тезауруси тих предметних областей, що цікавлять користувачів) [58]. Основою "МАІПС" є технології Semantic Web, зокрема мова подання онтологій OWL, і засоби його обробки. Для подання знань, що цікавлять користувача, використовуються онтології ПрО та тезауруси задач, що базуються на них: тезаурус будується користувачем за відповідною онтологією самостійно, а онтологія вибирається з набору онтологій, запропонованих на сайті розробниками системи.

"МАІПС" базується на онтологічній моделі, що здійснює опис семантики взаємодії користувачів і ресурсів "МАІПС" в інформаційному просторі Web [59]. Користувач має вибрати онтологію, що характеризує ПрО, яка його цікавить (якщо онтології немає, то її слід сформувати самостійно, наприклад, побудувати в Protege чи знайти у Web).

В обраній онтології користувач відбирає множину термінів, що стосуються його запиту, та формує з них *тезаурус запиту*. Користувач може позначити терміни, наявність яких у шуканому документі є бажаною або небажаною, а також задати більш складні операції, наприклад, автоматично позначати всі терміни, що знаходяться у певних відношеннях із термінами, позначеними раніше. Це дає змогу, зокрема, легко враховувати під час пошуку синоніми чи близькі за значенням слова, а також здійснювати пошук відразу кількома мовами. Унаслідок цього формується непорожня множина слів (чи словосполучень) $W = \{w_1, ..., w_m\}$, кожне з яких може мати свою позитивну або негативну вагу v_k, $k = \overline{1,m}$. Після цього для кожного документа i_j, $j = \overline{0,k}$ з множини $I`, I` \subseteq I$ формується коефіцієнт відповідності контексту пошуку:

$$s_j, j = \overline{0,k}, s_j = \sum_{k=1}^{m} v_k * f(i_j, w_k),$$

де $f(i_j, w_k) = \begin{cases} 1, w_k \in i_j \\ 0, w_k \in i_j \end{cases}$.

Чим вищим є цей коефіцієнт, тим, імовірно, вищою буде релевантність документа запиту користувача.

У "МАІПС" реалізуються такі базові операції над тезаурусами, як об'єднання, перетинання, різниця. Крім того, підтримується побудова тезауруса за документом, тобто формується множина слів, що використовуються у визначеному ІР. Також користувач може редагувати тезауруси вручну. Кожне слово в тезаурусі має визначену вагу (ціле позитивне або негативне число), що визначає важливість слова для запиту. Негативні значення відповідають термінам, які користувач вважає небажаними. Тезаурус може відображатися у вигляді хмари тегів (розмір шрифту відображає вагу терміну в тезаурусі, а терміни з негативною вагою позначені червоним кольором) [7]. Спосіб виконання пошуку залежить від специфіки конкретних ІР (рис. 1).

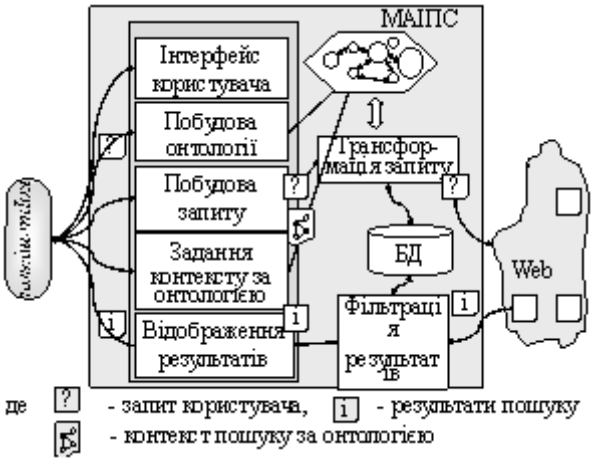


Рис. 1. Пошук в МАІПС на основі онтології

Проаналізувавши наявні відомості про інформаційну потребу користувача (для того, щоб обробка виконувалася на семантичному рівні, "МАІПС" використовує знання, що містяться у відповідних зовнішніх і внутрішніх онтологіях), система передає запит до зовнішніх ІПС. Сьогодні в ролі такої ІПС використовується Google, як найдосконаліша з сучасних пошукових систем. Отримавши у відповідь від зовнішньої ІПС набір інформаційних ресурсів, "МАІПС" намагається здобути з них потрібні користувачеві ві-

домості. У найпростішому варіанті, якщо потрібний користувачеві ІО є документом (можливо, певного типу), то система перевпорядковує отримані посилання на ІР з урахуванням персональних особливостей користувача та збережених у БД системи відомостей щодо цих ІР. В результаті виконання пошуку користувачу надається така інформація: тезаурусний рейтинг ІР, посилання на ІР, його назва, анотація й оцінки легкості читання (рис. 2). Припускається, що ІР, у яких зустрічається більше слів з тезаурусу, більш релевантні потребам користувача.

Рис. 2. Результати пошуку в системі “МАІПС”

“МАІПС” реалізована як серверне Інтернет-застосування мовою PHP версії 5. 0. Для збереження внутрішніх даних використовується XML (надалі планується використання СУБД MySQL). Онтології зберігаються у форматах RTF і OWL, тезауруси – у форматі XML.

Висновки

У дослідженні, результати виконання якого наведено в цій роботі, проаналізовано сучасний стан та перспективи розвитку систем семантичного пошуку, орієнтованих на обробку інформаційних ресурсів Web, розглянуто критерії їх класифікації. На основі цього аналізу запропоновано засоби та методи персоніфікації семантичного пошуку та зовнішні джерела інформації, з яких доцільно здобувати знання, які дозволяють підвищити пертинентність пошуку і більш повно та точно задовольняти сталі інформаційні потреби користувачів. Ці підходи реалізовано в системі семантичного пошуку “МАІПС”, яку також описано у даній роботі.

1. *Hendler J.* Web 3.0: The dawn of semantic search. Computer. 2010. 43(l)/. P. 77–80.
2. *Baeza-Yates R., A. Raghavan R.* Next generation Web search. S. Ceri and M. Brambilla, editors, Search Computing, Springer. 2010. P. 11–23.
3. *Lawrence S.* Context in the Web Search – http://citeser.nj.nec.com/ lawrence00context.html.
4. *Janowicz K., Wilkes M., Lutz M.* Similarity-based information retrieval and its role within spatial data infrastructures. Proc. GIScience-2008, Springer. 2008. P. 151–167.
5. *Broder A.* A taxonomy of web search, IBM Research, ACM SIGIR Forum archive. Vol. 36 , Issue 2 (Fall 2002). P. 3–10.
6. *Гришанова І.Ю.* Аналітичний огляд методів і засобів інформаційного пошуку в Semantic Web. Проблеми програмування. 2016. № 1. С. 51–72.
7. *Рогушина Ю. В.* Семантичний пошук у Web на основі онтологій: розробка моделей, засобів і методів. Мелітополь: МДУПУ ім. Б.Хмельницького. 2015. 291 с.
8. *Гладун А.Я., Рогушина Ю.В.* Семантичні технології: принципи та практики. К.: ТОВ "АДЕФ-Україна". 2016. 308 с.
9. *Brachman R., Schmolze J.* An overview of the KL-ONE knowledge representation system. Cognitive Science, 1985. 9(2).
10. *Bobrow D., Winograd T.* An overview of KRL, a knowledge representation language. Cognitive Science 1(1) (1977).
11. *Antoniou G., Van Harmelen F.* Web ontology language: Owl. In Handbook on ontologies. Springer Berlin Heidelberg. 2004. P. 67–92.
12. *Gruber T.* A Translation Approach to Portable Ontology Specifications. Knowledge Acquisition. 1993. N 5. P. 199–220.
13. *Cyganiak R., Wood D., Lanthaler M.*RDF 1. 1 Concepts and Abstract Syntax. W3C Recommendation 25 February 2014. http://www.w3.org/TR/2014/REC-rdf11-concepts-20140225/.
14. *Rogushina J.* Means of the semantic search personification on base of ontological approach. International Journal of Mathematical Sciences and Computing (IJMSC). 2016. Vol. 2. N 3. P. 1–20. – http://www.mecs-press.org/ijmsc/ijmsc-v2-n3/IJMSC-V2-N3-1.pdf.

15. *Rogushina J.* Use of the Ontological Model for Personification of the Semantic Search. International Journal of Mathematical Sciences and Computing (IJMSC). 2016. Vol. 2, N 1. http://www.mecs-press.org/ijmsc/ijmsc-v2-n1/IJMSC-V2-N1-1.pdf
16. *Corby O., Dieng-Kuntz R., Faron-Zucker C.* Querying the Semantic Web with Corese search engine. Proc. ECAI-2004, IOS Press. 2004. P. 705–709.
17. *Finin T.W., Ding L., Pan R., Joshi A., Kolari P., Java A., Peng Y.* Swoogle: Searching for knowledge on the Semantic Web. Proc. AAAI-2005,. AAAI Press / MIT Press, 2005. P. 1682–1683.
18. *Heflin J., Hendler J. A., Luke S.* SHOE: A blueprint for the Semantic Web. D. Fensel, W. Wahlster, and H. Lieberman, editors. Spinning the Semantic Web: Bringing the World Wide Web to Its Full Potential, MIT Press. 2003. P. 29–63.
19. *Kasneci G., Suchanek F.M., Ifrim G., Ramanath M., Weikum G.* NAGA: Searching and ranking knowledge. Proc. ICDE-2008, IIIEE Computer Society. 2008. P. 953–962.
20. *Oren E., Gueret C., Schlobach S.* Anytime query answering in RDF through evolutionary algorithms. Proc. ISWC- 2008, LNCS 5318, Springer. 2008. P. 98–113.
21. *Buitelaar P., Eigner T., Declerck T.* OntoSelect: A dynamic ontology library with support for ontology selection. Proc. Demo Session at ISWC-2004, 2004.
22. *Cheng G., Ge W., Qu Y.* Falcons: Searching and browsing entities on the Semantic Web. Proc. WWW-2008, ACM Press, 2008. P. 1101–1102.
23. *Harth A., Hogan A., Delbru R., Umbrich J., O'Riain S., Decker S.* SWSE: Answers before links. Proc. Semantic Web Challenge 2007, CEUR Workshop Proceedings 295. CEUR-WS.org, 2007.
24. *Lei Y., Uren V. S., Motta E.* SemSearch: A search engine for the Semantic Web. Proc. EKAW-2006, LNCS 4248, Springer, 2006. P. 238–245.
25. *Tran T., Cimiano P., Rudolph S., Studer R.* Ontology-based interpretation of keywords for semantic search. Proc. ISWC/ASWC-2007, LNCS 4825. Springer. 2007. P. 523–536.
26. *Zenz G., Zhou X., Minack E., Siberski W., Nejdl W.* From keywords to semantic queries. Incremental query construction on the Semantic Web. J. Web Sem., 7(3):, 2009. P. 166–176.
27. *Cimiano P., Haase P., Heizmann J., Mantel M., Studer R.*. Towards portable natural language interfaces to knowledge bases – The case of the ORAKEL system. Data Knowl. Eng., 65(2), 2008. P. 325–354.
28. *Damljanovic D., Agatonovic M., Cunningham H.* Natural language interface to ontologies: Combining syntactic analysis and ontology-based lookup through the user interaction. Proc. ESWC-2010, Part I, LNCS 6088. 2010. P. 106–120.
29. *Fernandez M., Lopez V., Sabou M., Uren V. S., Vallet D., Motta E., Castells P.* Semantic search meets the Web. Proc. ICSC-2008. 2008. P. 253–260.
30. *Fazzingaa B., Lukasiewicz T.* Semantic search on the Web. Semantic Web – Interoperability, Usability, Applicability. 2010. N 1. – P. 1–7. – https://www.researchgate.net/profile/Thomas_Lukasiewicz/publication/220575552_Semantic_search_on_the_Web/links/0046351e94ee8994bd000000.pdf.
31. *Heflin J., Hendler J. A., Luke S.* SHOE: A blueprint for the Semantic Web. In D. Fensel, W. Wahlster, and H. Lieberman, editors. Spinning the Semantic Web: Bringing the World Wide Web to Its Full Potential, MIT Press. 2003. P. 29–63. – http://www.cse.lehigh.edu/~heflin/pubs/swbook03.pdf.
32. *Finin T.W., Ding L., Pan R., Joshi A., Kolari P., Java A., Peng Y.* Swoogle: Searching for knowledge on the Semantic Web. Proc. AAAI-2005, AAAI Press / MIT Press. 2005. P. 1682–1683.
33. *Corby O., Dieng-Kuntz R., Faron-Zucker C.* Querying the Semantic Web with Corese search engine. Proc. ECAI-2004, IOS Press, 2004. P. 705–709.
34. *Oren E., Gueret C., Schlobach S.* Anytime query answering in RDF through evolutionary algorithms. Proc. ISWC- 2008, LNCS 5318, Springer. 2008. P. 98–113.
35. W3C. SPARQL Query Language for RDF, 2008. W3C Recommendation (15 January 2008). – http: / /www. w3.org/TR/rdf-sparql-query/.
36. *Thomas E., Pan J.Z., Sleeman D. H.* ONTOSEARCH2: Searching ontologies semantically. Proc. OWLED-2007, CEUR Workshop Proceedings 258. CEUR-WS.org, 2007.
37. *Novacek V., Groza T., Handschuh S.* CORAAL – Towards deep exploitation of textual resources in life sciences. Proc. ALME-2009, LNCS 5651, Springer, 2009. P. 206–215.
38. *Kasneci G., Suchanek F.M., Ifrim G., Ramanath M., Weikum G.* NAGA: Searching and

ranking knowledge. Proc. ICDE-2008, IIEE Computer Society, 2008. P. 953–962.
39. *Suchanek F.M., Kasneci G., Weikum G.* Yago: A core of semantic knowledge. Proc. WWW-2007, ACM Press, 2007. P. 697–706.
40. *Buitelaar P., Eigner T., Declerck T.* OntoSelect: A dynamic ontology library with support for ontology selection. Proc. Demo Session at ISWC-2004, 2004.
41. *Guha R.V., McCool R., Miller E.* Semantic search. Proc. WWW-2003, ACM Press, 2003. P. 700–709.
42. *Lei Y., Uren V.S., Motta E.* SemSearch: A search engine for the Semantic Web. Proc. EKAW-2006, LNCS 4248, Springer, 2006. P. 238–245.
43. *Tran T., Cimiano P., Rudolph S., Studer R.* Ontology-based interpretation of keywords for semantic search. Proc. ISWC/ASWC-2007, LNCS 4825, Springer. 2007. P. 523–536.
44. *Tummarello G., Cyganiak R., Catasta M., Danielczyk S., Delbru R., Decker S.* Sig.ma: Live views on the Web of data. Proc. WWW-2010, ACM Press, 2010. P. 1301–1304.
45. *Harth A., Umbrich J., Hogan A., Decker S.* YARS2: A federated repository for querying graph structured data from the Web. Proc. ISWC/ASWC-2007, LNCS 4825, Springer, 2007. P. 211–224.
46. YahooSearchBoss. http://developer.yahoo. com/ search/boss/.
47. *Delbru R., Polleres A., Tummarello G., Decker S.* Context dependent reasoning for semantic documents in Sindice. Proc. SSWS-2008, 2008.
48. *Zenz G., Zhou X., Minack E., Siberski W., Nejdl W.* From keywords to semantic queries – Incremental query construction on the Semantic Web. J. Web Sem., 2009. N 7(3). P. 166–176.
49. YahooISearchMonkey. – http ://developer. yahoo, com/searchmonkey.
50. *Lopez V., Sabou M., Motta E.* PowerMap: Mapping the real Semantic Web on the fly Proc. ISWC-2006, LNCS 4273, Springer, 2006. P. 414–427.
51. *Lopez V., Pasin M., Motta E.* AquaLog: An ontology- portable question answering system for the Semantic Web. Proc. ESWC-2005, LNCS 3532, Springer, 2005. P. 546–562.
52. *Castells P., Ferndndez M., Vallet D.* An adaptation of the vector-space model for ontology-based information retrieval. IEEE Trans. Knowl. Data Eng. 2007. N 19(2). P. 261–272.
53. *Damljanovic D., Agatonovic M., Cunningham H.* Natural language interface to ontologies: Combining syntactic analysis and ontology-based lookup through the user interaction. Proc. ESWC-2010, Part I, LNCS 6088, Springer, 2010. P. 106–120.
54. *Fazzinga B., Gianforme G., Gottlob G., Lukasiewicz T.* Semantic Web search based on ontological conjunctive queries. Proc. FoIKS-2010, LNCS 5956, Springer, 2010. P. 153–172.
55. *D'Amato C., Esposito F., Fanizzi N., Fazzinga B., Gottlob G., Lukasiewicz T.* Inductive reasoning and Semantic Web search. Proc. SAC-2010, ACM Press, 2010. P. 1446–1447.
56. *Mangol C.* A survey and classification of semantic search approaches. International Journal of Metadata, Semantics and Ontologies. 2007. N 2(1). P. 23–34.
57. *Рогушина Ю. В.* Семантический поиск как составляющая управления знаниями в Semantic Web. Материалы международной научно-технической конференции OSTIS-2012. Минск БГУИР. С. 239–244.
58. *Rogushina J.* Use of the Ontological Model for Personification of the Semantic Search. International Journal of Mathematical Sciences and Computing(IJMSC). 2016. Vol. 2. N 1, – http://www.mecs-press.org/ijmsc/ijmsc-v2-n1/IJMSC-V2-N1-1.pdf
59. *Рогушина Ю.В.* Разработка онтологической модели информационной потребности пользователя при семантическом поиске. Онтология проектирования. 2014. № 2(12). С. 61–82. – http://agora.guru.ru/scientific_ journal/files/Ontology_Of_Designing_2_2014 _shot.pdf.

References

1. Hendler J. Web 3.0: The dawn of semantic search. Computer, 2010, 43(l)/. P. 77–80.
2. Baeza-Yates R., A. Raghavan R. Next generation Web search // S. Ceri and M. Brambilla, editors, Search Computing, Springer, 2010. P. 11–23.
3. Lawrence S. Context in the Web Search – http://citeser.nj.nec.com/ lawrence00context.html.
4. Janowicz K., Wilkes M., Lutz M. Similarity-based information retrieval and its role within

spatial data infrastructures. Proc. GIScience-2008, Springer, 2008. P. 151–167.
5. Broder A. A taxonomy of web search, IBM Research, ACM SIGIR Forum archive, Vol. 36 , Issue 2 (Fall 2002), P. 3–10.
6. Grishanova I.Y. Analitic review of methods and tools of information search for Semantic Web. Problems in programming, 2016. N 1. P. 51–72. [in Ukrainian].
7. Rogushina J.V. Semantic retrieval for Web on base of ontologies: design of models, tools and methods. Melitopol: Bogdan Hmelnitsky MDUPU , 2015. 291 p. [in Ukrainian].
8. Gladun A.Y., Rogushina J.V. Semantic technologies: principles and practics. – K.: ADEF-Ukraine, 2016. 308 p. [in Ukrainian]
9. Brachman R., Schmolze J. An overview of the KL-ONE knowledge representation system. Cognitive Science, 1985, 9(2).
10. Bobrow D., Winograd T. An overview of KRL, a knowledge representation language. Cognitive Science 1(1) (1977).
11. Antoniou G., Van Harmelen F. Web ontology language: Owl. In Handbook on ontologies. Springer Berlin Heidelberg, 2004. P. 67–92.
12. Gruber T. A Translation Approach to Portable Ontology Specifications. Knowledge Acquisition. 1993. N 5. P. 199–220.
13. Cyganiak R., Wood D., Lanthaler M.RDF 1. 1 Concepts and Abstract Syntax. W3C Recommendation 25 February 2014. http://www.w3.org/TR/2014/REC-rdf11-concepts-20140225/.
14. Rogushina J. Means of the semantic search personification on base of ontological approach. International Journal of Mathematical Sciences and Computing (IJMSC), Vol. 2, N 3. 2016. P. 1–20. – http://www.mecs-press.org/ijmsc/ijmsc-v2-n3/IJMSC-V2-N3-1.pdf.
15. Rogushina J. Use of the Ontological Model for Personification of the Semantic Search. International Journal of Mathematical Sciences and Computing(IJMSC), Vol. 2, N 1, 2016. http://www.mecs-press.org/ijmsc/ijmsc-v2-n1/IJMSC-V2-N1-1.pdf/
16. Corby O., Dieng-Kuntz R., Faron-Zucker C. Querying the Semantic Web with Corese search engine. Proc. ECAI-2004, IOS Press, 2004. – P. 705–709.
17. Finin T. W., Ding L., Pan R., Joshi A., Kolari P., Java A., Peng Y. Swoogle: Searching for knowledge on the Semantic Web. Proc. AAAI-2005,. AAAI Press / MIT Press, 2005. P. 1682–1683.
18. Heflin J., Hendler J. A., Luke S. SHOE: A blueprint for the Semantic Web. D. Fensel, W. Wahlster, and H. Lieberman, editors. Spinning the Semantic Web: Bringing the World Wide Web to Its Full Potential, MIT Press, 2003. P. 29–63.
19. Kasneci G., Suchanek F.M., Ifrim G., Ramanath M., Weikum G. NAGA: Searching and ranking knowledge. Proc. ICDE-2008, IIIEE Computer Society, 2008. P. 953–962.
20. Oren E., Gueret C., Schlobach S. Anytime query answering in RDF through evolutionary algorithms. Proc. ISWC- 2008, LNCS 5318, Springer, 2008. P. 98–113.
21. Buitelaar P., Eigner T., Declerck T. OntoSelect: A dynamic ontology library with support for ontology selection. Proc. Demo Session at ISWC-2004, 2004.
22. Cheng G., Ge W., Qu Y. Falcons: Searching and browsing entities on the Semantic Web. Proc. WWW-2008, ACM Press, 2008. P. 1101–1102.
23. Harth A., Hogan A., Delbru R., Umbrich J., O'Riain S., Decker S. SWSE: Answers before links. Proc. Semantic Web Challenge 2007, CEUR Workshop Proceedings 295. CEUR-WS.org, 2007.
24. Lei Y., Uren V. S., Motta E. SemSearch: A search engine for the Semantic Web. Proc. EKAW-2006, LNCS 4248, Springer, 2006. P. 238–245.
25. Tran T., Cimiano P., Rudolph S., Studer R. Ontology-based interpretation of keywords for semantic search. Proc. ISWC/ASWC-2007, LNCS 4825, Springer, 2007. P. 523–536.
26. Zenz G., Zhou X., Minack E., Siberski W., Nejdl W. From keywords to semantic queries. Incremental query construction on the Semantic Web. J. Web Sem., 7(3):,2009. P. 166–176.
27. Cimiano P., Haase P., Heizmann J., Mantel M., Studer R.. Towards portable natural language interfaces to knowledge bases — The case of the ORAKEL system. Data Knowl. Eng., 65(2), 2008. P. 325–354.
28. Damljanovic D., Agatonovic M., Cunningham H. Natural language interface to ontologies: Combining syntactic analysis and ontology-based lookup through the user interaction. Proc. ESWC-2010, Part I, LNCS 6088, 2010. P. 106–120.
29. Fernandez M., Lopez V., Sabou M., Uren V. S., Vallet D., Motta E., Castells P. Semantic

search meets the Web. Proc. ICSC-2008, 2008. – P. 253–260.
30. Fazzingaa B., Lukasiewicz T. Semantic search on the Web. Semantic Web – Interoperability, Usability, Applicability, N 1, 2010. P. 1–7. https://www.researchgate.net/profile/Thomas_Lukasiewicz/publication/220575552_Semantic_search_on_the_Web/links/0046351e94ee8994bd000000.pdf.
31. Heflin J., Hendler J. A., Luke S. SHOE: A blueprint for the Semantic Web. In D. Fensel, W. Wahlster, and H. Lieberman, editors. Spinning the Semantic Web: Bringing the World Wide Web to Its Full Potential, MIT Press, 2003. P. 29–63. http://www.cse.lehigh.edu/~heflin/pubs/swbook03.pdf.
32. Finin T. W., Ding L., Pan R., Joshi A., Kolari P., Java A., Peng Y. Swoogle: Searching for knowledge on the Semantic Web. Proc. AAAI-2005, AAAI Press / MIT Press, 2005. P. 1682–1683.
33. Corby O., Dieng-Kuntz R., Faron-Zucker C. Querying the Semantic Web with Corese search engine. Proc. ECAI-2004, IOS Press, 2004. P. 705–709.
34. Oren E., Gueret C., Schlobach S. Anytime query answering in RDF through evolutionary algorithms. Proc. ISWC- 2008, LNCS 5318, Springer, 2008. P. 98–113.
35. W3C. SPARQL Query Language for RDF, 2008. W3C Recommendation (15 January 2008). – http://www. w3.org/TR/rdf-sparql-query/.
36. Thomas E., Pan J. Z., Sleeman D. H. ONTOSEARCH2: Searching ontologies semantically. Proc. OWLED-2007, CEUR Workshop Proceedings 258. CEUR-WS.org, 2007.
37. Novacek V., Groza T., Handschuh S. CORAAL – Towards deep exploitation of textual resources in life sciences. Proc. AlME-2009, LNCS 5651, Springer, 2009. P. 206–215.
38. Kasneci G., Suchanek F. M., Ifrim G., Ramanath M., Weikum G. NAGA: Searching and ranking knowledge. Proc. ICDE-
2008, IIEE Computer Society, 2008. P. 953–962.
39. Suchanek F. M., Kasneci G., Weikum G. Yago: A core of semantic knowledge. Proc. WWW-2007, ACM Press, 2007. P. 697–706.
40. Buitelaar P., Eigner T., Declerck T. OntoSelect: A dynamic ontology library with support for ontology selection. Proc. Demo Session at ISWC-2004, 2004.
41. Guha R. V., McCool R., Miller E. Semantic search. Proc. WWW-2003, ACM Press, 2003. – P. 700–709.
42. Lei Y., Uren V. S., Motta E. SemSearch: A search engine for the Semantic Web. Proc. EKAW-2006, LNCS 4248, Springer, 2006. P. 238–245.
43. Tran T., Cimiano P., Rudolph S., Studer R. Ontology-based interpretation of keywords for semantic search. Proc. ISWC/ASWC-2007, LNCS 4825, Springer, 2007. P. 523–536.
44. Tummarello G., Cyganiak R., Catasta M., Danielczyk S., Delbru R., Decker S. Sig.ma: Live views on the Web of data. Proc. WWW-2010, ACM Press, 2010. P. 1301–1304.
45. Harth A., Umbrich J., Hogan A., Decker S. YARS2: A federated repository for querying graph structured data from the Web. Proc. ISWC/ASWC-2007, LNCS 4825, Springer, 2007. P. 211–224.
46. YahooSearchBoss. – http://developer. yahoo.com/ search/boss/.
47. Delbru R., Polleres A., Tummarello G., Decker S. Context dependent reasoning for semantic documents in Sindice. Proc. SSWS-2008, 2008.
48. Zenz G., Zhou X., Minack E., Siberski W., Nejdl W. From keywords to semantic queries – Incremental query construction on the Semantic Web. J. Web Sem., N 7(3), 2009. P. 166–176.
49. YahooISearchMonkey. – http ://developer. yahoo, com/searchmonkey.
50. Lopez V., Sabou M., Motta E. PowerMap: Mapping the real Semantic Web on the fly Proc. ISWC-2006, LNCS 4273, Springer, 2006. P. 414–427.
51. Lopez V., Pasin M., Motta E. AquaLog: An ontology- portable question answering system for the Semantic Web. Proc. ESWC-2005, LNCS 3532, Springer, 2005. P. 546–562.
52. Castells P., Ferndndez M., Vallet D. An adaptation of the vector-space model for ontology-based information retrieval. IEEE Trans. Knowl. Data Eng., N 19(2), 2007. P. 261–272.
53. Damljanovic D., Agatonovic M., Cunningham H. Natural language interface to ontologies: Combining syntactic analysis and ontology-based lookup through the user interaction. Proc. ESWC-2010, Part I, LNCS 6088, Springer, 2010. P. 106–120.
54. Fazzinga B., Gianforme G., Gottlob G., Lukasiewicz T. Semantic Web search based on ontological conjunctive queries. Proc.

FoIKS-2010, LNCS 5956, Springer, 2010. P. 153–172.
55. D'Amato C., Esposito F., Fanizzi N., Fazzinga B., Gottlob G., Lukasiewicz T. Inductive reasoning and Semantic Web search. Proc. SAC-2010, ACM Press, 2010. P. 1446–1447.
56. Mangol C. A survey and classification of semantic search approaches. International Journal of Metadata, Semantics and Ontologies, N 2(1), 2007. P. 23–34.
57. Rogushina J.V. Semantic retrieval as a comonent of the Semantic Web knowledge management. Proc.of OSTIS-2012, Minsk . P. 239–244. [in Russian].
58. Rogushina J. Use of the Ontological Model for Personification of the Semantic Search // International Journal of Mathematical Sciences and Computing(IJMSC), Vol. 2, N 1, 2016. http://www.mecs-press.org/ ijmsc/ijmsc-v2-n1/IJMSC-V2-N1-1.pdf.
59. Rogushina J.V. Design of ontological model of user information need in semantic retrieval. Ontology of designing. P. 61–82. http://agora.guru.ru/scientific_journal/files/Ontolo-gy_Of_Designing_2_2014_shot.pdf. [in Russian].

Одержано 07.12.2016

Про автора:

Рогушина Юлія Віталіївна,
кандидат фізико-математичних наук,
старший науковий співробітник.
Кількість наукових публікації в українських виданнях – 120.
Кількість наукових публікацій в зарубіжних виданнях – 28.
Індекс Хірша – 10,
http://orcid.org/0000-0001-7958-2557.

Місце роботи автора:

Інститут програмних систем
НАН України,
03181, Київ-187,
проспект Академіка Глушкова, 40.
Тел.: 066 550 1999.
E-mail: ladamandraka2010@gmail.com

УДК 004.62

И.С. Чистякова

ИНТЕГРАЦИЯ АКСИОМАТИКИ ДЕСКРИПТИВНЫХ ЛОГИК С РЕЛЯЦИОННОЙ МОДЕЛЬЮ ДАННЫХ

Работа является логическим продолжением ранее опубликованных исследований, посвященных описанию отображений между дескриптивной логикой и реляционной моделью данных. На основе бинарной реляционной структуры данных, созданной ранее, осуществляется отображение аксиоматики дескриптивной логики ALC в реляционную модель данных (RDM). В работе используются полученные ранее результаты исследований, а именно структура данных RM^2 и отображения базовых концептов логики ALC в RDM.

Ключевые слова: семантическая интеграция данных, семантический веб, отображение данных, реляционная модель данных, онтология, ALC, дескриптивная логика, аксиомы дескриптивных логик.

Введение

Интеграция данных является одним из наиболее важных научных направлений в современном мире. Стремительное развитие информационных технологий привело к накоплению огромных объемов данных в различных источниках, разнородных, распределенных, представляющих информацию различными способами, содержащих взаимосвязанные и взаимно противоречивые данные.

В публикации [1] приведен подробный обзор комплексной проблемы интеграции данных в семантическом Web. В процессе анализа было выделено три её составляющих:

– выработка схем интеграции данных;

– выработка отображений между моделями;

– выработка способов манипулирования.

В данной работе мы продолжаем рассматривать вторую составляющую комплексной проблемы, а именно – отображения между дескриптивной логикой (DL) и реляционной моделью данных (RDM).

Исходя из ряда особенностей, изложенных в обзоре [1], в основу нашего подхода к созданию отображений между DL ALC и RDM легли следующие компоненты:

- центральная схема интеграции данных;
- GAV/LAV представление;
- формально-логический уровень абстракции.

Использование центральной схемы интеграции означает, что в системе присутствует одна глобальная «точка контроля», в основе которой лежит некоторая модель данных. В работе [2] ее называют глобальной схемой, а все остальные – локальными схемами, или схемами источников. Мы придерживаемся той же терминологии. В качестве глобальной модели данных мы рассматриваем дескриптивную логику ALC. Обоснование такого выбора приведено в работе [3]. Критическим моментом централизованной схемы остается разработка отображений между моделями, а именно схемами источников и глобальной схемой.

В данной публикации рассматривается только GAV представление, поскольку созданные нами механизмы отображений [3–5] разрешают взаимодействие от глобальной схемы к источнику. Создание отображений LAV представлений является предметом дальнейших исследований.

Решение проблемы происходит на формально-логическом уровне абстракции. Это означает, что для создания отображений мы не опускаемся на уровень прикладных реализаций (конкретных онтологий и реляционных баз данных), а решаем проблему на уровне моделей, лежащих в основе большинства современных информационных систем. Раздел 1 посвящен описанию аксиоматики дескриптивной логики ALC в контексте применяемого

механизма создания отображений. В разделе 2 рассматривается определение аксиоматики ALC. Раздел 3 описывает представление аксиоматики ALC в модели RM^2. В разделе 4 можно ознакомиться с основными выводами.

DL ALC и её аксиоматика

Согласно определению [6].

Дескриптивная логика – формальный язык для представления знаний и рассуждений в его терминах.

Рассмотрим более широкое определение.

Дескриптивная логика – это семейство формализмов для представления знаний прикладного домена, с помощью первоначального определения понятий домена (*терминологии*) и последующим использованием этих концептов для специфицирования свойств объектов и индивидов, возникающих в домене.

Мы даем упрощенное и более наглядное определение.

Дескриптивная логика – это семейство языков представления знаний, позволяющих описывать понятия предметной области в формализованном виде, обладающие развитыми выразительными возможностями и являющиеся фрагментом логики предикатов.

Для того, чтобы задать какую-либо DL, необходимо задать её *синтаксис* и *семантику*.

Синтаксис описывает, какие выражения (концепты, роли, аксиомы и т. п.) считаются правильно построенными в данной логике.

Семантика указывает, как интерпретировать эти выражения, т. е. придает им формальный смысл.

Существует множество различных дескриптивных логик, которые обладают своими специфичными синтаксисом и семантикой, применяющиеся в различных знание-ориентированных системах, выполняющих различное целевое назначение. При создании бинарной реляционной модели данных, нами была выбрана DL ALC. В работе [3] приведено обоснование данного выбора, поэтому мы не будем рассматривать это подробно.

Приведем **синтаксис** для логики ALC:

$$\top \mid \bot \mid A \mid \neg C \mid C \sqcap D \mid C \sqcup D \mid \exists R.C \mid \forall R.C,$$

где A – атомарный концепт, C и D – произвольные концепты, R – атомарная роль, $\top$ – концепт истина (Thing), $\bot$ – концепт ложь (Nothing), $\neg C$ – концепт-дополнение концепта C, $C \sqcap D$ и $C \sqcup D$ – концепты пересечение и объединение, $\exists R.C$ и $\forall R.C$ – концепты. Другие выражения не являются концептами.

Семантика логики ALC определяется через понятие интерпретации [7].

Интерпретация есть пара

$$I = (\Delta, \cdot^I),$$

состоящая из непустого множества Δ, называемого *областью* данной интерпретации и интерпретирующей функции $\cdot^I$, которая сопоставляет:

– каждому атомарному концепту $A \in CN$ – произвольное подмножество $A^I \in \Delta$;

– каждой атомарной роли $R \in RN$ – произвольное подмножество $RI \in \in \Delta \times \Delta$.

Интерпретирующая функция распространяется на множество всех концептов логики ALC однозначным образом – индукцией по построению концепта:

$$\top^I = \Delta;\ \bot^I = \varnothing;\ (\neg C)^I = \Delta \setminus C^I;$$

$$(C \sqcap D)^I = (C \cap D)^I;\ (C \sqcup D)^I = (C \cup D)^I;$$

$(\exists R.C)^I = \{e \in \Delta \mid$ существует $d \in \Delta$ такой, что $\langle e,d \rangle \in R^I$ и $d \in C^I\}$;

$(\forall R.C)^I = \{e \in \Delta \mid$ для всех $d \in \Delta$ таких, что $\langle e,d \rangle \in R^I$ выполнено $d \in C^I\}$.

Основу нашего механизма составляет следующий тезис: DL можно рас-

сматривать как модель данных. Обоснование данного утверждения приведено в [3], но необходимо напомнить некоторые существенные аспекты, которые пригодятся нам в текущей публикации.

Следующее определение гласит:

Модель данных – это интегрированный набор, а именно:

- структура данных, представляющая собой набор понятий для описания и обработки данных;
- операция, предоставляющая возможность манипулирования данными;
- ограничения целостности, накладываемых на данные.

Опираясь на данное определение, мы утверждаем, что любую DL можно рассматривать как модель данных, в соответствии со следующими положениями:

1) DL обладают синтаксисом и семантикой. Именно в семантике проявляется её структурная часть (возникают понятия множеств, бинарных отношений и т. д.);

2) DL обладает манипулятивной частью, которую представляют такие её компоненты, как конструкторы концептов и ролей;

3) DL обладает целостной частью, которую представляют такие её компоненты, как *аксиомы*.

Таким образом, в качестве центральной модели в интеграционной схеме была выбрана DL. В качестве локальных источников могут выступать любые другие модели (и онтологические в том числе). Мы остановили свой выбор на реляционной модели данных.

Описывая отображения DL в RDM [3–5], мы останавливались на первых двух пунктах, – это структура и манипулятивная часть. В текущей работе мы будем рассматривать последний компонент – аксиоматику DL.

Концепты DL интересны как инструмент записи знаний о предметной области, которые подразделяются на *интенсиональные* и *экстенсиональные*.

Интенсиональные – общие знания о понятиях и их взаимосвязях.

Экстенсиональные – знания об индивидуальных объектах, их свойствах и связях с другими объектами.

В соответствии с этим делением, знания, записываемые с помощью языка DL, подразделяются на:

– набор **терминологических аксиом** или TBox (T);

– набор **утверждений (фактов)** об индивидах или ABox (A);

– набор **аксиом для ролей** RBox (R), который является специальным расширением синтаксиса логики ALC, о котором подробнее будем говорить далее.

Терминологической аксиомой называется выражение вида $C \sqsubseteq D$ или $C \equiv D$, где C и D – произвольные концепты. Терминологией (или TBox) называется произвольный конечный набор аксиом данного вида.

Таким образом, в *синтаксис аксиоматики* ALC вводятся две аксиомы: $\sqsubseteq$ и $\equiv$ (вложенность и эквивалентность концептов).

Аксиоматика обладает своей *семантикой*. Аксиома $C \sqsubseteq D$ истинна в интерпретации *I*, если $C^I \subseteq D^I$, при этом I называют моделью данной аксиомы. Аналогично, $C \equiv D$ истинна в интерпретации *I*, если $C^I = D^I$.

Интерпретацию *I* называют **моделью** терминологии *T*, если *I* является моделью всех аксиом из T. Терминология T называется выполнимой (или совместной), если она имеет модель.

Концепт имеет модель, если существует такая интерпретация, при которой концепт C имеет модель, если существует такая C итое, при которой оно не равно пустому множеству.

Концепт C **выполним** в терминологии *T*, если существует модель I терми-

нологии *T*, такая что $C^I \neq \varnothing$. *Концепты* C и D эквивалентны в *T*, если в любой модели *I* терминологии T имеем $C^I = D^I$. Концепт **C вложен в D** в терминологии *T*, если в любой модели *I* терминологии *T* имеем $C^I \subseteq D^I$. Концепты C и D называются **непересекающимися** в *T*, если в любой модели *I* терминологии *T* имеем $C^I \cap D^I = \varnothing$.

На этом основании, мы делаем следующее утверждение: любой концепт в терминологии T может быть **эквивалентным другому концепту, быть в него вложенным или не пересекаться с ним.**

Как было сказано ранее, мы решаем проблему создания отображений на формально-логическом уровне абстракции, то есть – на уровне моделей, которые лежат в основе большинства современных информационных систем. В работах [3–5] мы говорили об отображениях семантики DL ALC, а также ее расширений в РМД, где описывали, что собой представляет модель ALC. Задача настоящей публикации определить, что собою представляют отображения для аксиом эквивалентности, вложенности и непересекаемости концептов.

Несмотря на то, что аксиома непересекаемости концептов выражается через аксиомы эквивалентности и вложенности, мы рассматриваем отображение для аксиомы непересекаемости, чтобы бинарная реляционная модель данных имела большую выразительность.

Системой фактов (или ABox) называется конечное множество *A* утверждений вида *a:C* и *aRb*, где $a,b \in \mathrm{IN}$ есть индивиды, C – произвольный концепт, R – роль.

Говорим, что факт *a:C* или *aRb* верен в интерпретации *I*, если $a^I \in C^I$ или $< a^I, b^I > \in R^I$ соответственно, при этом *I* – модель этого факта.

Для системы фактов ABox иногда вводится «соглашение об уникальности имен», которое означает, что разным именам индивидов интерпретация должна сопоставлять различные элементы из области интерпретации. Если соглашение не выдвигается, то есть допускается что один объект может иметь множество имен, то тогда вводится **аксиома равенства индивидов**, с помощью которой мы указываем, что некоторая пара имен именует один индивид. Она обозначается «=» и составляет аксиоматику ABox (отображения самой системы фактов было рассмотрено в работе [3]). Эта аксиома показывает, что у одного индивида может быть множество имен, то есть не выполняется принцип уникальности. Однако по-прежнему одно имя именует только один индивид. Отображение этой аксиомы составляет одну из задач настоящей публикации.

В работах [4–5] мы рассматривали отображения для так называемых расширенных дескриптивных логик. До этого момента мы рассматривали расширения путем добавления в синтаксис дескриптивных логик новых концептов или ролей. Однако, существует еще один способ расширения DL – это введение в логику аксиом для ролей, составляющих RBox.

Аксиомы RBox бывают следующие:

– **иерархия ролей (H)**: допускаются аксиомы вида $R \sqsubseteq S$, где R и S – произвольные роли;

– **транзитивные роли (S)**: допускаются аксиомы вида Tr(R), где R – произвольная роль;

– **эквивалентность роли (≡)** можно считать сокращением для двух аксиом $R \sqsubseteq S$ и $S \sqsubseteq R$.

Если I – интерпретация, то по определению полагаем:

– $I \models R \sqsubseteq S \Leftrightarrow R^I \subseteq S^I$;

– $I \models \mathrm{Tr}(R) \Leftrightarrow$ отношение R^I является транзитивным;

– $I \models R \equiv S \Leftrightarrow R^I = S^I$.

Когда дескриптивная логика расширяется аксиомами для ролей, то для

ее именования действуют те же принципы, что и для логик, расширенных путем добавления новых концептов или ролей. А именно, если в логике присутствует аксиома иерархии ролей, то буква H добавляется к имени логики. Но если в логике присутствуют транзитивные роли, то буква S пишется вместо букв ALC. Так, например, логика ALC с обратными ролями (I) и иерархией ролей (H) обозначается как ALCHI, а если расширить ее транзитивными ролями (S), то логика будет называться SHI.

В задачу текущей публикации входит нахождение отображений аксиом для ролей.

Отображение аксиоматики ALC в RM^2

Для того, чтобы начать описание отображений аксиоматики ALC, следует напомнить, что основополагающей частью данного механизма является центральная схема.

Мы уже приводили ее в работе [3] вместе с описанием основополагающих сущностей, однако, в контексте решаемой задачи, схема претерпела существенные изменения. Она показана на рисунке. Дадим некоторые пояснения.

В структуре присутствуют только такие сущности, которые представляют унарные и бинарные отношения, так как в DL ALC n-арные отношения отсутствуют, а также их экземпляры. Следует отметить, что DL с n-арными отношениями существуют, однако они представляют собой специальные расширения ALC, поэтому на данном этапе мы решили отказаться от них, сосредоточившись только на самой ALC.

В отличие от классической RDM, в нашей структуре представлена одновременно и модельная, и метамодельная часть. К метамодельной части мы относим сущности Concept и Role, которые представляют собой перечни имен концептов и ролей соответственно. Следует отметить, что в DL не может существовать отдельно роль, как самостоятельная единица. В нашей структуре роль может существовать отдельно, будучи представлена именем самой роли, именем концепта-домена и именем концепта-диапазона.

К модельной части относятся сущности ConceptIndividual и RoleIndividual, которые представляют собой индивиды концептов и индивиды ролей. В явном виде таких элементов, как индивиды ролей нет в DL, однако мы вводим данную сущность для связывания индивидов концептов с именами ролей.

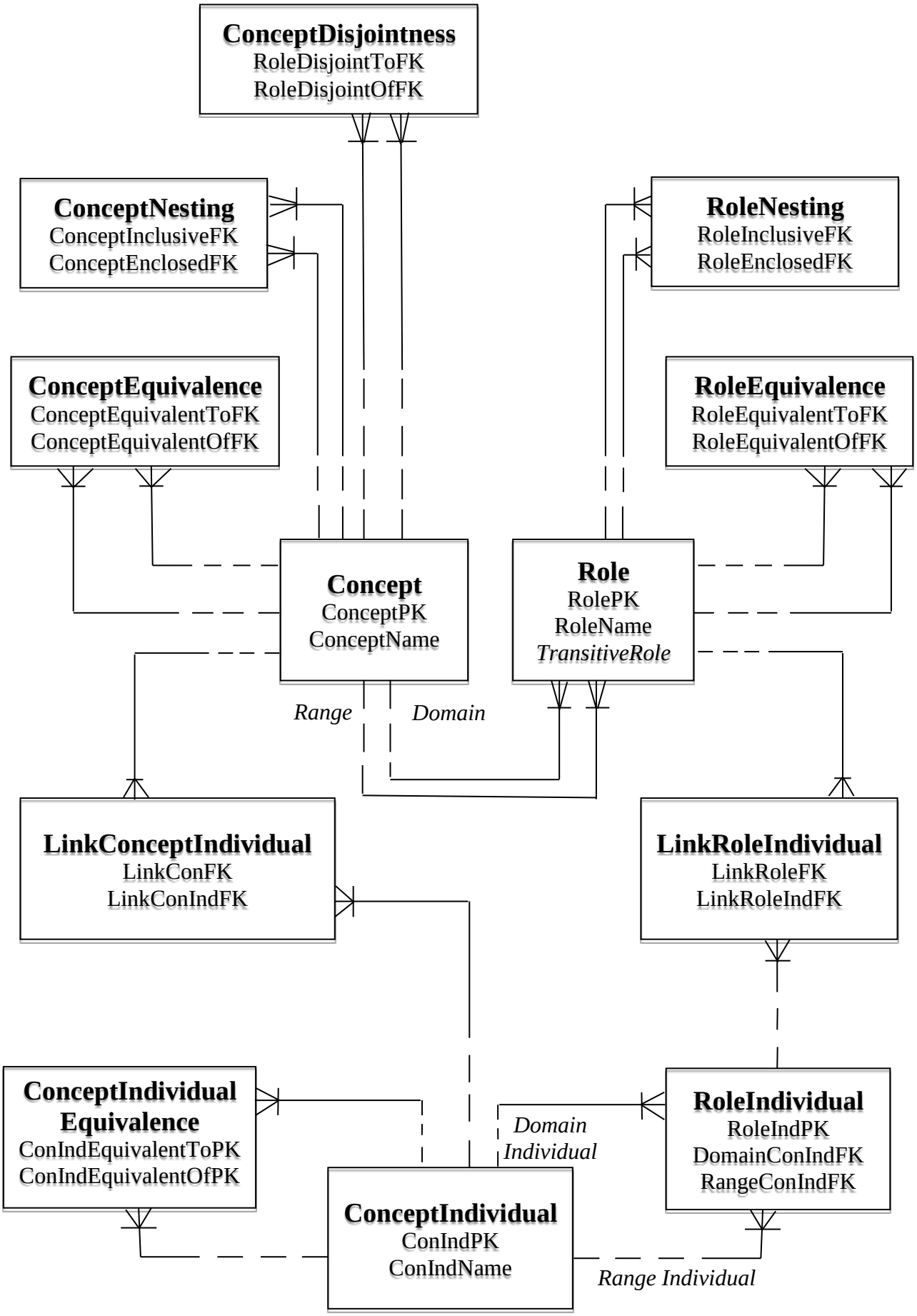
Поскольку один концепт может иметь множество индивидов, а один и тот же индивид может принадлежать нескольким концептам, то для разрешения связи многие-ко-многим мы вводим дополнительную сущность LinkConceptIndividual. Аналогично, сущность LinkRoleIndividual разрешает связь многие-ко-многим для роли и индивидов, которые она связывает. Каждая роль имеет много экземпляров, связывающих два индивида. С другой стороны, каждая конкретная пара индивидов может быть связана несколькими ролями.

В отличие от схемы, которую мы приводили в предыдущих работах, на текущей модели присутствует несколько полноценных сущностей, которые не были представлены в прошлый раз, а именно:

- ConceptEquivalence;
- ConceptNesting;
- ConceptDisjointness;
- ConceptIndividualEquivalence;
- RoleEquivalence;
- RoleNesting.

Эти сущности – есть не что иное, как отображение аксиоматики ALC, о чем мы упоминали в работе [4], в процессе описания схемы. Рассмотрим по отдельности каждую из них.

ConceptEquivalence – сущность, в которой содержатся кортежи, которые связывают два, эквивалентных между собой, концепта. Определяется парой внешних ключей:


ConceptDisjointness
RoleDisjointToFK
RoleDisjointOfFK
ConceptNesting
ConceptInclusiveFK
ConceptEnclosedFK
RoleNesting
RoleInclusiveFK
RoleEnclosedFK
ConceptEquivalence
ConceptEquivalentToFK
ConceptEquivalentOfFK
RoleEquivalence
RoleEquivalentToFK
RoleEquivalentOfFK
Concept
ConceptPK
ConceptName
Role
RolePK
RoleName
TransitiveRole
Range
Domain
LinkConceptIndividual
LinkConFK
LinkConIndFK
LinkRoleIndividual
LinkRoleFK
LinkRoleIndFK
ConceptIndividual Equivalence
ConIndEquivalentToPK
ConIndEquivalentOfPK
Domain Individual
RoleIndividual
RoleIndPK
DomainConIndFK
RangeConIndFK
ConceptIndividual
ConIndPK
ConIndName
Range Individual

Рисунок

1) ConceptEquivalentToFK – внешний ключ на имя концепта, которому ставится в эквивалентность другой концепт;

2) ConceptEquivalentOfFK – внешний ключ на имя концепта, который ставится в качестве эквивалента первому концепту.

ConceptNesting – сущность, в которой содержатся кортежи, которые связывают два иерархических (⊑) концепта. Определяется парой внешних ключей:

1) ConceptInclusiveFK – внешний ключ на концепт-родитель;

2) ConceptEnclosedFK –внешний ключ на концепт-потомок.

ConceptDisjointness – сущность, в которой содержатся кортежи, которые связывают два, непересекающихся между собой, концепта. Определяется парой внешних ключей:

1) ConceptDisjointToFK – внешний ключ на концепт, который не пересекается с другим концептом;

2) ConceptDisjointOfFK – внешний ключ на концепт, с которым не пересекается первый концепт.

ConceptIndividualEquivalence – сущность, в которой содержатся кортежи, связывающие, равные между собой, индивиды, подпадающие под соглашение о не уникальности имен. Определяется парой внешних ключей:

1) ConIndEquivalentToFK – внешний ключ на индивид, который равен другому индивиду;

2) ConIndEquivalentOfFK – внешний ключ на индивид, который ставится в качестве равного первому индивиду.

Следует напомнить, что для индивидов ролей (RoleIndividual) мы не вводим такую аксиому, потому что на данный момент этот вопрос недостаточно исследован, и у нас нет никаких оснований утверждать, что одна и та же связь между одними и теми же индивидами может иметь несколько имен. Поэтому, в нашей структуре правильно лишь утверждение о том, что экземпляр роли существует и представляет собой пару индивидов концептов, связанную определенной ролью.

RoleEquivalence – сущность, в которой содержатся кортежи, которые связывают две, эквивалентных между собой, роли. Определяется парой внешних ключей:

1) RoleEquivalentToFK – внешний ключ на имя роли, которой ставится в эквивалентность другая роль;

2) RoleEquivalentOfFK – внешний ключ на имя роли, которая ставится в качестве эквивалента первой роли.

RoleNesting – сущность, в которой содержатся кортежи, которые связывают две иерархичных (⊑) роли. Определяется парой внешних ключей:

1) RoleInclusiveFK – внешний ключ на роль-родитель;

2) RoleEnclosedFK – внешний ключ на роль-потомок.

TransitiveRole – атрибут отношения Role, который указывает, является ли данная роль транзитивной или нет. На рисунке выделен курсивом, чтобы наглядно обозначить, что данный атрибут является отображением аксиомы транзитивности, и не присутствовал в модели в предыдущих публикациях.

Выводы

В результате исследований, изложенных в данной работе, получен определенный результат в области интеграции семейств дескриптивных логик, в основе которых лежит DL ALC, в реляционную модель данных. Он охватывает создание бинарной реляционной структуры данных, описание механизмов отображения для базовых операций DL ALC, а также основных её расширений.

Текущая работа была посвящена описанию отображений аксиоматики дескриптивных логик. Показано, что ряд аксиом, вводимых в логику, специальным образом расширяют логику ALC, тем самым образуя новую дескриптивную логику, более выразительную по отношению к базовой. Однако, такие логики не требуют расширения набора операций традиционной реляционной алгебры.

Среди открытых вопросов остаются следующие:

– как отобразить n-арные расширения в бинарную реляционную структуру данных;

– как отобразить операции реляционной алгебры в конструкторы дескриптивной логики.

Интерпретация операций реляционной алгебры в конструкторы дескриптивной логики является объектом дальнейших исследований.

1. *Чистякова И.С.* Онтолого-ориентированная интеграция данных в семантическом вебе. Проблеми програмування. 2014. – № 2–3. С. 188–196.
2. *Lenzerini M.* Data Integration: A Theoretical Perspective. Proc. of the 21st ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS 2002). N. Y.: ACM Press, 2002. P. 233–246.
3. *Резниченко В.А., Чистякова И.С.* Отображение дескриптивной логики ALC в бинарную реляционную структуру данных. Проблеми програмування. 2015. – № 4. – С. 13–30.
4. *Резниченко В.А., Чистякова И.С.* Интеграция семейства расширенных дескриптивных логик с реляционной моделью данных. Проблеми програмування. 2016. № 2–3. С. 38–47.
5. *Чистякова И.С.* Интеграция логик с операциями над ролями с реляционной моделью данных. Проблеми програмування. 2016. № 4. С. 58–65.
6. *Baader F., Calvanese D., McGuinness D., Nardi D., and Patel-Schneider P. F., editors.* The Description Logic Handbook. Cambridge University Press, 2003.
7. *Evgeny Zolin* (2009) Description logics (lections). [Online] Available from: http://lpcs.math.msu.su/~zolin/dl/ [Accessed: 2009].

References

1. Chystiakova I.S. (2014). Ontology-oriented data integration on the Semantic Web (Онтолого-ориентированная интеграция данных в семантическом вебе). Problems in programming. N 2–3. P. 188–196.
2. Lenzerini M. (2002). Data Integration: A Theoretical Perspective. Proceedings of the 21st ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems. New York, NY, USA. 2002. New York: ACM.
3. Reznichenko V.A. Chystiakova I.S. (2015). Mapping of the Description Logics ALC into the Binary Relational Data Structure (Отображение дескриптивной логики ALC в бинарную реляционную структуру данных). Problems in programming. N 4. P. 13–30.
4. Reznichenko V.A. Chystiakova I.S. (2016). Integration of the family of extended description logics with relational data model (Онтолого-ориентированная интеграция данных в семантическом вебе). Problems in programming. N 2–3. P. 38–47.
5. CHYSTIAKOVA I.S. (2016). Integration of the description logics with extensions into relational data model (Интеграция логик с операциями над ролями с реляционной моделью данных). Problems in programming. N 4. P. 58–65.
6. BAADER F. et al. (2003). The Description Logic Handbook. P. 47–65.
7. EVGENY ZOLIN (2009) Description logics (lections). [Online] Available from: http://lpcs.math.msu.su/~zolin/dl/ [Accessed: 2009].

Получено 08.11.2016

Об авторе:

Чистякова Инна Сергеевна,
младший научный сотрудник.
Количество научных публикаций в украинских изданиях – 10.
orcid.org/0000-0001-7946-3611.

Место работы автора:

Институт программных систем
НАН Украины.
03187, г. Киев,
проспект Академика Глушкова, 40.
E-mail: inna_islyamova@ukr.net.
Тел.: +38(066) 847 7784.

УДК 681.3

А.Ю. Дорошенко, О.Г. Бекетов

МЕТОД ПАРАЛЕЛІЗАЦІЇ ЦИКЛІВ СІТКОВИХ ОБЧИСЛЮВАЛЬНИХ ЗАДАЧ ДЛЯ ГРАФІЧНИХ ПРИСКОРЮВАЧІВ

Розроблено формальне перетворення гнізда обчислювального циклу, що дозволяє здійснити перехід від послідовного алгоритму до паралельного, орієнтованого на виконання на пристрої з SIMD архітектурою, зокрема, на графічному прискорювачі із використанням технології CUDA та на гетерогенних кластерах.

Ключові слова: методи паралелізації, оптимізація циклів, обчислення загального призначення на графічних процесорах.

Вступ

Сучасні обчислювальні комплекси мають гетерогенну архітектуру і включають як багатоядерні процесори, так і графічні прискорювачі. Проте велику кількість існуючих програмних засобів створено для послідовного виконання на одноядерних машинах. Крім того, створення нових програмних засобів здебільшого проходить через етап послідовної програми. Таким чином, задача автоматичної паралелізації має велику актуальність. В більшості обчислювальних задач значну частину апаратних ресурсів витрачають обчислення, що здійснюються всередині циклів, тому використання автоматичної паралелізації на рівні потоків найбільш ефективне саме для них. Так, наприклад, більшість задач математичної фізики розв'язуються чисельними методами з використанням сіткових обчислень. До виникнення циклів призводять різницеві схеми, метод скінченних елементів, загалом, задачі, що вимагають виконання операцій над матрицями даних. Типовий для сіткової задачі обчислювальний цикл має вигляд:

$$
\begin{aligned}
&for \quad i_0 \in I_0 \\
&for \quad i_1 \in I_1 \\
&\quad \ldots \\
&\quad for \quad i_N \in I_N \\
&\quad\quad F(data, i_0, \ldots, i_N); \qquad (1)
\end{aligned}
$$

де $data$ – множина даних, що обробляються, $I_0, I_1, \ldots, I_N$ – множини значень індексів $i_0, i_1, \ldots, i_N$, $N+1$ – глибина вкладеності циклу, $data$ – множина даних, $F : (data, I_0, \ldots, I_N) \mapsto data$ – відображення, що здійснює перетворення даних. Цикли такого роду у випадку відсутності залежності між вхідними та вихідними даними природним чином розкладаються на паралельні потоки обчислень, оскільки на різних ітераціях цикл повторює однотипні операції над різними даними. Par4All [1] – програмний засіб, що дозволяє на рівні коду в автоматичному режимі здійснити перехід від послідовних програм, написаних мовами C та Fortran, до паралельних для різних апаратних платформ, в тому числі графічного прискорювача із використанням CUDA API. Par4All використовує механізм сіток для зниження глибини вкладеності циклів, проте не надає можливості обробляти обсяги даних, що перевищують обсяг пам'яті прискорювача, а також використовувати гетерогенні обчислювальні кластери.

1. Зміна індексації ітераторів циклу

Розглянемо складений цикл глибиною $N+1$, що має вигляд

$$
for \quad i_0 = 0..\#I_0
$$

$$
\begin{aligned}
&for \quad i_1 = 0..\#I_1 \\
&\quad ... \\
&\quad\quad for \quad i_N = 0..\#I_N \\
&\quad\quad\quad F(in, out, i_0, ..., i_N),
\end{aligned} \qquad (2)
$$

де in – множина вхідних даних, out – множина вихідних даних, причому перетворення $F : (in, I_0, ..., I_N) \mapsto out$ таке, що для кожного набору індексів $(i_0, ..., i_N)$ із гіперкуба $I_0 \times ... \times I_N$ виконується умова $in \cap out \subset \varnothing$ (*), а символом октоторпа позначено потужність відповідної множини. Таким чином, для проведення обчислень цикл (2) здійснює $C = \#(I_0 \times ... \times I_N)$ викликів F. Не обмежуючи загальності, цикл вигляду (2) можна вважати рівносильним циклу (1).

Пронумеруємо елементи гіперкуба $I_0 \times ... \times I_N$ наступним чином. Нехай елемент, що відповідає набору індексів $(i_0, i_1, ..., i_N)$, має номер $i_0 + \sum_{k=1}^{N} i_k \prod_{m=0}^{k-1} \#I_m$.

Введемо відображення $f(k)$, $0 \le k \le \#(I_0 \times ... \times I_N)$:

$$
f(k) = \left(D - \sum_{j=0}^{k-1} i_j \prod_{l=j+1}^{N} \#I_l \right) \Bigg/ \prod_{j=k+1}^{N} \#I_j, \quad k \ne 0,
$$

$$
f(0) = \left(D - \sum_{j=0}^{k-1} i_j \prod_{l=j+1}^{N} \#I_l \right) \bmod \prod_{j=k+1}^{N} \#I_j,
$$

$$
D = \prod_{k=0}^{N} I_k .
$$

Відображення $f(k)$ за номером елемента гіперкуба $I_0 \times ... \times I_N$ відновлює відповідний індекс i_k. Розглянемо цикл такого вигляду:

$$
\begin{aligned}
&for \quad k = 0..D \\
&\quad for \quad j = 0..N \\
&\quad\quad i_j' = f(j); \\
&\quad\quad F(in, out, i_0', ..., i_N').
\end{aligned} \qquad (3)
$$

При проходженні зовнішнього циклу в (3) індекси i_k пробігають ті самі значення, що і в (2), тому цикли (2) та (3) є рівносильними.

Зафіксуємо $0 < d < N$ і введемо відображення $g(k, id)$, $0 \le k \le N$, $0 \le id \le \prod_{j=0}^{d-1} \#I_j$:

$$
g(k, e) = \left(id - \sum_{j=0}^{k-1} i_j \prod_{l=j+1}^{d-1} \#I_l \right) \Bigg/ \prod_{j=k+1}^{d-1} \#I_j, \quad k \ne d-1,
$$

$$
g(d-1, e) = \left(id - \sum_{j=0}^{d-2} i_j \prod_{l=j+1}^{d-1} \#I_l \right) \Big/ \#I_{d-1} .
$$

Відображення $g(k, e)$ побудоване таким чином, що при проходженні наступного циклу індекси i_k пробігають ті самі значення, що і в (2):

$$
\begin{aligned}
&for \quad e = 0..\prod_{j=0}^{d-1} \#I_j \\
&\quad for \quad id = 0..\prod_{m=d}^{N} \#I_m \\
&\quad\quad for \quad k = d..N \\
&\quad\quad\quad i_j' = f(k); \\
&\quad\quad for \quad k = 0..d-1 \\
&\quad\quad\quad i_k' = g(k, id); \\
&\quad\quad\quad F(in, out, i_0', ..., i_N'),
\end{aligned} \qquad (4)
$$

тому останній наведений цикл (4) також рівносильний циклу (2). При цьому ітератори циклів, що мають глибину меншу, ніж d, лишаються сталими впродовж однієї ітерації внутрішнього циклу.

2. Підготовка початкових даних

Оскільки на F діє обмеження (*), операції над вхідними даними є незалежними, і можуть бути виконані паралель-

ними потоками. Припустимо, що F таке, що не містить умовних переходів, тоді різні потоки будуть здійснювати однотипні операції над різними даними і для виконання обчислень доцільно використовувати апаратну SIMD платформу. Пронумеруємо потоки символом $id \in [0, D]$. У випадку (3) кожен із потоків виконує окрему гілку зовнішнього циклу. У випадку (4) зовнішній цикл здійснюється окремим керуючим потоком. Таким чином, за допомогою заміни індексів циклу здійснюється перехід до багатопоточного алгоритму, при цьому не змінюючи F.

Крім того, у випадку, коли в початковому алгоритмі програми використовуються багатовимірні масиви даних, враховуючи той факт, що читання із пам'яті відбувається швидше із лінійних ділянок пам'яті, для оптимізації обробки великих обсягів даних доцільно використовувати перетворення серіалізації, що справедливо навіть для SISD пристроїв, не враховуючи витрат, необхідних для серіалізації даних.

3. Загальна схема перетворення алгоритму

Перехід від послідовного до паралельного алгоритму програми здійснюється у кілька етапів. Загальна схема переходу виглядає так.

1. Підготовчий етап. Полягає у зведенні вихідного алгоритму до циклу вигляду (2). При цьому слід позбавитись міжітераційних залежностей. На цьому етапі можуть використовуватись розбиття циклів, їх перестановка, афінні перетворення, попередній розрахунок даних тощо.

2. Визначення обсягу даних, що обробляються та кількості незалежних операцій, що виконуються всередині окремих циклів гнізда. Ці параметри можливо визначити за умови відсутності умовних переходів в обчисленнях.

3. Визначення параметра d згідно апаратних можливостей виконуючого пристрою. Параметр d обирається таким чином, щоб, з одного боку, обсяг даних, що обробляється циклом глибини d, не перевищував обсяг пам'яті виконуючого пристрою, і з іншого, щоб найбільш повно задіяти наявні апаратні потоки пристрою.

4. Заміна ітераторів згідно правил, описаних в розділі 1. При цьому програма залишається послідовною, змінюється лише спосіб індексації даних. Після цього перетворення залишаються лише два цикли – зовнішній (керуючий) та внутрішній.

5. Створення модуля серіалізації вхідних даних, який підготовляє порцію даних, що буде оброблятися поточною ітерацією керуючого циклу.

6. Створення модуля десеріалізації вихідних даних. Модуль приймає буфер вихідних даних, отриманих після виконання обчислень на поточній ітерації, та передає дані для подальшого використання.

7. Перехід до буферної схеми перенесення даних. На цьому етапі залучаються серіалізатор та десеріалізатор, що викликаються до початку та після виконання внутрішнього циклу відповідно. Змінні внутрішнього циклу переадресовуються на відповідні ділянки вхідного та вихідного буферів даних.

8. Створення ядра, тобто частини програми, що буде безпосередньо перенесена на виконуючий пристрій. Ядро є функцією, що приймає буфер вхідних даних та номер ітерації зовнішнього циклу, та повертає буфер вихідних даних. Функціонально ядро має виконувати ті самі операції, що і внутрішній цикл. Ітератор id внутрішнього циклу замінюється номером потоку виконуючого пристрою.

9. Залучення ядра. Внутрішній цикл замінюється на директиву виклику ядра. Також додаються операції перенесення вхідного буферу даних до виконуючого пристрою та повернення вихідного буферу даних назад в центральний обчислювальний пристрій.

10. Винесення роботи із буферними даними та з графічним прискорювачем в окремі потоки керуючого пристрою з метою уникнення простою виконуючого пристрою.

Проміжні етапи 4, 7, а також фінальні 9, 10 потребують тестових випробувань, які проводяться за допомогою засобів перевірки коректності результатів виконання обчислень. Наприклад, це може бути обробка тестового набору даних і порівняння із наперед відомими результатами. Етап 5 може бути здійснений автоматизовано за допомогою техніки переписуваних правил [2], що дозволить суттєво спростити здійснення перетворень. Для здійснення перетворень на етапі 9 доцільно задіяти Інтегрований інструментарій Проектування та Синтезу програм (ІПС) [3], що зокрема був налаштований на роботу із CUDA API [4, 5].

Роль параметра d циклу (4) полягає в тому, що він визначає глибину, починаючи з якої цикли виконуються безпосередньо виконуючим пристроєм. Таким чином, d встановлює кількість викликів та розподіл навантаження обчислювального ядра внутрішнього циклу. Така техніка надає перевагу в тому разі, якщо об'єм даних, що обробляються, перевищує ємність обчислювального пристрою. При значеннях $0 < d < N$ цикл (2) розбивається на $P = \prod_{i=0}^{d-1} N_i$ частин, кожна з яких оброблюється окремо. Відповідно, керуючий пристрій через змінну $0 \le e \le P$ передає виконуючому пристою номер ітерації, та необхідну на даній ітерації порцію вхідних даних. Порції вхідних даних підготовлюються окремим потоком. В свою чергу, обчислюючий пристрій за допомогою відображень f, g із e відновлює індекси елементів поданої порції даних, що відповідають ітерації зовнішнього керуючого циклу. Таким чином, при значенні $d = 0$ всі операції у циклі (2) будуть здійснюватись виконуючим пристроєм за один виклик ядра, і всі оброблювані дані мають бути передані виконуючому пристрою одночасно, і тоді випадок (4) приймає вигляд (3).

Проілюструємо наведений ланцюжок перетворень на прикладі. Розглянемо частковий випадок циклу вигляду (2) глибини 5:

$$
\begin{array}{l}
for \quad l = 0..L \\
\quad for \quad m = 0..M \\
\quad\quad for \quad n = 0..N \\
\quad\quad\quad for \quad k = 0..K \\
\quad\quad\quad\quad for \quad j = 0..J \\
\quad\quad\quad\quad\quad F(i_{lmnkj}, o_{lmnkj});
\end{array}
$$

де L, M, N, K, J – натуральні числа, i_{lmnkj}, o_{lmnkj} – елементи множин вхідних та вихідних даних ***I, O*** відповідно.

Обсяг вхідних даних рівний обсягу вихідних даних для кожного із циклів і становить $L \cdot M \cdot N \cdot K \cdot J \cdot s$, $M \cdot N \cdot K \cdot J \cdot s$, $N \cdot K \cdot J \cdot s$, $K \cdot J \cdot s$, $J \cdot s$ починаючи із зовнішнього циклу відповідно; s позначає розмір елемента даних. Оберемо параметр d рівним двом та виконаємо переіндексацію даних. Тоді останній цикл набуде наступного вигляду:

$$
\begin{array}{l}
for \quad e = 0..L \cdot M \\
\quad for \quad id = 0..N \cdot K \cdot J \\
\quad\quad l = e / T; \\
\quad\quad m = e \vdots T; \\
\quad\quad n = (id - l \cdot M \cdot N \cdot K \cdot J - \\
\quad\quad\quad - m \cdot N \cdot K \cdot J) / (K \cdot J); \\
\quad\quad k = (id - l \cdot M \cdot N \cdot K \cdot J - \\
\quad\quad\quad - m \cdot N \cdot K \cdot J - n \cdot K \cdot J) / J; \\
\quad\quad j = (id - l \cdot M \cdot N \cdot K \cdot J - \\
\quad\quad\quad - m \cdot N \cdot K \cdot J \\
\quad\quad\quad - n \cdot K \cdot J - k \cdot J) \vdots J; \\
\quad\quad F(i_{lmnkj}, o_{lmnkj}).
\end{array}
$$

Відображення-серіалізатор $Serialize\ (I,\ e,\ inbuf)$ будується наступним чином:

$$
\begin{array}{l}
l = e / T; \\
m = e \vdots T; \\
for \quad n = 0..N \\
\quad for \quad k = 0..K
\end{array}
$$

$$for \quad j = 0..J$$
$$index = n \cdot K \cdot J + k \cdot J + j;$$
$$inbuf_{index} = i_{lmnkj};$$

де $inbuf$ – буфер вхідних даних. Дані вміщуються у єдиний буфер, що згодом дасть зручну можливість переносити дані в виконуючий пристрій однією операцією перенесення.

Десеріалізатор

$$Deserialize\ (e,\ outbuf,\ O)$$

будується аналогічно серіалізатору:

$$l = e / T;$$
$$m = e \vdots T;$$
$$for \quad n = 0..N$$
$$for \quad k = 0..K$$
$$for \quad j = 0..J$$
$$index = n \cdot K \cdot J + k \cdot J + j;$$
$$o_{lmnkj} = outbuf_{index};$$

де $outbuf$ – буфер вихідних даних.

Після впровадження серіалізатора та десеріалізатора цикл набуває наступного вигляду:

$$for \quad e = 0..L \cdot M$$
$$Serialize\ (I,\ e,\ inbuf);$$
$$for \quad id = 0..N \cdot K \cdot J$$
$$l = e / T;$$
$$m = e \vdots T;$$
$$n = (id - l \cdot M \cdot N \cdot K \cdot J - m \cdot N \cdot K \cdot J) / (K \cdot J);$$
$$k = (id - l \cdot M \cdot N \cdot K \cdot J - m \cdot N \cdot K \cdot J - n \cdot K \cdot J) / J;$$
$$j = (id - l \cdot M \cdot N \cdot K \cdot J - m \cdot N \cdot K \cdot J - n \cdot K \cdot J - k \cdot J) \vdots J;$$
$$F(inbuf_{nkj}, outbuf_{nkj});$$
$$Deserialize\ (e,\ outbuf,\ O).$$

Ядро $Kernel(e,\ inbuf,\ outbuf)$ виконує ті самі операції, що і внутрішній цикл:

$$id = threadId();$$
$$if \quad id < N \cdot K \cdot J$$
$$l = e / T;$$
$$m = e \vdots T;$$
$$n = (id - l \cdot M \cdot N \cdot K \cdot J - m \cdot N \cdot K \cdot J) / (K \cdot J);$$
$$k = (id - l \cdot M \cdot N \cdot K \cdot J - m \cdot N \cdot K \cdot J - n \cdot K \cdot J) / J;$$
$$j = (id - l \cdot M \cdot N \cdot K \cdot J - m \cdot N \cdot K \cdot J - n \cdot K \cdot J - k \cdot J) \vdots J;$$
$$F(inbuf_{nkj}, outbuf_{nkj}),$$

де процедура $threadId()$ повертає номер потоку. Таким чином, після всіх перетворень вихідний цикл набуває остаточного вигляду:

$$for \quad e = 0..L \cdot M$$
$$Serialize\ (I,\ e,\ inbuf);$$
$$CopyInput\ (inbuf,\ inbuf');$$
$$Kernel(e,\ inbuf',\ outbuf');$$
$$CopyOutput\ (outbuf',\ outbuf);$$
$$Deserialize\ (e,\ outbuf,\ O);$$

де процедури *CopyInput* та *CopyOutput* переносять буфер вхідних та буфер вихідних даних від керуючого до обчислювального пристрою та в зворотному порядку відповідно. Процедуру *Serialize* доцільно винести в окремий потік керуючого пристрою. Для цього створюється окремий додатковий буфер, і робота *Serialize* виконується почергово:

$$Serialize\ (I,\ e,\ inbuf_0);$$
$$for \quad e = 0..L \cdot M$$
$$thread_0:$$

Serialize (*I*, *e*, $inbuf_{(e+1):2}$);

$thread_1$:

CopyInput ($inbuf_{e:2}$, $inbuf'$);

Kernel(*e*, $inbuf'$, $outbuf'$);

CopyOutput ($outbuf'$, $outbuf_{e:2}$);

$thread_3$:

Deserialize (*e*, $outbuf_{(e+1):2}$, *O*).

Крім того, для проведення обчислень можливо залучити гетерогенний кластер, до складу якого входять кілька SIMD пристроїв. В такому випадку окремі ітерації викликів *Kernel*() керуватимуться окремими потоками керуючого пристрою.

4. GPU та технологія CUDA

Як обчислювальний пристрій будемо використовувати графічний прискорювач. В такому випадку керуючий потік виконується на CPU, а решта потоків переносяться на GPU та виконуються в межах ядра. Для роботи з графічним прискорювачем будемо використовувати програмно-апаратну технологію CUDA [6], що дозволяє використовувати GPU як обчислювальний пристрій. CUDA відрізняється від інших GPGPU технологій тим, що надає безпосередній доступ до пристрою через програмний інтерфейс, що дозволяє гнучко керувати окремими потоками та всіма рівнями пам'яті графічного прискорювача. Крім того, саме для архітектури CUDA створюються спеціалізовані графічні прискорювачі для наукових та технічних обчислень загального призначення [7]. CUDA накладає обмеження на розмірність масивів даних, що не має перевищувати трьох. Це обмеження оминається на етапі серіалізації даних. Проблема, що виникає у тому випадку, якщо необхідна для проведення обчислень кількість потоків більша, ніж кількість апаратних потоків графічного прискорювача, автоматично оминається механізмом сіток CUDA. Таким чином, при використанні графічних прискорювачів, що дозволяють використання технології CUDA, основним обмеженням виступає обсяг пам'яті графічного прискорювача.

5. Експеримент

Проведено експеримент із циклом, що був взятий у задачі прогнозування погоди [8, 9]. Гніздо циклу складене із чотирьох вкладених циклів і обробляє чотиривимірний масив даних числових значень типу float. Навантаження масштабується шляхом зміни розмірності задачі. Було здійснено перехід від послідовної реалізації циклу до паралельної із застосуванням описаної методології та порівняно часові показники для різного масштабу навантаження та різних значень параметра *d*.

Випробування проводились із використанням процесора i5-3570 (у 64-бітному режимі) та графічного прискорювача GeForce GTX 650 Ti, що має наступні характеристики:

- 768 stream-процесори (CUDA-ядра), базова частота 928 MHz;
- обсяг глобальної пам'яті 1024 Mb;
- базова частота пам'яті 5400 MHz.

Обсяг пам'яті ОЗП становив 8 Гб.

При випробуваннях досліджено поведінку послідовної та паралельної програм при фіксованому значенні параметра d, та зміні розмірності задачі, що відповідає зміні кількості викликів ядра при сталому навантаженні на нього, та при зміні розмірності задачі, що відповідає зміні навантаження на один виклик ядра.

Графік на рисунку показує залежність часу виконання обчислень від обсягу даних для $d = 1$, при сталому навантаженні на ядро. При цьому відносне прискорення є сталим та становить близько 4,2.

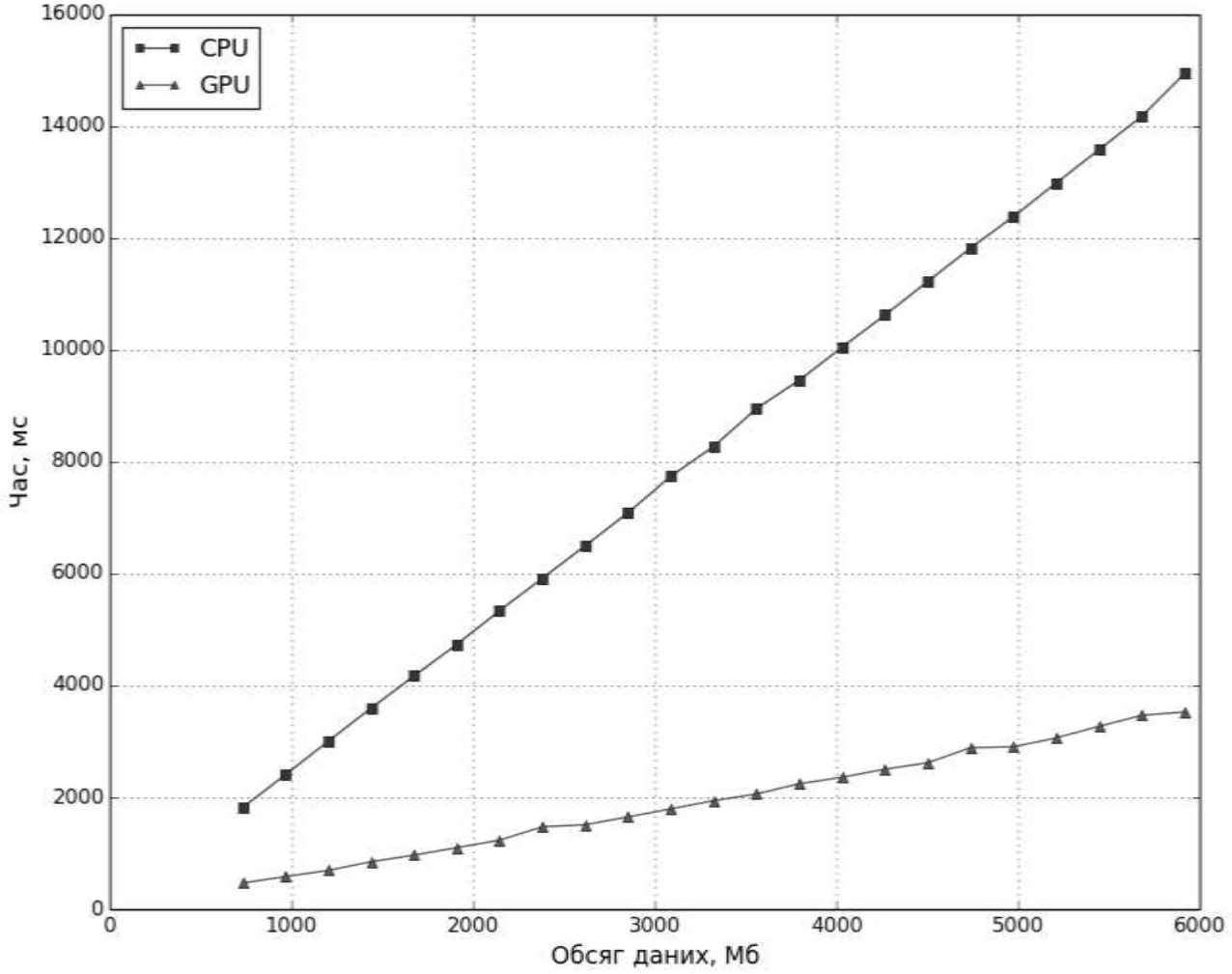


Рисунок. Графік залежності часу виконання послідовної та паралельної програм від обсягу даних, що обробляються

Висновки

Побудовано перетворення алгоритму виконання складеного циклу, що дозволяють здійснити перехід від послідовного до паралельного алгоритму проведення обчислень, що дозволяє залучення гетерогенної обчислювальної платформи. Перевагою застосування методу є те, що він дозволяє здійснювати перетворення над даними, обсяг яких перевищує обсяг пам'яті виконуючого пристрою, а також є автоматизовним. Проведено експеримент, що підтверджує доцільність запропонованого підходу. Таким чином, розроблено основу для подальшої практичної реалізації автоматизованої системи паралелізації вкладених циклів.

1. *Дорошенко А.Е., Шевченко Р.С.* Система символьных вычислений для программирования динамических приложений. Проблеми програмування. 2005. № 4. С. 718–727.
2. *Яценко Е.А.* Интеграция инструментальных средств алгебры алгоритмов и переписывания термов для разработки эффективных параллельных программ. Проблеми програмування. 2013. № 2. С. 62–70.
3. *Дорошенко А.Ю., Бекетов О.Г., Прусов В.А., Тирчак Ю.М., Яценко О.А.* Формалізоване проектування та генерація паралельної програми чисельного прогнозування погоди. Проблеми програмування. 2014. № 2–3. С. 72–81.
4. *Дорошенко А.Ю., Бекетов О.Г., Іванів Р.Б., Іовчев В.О., Мироненко І.О., Яценко О.А.* Автоматизована генерація паралельних програм для графічних прискорювачів на основі схем алгоритмів. Проблеми програмування. 2015. № 1. С. 19–28.

5. CUDA [Електронний ресурс] – Режим доступу до ресурсу: http://www.nvidia.com/object/cuda_home_new.html.
6. TESLA [Електронний ресурс] – Режим доступу до ресурсу: http://www.nvidia.com/object/tesla-servers.html.
7. PIPS: Automatic Parallelizer and Code Transformation Framework [Електронний ресурс] – Режим доступу до ресурсу: http://pips4u.org/.
8. *Прусов В.А., Дорошенко А.Ю.* Моделювання природних і техногенних процесів в атмосфері. Київ: Наукова думка. 2006. 542 с.
9. *Прусов В.А., Сніжко С.І.* Математичне моделювання атмосферних процесів. Київ: Ніка-Центр. 2005. 496 с.

References

1. Doroshenko, A.Yu. & Shevchenko R.S. (2005) Symbolic computation system for dynamical application programming. Problems in programming. (4). P. 718–727. (in Russian)
2. Yatsenko, O.A. (2013) Integration of Software Tools of Algebra of Algorithms and Rewriting Terms for Development of Effective Parallel programs. Problems in programming. (2). P. 62–70. (in Russian).
3. Doroshenko, A.Yu., Beketov, O.G , Prusov, V.A., Tyrchak, Yu.M. & Yatsenko, O.A. (2014) Formalized design and generation of parallel programs for numerical weather forecast. Problems in programming. (2-3). P. 72–81. (in Ukrainian).
4. Doroshenko, A.Yu., Beketov, O.G., Ivaniv, R.B., Iovchev, V.O., Mironenko, I.O. & Yatsenko, O.A. (2015) Automated generation of parallel programs for graphics processing units based on algorithm schemes. Problems in programming. (1). P. 19–28. (in Ukrainian).
5. CUDA [Online] – Available from: http://www.nvidia.com/object/cuda_home_new.html.
6. TESLA [Online] – Available from: http://www.nvidia.com/object/tesla-servers.html.
7. PIPS: Automatic Parallelizer and Code Transformation Framework [Online] – Available from: http://pips4u.org/.
8. Prusov V.A. & Doroshenko A.Yu. (2006) Simulation of natural and anthropogenic processes in the atmosphere. Kyiv: Naukova Dumka. (in Ukrainian).
9. Prusov, V.A. & Snizhko, S.I. (2005) Mathematical modeling of atmospheric processes. Kyiv: NikaTsentr. (in Ukrainian).

Одержано 01.02.2017

***Про авторів*:**

Дорошенко Анатолій Юхимович,
доктор фізико-математичних наук,
професор, завідувач відділу теорії
комп'ютерних обчислень,
професор кафедри автоматики
та управління в технічних системах
НТУ України "КПІ".
Кількість наукових публікацій в
українських виданнях – понад 200.
Кількість наукових публікацій в
зарубіжних виданнях – понад 50.
Індекс Хірша – 5.
http://orcid.org/0000-0002-8435-1451,

Бекетов Олексій Геннадійович,
молодший науковий співробітник.
Кількість наукових публікацій в
українських виданнях – 10.
Кількість наукових публікацій в
зарубіжних виданнях – 1.
http://orcid.org/0000-0003-4715-5053.

Місце роботи авторів:

Інститут програмних систем
НАН України.
проспект Академіка Глушкова, 40.
03187, Київ,
НТУ України «КПІ»
Тел.: (044) 526 3559.
E-mail: dor@isofts.kiev.ua,
beketov.oleksii@gmail.com

УДК 004.82

Г.Ю. Проскудина, К.А. Кудим

О ТЕХНОЛОГИИ ИСПОЛЬЗОВАНИЯ ВНЕШНИХ ДАННЫХ ПРИ СОЗДАНИИ И РЕДАКТИРОВАНИИ ЭНЦИКЛОПЕДИЧЕСКИХ ТЕКСТОВ

В работе обсуждается развивающийся проект Викиданные, веб-сервис запросов и язык запросов. Работа веб-сервиса, языка запросов и форм вывода результатов демонстрируется на многочисленных примерах. Разработана технология использования Викиданных сторонними системами. Учитывая это, рассматривается расширение ExternalData, разработанное для программного обеспечения MediaWiki. В ходе тестовой эксплуатации расширение ExternalData было доработано. Расширение используется для вставки запросов данных к внешним источникам, в нашем случае к базе знаний Викиданные, и их результатов в вики-разметку создаваемых текстов статей. Разработана процедура создания страницы-списка.

Ключевые слова: электронная энциклопедия, база Викиданные, сервис запросов, язык запросов.

Введение

Технология создания и поддержки функционирования *электронной Большой украинской энциклопедии*[1] предполагает оперативное и своевременное ее наполнение из уже имеющихся внешних источников. Важным и необходимым источником данных может стать *Википедия*, *Свободная энциклопедия* – один из крупнейших и популярнейших веб-сайтов мира[2].

Концепция Википедии хорошо известна и довольно проста – это открытая энциклопедия знаний, где любой человек может вносить и редактировать информацию. На сегодняшний день ее англоязычный раздел (английская Википедия) насчитывает более 5 млн. статей, 11 Википедий, в том числе немецкая, французская, русская, итальянская и некоторые другие насчитывают от 1 до 4 миллионов статей. Украинский раздел Википедии по состоянию на 9 января 2017 года содержит 671 977 статей различной тематики; по данному показателю украинская Википедия занимает 16 место из более чем 287 языковых разделов.

В настоящее время незаметно для большинства своих читателей эта популярная онлайновая система претерпевает значительные изменения, учитывая то, что связанный с Википедией проект, *Викиданные*[3], управляя фактологической информацией Википедии, вводит новую многоязычную так называемую "*Википедию данных*" – свободную и открытую базу данных, которая собирает многоязычные структурированные данные Википедии в одном месте и представляет их в общий свободный доступ для копирования, использования и распространения, открывая новые возможности для многих других приложений[4].

Как и Википедия, Викиданные организованы постранично. Каждый экземпляр данных имеет свою страницу, на которой можно редактировать его свойства. Свойства описывают объект и связывают его с другими страницами данных, например, с классом, представителем которого он является, как то: персона, место, событие и многие другие. Например, элемент города *Рим* может иметь свойство *Население* со значением 2.777.979. Можно задавать горные вершины, места и географические координаты зданий. Можно связать человека с его или ее местом рождения, родом занятий или с его номером в базе данных органа управления; связать поли-

[1]Работа выполнена в рамках первого этапа проекта "Разработка проекта компьютерного варианта и технологии создания и поддержки функционирования Большой украинской энциклопедии" программы информатизации НАНУ.

[2]https://ru.wikipedia.org/wiki/Википедия

[3]https://ru.wikipedia.org/wiki/Викиданные

[4]https://www.wikidata.org/wiki/Wikidata:Introduction/ru

тика со своей политической партией; связать населенный пункт со своей более высокой административной единицей; связать страну с ее высшим руководством и ее национальным гимном и т. д. Данная информация может быть показана на любом языке, даже если данные собраны на другом языке. При доступе к этим значениям вики-клиент покажет самые последние данные в актуальном состоянии.

Сегодня почти каждая страница Википедии на разных языках включает в себя содержимое от Викиданных. Все больше и больше редактируемых вручную *инфобоксов* (таблицы с основной, фактической информацией по теме статьи) используют Викиданные в качестве базы данных серверной части, поэтому отображаемая информация будет одинакова во всех изданиях Википедии.

Цель данной работы – использовать внешний надежный источник данных такой, как Википедия, при создании страниц электронной Большой украинской энциклопедии. Для чего предлагается технология включения внешних данных, которую можно рассматривать в качестве одной из составляющих технологии создания статей для энциклопедии. Это позволяет на своем ресурсе запрашивать нужную информацию из Википедии и формировать на основе полученных данных свои страницы или фрагменты страниц.

Разработке такой технологии и посвящена настоящая работа. Здесь подробно обсуждается новый перспективный и развивающийся проект Википедии – Викиданные, веб-сервис запросов к базе данных Викиданные и язык запросов. Работа вебсервиса, языка запросов и форм вывода результатов демонстрируется на многочисленных примерах. Затем речь пойдет о том, как можно использовать Викиданные в сторонних системах, а не только в Википедии. При этом используется разработанное для программного обеспечения MediaWiki расширение ExternalData для вставки запросов данных к *внешним источникам* (в нашем случае к базе данных Викиданные) и их результатов в викиразметку создаваемых текстов статей. В ходе тестовой эксплуатации расширение ExternalData было доработано. Разработана и приводится процедура создания страницы-списка.

В нашей экспериментальной энциклопедии[5] можно посмотреть ряд статей, имеющих на своих страницах запросы к внешней базе данных Викиданные: "Реки и озера Украины", "Национальные парки Украины", "Города Украины", “ВУЗы Украины", "Писатели и поэты, родившиеся в Украине", "Знаменитые музыканты Украины", "Выдающиеся ученые и инженеры Украины".

1. Викиданные

Первоначально задуманная как текстовый ресурс, Википедия собирает растущее количество *структурированных данных*: числа, даты, координаты, разные типы отношений. Эти данные стали ресурсом огромной ценности с потенциальным применением во всех областях науки, техники и культуры.

Такое развитие событий не удивительно, учитывая, что Википедия представляет "всеобщее видение мира, в котором каждый человек может свободно обмениваться суммой всех знаний". Тогда не шла речь о том, что эти знания должны включать в себя данные, которые могут быть найдены, проанализированы и использованы повторно.

Может вызвать удивление тот факт, что Википедия не обеспечивает прямого доступа к большинству этих данных, ни через сервисы запросов, ни через выгружаемый экспорт данных. Фактически использование данных происходит редко и часто ограничено очень специфическими частями информации, такими, например, как гео-теги статей Википедии, используемые в Google Maps. Причина этого поразительного разрыва между видением и реальностью является то, что данные Википедии спрятаны внутри 30 млн. статей на 287 языках, откуда очень трудно извлечь нужную информацию.

Такая ситуация, во-первых, не устраивает тех, кто будет использовать данные, но и, во-вторых, возрастает опас-

[5] http://sew.isofts.kiev.ua

ность для основной цели Википедии – обеспечение современными и точными энциклопедическими знаниями. Одна и та же информация часто появляется в статьях на разных языках и во многих статьях в пределах одного языка. Численность населения Рима, например, можно найти в английской и итальянской статье о Риме, но также и в английской статье о городах Италии. Все эти цифры могут быть отличаться друг от друга.

Цель Викиданные – преодолеть эти проблемы путем создания новых средств Википедии управлять своими данными в глобальном масштабе. Результат этих продолжающихся усилий можно увидеть на сайте wikidata.org.

Викиданные – самый новый проект Викимедиа, это совместно редактируемая, свободная база знаний, которую можно читать и редактировать людьми и машинами. Хороший обзор на эту тему представлен в [1]. На данный момент достигнуты следующие результаты:

– централизация связей между разноязычными изданиями Википедии и другими сайтами проекта Викимедиа. К примеру все статьи Википедии об "энциклопедии" (на любом языке) связаны с одним элементом Викиданных с идентификатором Q5292. Эти так называемые *ссылки на сайты* и другие данные о сущности, известной как "энциклопедия", можно посмотреть на странице https://www.wikidata.org/wiki/Q5292;

– централизация инфобоксов. Все больше и больше измененных вручную инфобоксов, таблиц с основной, фактической информацией по теме статьи, намереваются использовать Викиданные в качестве базы данных серверной части, поэтому отображаемая информация будет одинакова во всех изданиях Википедии;

– обеспечение интерфейса для различных запросов. Содержание Викиданных можно запросить через открытый интерфейс SPARQL на сервисе https://query.wikidata.org. В дальнейшем результаты запроса планируется интегрировать на страницы в Википедии и других проектов, как списки, таблицы, карты и другие формы.

Модель данных Викиданных не реляционная и не на основе RDF, хотя и существуют отображения в RDF, но она отражает стратегию Викиданных на хранение утверждений вместо фактов. Каждое утверждение должно быть получено с помощью ссылок, а противоречивые утверждения намеренно не запрещены. Утверждения могут дополнительно контролироваться уточнителями, такими как домен или дата действия, в конечном счете, поддерживая п-арные отношения между сущностями (элементами) Викиданных. Свойства Викиданных определяются консенсусом сообщества. Например, P571 идентифицирует свойство *зарождение* (*inception*), чтобы заявить дату, когда-то что-то было создано или основано. Названия (labels) и область применения (scope notes) могут быть отредактированы независимо от утверждений с поддержкой синонимов и омонимов.

1.1. Руководящие принципы базы данных Викиданные. Приведем перечень конструктивных решений, характеризующий подход, принятый в базе данных Викиданные.

Открытое редактирование. Также как и Википедия, Викиданные позволяют каждому пользователю сайта расширять и редактировать сохраненную информацию, даже без создания учетной записи. Интерфейс на основе форм делает редактирование легким и удобным.

Контроль сообщества. Под контролем сообщества вкладчиков находятся не только фактологические данные, но и схема данных. Авторы, редактирующие численность населения Рима, в первую очередь на их взгляд вносят самое правильное число.

Множественность. Поскольку многие факты оспариваются или просто не определены, было бы наивно ожидать глобального соглашения об "истинных" данных. Викиданные позволяют противоречивым данным сосуществовать и обеспечивают механизмы для организации такого множества данных.

Вторичные данные. Викиданные собирают факты, опубликованные в пер-

вичных источниках, вместе со ссылками на эти источники. Там нет такого понятия, как "истинное население Рима", но есть – "население Рима, опубликованное в городе Риме в 2011 году".

Многоязычность данных. Большинство данных не привязаны к одному языку: цифры, даты и координаты имеют универсальное значение; заголовки (labels), например, Рим или Население, переведены на многие языки. Викиданные – это многоязычный проект. Есть только один сайт Викиданные, в то время как Википедия имеет независимые издания для каждого языка, т.е. Википедия имеет множество сайтов.

Легкий доступ. Цель Викиданные – предоставлять данные не только Википедии, но и другим внешним приложениям. Данные экспортируются через вебсервисы или API в нескольких форматах, включая XML, JSON, RDF. Данные публикуются в соответствии с юридическими условиями по лицензии CC0[6], позволяющие максимально широкое повторное использование.

Непрерывная эволюция. В лучших традициях Википедии, Викиданные растут вместе с сообществом и задачами. Вместо того, чтобы разработать совершенную систему, которая была бы представлена миру через несколько лет, новые возможности разворачиваются постепенно и как можно раньше. Все эти свойства характеризуют Викиданные как специфический вид curated (специально отобранных) баз данных [2].

1.2. Краткая история проекта Викиданные. Проект Викиданные был запущен в октябре 2012 года. Тогда редакторы могли только создавать элементы (items) и соединять их со статьями Википедии. В январе 2013 года три Википедии, сначала венгерская, затем еврейская (на иврите) и итальянская, подключились к Викиданные. Между тем, сообщество уже создало более трех миллионов элементов. В феврале присоединилась английская Википедия, а в марте 2013 года уже все существующие Википедии были подключены к БД Викиданные.

По состоянию на февраль 2014 года Викиданные получали информацию от более чем 40 тыс. участников. Начиная с мая 2013 года с Викиданные постоянно работали более 3.5 тыс. активных участников – это те вкладчики, которые делают по крайней мере пять изменений в течение месяца. Учитывая это можно сделать вывод, что в настоящее время Викиданные – один из наиболее активных проектов Викимедиа.

В марте 2013 года в качестве языка сценариев Википедии введен язык Lua[7], который может использоваться для автоматического создания и обогащения некоторых частей статьи, например, упомянутых инфобоксов. Скрипты Lua могут получить доступ к Викиданные, позволяя редакторам Википедии извлекать, обрабатывать и отображать эти данные.

В настоящее время продолжаются работы, относящиеся к поддержке произвольных поисковых запросов с возможностью использовать их результаты при автоматическом обновлении различных списков в статьях Википедии.

1.3. Из многих, один. Первоначальной задачей Викиданных было согласовать 287 языковых разделов Википедии [1]. Для Викиданных, чтобы быть действительно многоязычными, объект, представляющий «Рим», должен быть одним и тем же во всех языках. К счастью, Википедия уже имеет механизм тесно связанный с этим вопросом: ссылки на язык, отображаемые слева каждой статьи, соединяют статьи на разных языках. Эти ссылки были созданы из отредактированных пользователем исходных текстов в нижней части каждой статьи, приводят к квадратному числу таких ссылок по теме: каждая из 207 статей о Риме содержит список из 206 ссылок на все другие статьи о Риме – это в общей сложности 42,642 строк текста. В итоге до появления проекта Викиданные в статьях Википедии содержалось больше

[6] https://creativecommons.org/choose/zero/

[7] https://www.lua.org/pil/p1.html

текста для разноязычных ссылок, чем фактического содержания статьи.

Соответственно, лучше хранить и управлять разноязычными ссылками в одном месте, и это была первая задача для Викиданных. Для каждой статьи Википедии, создана отдельная страница на Викиданных, где ссылки на соответствующие статьи Википедии указаны на разных языках. Такие страницы на Викиданных называются *элементами* (items). Первоначально для каждого элемента могло храниться только ограниченное число данных: список языковых ссылок-связей, имена, список псевдонимов, краткое описание. Имена, псевдонимы и описания могут определяться отдельно (на данный момент до 358 языков).

Сообщество Викиданных создало *боты* для того, чтобы переместить языковые ссылки из Википедии в Викиданные, вследствие чего из Википедии можно было удалить более 240 миллионов ссылок. И сегодня, большинство языковых ссылок, отображаемых в статьях Википедии, подаются с Викиданных. В статью еще можно добавить пользовательские ссылки, которые необходимы в тех редких случаях, когда ссылки не являются двунаправленным, например, некоторые статьи связаны с более общими статьями на других языках, при этом Викиданные намеренно соединяет только те страницы, которые охватывают один и тот же предмет. Импортируя языковые ссылки, Викиданные получили огромное множество исходных элементов, которые "обоснованы" реальными страницами Википедии.

1.4. Модель данных. Как и Википедия, Викиданные организованы постранично, и такая организация также совпадает со структурой самих данных [3]. Каждый *предмет* (subject), по которому Викиданные структурируют свои данные называется *сущностью* (entity), и каждая сущность имеет свою страницу. Система пока что различает два типа сущностей: *элементы* (items) и *свойства* (properties). Практически каждая статья Википедии на любом языке имеет соответствующий элемент, представляющий собой предмет (или тему) данной статьи. Каждый элемент имеет страницу, на которой пользователи могут просматривать и вводить данные. Так, например, страницу элемента английского писателя Дугласа Адамса можно увидеть по ссылке: https://www.wikidata.org/wiki /Q42 (рис. 1).

Каждый элемент Викиданных имеет *название* (label), *описание* (description) и, вероятно, один или несколько *псевдонимов* (aliases). *Ссылки на сайты* (sitelinks) связывают каждый элемент с соответствующими статьями на всех клиентах Википедии. *Утверждения* (statements) описывают детальные характеристики для каждого элемента. Каждое утверждение состоит из *свойства* (property) и его *значения* (value).

В нашем примере, название страницы – "Q42", а не "Дуглас Адамс", так как Викиданные – многоязычный сайт. Поэтому элементы не идентифицируются названием на конкретном языке, а нейтральным идентификатором элемента, который автоматически назначается при его создании, и который не может быть изменен в дальнейшем. Идентификаторы элемента всегда начинаются с буквы "Q" с последующим числом. Каждая страница элемента содержит следующие основные части:

– название (например, "Дуглас Адамс");

– краткое описание (например, "английский писатель и юморист");

– список псевдонимов (например, "Дуглас Ноэль Адамс");

– список утверждений (самая обширная часть данных, см. далее);

– список ссылок на сайты (ссылки на страницы Википедии и другие проекты).

Первые три части данных (название, описание, псевдонимы) известны под общим названием термины. Они в основном используются для поиска и отображения элементов. Элемент может иметь название на любом языке, поддерживаемом Викиданными. То, что отображается на страницах, зависит от настройки языка

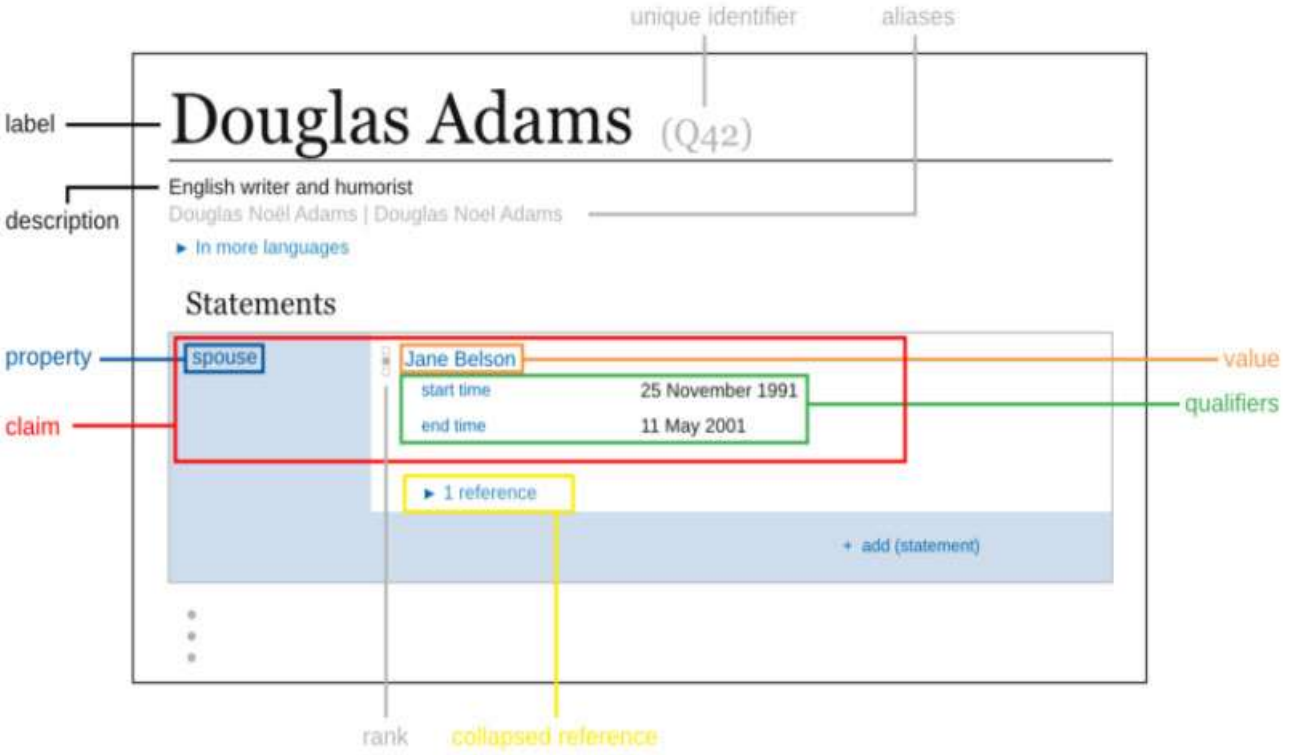


Рис. 1. Пример элемента Викиданные "Дуглас Адамс" Q42

пользователя. Ссылки могут быть предоставлены для любой из 286 языковых версий Википедии, а также для нескольких родственных проектов, таких например, как Викигид и Викисклад. Ссылки на сайты являются функционалом (не более одной ссылки на сайт) и обратным функционалом (не более одного элемента для любой ссылки). В отличие от прежней системы языковых ссылок Википедии, ссылки должны использоваться только для статей, которые точно на эту тему, а не на более широкую или более узкую, или иным образом связанных с темой. Некоторые элементы не имеют каких-либо ссылок, например, элемент "Женский" (Q6581072), который используется в качестве возможного значения для пола лиц.

1.4.1. Свойства и типы данных. На рис. 2 показан простой пример утверждения (statement), которое близко напоминает RDF-тройку, где предмет – “Дуглас Адамс” (Q42), свойство (property) – супруга (spouse) и значение (value) – Jane Belson.

Свойства, как предметы, описаны на страницах и имеют идентификаторы, начинающиеся с "P". Например, свойство супруг(а) на самом деле P26. Свойства тоже имеют названия, псевдонимы и описания, но у них нет ссылок на сайты.

Кроме того, свойства Викиданных также имеют тип, ограничивающий принимаемое значения. Тип данных *дата рождения* – время, *супруг*(*а*) – связан с другим элементом. В табл. (рис. 2, [3]) в левой колонке приведен список всех допустимых типов данных. Общие медиа (Commons Media) представляет собой особый тип данных для ссылок на медиафайлы в хранилище медиа-ресурсов Викисклад, использующееся всеми Википедиями. Типы данных определяют структуру значений, принимаемых свойствами. Свойство может иметь простое значение (как в случае типа *элемент*, *item* или как для типа *строка*) или комплексное значение, которое требует несколько полей, как для времени, сферических координат и количества. Колонка справа (рис. 2) показывает возможные компоненты каждого значения.

Table 1. Wikidata datatypes and their current member fields and field types

Datatype	Member fields
Item	item id (IRI)
String	string
URL	URL (IRI)
Commons Media	article title (string)
Time	point in time (dateTime), timezone offset (int), preferred calendar (IRI), precision (byte), before tolerance (int), after tolerance (int)
Globe coordinates	latitude (decimal), longitude (decimal), globe (IRI), precision (decimal)
Quantity	value (decimal), lower bound (decimal), upper bound (decimal)

Рис. 2. Типы данных в Викиданных, их наборы и типы полей

Для времени, сохраняется дополнительное смещение (в минутах) для часового пояса и ссылка на календарную модель, являющаяся предпочтительной для отображения (например, по Юлианскому календарю, Q1985786). Также можно указать точность, чтобы выразить неопределенные значения, такие как "сентябрь 1547" или "3-е столетие". Детали здесь не существенны. Для наиболее распространенных типов неточностей (точность до дня, месяца, года), чтобы закодировать эту информацию непосредственно в литералах, задавая основной момент времени, используются специальные типы данных схемы XML (xsd:date, xsd:gYearMonth, xsd:gYear).

Для получения сферических координат, в таблице представлено необычное поле – *шар* (globe), которое задает небесное тело, например, координаты относятся к Земле (Q2).

1.4.2. Сложные утверждения и ссылки. Полная модель данных утверждений Викиданных немного сложнее, чем можно предложить из рис. 1. С одной стороны, утверждения могут иметь так называемые *квалификаторы* (или уточнители), предоставляющие дополнительную контекстную информацию для данного утверждения. С другой стороны, каждое утверждение может включать в себя одну или несколько ссылок, в поддержку этого утверждения. Утверждение, где представлены оба аспекта, показано на рис. 3.

Основная пара свойство-значение в данном утверждении – "spouse: Jane Belson" (P26: Q14623681), но здесь существует и контекстная информация.

Уточнители на рис. 3 – "дата начала: 25 ноября 1991" и "дата окончания: 11 мая 2011", утверждают, что Дуглас Адамс был женат на Джейн Белсон с 1991 года до своей смерти в 2011 году. Здесь используются свойства *дата начала* (P580) и *дата окончания* (P582) соответствующих типов времени. Эти пары свойство-значение относятся к основной части утверждения, а не к элементу на странице (Дуглас Адамс).

В Викиданных уточнители используются в нескольких ситуациях. Наиболее распространенным является указание на время действия утверждения, так что случай на рис. 3 – довольно типичный. Тем не менее, Викиданные использует многие другие виды аннотаций, предоставляющих контекстную информацию об утверждении. Например, *автор таксона* (P405, важный контекст для биологических названий таксонов) и таксономия *астероидов* (P1016, контекстуализировать

Рис. 3. Часть сложного утверждения о жене Дугласа Адамса,
как показано в Викиданных

спектральную классификацию астероидов). В некоторых случаях, уточнители-предоставляют дополнительные аргументы отношений, которые имеют более двух участников. Например, свойство (P553) учетная запись веб-сайта определяет веб-сайт (например, Twitter, Q918), но, оно как правило, используется с уточнителем P554, задающего имя учетной записи, используемого элементом на этом сайте. Можно утверждать, что это тернарные отношения, но граница между аннотацией контекста и n-арной связью размыта. Например, американский сериал, *Звёздный путь: Следующее поколение* (Q16290) имеет свойство *в ролях* (P161) *Брент Спайнер* (Q311453) с двумя значениями уточнителя *играет роль персонажа* (P453): *Дейта* (Q22983) и *Lore* (Q2609295). Обратите внимание, что такое же свойство может быть использовано в нескольких уточнителях в одном утверждении.

Мы использовали пары свойство-значение во многих местах: и в качестве основных частей утверждений, и в качестве уточнителей, и в ссылках. В каждом из этих случаев Викиданные также поддерживают два специальных “значения”: ни один и некоторый (none и some). Значение ни один используется, например, в утверждении: "Королева Англии Елизавета I не имела супруга". Что позволило получить простую форму для отрицания и отличить его от случаев, когда информация является просто неполной. Это также позволяет добавлять ссылки на негативные утверждения. Иногда это используется, когда известно, что это свойство имеет значение, но нет возможности предоставить более подробную информацию, как, например, в утверждении: "Папа Линус имел дату рождения, но она нам неизвестна". Оба данных специальных "значения" можно использовать во всех местах, где разрешены обычные значения свойств, поэтому они, как правило, не упоминаются в явном виде.

1.4.3 Порядок и ранг. Все данные в Викиданных упорядочены – псевдонимы, утверждения, пары свойство-значение в качестве ссылки и т. д. Информация о порядке в Викиданные используется только для представления, и не считается значимой для ответа на запросы.

Даже если в ответах на запрос не нужно использовать порядок утверждений, иногда необходимо выделять некоторые из утверждений от остальных.

Например, Викиданные содержат много исторических данных с подходящими уточнителями, например, численность населения городов в разное время. Такие данные имеют множество применений, но простой запрос для населения города не должен возвращать длинный список чисел. Чтобы упростить базовую фильтрацию данных, утверждениям Викиданных можно присвоить один из трех рангов: *нормальный* (используется по умолчанию), *привилегированный* (когда нужно выделить предпочтительные значения) и *вызывает возражение* (когда нужно пометить неправильно или непригодную информацию, но по какой-то практической причине ее хранят в системе).

1.5. Викиданные в цифрах. С момента своего запуска в октябре 2012 года база данных Викиданные значительно выросла. Некоторые статистические факты о ее текущем содержании показаны на рис. 4.

Статистика Викиданных сгенерирована внешним приложением, в данном случае – SQID (https://tools.wmflabs.org/sqid/#/status). Внешние инструменты – программы, которые запускаются не на серверах Викиданных, а на сторонних серверах. Большинство из них полезны для извлечения данных, предоставляемых Викиданными [4–5].

1.6. Примеры использования Викиданных сторонними приложениями. Информация, собранная в Викиданных интересна в своем собственном виде, поэтому для более удобного и эффективного доступа к ней могут быть построены многие внешние приложения. Например, приложения общего просмотра данных, такие как на рис. 5. Здесь страница Иоганн Себастьян Бах автоматически сгенерирована на основе данных, что были извлечены из Викиданных. Или инструменты специального назначения, например, древо жизни, таблицы элементов, а также различные инструменты для отображения.

Приложения могут использовать API Викиданных для просмотра, запроса и даже редактирования данных. Если не достаточно возможностей простых запросов, то в таком случае требуется локальная копия базы Викиданных.

Statistics based on Wikidata dump 31.10.2016

	Items	Properties	Total
Number	24324155	2844	24326999
Statements	120600907	20106	120621013
Labels	125873851	69604	125943455
Descriptions	201000976	30174	201031150
Aliases	12800682	39137	12839819
Site links	53711422	0	53711422

Data Freshness

Statistical data is computed from the data dump about once per week. Basic statisics (class and property names, usage counts for properties, direct instances of classes) are refreshed more frequently, about once per hour. All other data is live.

Dump date	31.10.2016
Property data refresh	07.11.2016, 12:12:06
Class data refresh	07.11.2016, 11:12:16

Рис. 4. Статистика базы данных Викиданные

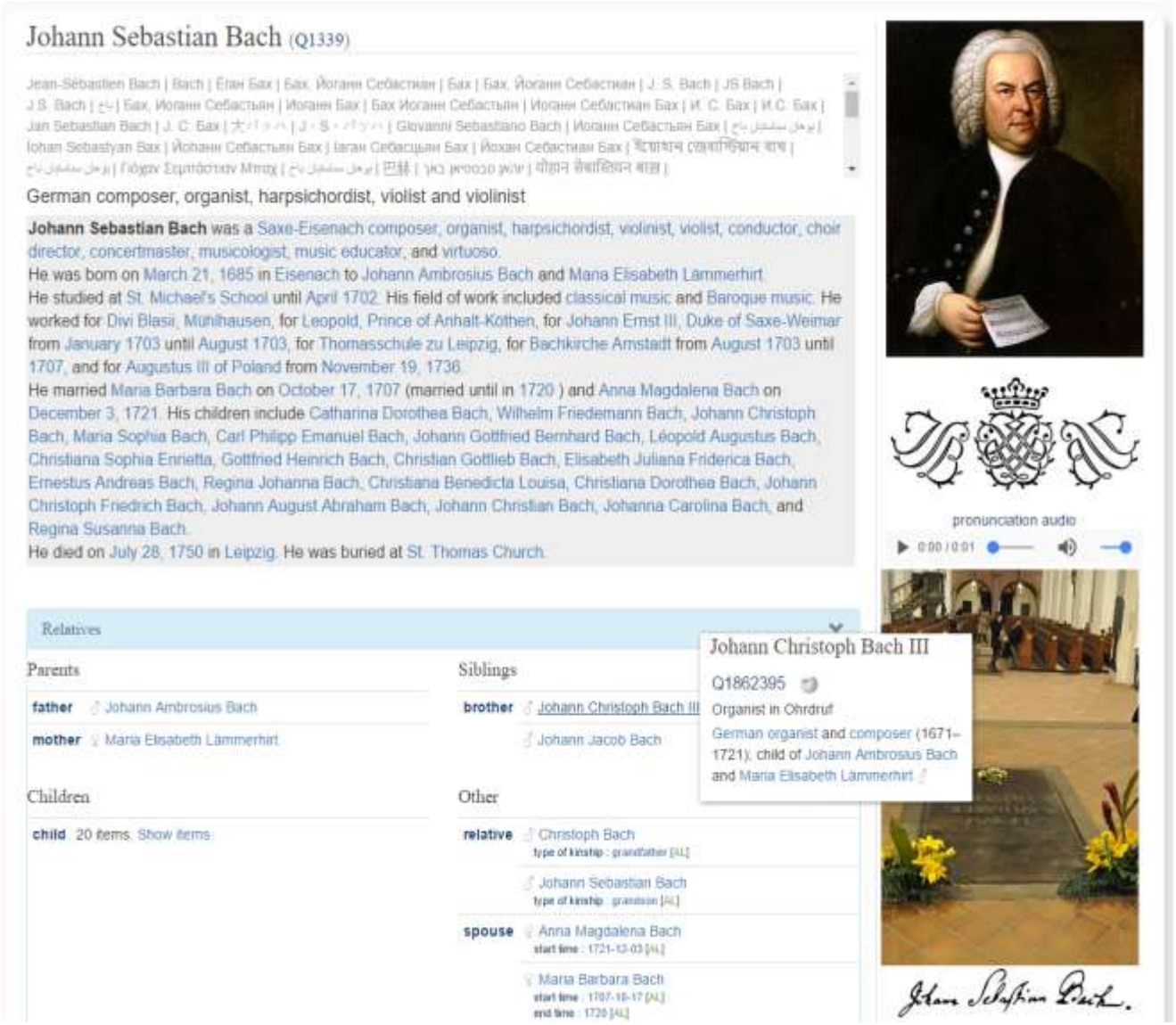


Рис. 5. Викиданные во внешних приложениях: "Резонатор" – программа просмотра данных

2. Сервис запросов к Викиданным

Wikidata Query Service (WDQS) представляет собой пакет программного обеспечения и публичный сервис[8], предназначенный для выполнения SPARQL-запросов, позволяющий запрашивать данные из базы данных Викиданные. Время выполнения каждого запроса ограничено 30 секундами. Это справедливо как для графического интерфейса пользователя (GUI), так и публичной точки доступа SPARQL.

2.1. Набор данных. Сервис запросов к Викиданным работает на множестве данных из wikidata.org, представленных в RDF. Предоставляется возможность скачать еженедельную копию всех данных[9], представленных в Викиданные.

2.2. Семантическая тройка – "Предмет – Предикат – Объект" известна как тройка или как утверждение о данных. Утверждение "Небо имеет голубой цвет", состоит из предмета "небо", предиката "имеет цвет" и объекта "голубой". Тройка также используется в качестве формы основной синтаксической схемы запросов в WDQS. Допускается расширенное использование троек, в том числе с использованием троек в качестве объектов или предметов других троек.

[8] http://query.wikidata.org/

[9] https://dumps.wikimedia.org/wikidatawiki/entities/

2.3. Графический интерфейс пользователя. Домашняя страница GUI позволяет редактировать и передавать SPARQL-запросы механизму выполнения запросов, результаты которых отображаются в виде таблицы HTML. Каждый запрос имеет уникальный URL, который может быть закладкой для последующего использования. Переход к этому URL вносит запрос в окно редактирования, но без его выполнения (для его выполнения нужно нажать кнопку "Выполнить").

Можно также генерировать короткий URL для текущего запроса через сервис укорачивания URL, выбрав опцию *Short URL to result* на ссылке *Link* справа. Также по этой ссылке имеется еще две полезные опции: *SPARQL-endpoint* (точка доступа SPARQL), по которой можно получить результирующий XML-файл текущего запроса и *Embed result* (встроить результат), когда по полученному коду результат текущего запроса можно непосредственно вставлять в вики-разметку вновь создаваемых или редактируемых страниц приложений.

Кнопка "Добавить префиксы" формирует заголовок, содержащий стандартные префиксы для SPARQL-запросов. Полный список полезных префиксов указан в документации о формате RDF[10]. Наиболее распространенные префиксы работают в автоматическом режиме.

GUI также имеет простой механизм более детального анализа сущности, который может быть активирован, нажав на символ  перед сущностью в результирующей таблице (рис. 6). Щелчок на идентификаторе Q-ID сущности приводит к обращению на страницу самой сущности в wikidata.org.

При выполнении запроса WDQS в GUI можно выбрать вид представления его результатов, указав в начале запроса комментарий: #defaultView:viewName. Результаты запросов могут быть представлены в виде таблицы, карты, сетки изображений, временной шкалы, графа, линейной диаграммы, гистограммы, точечной диаграммы.

[10] RDF format documentation

Рис. 6. Просмотр свойств, установленных для сущности с идентификатором Q2906022

2.4. Точка доступа SPARQL (API). SPARQL запросы могут быть переданы непосредственно в точку доступа SPARQL запросом GET к https://query.wikidata.org/sparql ?query=SPARQL (POST и другие методы запросов запрещены). Результат возвращается в виде XML по умолчанию, или как JSON, если установлен либо параметр запроса format=json, либо заголовок Accept: application/sparql-results+json. Формат JSON является стандартным[11]. В настоящее время точкой доступа SPARQL поддерживаются следующие форматы вывода результата запросов: XML, JSON, TSV, CSV, бинарный RDF.

2.5. Автономный сервис. Поскольку представляемый сервис запросов – программа с открытым исходным кодом, его можно запустить на сервере любого пользователя, используя инструкции[12].

2.6. Примеры выполнения SPARQL-запросов в WDQS. В документации[13] приводится множество примеров

[11] SPARQL 1.1 Query Results JSON Format

[12] https://www.mediawiki.org/wiki/Wikidata_query_service/User_Manual#Standalone_service

[13] https://www.wikidata.org/wiki/Wikidata:SPARQL_query_service/queries/examples

SPARQL-запросов со ссылками на их выполнение в представляемом сервисе WDQS.

Мы рассказали, как создавать и отображать запросы к Викиданным в сервисе WDQS. Далее будем встраивать запросы и их результаты в разметку страниц создаваемых энциклопедических статей. Для этого можно воспользоваться одним из расширений программной системы MediaWiki, на основе которой строится наш энциклопедический сайт – ExternalData (https://www.mediawiki.org/wiki/Extension:External_Data).

Расширение ExternalData позволяет использовать и отображать значения, извлеченные из различных источников: внешних URL-адресов, локальных вики-страниц и локальных файлов.

3. Процедура создания страницы-списка

Рассмотрим процедуру создания страницы-списка "Музыканты, родившиеся в Украине" на примере одноименного запроса (http://tinyurl.com/zmz8g27) к Викиданным.

Шаг 1. Формулировка запроса на языке SPARQL (рис. 7).

```
SELECT DISTINCT ?label ?subj ?placeLabel (year(?dateOfBirth) as ?yearOfBirth)
(coalesce(year(?dateOfDeath), '_') as ?yearOfDeath) ?pic  WHERE {
  ?subj wdt:P106 wd:Q639669 .
  ?subj wdt:P19 ?place .
  ?subj wdt:P569 ?dateOfBirth.
  ?subj wdt:P570 ?dateOfDeath.
  ?subj wdt:P18 ?pic.
  ?place wdt:P17 wd:Q212 .
  ?subj rdfs:label ?label filter (lang(?label) = "ru")
 SERVICE wikibase:label {
 bd:serviceParam wikibase:language "ru" .
 }
} ORDER BY ASC(?label)
```

Рис. 7. Запрос "Музыканты, родившиеся в Украине" на языке SPARQL

Шаг 2. Проверка и отладка запроса в сервисе WDQS (рис. 8).

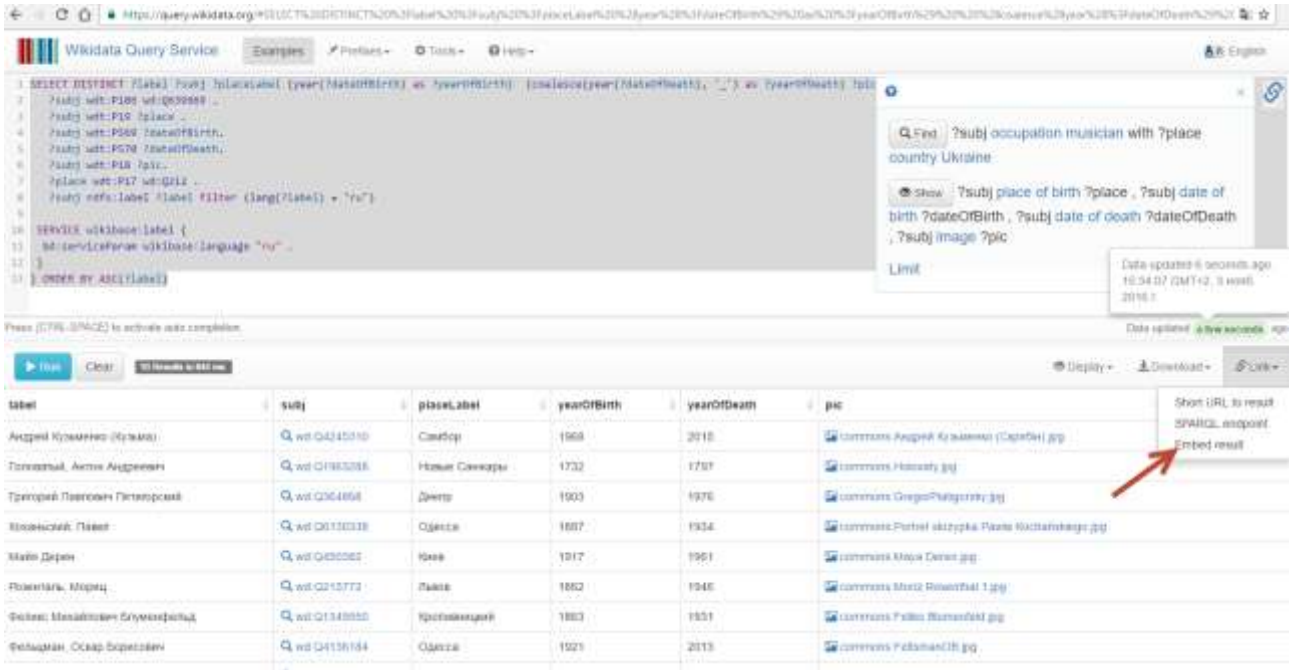


Рис. 8. Выполнение запроса "Музыканты, родившиеся в Украине" в сервисе WDQS

Шаг 3. URL Decoder/Encoder. Далее отлаженный текст запроса нужно закодировать. Существует как минимум два варианта выполнения этой операции. В окне результата WDQS, во всплывающем списке Link выбрать опцию Embed result и в появившемся окне выбираем закодированный код данного запроса. Либо, используя сторонний инструмент, например (http://meyerweb.com/eric/tools/dencoder/).

Шаг 4. Получение результата в виде XML-файла. Выбрав опцию SPARQL-endpoint (точка доступа PARQL) на ссылке Link справа в окне результата запроса сервиса WDQS получаем ответ в отдельном XML-файле. Фрагмент полученного файла показан на рис. 9.

Шаг 5. Задание функции #get_web_data расширения ExternalData. Расписываем составляющие функции #get_web_data для данного примера и вносим в вики-разметку создаваемой страницы "Музыканты, родившиеся в Украине" (рис. 10).

```
<?xml version='1.0' encoding='UTF-8'?>
<sparql xmlns='http://www.w3.org/2005/sparql-results#'>
    <head>
        <variable name='label'/>
        <variable name='subj'/>
        <variable name='placeLabel'/>
        <variable name='yearOfBirth'/>
        <variable name='yearOfDeath'/>
        <variable name='pic'/>
    </head>
    <results>
        <result>
            <binding name='label'>
                <literal xml:lang='ru'>Головатый, Антон Андреевич</literal>
            </binding>
            <binding name='subj'>
                <uri>http://www.wikidata.org/entity/Q1963288</uri>
            </binding>
            <binding name='placeLabel'>
                <literal xml:lang='ru'>Новые Санжары</literal>
            </binding>
            <binding name='yearOfBirth'>
                <literal datatype='http://www.w3.org/2001/XMLSchema#integer'>1732</literal>
            </binding>
            <binding name='yearOfDeath'>
                <literal datatype='http://www.w3.org/2001/XMLSchema#integer'>1797</literal>
            </binding>
            <binding name='pic'>
                <uri>http://commons.wikimedia.org/wiki/Special:FilePath/Holovaty.jpg</uri>
            </binding>
        </result>
```

Рис. 9. Фрагмент результата запроса в виде XML-файла

```
{{#get_web_data:url=https://query.wikidata.org
/sparql?query=SELECT%20DISTINCT%20%3Flabel
%20%3Fsubj%20%3FplaceLabel%20(year(%3Fdat
eOfBirth)%20as%20%3FyearOfBirth)%20%20(coal
esce(year(%3FdateOfDeath)%2C%20%27_%27)%
20as%20%3FyearOfDeath)%20%3Fpic%20%20W
HERE%20%7B%0A%20%20%20%3Fsubj%20wdt
%3AP106%20wd%3AQ639669%20.%0A%20%20
%20%3Fsubj%20wdt%3AP19%20%3Fplace%20.%
0A%20%20%20%3Fsubj%20wdt%3AP569%20%3
FdateOfBirth.%0A%20%20%20%3Fsubj%20wdt%
3AP570%20%3FdateOfDeath.%0A%20%20%20%
3Fsubj%20wdt%3AP18%20%3Fpic.%20%0A%20
%20%20%3Fplace%20wdt%3AP17%20wd%3AQ2
12%20.%0A%20%20%20%3Fsubj%20rdfs%3Alab
el%20%3Flabel%20filter%20(lang(%3Flabel)%20
%3D%20%22ru%22)%0A%20%20%20%20%20%
20%20%20%20%0A%20SERVICE%20wikibase%3
Alabel%20%7B%0A%20%20bd%3AserviceParam
%20wikibase%3Alanguage%20%22ru%22%20.%0
A%20%7D%0A%7D%20ORDER%20BY%20ASC(%
3Flabel)limit%20343&format=xml

|format=xml

|use xpath

|data=name=//binding[@name='label']/literal,item
=//binding[@name='subj']/uri,place=//binding[@n
ame='placeLabel']/literal,yearOfBirth=//binding[@n
ame='yearOfBirth']/literal,yearOfDeath=//binding[
@name='yearOfDeath']/literal,pic=//binding[@nam
e='pic']/uri}}
```

Рис. 10. Пример задания функции #get_web_data в вики-разметке

Шаг 6. Отображение таблицы значений, задание функции #for_external_table расширения ExternalData. Добавляем в вики-разметку создаваемой страницы следующий фрагмент, содержащий описание заглавия таблицы результата и имена внешних переменных (рис. 11).

```
{| class="wikitable"
! Имя
! URI
! Место рождения
! Годы жизни
! Фото
{{#for_external_table:<nowiki/>
{{!}}-
{{!}} {{{name}}}
{{!}} {{{item}}}
{{!}} {{{place}}}
{{!}} {{{yearOfBirth}}}-{{{yearOfDeath}}}
{{!}} <img src={{{pic}}} width=200px>
}}
```

Рис. 11. Пример задания функции #for_external_table в вики-разметке

Сохраняем его и переходим в режим просмотра этой страницы[14]. Все данные таблицы получены из Википедии и сопутствующих проектов. Далее показан внешний вид страницы "Музыканты, родившиеся в Украине", рис. 12.

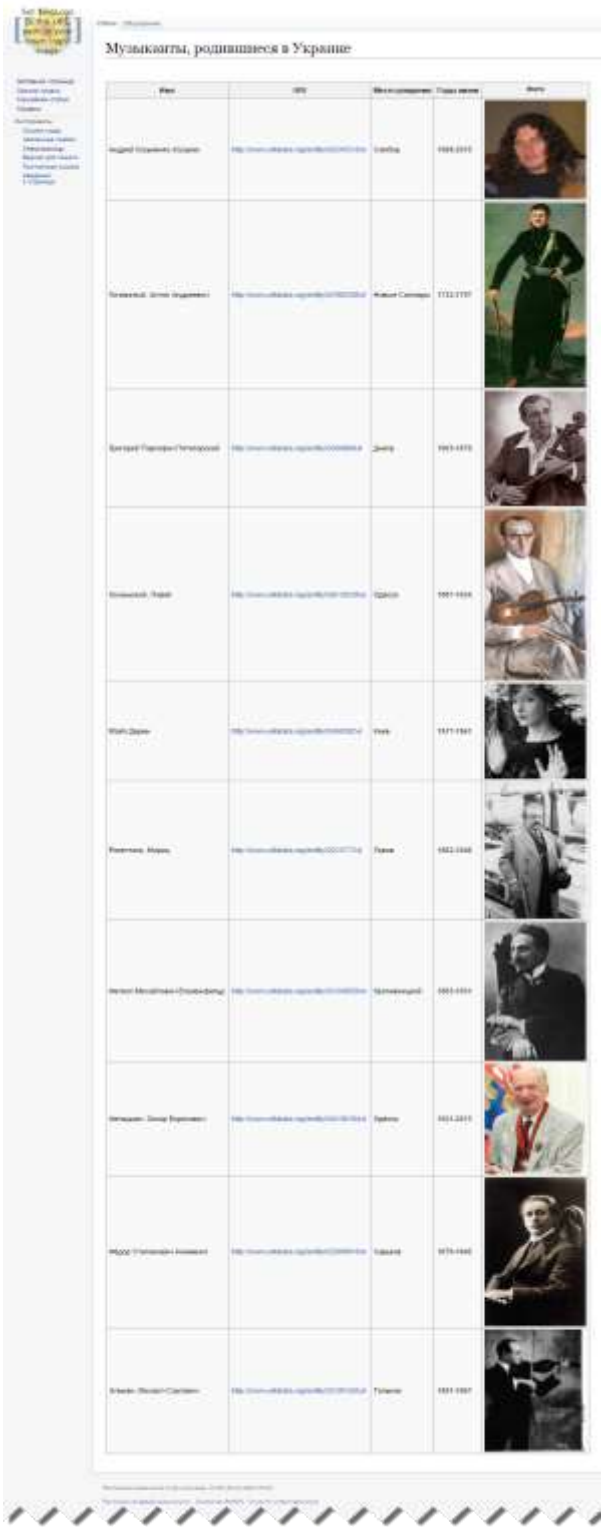


Рис. 12. Просмотр страницы

[14] http://sew.isofts.kiev.ua/index.php/Музыканты,_родившиеся_в_Украине

Аналогичным образом можно создать страницы-списки "Поэты, родившиеся в Украине", "Писатели, родившиеся в Украине" или страницы включающие другие подобные запросы.

Выводы

База Викиданные, её содержание и основное программное обеспечение находятся в стадии постоянного развития, исход которого трудно предвидеть. Учитывая важную роль, которую играет база Викиданные для Википедии, можно быть уверенным в том, что этот проект будет продолжать расти по размеру и качеству. Многие захватывающие возможности использования этих данных еще предстоит исследовать.

1. *Denny Vrandeciˇ, Markus Krötzsch* Wikidata: A Free Collaborative Knowledgebase. In. Proc. CACM-2014 – Communications of the ACM. October 2014. Vol. 57, N 10. P. 78–85. http://korrekt.org/papers/Wikidata-CACM-2014.pdf
2. *Peter Buneman, James Cheney, Wang-Chiew Tan, Stijn Vansummeren*. Curated databases. In Maurizio Lenzerini and Domenico Lembo, editors. In Proc. 27th Symposium on Principles of Database Systems. PODS'09. P. 1–12. http://citeseerx.ist.psu.edu/viewdoc/download;jses sionid=9A9267DB6C3139BA98E3C309E5DFA81F?doi=10.1.1.168.2515&rep=rep1&type=pdf
3. *Fredo Erxleben, Michael Günther, Markus Krötzsch, Julian Mendez and Denny Vrandeciˇ* Introducing Wikidata to the Linked Data Web. In Proc. The Semantic Web – ISWC 2014. Lecture Notes in Computer Science. Vol. 8796. P. 50–65. http://korrekt.org/papers/Wikidata-RDF-export-2014.pdf
4. *Lucie-Aimée Kaffee* Generating Article Placeholders from Wikidata for Wikipedia: Increasing Access to Free and Open Knowledge. HTW Berlin University of Applied Sciences. International Media and Computing Faculty IV. A thesis for the degree of Bachelor of Science. March 4, 2016. 62 p. https://upload.wikimedia.org/wikipedia/commons/9/99/Generating_Article_Placeholders_from_Wikidata_for_Wikipedia_-_Increasing_Access_to_Free_and_Open_Knowledge.pdf
5. *Jakob Voß* Classification of Knowledge Organization Systems with Wikidata. In Proc. 15th European Networked Knowledge Organization Systems Workshop. NKOS 2016. Hannover. September 9, 2016. Vol. 1676. P. 15–22. http://ceur-ws.org/Vol-1676/paper2.pdf

References

1. Denny Vrandeciˇ, Markus Krötzsch Wikidata: A Free Collaborative Knowledgebase. In. Proc. CACM-2014 – Communications of the ACM. October 2014. Vol. 57, N 10. P. 78–85. http://korrekt.org/papers/Wikidata-CACM-2014.pdf
2. Peter Buneman, James Cheney, Wang-Chiew Tan, Stijn Vansummeren. Curated databases. In Maurizio Lenzerini and Domenico Lembo, editors. In Proc. 27th Symposium on Principles of Database Systems. PODS'09. P. 1–12. http://citeseerx.ist.psu.edu/viewdoc/download;jses sionid=9A9267DB6C3139BA98E3C309E5DFA81F?doi=10.1.1.168.2515&rep=rep1&type=pdf
3. Fredo Erxleben, Michael Günther, Markus Krötzsch, Julian Mendez and Denny Vrandeciˇ Introducing Wikidata to the Linked Data Web. In Proc. The Semantic Web – ISWC 2014. Lecture Notes in Computer Science. Vol. 8796. P. 50–65. http://korrekt.org/papers/Wikidata-RDF-export-2014.pdf
4. Lucie-Aimée Kaffee Generating Article Placeholders from Wikidata for Wikipedia: Increasing Access to Free and Open Knowledge. HTW Berlin University of Applied Sciences. International Media and Computing Faculty IV. A thesis for the degree of Bachelor of Science. March 4, 2016. 62 p. https://upload.wikimedia.org/wikipedia/commons/9/99/Generating_Article_Placeholders_from_Wikidata_for_Wikipedia_-_Increasing_Access_to_Free_and_Open_Knowledge.pdf
5. Jakob Voß Classification of Knowledge Organization Systems with Wikidata. In Proc.

15th European Networked Knowledge Organization Systems Workshop. NKOS 2016. Hannover. September 9, 2016. Vol. 1676. P. 15–22. http://ceur-ws.org/Vol-1676/paper2.pdf

Получено 10.01.2017

Об авторах:

Проскудина Галина Юрьевна,
научный сотрудник,
Количество научных публикаций в украинских изданиях – 28.
Количество научных публикаций в зарубежных изданиях – 15.
http://orcid.org/0000-0001-9094-1565.

Кудим Кузьма Алексеевич,
младший научный сотрудник,
Количество научных публикаций в украинских изданиях – 12.
Количество научных публикаций в зарубежных изданиях – 7.
http://orcid.org/0000-0001-9483-5495.

Место работы авторов:

Институт программных систем
НАН Украины,
03187, Киев-187,
проспект Академика Глушкова, 40.
Тел.: +38(044)526 6033.

E-mail: gupros@isofts.kiev.ua,
kuzma@isofts.kiev.ua

УДК 004.942+004.415.2

А.Л. Яловець

АРХІТЕКТУРА ТА ФУНКЦІОНАЛЬНІ МОЖЛИВОСТІ МУЛЬТИАГЕНТНОЇ СИСТЕМИ НАВІГАЦІЯ

Розглядаються особливості проектування, розробки та функціонування мультиагентної системи Навігація. Наводиться архітектура системи та обґрунтовується вибір мови реалізації системи. Детально аналізуються функціональні можливості підсистем мультиагентної системи Навігація. На змістовному прикладі порівнюються результати мультиагентного моделювання процесів переслідування/утікання агентів засобами мультиагентної системи за різними режимами її функціонування.
Ключові слова: архітектура, мультиагентна система, функції підсистем, мультиагентне моделювання.

Вступ

Мультиагентна система (МАС) Навігація [1, 2] створена на основі методів, викладених в [3–8]. МАС Навігація призначена для моделювання поведінки множини інтелектуальних агентів у складному динамічному середовищі. Змістовно під динамічним середовищем розуміється ділянка морського кордону, в межах якої протидіють два класи кораблів: кораблі-порушники (утікачі) та кораблі берегової охорони (переслідувачі). Відповідно до цього, розглядаються два класи антагоністичних агентів (агенти-утікачі та агенти-переслідувачі), які протидіють у динамічному середовищі. Перед початком процесу моделювання переслідування агенти-переслідувачі засобами системи оптимально розподіляються по групах, кожна з яких націлена на захоплення цілком визначеного агента-утікача. Кожний з агентів-переслідувачів, що належить окремій групі, реагує на дії відповідного агента-утікача, а агенти-утікачі, в свою чергу, реагують на дії агентів-переслідувачів, які призначені для їх захоплення. Вважається, що кожний агент-утікач може перебувати в одному з двох станів: або він знає своїх переслідувачів, або він не знає своїх переслідувачів. У другому випадку кожний агент-утікач у кожний момент часу свого руху формує припущення щодо своїх вірогідних переслідувачів, на дії яких він і реагує. Будь-який агент намагається діяти найоптимальнішим чином: агенти-переслідувачі використовують оптимальні стратегії для найшвидшого захоплення відповідних агентів-утікачів, а агенти-утікачі використовують відповідні стратегії для збільшення часу їх захоплення (або уникнення цього) агентами-переслідувачами. З метою як найшвидшого захоплення агентів-утікачів агенти-переслідувачі в процесі переслідування можуть вступати між собою у перемовини, за результатами яких можуть переходити в інші групи. Агенти для запобігання зіткнень з іншими агентами використовують методи маневрування, які відповідають вимогам Міжнародних правил попередження зіткнення суден у морі [9].

1. Архітектура та мова реалізації МАС Навігація

Узагальнену архітектуру МАС Навігація показано на рис. 1. В архітектурі МАС можна умовно виділити дві укрупнені функціонально орієнтовані складові (середовище формування даних для мультиагентного моделювання та середовище мультиагентного моделювання процесів переслідування/утікання), пов’язані між собою через підсистему керування процесом моделювання (див. рис. 1).

До складу середовища формування даних для мультиагентного моделювання входять три підсистеми, засобами яких виконується завантаження, підготовка, редагування та збереження початкових даних, потрібних для функціонування середовища мультиагентного моделювання процесів переслідування/утікання, яке, в свою чергу, включає до свого складу дві головні підсистеми МАС Навігація. Функціональні можливості підсистем, показаних на рис. 1,

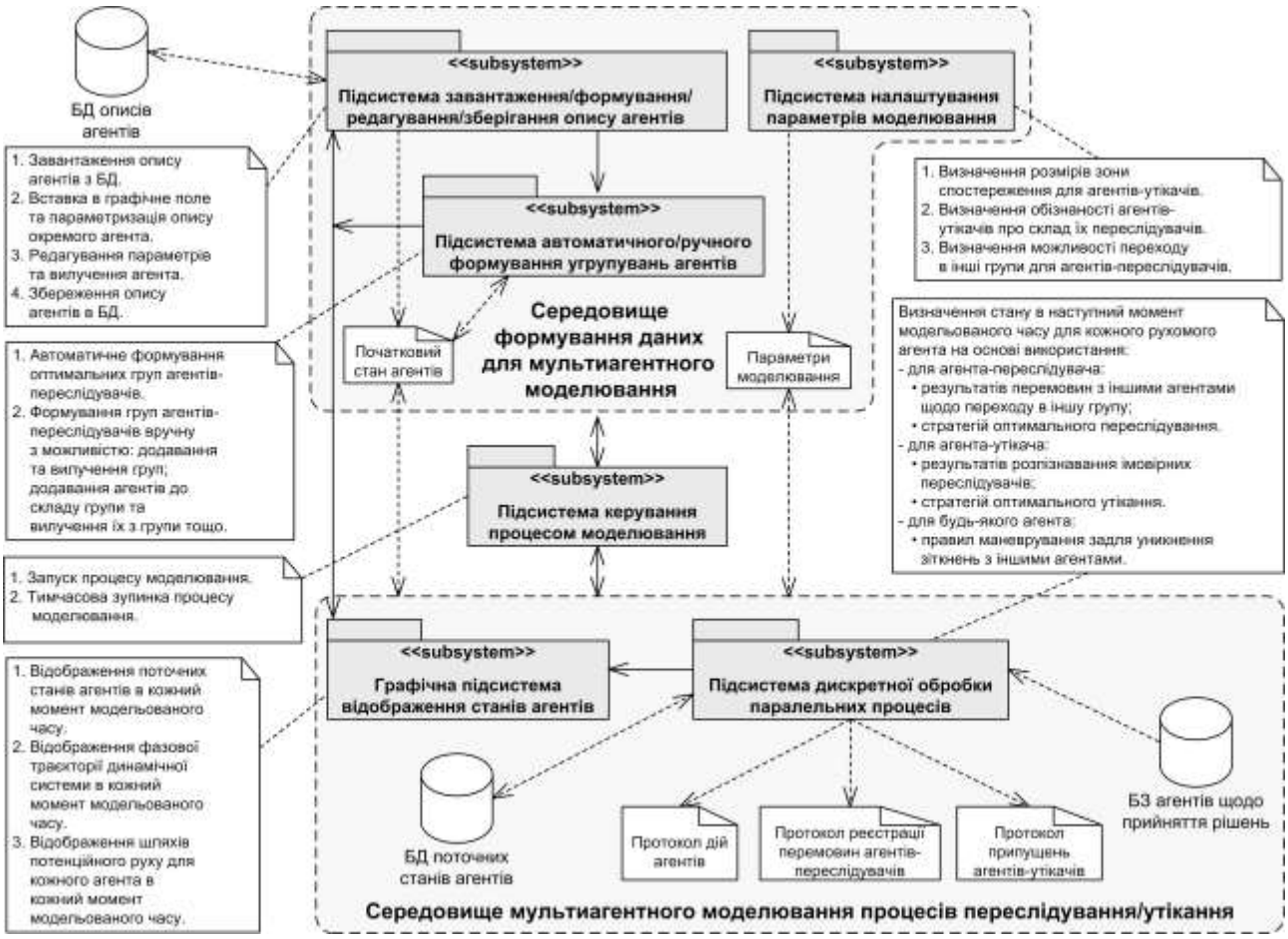


Рис. 1. Узагальнена архітектура МАС Навігація

розглянуто в п. 2 даної роботи.

Для реалізації МАС Навігація використано мову PDC Visual Prolog 5.2 (VIP) з наступних міркувань: по-перше, VIP – це реляційна мова, завдяки чому суттєво спрощується створення, обробка та збереження різноманітних баз даних (БД), потрібних для опису агентів та їх станів; по-друге, VIP – це мова логічного програмування, за допомогою якої забезпечується адекватний опис логіки поведінки агентів у різноманітних ситуаціях та подання знань агентів про їх дії у різних обставинах.

2. Функціональні можливості підсистем МАС Навігація

2.1. Підсистема завантаження/формування/редагування/зберігання опису агентів дозволяє побудувати нову БД описів агентів або завантажити раніше збережену БД. Під побудовою БД описів агентів розуміється процес послідовного формування множини агентів у графічному полі. Під формуванням агента розуміється упорядкована сукупність операцій, що включає (див. рис. 2): вибір вида агента (з панелі інструментів), параметризацію агента та його відображення у вибраній точці графічного поля. Як видно з рис. 2, як інтерфейс ця підсистема разом з діаловим інтерфейсом використовує інтерфейс графічної підсистеми відображення станів агентів (див. п. 2.6).

Сформовані описи агентів можуть бути відредаговані за допомогою pull-down меню, що відкривається при виділенні агента у графічному полі і натисканні на праву кнопку миші. Вибір опції меню „Вилучити” призводить до вилучення агента з графічного поля (та його опису з БД). Вибір опції „Редагувати” призводить до завантаження діалогу параметризації (див. рис. 2), в якому користувач має можливість внести зміни до опису агента.

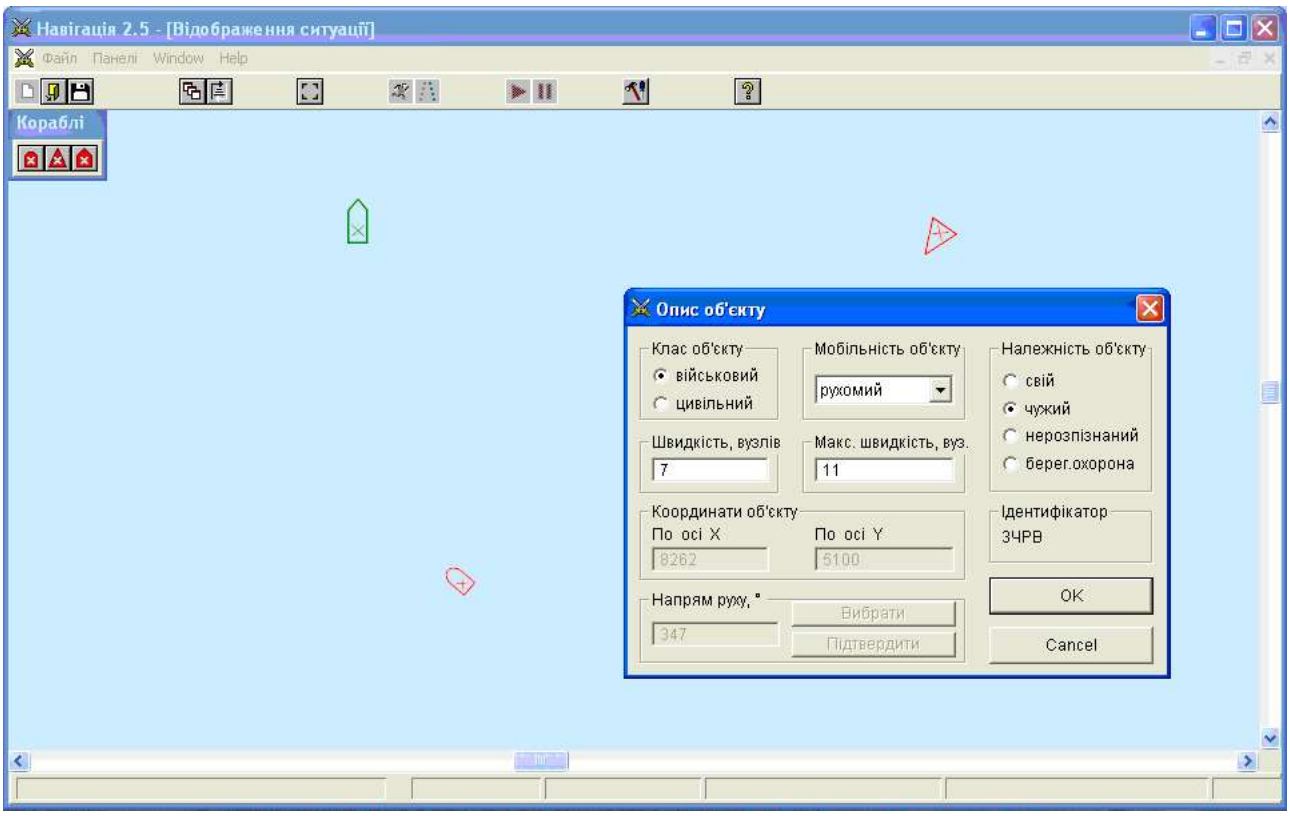


Рис. 2. Приклад виконання параметризації агента

Описи агентів утворюють БД описів агентів, яка може бути збережена та повторно завантажена. Сформована (завантажена) БД описів агентів утворює *початковий стан агентів* (див. рис. 1), використовуваний для подальшого моделювання процесів переслідування/утікання.

2.2. Підсистема налаштування параметрів моделювання дозволяє сформувати значення додаткових параметрів, які впливають на результати модельованого процесу переслідування/утікання агентів. Як інтерфейс ця підсистема використовує діалог «Установки параметрів» (див. рис. 3).

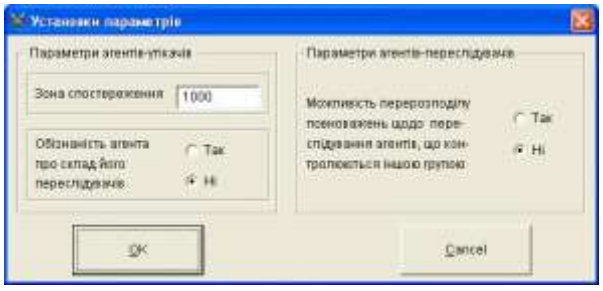


Рис. 3. Діалог налаштування параметрів

Засобами цього діалогу задаються додаткові параметри властивостей як агентів-утікачів, так і агентів-переслідувачів. Для агентів-утікачів задаються значення двох параметрів: зони спостереження та обізнаності щодо складу їх переслідувачів.

За допомогою зони спостереження моделюється простір видимості для агентів-утікачів і вони спостерігають тільки тих інших агентів, які в ній перебувають. Зазначимо, що агенти-утікачі можуть перебувати в одному з двох станів: або вони знають своїх агентів-переслідувачів, або ні. В другому випадку кожний агент-утікач для визначення складу своїх вірогідних агентів-переслідувачів (які потрапили в його зону спостереження) в кожний момент модельованого часу формує припущення.

В свою чергу, агенти-переслідувачі для зменшення часу переслідування можуть або переходити в інші групи, або ні, що і задається за допомогою діалогу.

Як наслідок, в результаті відповідних призначень параметрів можливі чотири альтернативи з моделювання поведінки агентів: «агенти-утікачі знають своїх агентів-переслідувачів та агенти-переслідувачі не перегруповуються»; «агенти-утікачі не знають своїх агентів-переслідувачів та

агенти-переслідувачі не перегруповуються»; «агенти-утікачі знають своїх агентів-переслідувачів та агенти-переслідувачі перегруповуються»; «агенти-утікачі не знають своїх агентів-переслідувачів та агенти-переслідувачі перегруповуються». Зазначені альтернативи суттєво розширяють функціональні можливості МАС Навігація щодо моделювання процесів переслідування/утікання на площині (приклади моделювання див. п. 3 даної роботи).

2.3. Підсистема автоматичного/ручного формування угрупувань агентів дозволяє сформувати групи агентів двома шляхами: автоматично (для цього використовується метод, викладений в [8]) або вручну.

Автоматичне формування груп агентів відбувається шляхом отримання оптимального рішення щодо стану груп, при якому мінімізується загальний час захоплення всіх утікачів переслідувачами з урахуванням обмеження, що швидкість кожного переслідувача у групі перебільшує швидкість відповідного утікача. При цьому, якщо швидкість окремого переслідувача у групі менша за швидкість відповідного утікача, підсистема діагностує помилку, для усунення якої необхідно засобами підсистеми завантаження/формування/редагування/зберігання опису агентів (див. п. 2.1) відредагувати швидкості агентів. Тобто ця підсистема може також використовувати інтерфейс підсистеми, описаної в п. 2.1.

Результати автоматичного формування груп агентів користувач може переглянути за допомогою діалогу „Формування груп” (див. рис. 4), який є інтерфейсом даної підсистеми.

В процесі формування груп вручну користувач може створювати групи за своїми уподобаннями, редагувати склад груп та переглядати розташування агентів у графічному полі (на рис. 4 показано приклад виділення агента для перегляду). Тобто дана підсистема може також використовувати інтерфейс графічної підсистеми відображення станів агентів (див. п. 2.6 та рис. 1).

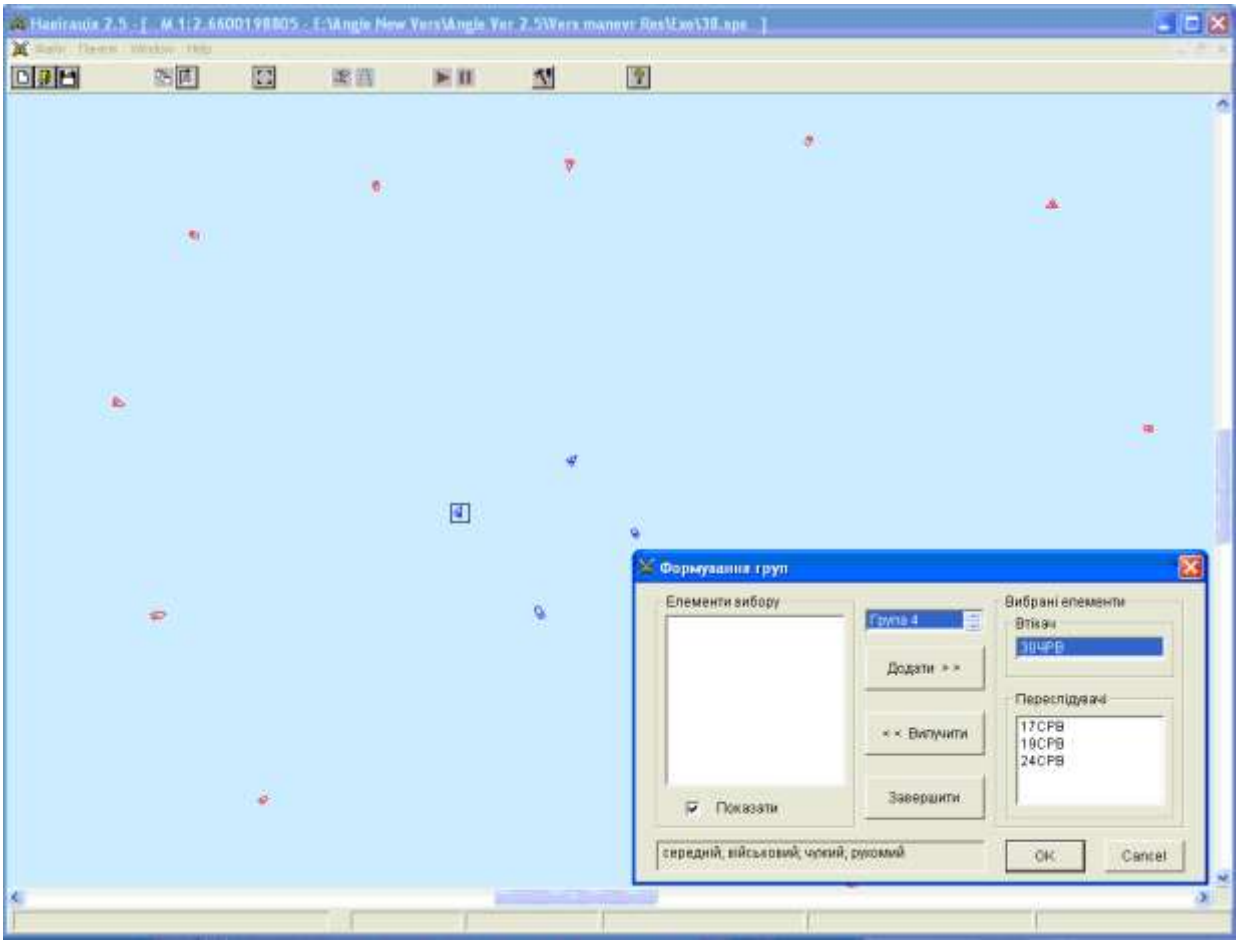


Рис. 4. Діалог формування/редагування стану груп агентів

2.4. Підсистема керування процесом моделювання стає доступною після успішного завершення роботи підсистеми автоматичного/ручного формування угрупувань агентів (див. п. 2.3), – при цьому стає активною кнопка „Запуск переслідування” панелі інструментів головного вікна МАС. При натисканні на цю кнопку керування передається підсистемі дискретної обробки паралельних процесів (див. рис. 1 та п. 2.5 даної роботи). При цьому стає активною (enabled) кнопка „Зупинити переслідування” панелі інструментів головного вікна МАС, а кнопка „Запуск переслідування” стає неактивною (disabled).

Користувач МАС Навігація в будь-який момент модельованого часу має можливість призупинити процес моделювання шляхом натискання на кнопку „Зупинити переслідування”. При цьому стає неактивною кнопка „Зупинити переслідування”, а кнопка „Продовжити переслідування” панелі інструментів головного вікна МАС стає активною.

При натисканні на кнопку „Зупинити переслідування” відбувається знищення таймера, запущеного на початку процесу моделювання, поточний стан агентів запам’ятовується як початковий стан агентів (див. рис. 1) та запам’ятовуються траєкторії здійснених рухів всіх агентів, що разом утворюють фазову траєкторію динамічної системи на момент зупинки модельованого часу. Зупинка процесу моделювання надає користувачу можливість змінити поточні параметри моделювання та проаналізувати стан агентів на даний момент модельованого часу.

Поновлення процесу моделювання відбувається шляхом натискання на кнопку „Продовжити переслідування”, при цьому ця кнопка стає неактивною, а кнопка „Зупинити переслідування” – активною. При натисканні на кнопку „Продовжити переслідування” відбувається запуск таймера і процес моделювання засобами підсистеми дискретної обробки паралельних процесів поновлюється зі збереженого початкового стану агентів з урахуванням поточних параметрів моделювання (див. рис. 1).

2.5. Підсистема дискретної обробки паралельних процесів є головною підсистемою МАС Навігація, в рамках якої реалізовано методи, викладені в [3–7]. Засобами цієї підсистеми підтримуються всі аспекти прийняття рішень агентами в межах модельованих процесів.

При активації даної підсистеми відбувається запуск таймера і в кожний момент модельованого часу кожний рухомий агент аналізує свій поточний стан та стани інших агентів з метою вирішення поставлених перед ним задач.

На рис. 5 показано діаграму діяльності *i*-го агента в довільний момент модельованого часу (в нотації UML). Ця діаграма узагальнено характеризує всі основні аспекти прийняття рішень довільним агентом (додатково зазначимо, що діаграму діяльності щодо виконання маневрування агентами наведено в [5]). Для загальності подання, в діаграмі розглянуто будь-якого агента (як рухомого, так і нерухомого), хоча в реалізації поточні стани *нерухомих* агентів підсистемою не розглядаються та не поновлюються.

Засобами даної підсистеми в кожний момент модельованого часу послідовно розглядаються всі рухомі агенти відповідно до наведеної діаграми їх діяльності.

З точки зору програмної реалізації, це відбувається в процесі виконання окремого періоду таймера події e_Timer. Після завершення виконання періоду таймера події e_Timer, виконується виклик події e_Update, в рамках якої відбувається перемальовування поновленого стану всіх агентів засобами підсистеми відображення станів агентів (див. п. 2.6 та рис. 1), після чого керування знов передається події e_Timer і так далі, до моменту захоплення всіх утікачів або призупинення процесу моделювання засобами підсистеми керування процесом моделювання (див. п. 2.4).

Зазначимо, що множина знань агентів, яка включає (див. рис. 5) правила прийняття рішень агентами та алгоритми, використовувані ними для оптимізації власних дій, узагальнено утворюють БЗ агентів щодо прийняття рішень (див. рис. 1).

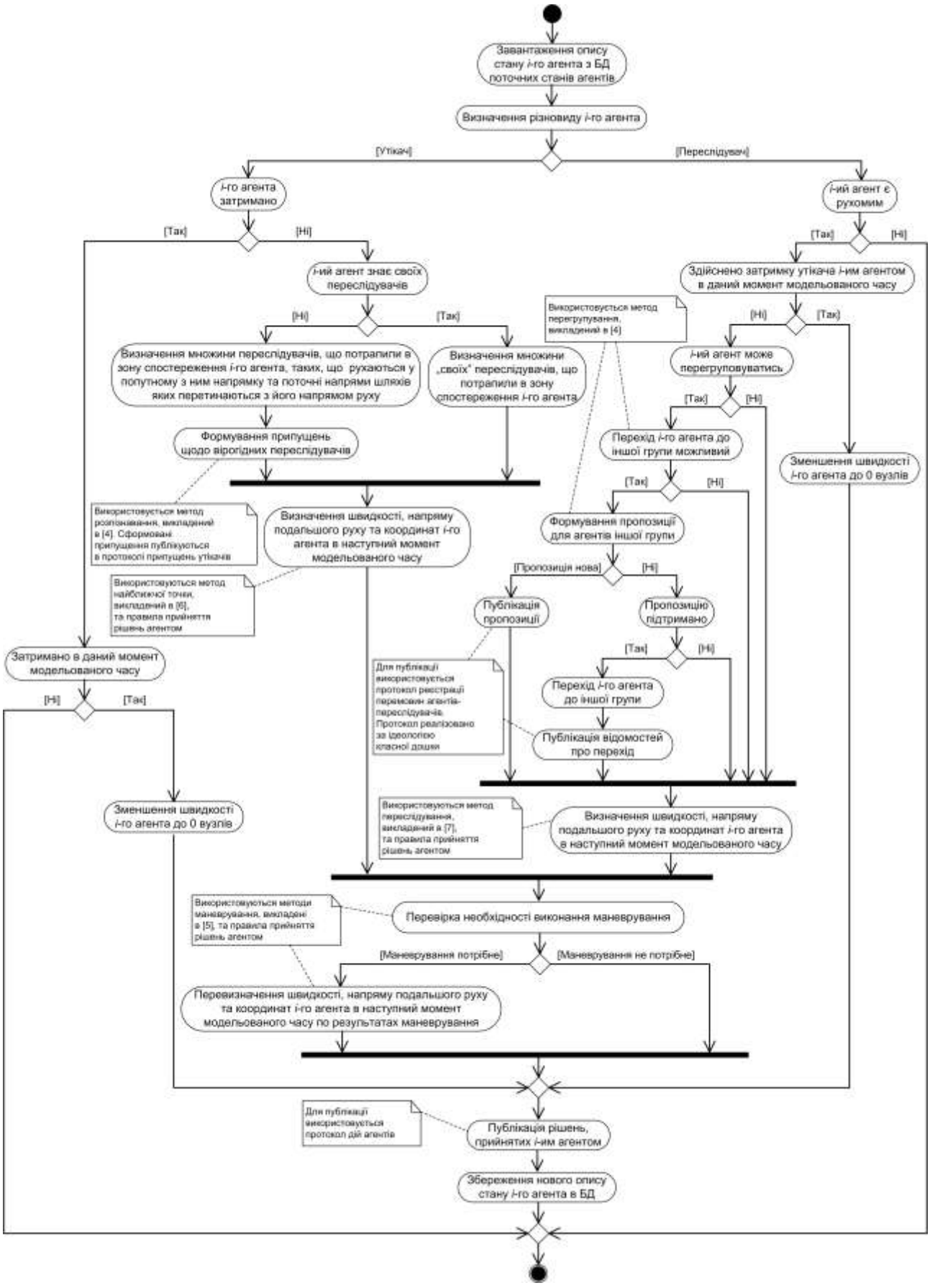


Рис. 5. Діаграма діяльності *i*-го агента в довільний момент модельованого часу

Слід підкреслити, що дана підсистема не має власного інтерфейсу, але використовує чотири зовнішні інтерфейси (див. рис. 1): графічної підсистеми відображення станів агентів (графічне вікно для малювання поточних станів агентів у кожний момент модельованого часу) та протоколів (три графічних вікна для відображення текстових протоколів дій, перемовин та припущень агентів).

2.6. Графічна підсистема відображення станів агентів, як вище показано, використовується декількома підсистемами як інтерфейс (див. рис. 1) та призначена для відображення поточних станів агентів як на етапі формування описів агентів (див. п. 2.1), так і в процесі мультиагентного моделювання (див. п. 2.5). Водночас основним призначенням цієї підсистеми є надання користувачу МАС Навігація головного інтерфейсу мультиагентного моделювання, оскільки саме завдяки їй користувач може в реальному масштабі часу спостерігати всі візуальні аспекти модельованих процесів. У даній підсистемі реалізовано основні функціональні можливості, притаманні графічним системам (у тому числі масштабування, здвиг, знаходження та виділення об'єктів моделювання тощо).

3. Функціональні можливості МАС Навігація

Функціональні можливості МАС Навігації щодо мультиагентного моделювання процесів переслідування/утікання розглянемо на прикладі, в якому засобами підсистеми налаштування параметрів моделювання (див. п. 2.2 даної роботи) задано чотири різні режими моделювання:

1) утікачам відомі їх переслідувачі, які не можуть перегруповуватись (рис. 7);

2) утікачам невідомі їх переслідувачі, які не можуть перегруповуватись (рис. 8);

3) утікачам відомі їх переслідувачі, які можуть перегруповуватись (рис. 9);

4) утікачам невідомі їх переслідувачі, які можуть перегруповуватись (рис. 10).

На рис. 6 показано приклад початкової дислокації агентів, який ми будемо далі використовувати для демонстрації розбіжностей при різних режимах моделювання процесів переслідування/утікання агентів на площині. На рисунку в центрі дислокації розташовані утікачі (обведені колом), які оточені переслідувачами.

Результати мультиагентного моделювання процесів переслідування/утікання, виконаного засобами МАС Навігація, для чотирьох вищезазначених режимів моделювання наведено в таблицях 1–4.

Зазначимо, що час, витрачений агентами-переслідувачами на захоплення агентів-утікачів, обчислюється в секундах модельованого часу (див. табл. 1–4), який визначається внаслідок здійснення мультиагентного моделювання засобами МАС Навігація при встановленому періоді таймера зі значенням 0.2 сек. Абревіатури, що позначають агентів, є їх унікальними ідентифікаторами, які автоматично генеруються при первинному формуванні описів агентів (наприклад, 28ЧРВ сформовано шляхом конкатенації порядкового номера агента (28) та ознак «чужий» (Ч), «рухомий» (Р) та «військовий» (В)).

Порівняння результатів моделювання при режимах, в яких переслідувачі не вступали у перемовини (не перегруповувались), а утікачі знали (див. рис. 7 та табл. 1) або не знали (див. рис. 8 та табл. 2) своїх переслідувачів, дозволяє зробити висновок, що у даному випадку наявність знань в утікачів щодо їх переслідувачів, по-перше, значно збільшує як мінімальний час їх захоплення (до 90 % – див., наприклад, мінімальний час захоплення утікача 28ЧРВ в табл. 1 та 2), так і загальний максимальний час переслідування (до 25 %), по-друге, призводить до різних траєкторій руху утікачів, внаслідок чого їх первинне захоплення здійснюється різними переслідувачами (за винятком утікача 31ЧРВ, у якого траєкторія руху та переслідувач з мінімальним часом захоплення в обох режимах збігаються, що пояснюється тим, що даний утікач, на відміну від інших утікачів, у процесі свого руху не відчував впливу переслідувачів з інших груп).

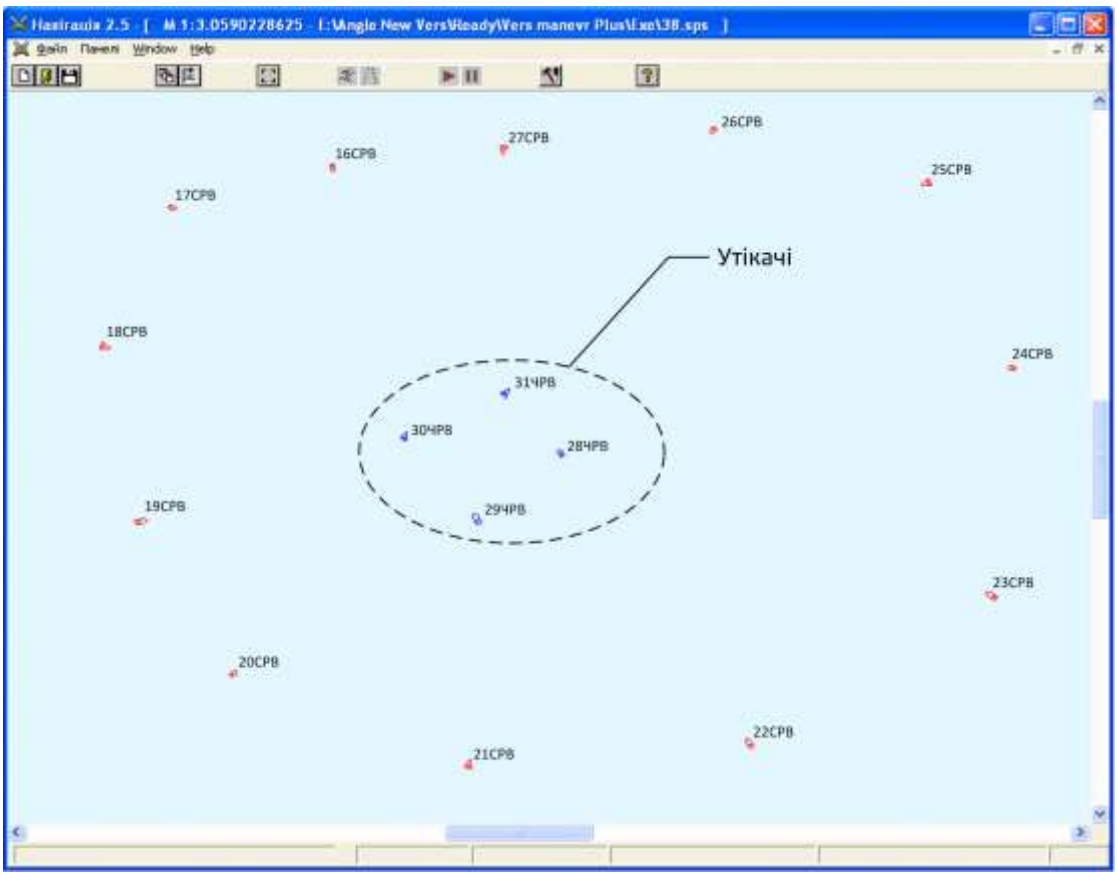


Рис. 6. Приклад початкової дислокації агентів

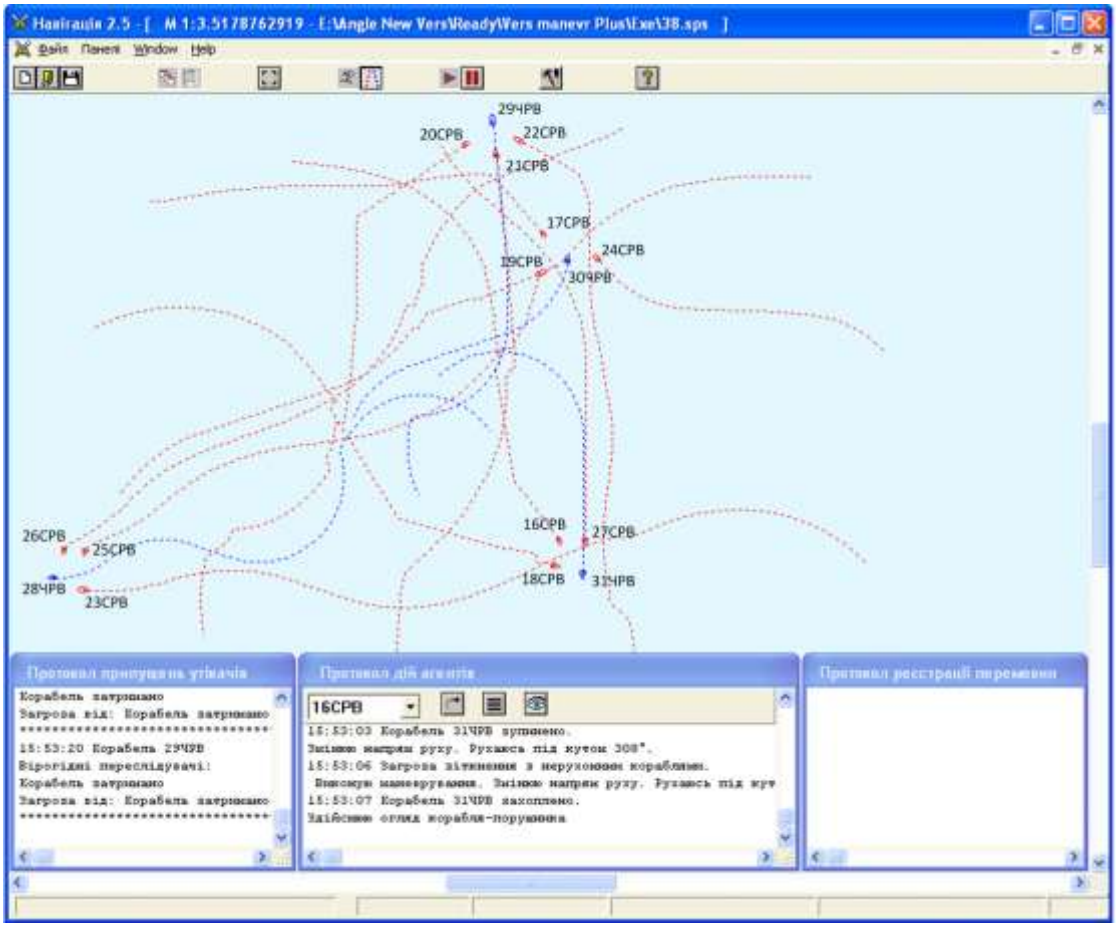


Рис. 7. Утікачі знають своїх переслідувачів; переслідувачі не можуть перегруповуватись

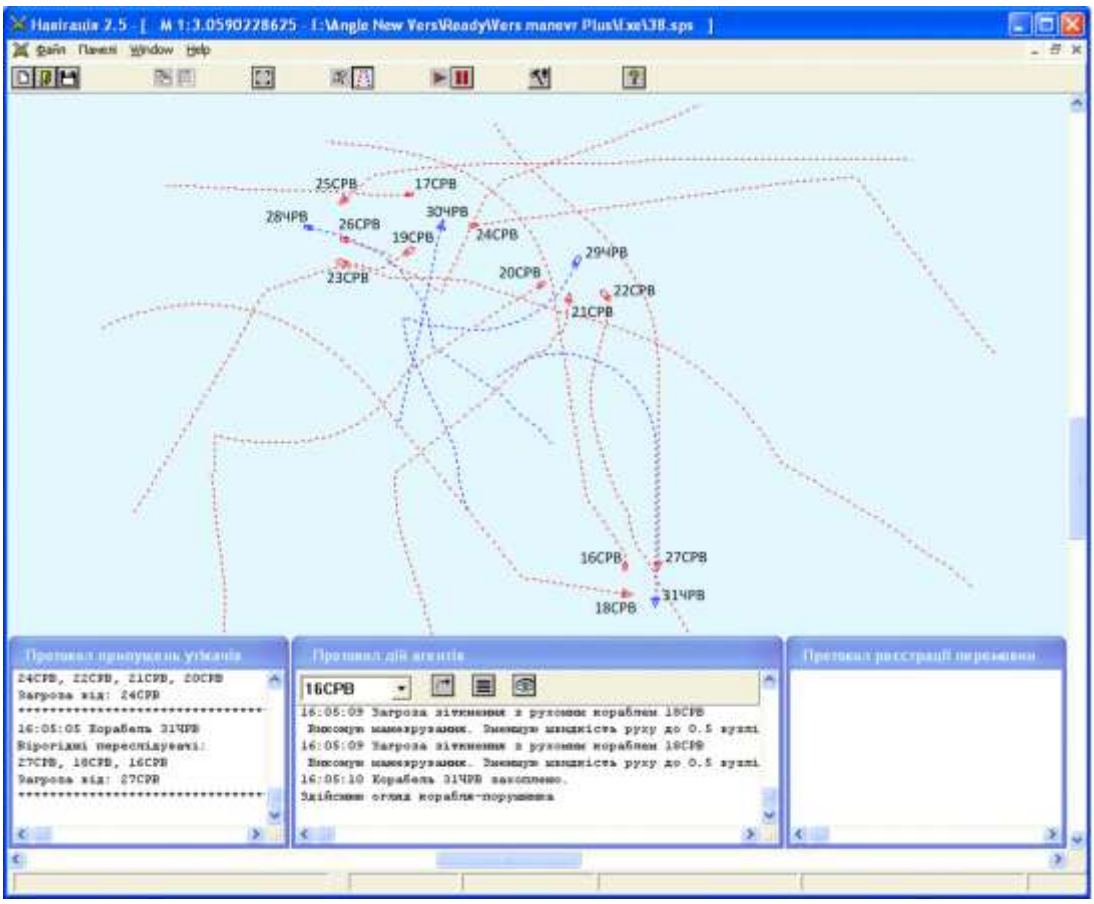


Рис. 8. Утікачі не знають своїх переслідувачів; переслідувачі не можуть перегруповуватись

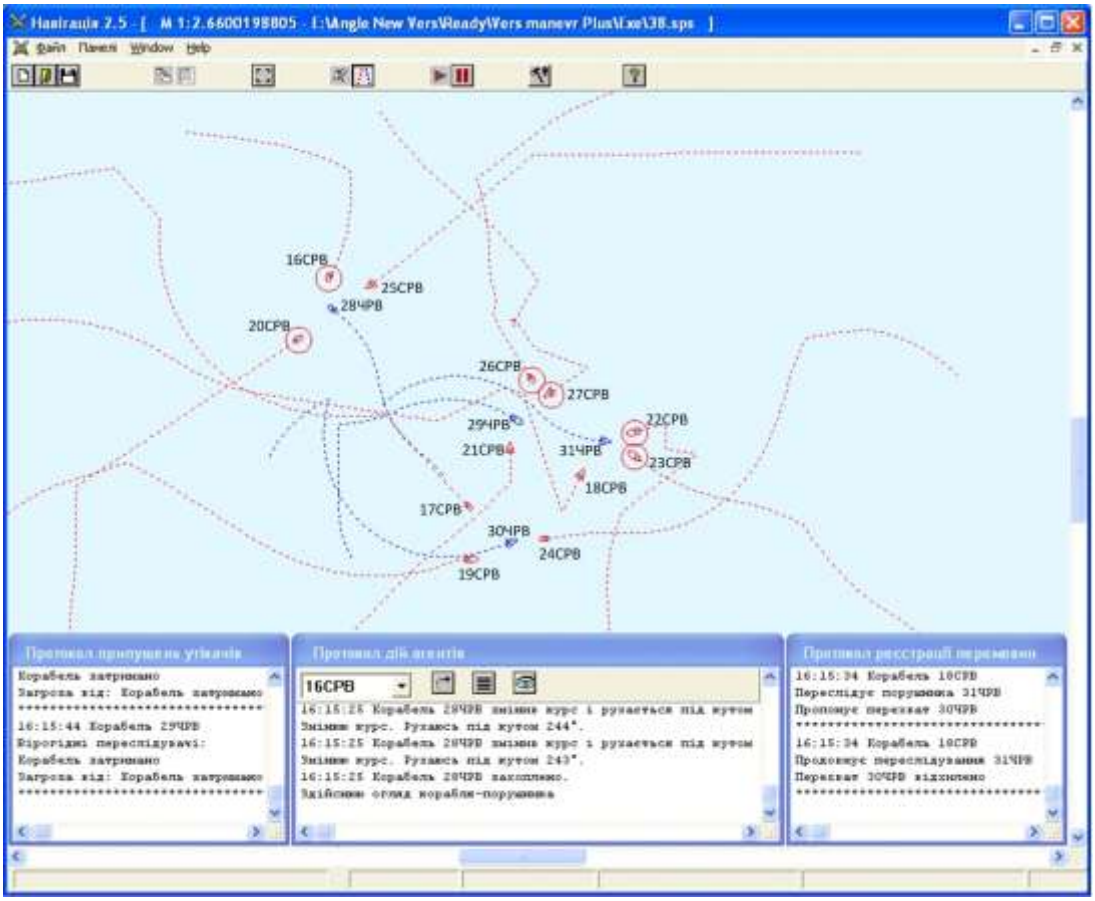


Рис. 9. Утікачі знають своїх переслідувачів; переслідувачі можуть перегруповуватись

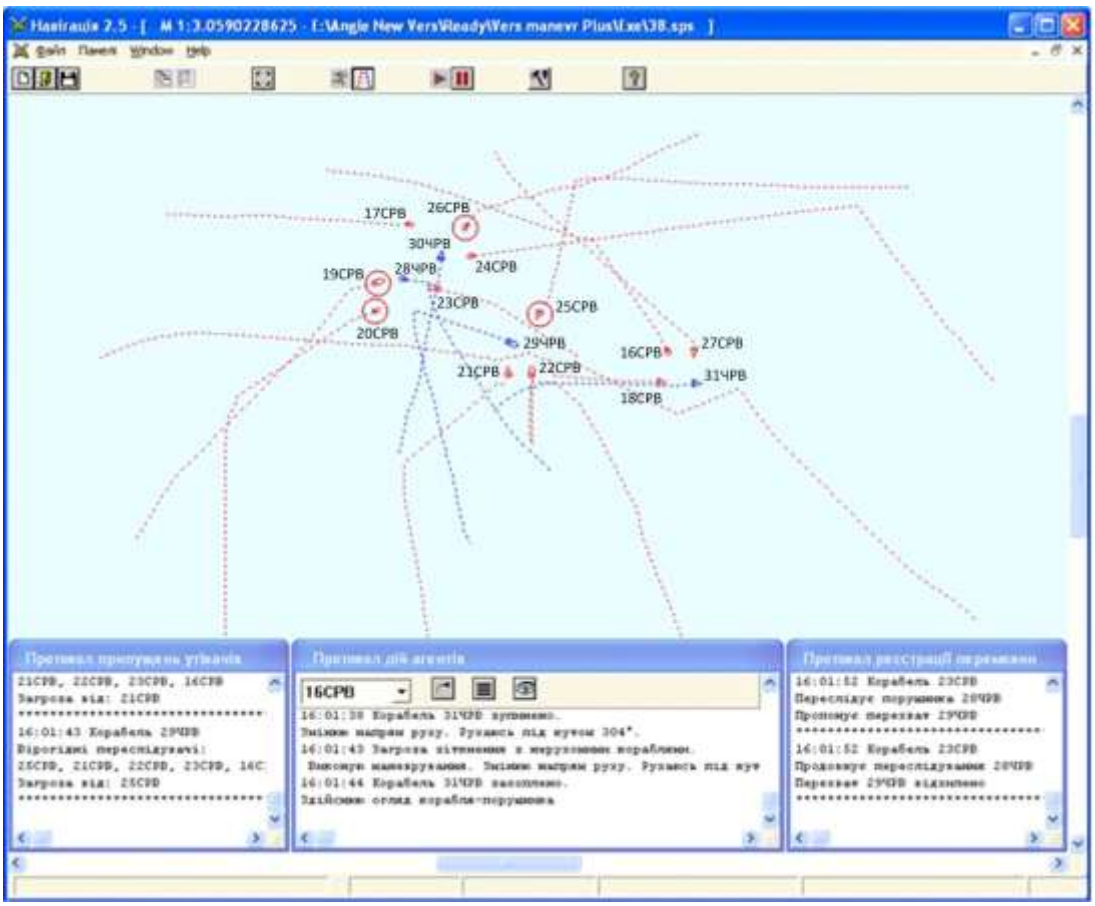


Рис. 10. Утікачі не знають своїх переслідувачів; переслідувачі можуть перегруповуватись

Таблиця 1

Результати мультиагентного моделювання процесів переслідування/утікання для випадку, коли утікачам відомі їх переслідувачі, переслідувачі не можуть перегруповуватись

Номер групи	Утікач	Склад оптимальної групи переслідувачів	Переслідувач/ min час захоплення (с)	Переслідувач/ max час захоплення (с)	Загальний max час переслідування (с)
1	28ЧРВ	23CPB 25CPB 26CPB	23CPB/40	26CPB/40	40
2	29ЧРВ	20CPB 21CPB 22CPB	21CPB/28	20CPB/36	
3	30ЧРВ	17CPB 19CPB 24CPB	24CPB/18	19CPB/26	
4	31ЧРВ	16CPB 18CPB 27CPB	27CPB/23	18CPB/32	

Таблиця 2

Результати мультиагентного моделювання процесів переслідування/утікання для випадку, коли утікачам невідомі їх переслідувачі, переслідувачі не можуть перегруповуватись

Номер групи	Утікач	Склад оптимальної групи переслідувачів	Переслідувач/ min час захоплення (с)	Переслідувач/ max час захоплення (с)	Загальний max час переслідування (с)
1	28ЧРВ	23СРВ 25СРВ 26СРВ	26СРВ/21	23СРВ/32	32
2	29ЧРВ	20СРВ 21СРВ 22СРВ	22СРВ/23	20СРВ/31	
3	30ЧРВ	17СРВ 19СРВ 24СРВ	17СРВ/11	24СРВ/31	
4	31ЧРВ	16СРВ 18СРВ 27СРВ	27СРВ/23	18СРВ/28	

Таблиця 3

Результати мультиагентного моделювання процесів переслідування/утікання для випадку, коли утікачам відомі їх переслідувачі, переслідувачі можуть перегруповуватись

Номер групи	Утікач	Склад оптимальної групи переслідувачів	Склад групи після перегрупування		Переслідувач/ min час захоплення (с)	Переслідувач/ max час захоплення (с)	Загальний max час переслідування (с)
1	28ЧРВ	23СРВ 25СРВ 26СРВ	+	16СРВ 20СРВ 25СРВ	16СРВ/11	20СРВ/23	30
2	29ЧРВ	20СРВ 21СРВ 22СРВ	+	21СРВ 26СРВ 27СРВ	26СРВ/16	27СРВ/20	
3	30ЧРВ	17СРВ 19СРВ 24СРВ	–	17СРВ 19СРВ 24СРВ	19СРВ/21	17СРВ/26	
4	31ЧРВ	16СРВ 18СРВ 27СРВ	+	18СРВ 22СРВ 23СРВ	23СРВ/14	18СРВ/30	

Таблиця 4

Результати мультиагентного моделювання процесів переслідування/утікання для випадку, коли утікачам невідомі їх переслідувачі, переслідувачі можуть перегруповуватись

Номер групи	Утікач	Склад оптимальної групи переслідувачів	Склад групи після перегрупування		Переслідувач/ min час захоплення (с)	Переслідувач/ max час захоплення (с)	Загальний max час переслідування (с)
1	28ЧРВ	23СРВ 25СРВ 26СРВ	+	19СРВ 20СРВ 23СРВ	19СРВ/16	23СРВ/29	31
2	29ЧРВ	20СРВ 21СРВ 22СРВ	+	21СРВ 22СРВ 25СРВ	25СРВ/19	22СРВ/28	
3	30ЧРВ	17СРВ 19СРВ 24СРВ	+	17СРВ 24СРВ 26СРВ	17СРВ/11	24СРВ/31	
4	31ЧРВ	16СРВ 18СРВ 27СРВ	–	16СРВ 18СРВ 27СРВ	27СРВ/13	18СРВ/27	

Якщо ж порівнювати результати моделювання при режимах, в яких переслідувачі вступали у перемовини (могли перегруповуватись), а утікачі знали (див. рис. 9 та табл. 3) або не знали (див. рис. 10 та табл. 4) своїх переслідувачів, то у даному випадку наявність знань в утікачів щодо їх переслідувачів не мала такого значення, як у попередньому порівнянні. По-перше, на дії утікачів (і, як наслідок, на час їх захоплення), які знали своїх переслідувачів, впливала необхідність маневрування між собою (на відміну від утікачів, які не знали своїх переслідувачів). По-друге, виконане перегрупування переслідувачів (в табл. 3, 4 групи, в яких відбулось перегрупування, позначено символом «+»; на рис. 9 та 10 переслідувачі, які перегрупувалися, обведені колами) ще більше зменшило час захоплення відповідних утікачів. Водночас, невиконання перегрупування та наявність знань щодо своїх переслідувачів (див., наприклад, мінімальний час захоплення утікача 30ЧРВ в табл. 3 та 1) призводить до збільшення часу первинного захоплення цього утікача на 17 %.

Ефекти від перегрупування віддзеркалює і попарне порівняння режимів, коли утікачам відомі їх переслідувачі, що або перегруповуються, або ні (див. табл. 3 і рис. 9 та табл. 1 і рис. 7), та коли утікачам невідомі їх переслідувачі, що або перегруповуються, або ні (див. табл. 4 і рис. 10 та табл. 2 і рис. 8). В першому випадку внаслідок перегрупування мінімальний час захоплення зменшується від 40 % до 73 %, а максимальний час захоплення – від 6 % до 44 % (порівн. табл. 1 та табл. 3); у другому випадку – мінімальний час захоплення зменшується від 17 % до 43 %, а максимальний час захоплення – від 4 % до 10 % (порівн. табл. 2 та 4).

Слід окремо підкреслити суттєвий вплив процесів маневрування на процеси переслідування/утікання агентів (і, як наслідок, на час захоплення утікачів): в процесі маневрування агенти не тільки змінюють напрям руху з метою уникнення зіткнень, а й за необхідністю зменшують швидкість свого руху (аж до повної

зупинки), що значно впливає на часові характеристики процесів переслідування/утікання. Крім того зауважимо, в МАС Навігація реалізовано автоматичні зміни швидкостей агентів у випадках, коли агент-утікач вперше розпізнав, що в зону його спостереження потрапив агент-переслідувач, який гіпотетично може бути його переслідувачем. В такій ситуації даний агент-утікач збільшує свою швидкість до максимально можливої, при цьому його фактичні агенти-переслідувачі також збільшують свої швидкості до максимально можливих.

З вищевикладеного стає очевидним, що формування груп за допомогою методів оптимізації (див. [8]) в розглянутих динамічних системах не дозволяє гарантувати отримання дійсно оптимальних рішень (за критерієм часу захоплення), оскільки, як показано вище, на поведінку об'єктів оптимізації впливають різні чинники, що ускладнюють визначення чітких критеріїв оптимізації.

Як наслідок, стає очевидним, що на основі методів оптимізації можна виконати лише первинний розподіл агентів по групах, який у подальшому може бути суттєво покращений завдяки перегрупуванню агентів за результатами перемовин.

Висновки

В статті описано особливості проектування, реалізації та функціонування МАС Навігації, як системи, засобами якої здійснюється моделювання процесів переслідування/утікання агентів на площині для загального випадку (n переслідувачів, m утікачів, де $n \geq m$). Для МАС Навігації наведено її архітектуру з детальною характеристикою основних функціональних компонентів; подано діаграму діяльності довільного агента, яка дозволяє отримати уявлення про «приховані» процеси прийняття рішень агентами; продемонстровано функціональні можливості МАС Навігація за різними режимами її функціонування. МАС Навігацію можна розглядати як інструментарій для практичної перевірки теоретичних результатів, отримуваних у рамках досліджень з теорії диференційних ігор.

1. *Яловець А.Л., Кондращенко В.Я., Арістов В.В*. Свідоцтво № 46897 про реєстрацію авторського права на твір «Комп'ютерна програма – "Мультиагентна система "Навігація", версія 2.0"». Державна служба інтелектуальної власності України, 2012.
2. *Яловець А.Л., Кондращенко В.Я., Арістов В.В*. Свідоцтво № 57880 про реєстрацію авторського права на твір «Комп'ютерна програма – "Мультиагентна система "Навігація", версія 2.5" ("МАС Навігація 2.5")». – Державна служба інтелектуальної власності України, 2014 р.
3. *Яловець А.Л.* До постановки задачі переслідування на площині. Проблеми програмування. 2013. № 2. С. 95–100.
4. *Яловец А.Л.* Методы моделирования поведения агентов в мультиагентной системе «Навигация». Проблеми програмування. 2014. № 2–3. С. 212–220.
5. *Яловець А.Л.* Проблема моделювання маневрування агентів в задачах переслідування/утікання на площині. Проблеми програмування. 2015. № 2. С. 86–100.
6. *Яловець А.Л.* Про метод найближчої точки як метод управління стратегіями переслідування/утікання агентів. Проблеми програмування. 2013. № 4. С. 94–99.
7. *Яловець А.Л.* Про один метод переслідування на площині. Проблеми програмування. 2013. № 3. С. 117–124.
8. *Яловець А.Л.* Проблема формування угрупувань агентів у задачах переслідування/утікання на площині. Проблеми програмування. 2014. № 1. С. 108–118.
9. *Коккрофт А.Н., Ламеер Дж.Н.Ф*. Руководство по Правилам предупреждения столкновения. СПб: ООО «МОРСАР». 2005. 320 с.

References

1. Yalovets A.L., Kondraschenko V.Ya., Aristov V.V. 46897 Certificate of registration of copyright in a product "Computer program – "Multi-agent systems "Navigation" version 2.0" ". The State Service of Intellectual Property of Ukraine, 2012. (in Ukrainian).
2. Yalovets A.L., Kondraschenko V.Ya., Aristov V.V. 57880 Certificate of registration of copyright in a product "Computer program - "Multi-agent systems "Navigation" version

2.5" ("MAS Navigation 2.5")". The State Service of Intellectual Property of Ukraine, 2014. (in Ukrainian).
3. Yalovets A.L. To problem definition of prosecuting in the plane. Problems in programming. 2013. N 2. P. 95–100. (in Ukrainian).
4. Yalovets A.L. Methods of modelling the behaviour of agents in multi-agent system "Navigation". Problems in programming. 2014. N 2–3. P. 212–220. (in Russian).
5. Yalovets A.L. The problem of modelling of maneuvering of agents in the tasks of pursuit/escape on a plane. Problems in programming. 2015. N 2. P. 86–100. (in Ukrainian).
6. Yalovets A.L. About the method of the nearest point as a method of strategies management of pursuit/escape of agents. Problems in programming. 2013. N 4. P. 94–99. (in Ukrainian).
7. Yalovets A.L. About one method of the persecution on a plane. Problems in programming. 2013. N 3. P. 117–124. (in Ukrainian).
8. Yalovets A.L. The problem of the formation of groups of agents in the tasks of pursuit/escape on a plane. Problems in programming. 2014. N 1. P. 108–118. (in Ukrainian).
9. Cockroft A.N., Lameijer J.N.F. A Guide to the Collision Avoidance Rules. Sankt-Petersburg: “MORSAR” Ltd. 2005. 320 p. (in Russian).

Одержано 29.12.2016

Про автора:

Яловець Андрій Леонідович,
доктор технічних наук,
заступник директора інституту.
Кількість наукових публікацій в українських виданнях – понад 100.
Кількість наукових публікацій в зарубіжних виданнях – 5.
http://orcid.org/0000-0001-6542-3483

Місце роботи автора:

Інститут програмних систем
НАН України.
03187, Київ-187,
проспект Академіка Глушкова, 40.
Тел.: (044) 526 15 38.
E-mail: yal@isofts.kiev.ua

УДК 004.855:519.216

О.С. Балабанов

ПРИНЦИПИ ТА АНАЛІТИЧНІ ЗАСОБИ РЕКОНСТРУКЦІЇ СТРУКТУР ЙМОВІРНІСНИХ ЗАЛЕЖНОСТЕЙ У СПЕЦІАЛЬНОМУ КЛАСІ

Запропоновано та обґрунтовано набір емпіричних резолюції, які спираються виключно на безумовні залежності двох змінних та забезпечують ідентифікацію безпосередніх зв’язків (ребер) у структурах залежностей в класі монопотокових графів. Цей клас структур є підкласом ациклонних орграфів та суперкласом для полі-лісів. Охарактеризовано властивості монопотокових моделей. Коректність розроблених емпіричних резолюцій ґрунтується на емпірично надійному припущенні безумовної (маргінальної) реберної неоманливості.

Ключові слова: відтворення структури залежностей, монопотоковий граф, безумовна залежність, емпіричне проявлення залежностей, емпіричні резолюції ідентифікації ребер.

Вступ

Відтворення структур зв’язків та впливів між характеристиками середовища, які неявно відбиті в багатовимірних масивах статистичних даних – одна з центральних задач глибокого аналізу даних та відкриття знань у базах даних. Хоча за останню чверть століття отримано багато результатів для розв’язання цієї важкої задачі, інтенсивні дослідження й розробки тривають [1–5]. Найбільш важка постановка задачі постає в ситуації, коли на структуру ймовірнісних залежностей не накладено жодних обмежень і зовсім нічого не відомо про цю структуру. Зазвичай передбачається єдине обмеження – в структурі немає орієнтованих циклів (циклонів). Відсутність циклонів можна вважати вимогою коректного збору даних. Тобто в процесі генерації одного запису даних кожна змінна X вимірюється досить швидко, так що «сигнал» від X до інших змінних не може встигнути «оббігти коло» й вплинути на X у цьому записі. Структури ймовірнісних залежностей, де заборонено циклони, об’єднуються в клас ациклонних орієнтованих графів (АОГ). Задача відтворення автентичної (генеративної) АОГ-структури за відсутності апріорних знань у найгіршому випадку є експоненційно важкою [4, 5]. Натомість якщо відомо, що структура генеративної моделі задовольняє певним «топологічним» обмеженням, то відтворення такої структури значно спрощується. Для різних спеціальних класів структур й моделей розроблено спеціалізовані методи відтворення моделі. Коли генеративна модель дійсно належить обраному класу, то така задача характеризується як реконструкція (відтворення) моделі. Коли генеративна модель виходить за межі обраного класу, то задачу називають «редукція» (структурна апроксимація, «покриття») моделі. В статті розглянуто постановку, коли відомо, що структура моделі належить до класу монопотокових графів залежностей. Строго обґрунтовано нові засоби вдосконалення методів реконструкції моделей у цьому класі.

Характеристика монопотокових моделей залежностей та огляд методів їх відтворення

Ребро відображає безпосередній зв’язок між двома змінними (вершинами). В більшості практичних задач працюють з моделями в підкласі ациклонних орієнтованих графів, де всі ребра є односпрямовані (тобто мають вигляд $X \to Y$). На основі таких графів (задавши параметри локальних залежностей) утворюється клас ординарних АОГ-моделей (оАОГ-моделей), у тому числі байєсові та гауссові мережі [1, 3, 6, 7]. Спеціальні підкласи оАОГ-моделей визначаються певними «топологічними» обмеженнями на структуру (вводиться заборона циклів або шля-

хів певних типів). Нагадаємо необхідні графові поняття.

Ребро зображується як неорієнтоване $X — Y$, коли орієнтація цього ребра є невідома або несуттєва в даному контексті. *Цикл* у графі – це шлях з ребер $X - Y - \cdots - X$, де всі проміжні (некінцеві) вершини – різні, а кінцеві вершини – тотожні. *Колізор* – це шлях вигляду $X \to Y \leftarrow Z$. *Ланцюг* – це шлях, на якому немає жодного колізора. Оршлях – це ланцюг вигляду $X \to \cdots \to Z \to Y$. Циклон – це оршлях, де остання вершина збігається з першою.

Мабуть найбільш відомим підкласом оАОГ є ліси залежностей. В лісі (дереві) залежностей немає ані циклів, ані колізорів. Його розширенням є полі-ліси; в них заборонені цикли (а колізори – дозволені). Ліси й полі-ліси підтримують прості паттерни залежностей, недостатні для моделювання багатьох об'єктів. Проміжне місце між полі-лісами та загальним випадком оАОГ посідає клас так званих монопотокових графів залежностей (МПГЗ). Моделі на базі МПГЗ можуть мати цикли і є значно експресивнішими за полі-ліси. (В літературі вони називалися простими або спрощеними оАОГ). Аксіоми та властивості МПГЗ-моделей описані в [6, 7]. Можна дати кілька еквівалентних варіантів **визначення** МПГЗ:

1) якщо в МПГЗ існує ланцюг $X \to Q \to Y$ або ребро $X — Y$, то між вершинами X та Y не існує жодного іншого ланцюга;

2) між батьками однієї й тієї ж вершини в МПГЗ немає жодного ланцюга;

3) в МПГЗ неможливий цикл з одним колізором.

Наслідками визначення є властивості МПГЗ.

Властивість 1: якщо в МПГЗ між вершинами X та Y існує більше одного ланцюга, то серед тих ланцюгів немає жодного оршляху, тобто всі ті ланцюги мають вигляд $X \leftarrow \cdots \to Y$. Візьмемо довільно два таких ланцюга λ та μ. На ланцюгах λ та μ є змінні відповідно Q та R, такі, що між Q та R не існує жодного ланцюга.

Властивість 2: якщо в МПГЗ між вершинами X та Y існують різні ланцюги λ та μ, з тим, що вони проходять через спільну вершину Q, яка не тотожна ані X, ані Y, то тоді ланцюги λ та μ закінчуються спільною дугою $X \leftarrow L$ або (та) спільною дугою $R \to Y$. (Можливо, що $Q \equiv L$ або $Q \equiv R$).

Сенс цієї властивості: коли в МПГЗ кілька ланцюгів між X та Y «зійшлися», вони вже не можуть розійтися (розділитися), бо це породило б одноколізорний цикл.

Залежність між змінними забезпечується неблокованим шляхом між змінними. Змінні X та Y є безумовно залежні тоді й тільки тоді, якщо між X та Y існує хоча б один ланцюг. Марковські властивості класу оАОГ-моделей визначаються критерієм d-сепарації [1, 3, 6, 8]. Кожному факту d-сепарації відповідає умовна незалежність. Процедурі перевірки факту d-сепарації можна зіставити статистичний тест незалежності. Ранг тесту незалежності визначається кількістю змінних в умові тесту. (Тест безумовної незалежності має нульовий ранг).

Відомим показником тісноти (сили) залежності між дискретними змінними X та Y є взаємна інформація (за Шенноном). Безумовну взаємну інформацію позначимо $\mathrm{Info}(X,Y)$. Елементарний наслідок марковської властивості формулюється наступним чином. Якщо дана умовна незалежність X та Y за умови на Q, то буде

$$\mathrm{Info}(X,Y) \le \mathrm{Info}(X,Q)$$

та

$$\mathrm{Info}(X,Y) \le \mathrm{Info}(Y,Q).$$

Безумовну залежність також називають асоціацією. Наступне твердження випливає з властивості 2 та критерія d-сепарації.

Властивість 3. Якщо в МПГЗ-моделі всі ланцюги між змінними X та Y взаємно перетинаються на одній змінній

Q, то буде $\mathrm{Info}(X,Y) \le \mathrm{Info}(X,Q)$ й $\mathrm{Info}(X,Y) \le \mathrm{Info}(Q,Y)$.

З властивостей 2 та 3 випливає

Властивість 4. Якщо в МПГЗ-моделі для заданої пари асоційованих змінних (X, Y) немає жодної змінної Q такої, що

$$\mathrm{Info}(X,Q) \ge \mathrm{Info}(X,Y)$$

та

$$\mathrm{Info}(Q,Y) \ge \mathrm{Info}(X,Y),$$

то не існує такої змінної, через яку проходили б усі ланцюги між X та Y.

Властивість 5. Якщо в МПГЗ-моделі між змінними X та Y існує єдиний ланцюг, то для кожного ребра $Q — R$ на цьому ланцюзі буде

$$\mathrm{Info}(X,Y) \le \mathrm{Info}(Q,R).$$

Визначення 1. В МПГЗ-моделі асоціація між змінними X та Y називається *обманною нереберною*, якщо немає ребра $X — Y$ і якщо на кожному ланцюзі між X та Y існує щонайменше одне ребро $R — Z$ таке, що

$$\mathrm{Info}(R,Z) < \mathrm{Info}(X,Y).$$

Визначення 2. В МПГЗ-моделі асоціація між змінними X та Y називається *двійниковою*, якщо вона обманна нереберна і якщо не існує змінної Q, яка сепарує X та Y.

Кожна обманна нереберна асоціація між X та Y або є двійниковою, або утворена конкатенацією двійникової асоціації з оршляхом. (В другому випадку всі ланцюги між X та Y проходять через певну спільну вершину (вершини)).

Зрозуміло, що задача відтворення моделі з даних стає найпростішою, коли відомо, що генеративна модель належить класу лісів або полі-лісів. Для лісів є відомий алгоритм квадратичної складності, запропонований C. Chow та C. Liu [9, 10]. Принцип алгоритму Chow&Liu – встановлення ребер у порядку зменшення міцності залежностей. Збагативши цей алгоритм засобами орієнтації ребер, можна отримати завершений алгоритм відтворення полі-лісів залежностей з даних. (Варіант такого алгоритму – 'SpaPolyTree' [6]). Моделі в класі лісів або полі-лісів можна відтворювати також методами, основаними на тестах незалежності. Для цього потрібні тести тільки нульового та першого рангів. (Прототипи таких алгоритмів – процедури 'Branch' («Гілка») та 'PolyForesyn' [6]).

Отже, є принциповий розрив між двома групами методів відтворення структури моделі з даних. Маємо елементарні спеціалізовані методи для однозв'язаних структур (якими є полі-ліси); і маємо універсальні комбінаторні методи для всього класу оАОГ-моделей. Втім, як показано в [11], можна надати універсальному методу здатності автоматично розпізнавати той факт, що генеративна модель є полілісом, і в такому разі наближатися до спеціалізованих методів за обчислювальними витратами.

Задача реконструкції моделі в класі монопотокових графів залежностей потребує нестандартних рішень. Відтворювати МПГЗ за допомогою універсальних методів – вкрай нерозумно. Водночас елементарні методи не працюють, бо в МПГЗ існують цикли, а сепаратори можуть бути великорозмірними, як і в довільних оАОГ.

Запропоновано кілька високоспеціалізованих алгоритмів реконструкції МПГЗ, які базуються на особливостях монопотокових графів. Перший алгоритм реконструкції МПГЗ, описаний в [12], логічно простий і потребує малої кількості тестів. Питання про присутність будь-якого ребра $X — Y$ вирішується за допомогою двох тестів. 1) тест безумовної незалежності; 2) тест умовної незалежності X та Y, причому в умову включено всі змінні, окрім X та Y. Це призводить до тестів умовної незалежності високого рангу, які є ненадійними. Отже, і сам алгоритм має низьку надійність. Уявимо, що змінних багато і що в генеративній моделі ребра $X — Y$ немає, але взаємозалежність змінних X та Y – сильна. Тоді (навіть за використання вибірки даних реалістичного обсягу) тест умовної незалежності може помилково не виявити незалежність (з огляду на складний формат). Тож

буде поставлене помилкове ребро. Зазначимо, що сильна безумовна залежність може виникати, коли змінні пов'язані ланцюгом з двох ребер, або коли між змінними є двійникова асоціація.

Згодом з'ясовано, що для відтворення монопотокових моделей достатньо спиратися на тести нульового та першого рангу [13–15]. Методи серії «Генеалогія», зокрема, алгоритм «Генеалог-2», описаний в [16], та більш досконалий алгоритм «Генеалог-С» [6] – непридатні для стандартних обставин, бо потребують розпізнавання слабких безумовних залежностей, у тому числі тих, що утворені довгими ланцюгами [6, 17]. Метод CH1, описаний в [13], для кожної пари залежних змінних виконує всі можливі тести першого рангу. Це призводить до надто великої кількості тестів. Можна уникнути цього недоліку, якщо змінити головний принцип методу, а саме, перейти від пошуку сепаратора до використання провокації залежності [14, 15, 17]. Замість інтенсивного пошуку спростування ребра можна знаходити підтвердження присутності ребра.

Феномен провокованої залежності полягає у наступному [18]. Якщо змінні X та Z – взаємонезалежні, і вони разом впливають на змінну Y, то кондиціонування змінної Y зазвичай породжує залежність між X та Z. Тобто в результаті введення умови на змінну Y виникає умовна залежність між X та Z. Згідно результатів стохастичної симуляції дискретних моделей, сила (рівень) провокованої залежності в середньостатистичному сенсі приблизно дорівнює силі маргінальної (реберної) залежності між Y та змінною X або Z. На провоковану залежність припадає біля $1/3$ повної «родинної» інформації колізора [18]. Провокована залежність може колапсувати (не виникати) тільки в особливих випадках у дискретних моделях.

Позначимо як $\mathrm{Ind}(X,Y\,|\,\mathbf{S})$ факт умовної незалежності змінних X та Y за умови $\mathbf{S}$. Факт умовної залежності виражається як $\sim\mathrm{Ind}(X,Y\,|\,\mathbf{S})$. Безумовна незалежність та залежність позначаються відповідно $\mathrm{Ind}(X,Y)$ та $\sim\mathrm{Ind}(X,Y)$. Теоретично провокована залежність визначається як сполучення (паттерн) $\mathrm{Ind}(X,Z)$ & & $\sim\mathrm{Ind}(X,Z\,|\,Y)$. Оскільки в МПГЗ, згідно їх визначення, в кожному колізорі $X\to Y\leftarrow Z$ змінні X та Z – взаємонезалежні, то на базі кожного колізора має породжуватися провокована залежність.

В групі методів, що спираються на інструмент провокованих залежностей, найбільш ефективним можна вважати алгоритм «Proliferator-D» [6, 17]. За задумом, цей алгоритм оминає тести сепарації, але виконує тести провокації. Виведення структури МПГЗ-моделі виконується в два основні етапи. На першому етапі обчислюються всі парні асоціації та ідентифікуються всі провоковані залежності. Для уникнення зайвої роботи провокація залежності виконується тільки за наявності передумов – коли трійка змінних утворює квазі-колізорний паттерн. На другому етапі алгоритм конструює структуру моделі, спираючись, зокрема, на принцип Chow&Liu. При цьому (за достатніх підстав) відразу встановлюються орієнтовані ребра. Хоча алгоритм «Proliferator-D» переважає своїх попередників, ретельний аналіз показує, що він теж недосконалий й часто виконує необов'язкові обчислення. «Proliferator-D» є алгоритмом субкубічної складності. Але в багатьох нетривіальних випадках МПГЗ-модель можна відтворити, виконавши роботу лінійної складності. Потужним резервом підсилення методів відтворення структури моделі є правила мінімальної сепарації [6, 8]. Більшість цих правил виведено для всього класу оАОГ-моделей, але декілька створено спеціально для МПГЗ-моделей [6]. Правила мінімальної сепарації обґрунтовані в апараті графів, а для обґрунтування емпіричних еквівалентів («зліпків») цих правил необхідні відповідні форми припущення каузальної неоманливості [6]. Ті форми припущення неоманливості можна оцінити як недостатньо надійні, тож й аргументація ефективності правил мінімальної сепарації може сприйматися як непереконлива. В даній роботі пропонується більш переконливий шлях гаран-

тування надійності відтворення моделі з даних. Спочатку формулюються найбільш надійні припущення про емпіричні (статистично-ймовірнісні) прояви залежностей, а потім на базі цих припущень будуються емпіричні резолюції та засоби для методів відтворення МПГЗ.

Базові припущення та інструментарій

Статистична залежність може бути слабкою. У ході аналізу реалістичних даних статистичний тест може відрізнити залежність від незалежності тільки якщо емпірична залежність досить міцна (велика). Метод відтворення моделі з даних може спиратися тільки на статистично значущі залежності. Позначатимемо предикатом $\mathrm{Dep}(X,Y)$ факт *значущої безумовної залежності* між X та Y. (Домовимося, що предикат $\mathrm{Dep}(X,X)$ завжди істинний).

Квазі-колізорна трійка змінних – це паттерн $\mathrm{Dep}(X,Y)\,\&\,\mathrm{Dep}(Z,Y)\,\&\,\mathrm{Ind}(X,Z)$.

Емпірична провокована залежність – це сполучення $\sim\mathrm{Dep}(X,Z)\,\&\,\mathrm{Dep}(X,Z\,|\,Y)$.

Обчислювальна складність алгоритму 'Proliferator-D' в основному визначається кількістю тестів провокації залежності, а значить, кількістю квазі-колізорних трійок. На рис. 1 показано приклади структур МПГЗ, які породжують велику кількість квазі-колізорних трійок. В першій структурі на рис. 1, *а* теоретично маємо $(n-1)(n-2)/2$ квазі-колізорних трійок. В другій структурі на рис. 1, *б* ця кількість менша. В третій структурі на рис. 1, *в* кількість квазі-колізорних трійок дорівнює $(n-2)(n-3)$. І хоча теоретично кількість квазі-колізорних трійок не може бути вище за квадратичну від кількості змінних, у багатьох випадках 'Proliferator-D' буде виконувати зайву роботу.

Дійсно, модель, показана на рис. 1, *а*, – дуже простий випадок полі-лісу й може бути тривіально відтворена на основі тестів нульового рангу та логіки. Інші моделі, показані на цьому рисунку теж реконструюються просто.

Бажано базувати метод відтворення моделі з даних на надійних припущеннях про емпіричні прояви залежностей [1, 3, 6]. Мабуть, найбільш надійним припущенням, яке можна сформулювати, є припущення *безумовної (маргінальної) реберної неоманливості* (0-РНО): будь-які дві змінні, поєднані ребром, є безумовно залежні. Формально: якщо існує (X — Y), то буде $\mathrm{Dep}(X,Y)$.

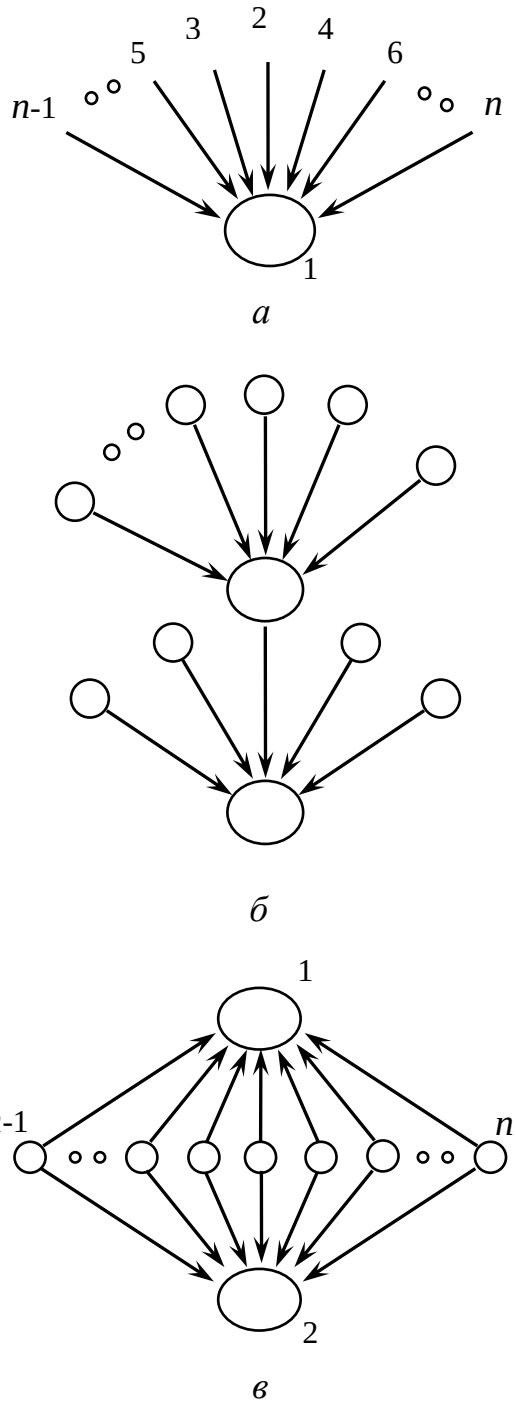


Рис. 1. Приклади МПГЗ:
а, б – полі-ліси; *в* – МПГЗ з циклами

Для МПГЗ-моделей це припущення – більш надійне, ніж для загального випадку оАОГ-моделей, тому що в оАОГ можуть бути одночасно присутні ребро $X — Y$ і ще якісь ланцюги між вершинами X та Y. (Тоді може статися взаємна компенсація залежностей). В МПГЗ таке виключено.

Припущення 0-РНО є необхідним та може виявитися недостатнім для ідентифікації всіх ребер. Тоді знадобиться залучити дещо складніший варіант. Припущення *реберної неоманливості першого рангу* (1-РНО): будь-які дві змінні, поєднані ребром, є умовно залежні, з умовою на довільну третю змінну.

Можна передбачити, що надійність припущення 1-РНО також буде вище для МПГЗ-моделей, ніж для загального випадку оАОГ-моделей, бо в оАОГ існує більше можливостей для контр-прикладів. Ризик порушення 1-РНО є досить високий, зокрема, у трикутнику $X \to Y \to$ $\to Z \leftarrow X$, де кондиціонування змінної Z створює провоковану залежність, яка може анігілювати з залежністю, створеною ребром $X — Y$.

Для обґрунтування здатності ідентифікувати спрямування (орієнтацію) ребер потрібні додаткові припущення. Для класу МПГЗ-моделей таким може бути одне з двох наступних припущень.

Припущення *безумовної каузальної неоманливості двох-реберних ланцюгів* (0-Л2НО): якщо в моделі між змінними X та Z присутній якийсь ланцюг довжиною у два ребра, то буде безумовна залежність $\mathrm{Dep}(X, Z)$. Іншими словами, якщо існує ланцюг $X \to Y \to Z$ або $X \leftarrow Y \to Z$, то він забезпечує значущу транзитну залежність між X та Z.

Припущення *елементарної провокації залежності на колізорі* (ЕПЗК): якщо в моделі присутній колізор $X \to Y \leftarrow Z$, то буде умовна залежність $\mathrm{Dep}(X, Z \mid Y)$. Для МПГЗ-моделей припущення ЕПЗК – більш надійне, ніж для загального випадку оАОГ-моделей, тому що для колізора $X \to Y \leftarrow Z$ в складі МПГЗ-моделі змінні X та Z обов'язково взаємонезалежні (а в оАОГ – не обов'язково).

(Зауважимо, що для орієнтації ребер в оАОГ-моделях цих припущень недостатньо).

Доцільно також зафіксувати варіант припущення, спрощений відносно 0-Л2НО. Припущення *безумовної каузальної неоманливості елементарних двох-реберних ланцюгів* (0-ЕЛ2НО): якщо в моделі між змінними X та Z присутній один й тільки один ланцюг довжиною у два ребра, то буде безумовна залежність $\mathrm{Dep}(X, Z)$. Можна очікувати, що це припущення – більш надійне, ніж 0-Л2НО, оскільки воно уникає ситуацій з певними ускладненнями.

Варто зазначити, що усі методи відтворення моделі з даних постулюють (за замовчуванням) наступне припущення: будь-які дві змінні, не поєднані жодним ланцюгом (ребром), тестуються як безумовно незалежні. (Звісно, статистичні тести мають ризики помилок, оцінювані значенням p-value).

Визначення 3. *Набір близьких до змінної* X визначається як $\mathrm{Clo}(X) = \{Y \mid \mathrm{Dep}(X, Y)\}$, тобто як множина всіх тих змінних, які безумовно залежать від X (включаючи саму X).

Тривіально $\mathrm{Clo}(X) \supseteq \{X\}$. Згідно припущення 0-РНО, якщо існує $X — Y$, то $Y \in \mathrm{Clo}(X)$. Якщо вершина X дотична до k ребер, то $|\mathrm{Clo}(X)| \geq (k+1)$.

Висока ефективність згаданих вище алгоритмів відтворення лісів та полілісів завдячує принципу Chow&Liu (який повторює відомий алгоритм Крускала). Коректність принципу Chow&Liu для полі-лісів забезпечується відсутністю циклів у структурі. На жаль, цей принцип не є коректним для МПГЗ-моделей в загальному випадку (бо в МПГЗ може бути кілька паралельних ланцюгів). Встановлення ребер в МПГЗ-моделі за принципом Chow&Liu призводить до помилок у певних ситуаціях [6, 7]. Виконання алгоритму Chow&Liu призведе до встановлення ребра на місці двійникової чи обманної нереберної асоціації (а певне автентичне ребро буде втрачене) [6, 7, 17]. Тому для використання принципу Chow&Liu необ-

хідно озброїти метод відтворення МПГЗ-моделей засобами розпізнавання двійникових асоціацій. Далі буде показано, як можна ідентифікувати багато ребер у МПГЗ-моделях на основі систематичного аналізу сукупності безумовних залежностей. Коректність пропонованих засобів ґрунтується на найбільш надійному припущенні – безумовної (маргінальної) реберної неоманливості (0-РНО). Для коректності резолюцій, які вдаються до принципу Chow&Liu, потрібне також припущення *монотонності залежності на марковському ланцюзі*. Формулювання: якщо маємо $\mathrm{Dep}(X,Y)$, $\mathrm{Dep}(Y,Z)$ та $\mathrm{Ind}(X,Z\,|\,Y)$, то чинне

$$\mathrm{Info}(X,Z) \le \min\{\mathrm{Info}(X,Y),\mathrm{Info}(Y,Z)\}.$$

Останнє припущення та припущення 0-РНО не поглинають один одного. (Властивість 3 треба розуміти в асимптотичному сенсі).

Емпіричні резолюції ідентифікації ребер в МПГЗ-моделях

З визначення двійникової асоціації та властивості 4 випливає наступний факт.

Факт 1. Якщо для МПГЗ-моделі асоціація між змінними X та Y – «двійникова», то $|\mathrm{Clo}(X)|\ge 4$ та $|\mathrm{Clo}(Y)|\ge 4$.

З властивостей 1 та 5 випливає наступний факт.

Факт 2. Якщо в МПГЗ-моделі асоціація між змінними X та Y – двійникова, то між вершинами X та Y існує не менше двох ланцюгів λ та μ таких, що кожна некінцева змінні на ланцюзі λ незалежна від кожної некінцевої змінної на ланцюзі μ.

Наслідок. Якщо для МПГЗ-моделі маємо $Y\in\mathrm{Clo}(X)$ і серед змінних, що входять до $\mathrm{Clo}(X)$, або серед змінних, що входять до $\mathrm{Clo}(Y)$, немає принаймні двох взаємонезалежних, то асоціація між X та Y не може бути двійниковою.

Якщо спиратися виключно на припущення 0-РНО, то серед усіх виведених [6, 8] правил мінімальної сепарації можна транслювати в емпіричну форму тільки одне. Це правило єдиного родича, або безальтернативного ребра [6] (воно чинне для всього класу оАОГ-моделей). В емпіричній формі воно формулюється наступним чином.

Резолюція єдиного близького. Якщо в оАОГ-моделі маємо $\mathrm{Clo}(X)=\{X,Y\}$, то існує ребро $X\,—\,Y$.

Для класу МПГЗ-моделей є чинним більш сильний результат.

Резолюція перемички (відсутність спільної суміжної). Нехай для МПГЗ-моделі маємо $Y\in\mathrm{Clo}(X)$ та $\big(\mathrm{Clo}(X)\setminus\{X\}\big)\cap\big(\mathrm{Clo}(Y)\setminus\{Y\}\big)=\varnothing$. Тоді, якщо $|\mathrm{Clo}(X)|\le 3$ або $|\mathrm{Clo}(Y)|\le 3$, то в моделі присутнє ребро $X\,—\,Y$. Інакше (тобто якщо $|\mathrm{Clo}(X)|>3$ та $|\mathrm{Clo}(Y)|>3$) є дві можливі альтернативи:

а) в моделі присутнє ребро $X\,—\,Y$, або

б) всі шляхи між X та Y є ланцюгами, з тим, що кожний ланцюг між X та Y має не менше трьох ребер, і існує не менше двох таких ланцюгів.

Ясно, що альтернатива «б» імплікує існування двійникової асоціації.

З умови «$Y\in\mathrm{Clo}(X)$ та $\big(\mathrm{Clo}(X)\setminus\{X\}\big)\cap\big(\mathrm{Clo}(Y)\setminus\{Y\}\big)=\varnothing$» випливає (властивість 3 та 4), що не існує такої змінної Z, що всі ланцюги між X та Y перетинаються на Z. Якщо немає ребра $X\,—\,Y$, то між X та Y існує не менше двох ланцюгів (властивість 5). Якби якийсь ланцюг між X та Y мав два ребра $X-Q-Y$, то згідно припущення 0-РНО було б $Q\in\mathrm{Clo}(X)$ та $Q\in\mathrm{Clo}(Y)$.

Резолюція кінцевого двох-реберного ланцюга. Нехай для МПГЗ-моделі маємо $\mathrm{Clo}(X)=\mathrm{Clo}(Y)=\{X,Y,Z\}$. Тоді змінні X,Y,Z поєднані в генеративній моделі двома ребрами, які утворюють ланцюг й ідентифікуються згідно принципу Chow&Liu.

Доведення. Якби якісь дві змінні з трьох X,Y,Z були поєднані через ланцюг, який проходить через якусь четверту

змінну W, то було б $W \in \mathrm{Clo}(X)$ або $W \in \mathrm{Clo}(Y)$, що суперечить умовам. Якби змінні X, Y, Z були поєднані як колізор, то дві з них були б взаємонезалежні, що суперечить умовам. Оскільки трикутник ребер в МПГЗ неможливий, отримуємо бажаний висновок. (Вказані два ребра ідентифікуються згідно властивості 5).

Зауважимо, що в такій конфігурації одна з трьох змінних (в даному разі Z) може мати «додаткові» близькі змінні, тобто можливе $|\mathrm{Clo}(Z)| > 3$.

Об'єднуючи властивість 3 та наслідок з факту 2, отримуємо результат.

Резолюція виключення нереберної асоціації. Нехай для МПГЗ-моделі маємо $Y \in \mathrm{Clo}(X)$, і серед змінних, що входять до $\mathrm{Clo}(X) \setminus \{X, Y\}$, або серед змінних, що входять до $\mathrm{Clo}(Y) \setminus \{X, Y\}$, немає жодної пари взаємонезалежних. Тоді, якщо серед змінних, що входять до $\mathrm{Clo}(Y) \setminus \{X, Y\}$, немає хоча б одної змінної Q, такої, що $\mathrm{Info}(X; Q) \geq \mathrm{Info}(X; Y)$ та $\mathrm{Info}(Q, Y) \geq \mathrm{Info}(X, Y)$, то присутнє ребро $X — Y$.

Об'єднавши резолюцію перемички та резолюцію виключення обманної нереберної асоціації, отримаємо більш сильний результат.

Резолюція перемички (об'єднана). Нехай для МПГЗ-моделі маємо $Y \in \mathrm{Clo}(X)$ і виконується $(\mathrm{Clo}(X) \setminus \{X\}) \cap \cap (\mathrm{Clo}(Y) \setminus \{Y\}) = \varnothing$. Для ідентифікації присутності ребра $X — Y$ в моделі достатнім є виконання принаймні однієї з наступних умов:

а) маємо $|\mathrm{Clo}(Y)| \leq 3$ або $|\mathrm{Clo}(Y)| \leq 3$;

б) серед змінних, що входять до $\mathrm{Clo}(X)$ немає жодних двох взаємонезалежних;

в) серед змінних, що входять до $\mathrm{Clo}(Y)$, немає жодних двох взаємонезалежних.

Ідея доведення. Якби асоціація між X та Y забезпечувалася єдиним ланцюгом або була обманною нереберною (але не двійниковою) асоціацією, то існував 1-сепаратор для пари (X, Y), а тоді, не виконувалася б передумова правила. Додаткові умови («а», «б, «в») резолюції виключають існування двійникової асоціації.

Резолюція слабкої пари ребер. Якщо для МПГЗ-моделі маємо $\mathrm{Clo}(Y) = \{X, Y, Z\}$ й $Z \notin \mathrm{Clo}(X)$, то в моделі присутня пара ребер $X — Y — Z$.

Доведення. Припустимо від протилежного, що ребра $X — Y$ немає. Тоді, з огляду на умову $\mathrm{Clo}(Y) = \{X, Y, Z\}$, суміжною до змінної Y буде тільки одна змінна Z. Отже, змінні X та Y мають поєднуватися ланцюгами (ланцюгом), які проходять через ребро $Z — Y$. Тоді, згідно властивості 3, буде $\mathrm{Info}(X, Y) \leq \mathrm{Info}(X, Z)$. А це разом з фактом $Y \in \mathrm{Clo}(X)$ тягне $Z \in \mathrm{Clo}(X)$, що суперечить умовам. З огляду на симетрію, доведення для другого ребра аналогічне.

Резолюцію слабкої пари ребер можна узагальнити.

Резолюція центральної вершини (*єдиного вузла*). Нехай для МПГЗ-моделі всі змінні, що входять до складу множини $\mathrm{Clo}(Q) \setminus \{Q\}$, є взаємонезалежними. Тоді, якщо $|\mathrm{Clo}(Q)| = 3$ або серед змінних множини $\mathrm{Clo}(Q) \setminus \{Q\}$ немає жодної такої Z, що $|\mathrm{Clo}(Z)| \geq 4$, то в моделі присутні ребра $Q — X$ для всіх змінних $X \in \mathrm{Clo}(Q) \setminus \{Q\}$.

Доведення. У випадку $|\mathrm{Clo}(Q)| = 3$ отримуємо резолюцію слабкої пари ребер. Для іншого випадку припустимо від протилежного, що є змінна W, $W \in \mathrm{Clo}(Q) \setminus \{Q\}$ несуміжна до Q. Тоді асоціація між Q та W була б двійникова (інші варіанти нереберної асоціації відпадають, бо потребують існування деякої спільної близької до Q та W змінної, що суперечить умовам згідно властивості 3). Але для можливості вказаної двійникової асоціації (згідно факту 1) необхідно $|\mathrm{Clo}(W)| \geq 4$. Суперечить умовам.

Резолюція крайнього трикутника асоціацій. Нехай для МПГЗ-моделі маємо

$\mathrm{Clo}(X)=\{X,Y,Z\}$, $\mathrm{Clo}(Y)\supset\{X,Y,Z\}$ та $\mathrm{Clo}(Z)\supset\{X,Y,Z\}$ (а асоціація між Y та Z може задовольняти необхідним вимогам обманної нереберної асоціації). Тоді чинне:

а) якщо $\mathrm{Info}(X;Y)>\mathrm{Info}(X;Z)>$ $>\mathrm{Info}(Y;Z)$, то в моделі присутній ланцюг $Y—X—Z$;

б) якщо $\mathrm{Info}(X;Y)>\mathrm{Info}(Y;Z)>$ $>\mathrm{Info}(X;Z)$ або $\mathrm{Info}(Y;Z)>\mathrm{Info}(X;Y)>$ $>\mathrm{Info}(X;Z)$, то в моделі присутнє ребро $X—Y$, а питання щодо другого ребра ланцюга залишається відкритим.

Нагадаємо, що *кліка* асоційованих змінних – це така множина змінних $\mathbf{C}$, що для кожних двох її елементів $X,Y\in\mathbf{C}$ чинне $Y\in\mathrm{Clo}(X)$. *Ізольована кліка* – це така множина змінних $\mathbf{C}$, що для кожної $X\in\mathbf{C}$ чинне $\mathrm{Clo}(X)=\mathbf{C}$.

Резолюція ізольованої кліки. Якщо для МПГЗ-моделі деяка підмножина змінних $\mathbf{C}$ утворює ізольовану кліку, то в генеративній моделі підмножина змінних $\mathbf{C}$ поєднана деревом ребер, і це дерево повністю відтворюється за принципом Chow&Liu.

Дійсно, якби якісь три змінні утворювали б колізор, то була б пара взаємно незалежних змінних. МПГЗ без колізорів – це дерево (ліс).

Можна розширити резолюцію ізольованої кліки (і охопити резолюцію кінцевого двох-реберного ланцюга як спеціальний випадок). Введемо допоміжні поняття.

Граф парних асоціацій (ГПА) для даних (генерованих з моделі з набором змінних $\mathbf{V}$) – це неорієнтований граф з вершинами $\mathbf{V}$, в якому встановлене ребро $X—Y$ для кожної пари змінних (X, Y), для якої чинне $\mathrm{Dep}(X,Y)$, й немає ребра $Q—Z$ для кожної пари змінних (Q, Z), якщо чинне $\sim\mathrm{Dep}(X,Y)$.

Факт 3. Якщо в МПГЗ-моделі асоціація між змінними X та Y – обманна нереберна, то в ГПА існує цикл, який проходить через X та Y і який не входить у склад жодної кліки.

Нехай дві змінні X та Y поєднані в МПГЗ двома чи більше ланцюгами і входять до деякої компоненти ГПА $\mathbf{K}$. Тоді ясно, що за припущення 0-РНО компонента $\mathbf{K}$ не може бути клікою.

Максимальна кліка змінних – це така кліка $\mathbf{C}$, що не існує жодної змінної $X\notin\mathbf{C}$ такої, що $\mathbf{C}\cup\{X\}$ теж є клікою змінних.

Резолюція кліки асоціацій у деревовидному оточенні. Нехай для МПГЗ-моделі маємо в ГПА максимальну кліку $\mathbf{C}$. Далі, нехай для кожної пари змінних X,Y з кліки $\mathbf{C}$ чинне наступне: між змінними X та Y немає жодного шляху в ГПА поза клікою $\mathbf{C}$. Тоді в генеративній моделі всі змінні кліки $\mathbf{C}$ з'єднані деревом, яке можна ідентифікувати за принципом Chow&Liu.

Умови резолюції формалізуються так: $\mathbf{C}=\{X,Y,Z,..\}$; для всіх $X\in\mathbf{C}$ маємо $\mathrm{Clo}(X)\supseteq\mathbf{C}$; для будь-якої пари $X,Y\in\mathbf{C}$ між X та Y *не існує* жодної послідовності $Z_1\in\mathrm{Clo}(X)$, $Z_2\in\mathrm{Clo}(Z_1)$, ... $,Z_k\in\mathrm{Clo}(Z_{k-1})$, …, $Y\in\mathrm{Clo}(Z_n)$, де $Z_k\notin\mathbf{C}$.

Дійсно, якби між якимись змінними $X,Y\in\mathbf{C}$ в генеративній моделі існувало б два чи більше ланцюгів, то, згідно властивості 1, ці ланцюги не входили б в жодну кліку. Тоді був би шлях в ГПА між X та Y поза клікою $\mathbf{C}$. Отже, в генеративній моделі в рамках множини $\mathbf{C}$ немає циклів, тобто всі змінні кліки $\mathbf{C}$ поєднані в генеративній моделі деревом. (Вимога, щоб кліка $\mathbf{C}$ була максимальною, є технічною. Якщо кліка не максимальна, неможливо виконати решту вимог резолюції).

Можна зауважити, що більшість наведених резолюцій автоматично виконується алгоритмом Chow&Liu. Проте ці резолюції гарантують коректну реконструкцію ребер (а алгоритм Chow&Liu – ні).

Пропоновані резолюції значно прискорюють реконструкцію моделей, структура яких показана на рис. 1. Для випадку рис. 1, а всі ребра ідентифікуються резолюцією єдиного близького. Ця са-

ма резолюція дозволяє відтворити ребра «нижнього ярусу» в структурі рис. 1, б (крім одного ребра). Всі ребра «верхнього ярусу» цієї структури відтворюються за допомогою резолюції крайнього трикутника асоціацій. Припустимо, що в моделі із структурою рис. 1, в асоціація між змінними $X1$ та $X2$ – значуща. Досить, щоб для однієї трійки змінних виконалися умови п. «а» резолюції крайнього трикутника асоціацій, і тоді всі ребра швидко ідентифікуються.

Експериментальне випробування

Продемонструємо ефективність роботи запропонованих резолюцій на прикладі. Структура генеративної моделі (в класі МПГЗ) для експерименту показана на рис. 2. Модель має 32 змінні та 37 ребер. Всі змінні – тризначні. Для цієї структури було генеровано (стохастичним механізмом) три варіанти параметризації, і для кожної параметризації генеровано кілька вибірок даних обсягом 500, 1000, 2000, 3000 та 5000 записів. Тестування безумовної незалежності виконувалось за спрощеною схемою – як порівняння взаємної інформації з пороговим значенням. Попередній експеримент проведено для перевірки надійності тестів та припущення 0-РНО. Для цього фіксувалися помилкові рішення двох типів:

1) ідентифікація залежності змінних, які згідно генеративної моделі є взаємонезалежними;

2) ідентифікація незалежності змінних, поєднаних ребром в генеративній моделі. Результати цього експерименту наведено в табл. 1.

Підсумуємо результати експерименту. В середньому для вибірки розміром 500 записів і величини порогу 0,01 ризик втрати ребра досяг 13 %, а кількість помилкових залежностей дорівнює 11. Аналогічні показники для величини порогу 0,02 становлять 28 % та 0,17 частки залежності. Коли вибірка має 1000 записів або більше, не з’явилося жодної помилкової залежності. Для вибірки розміром

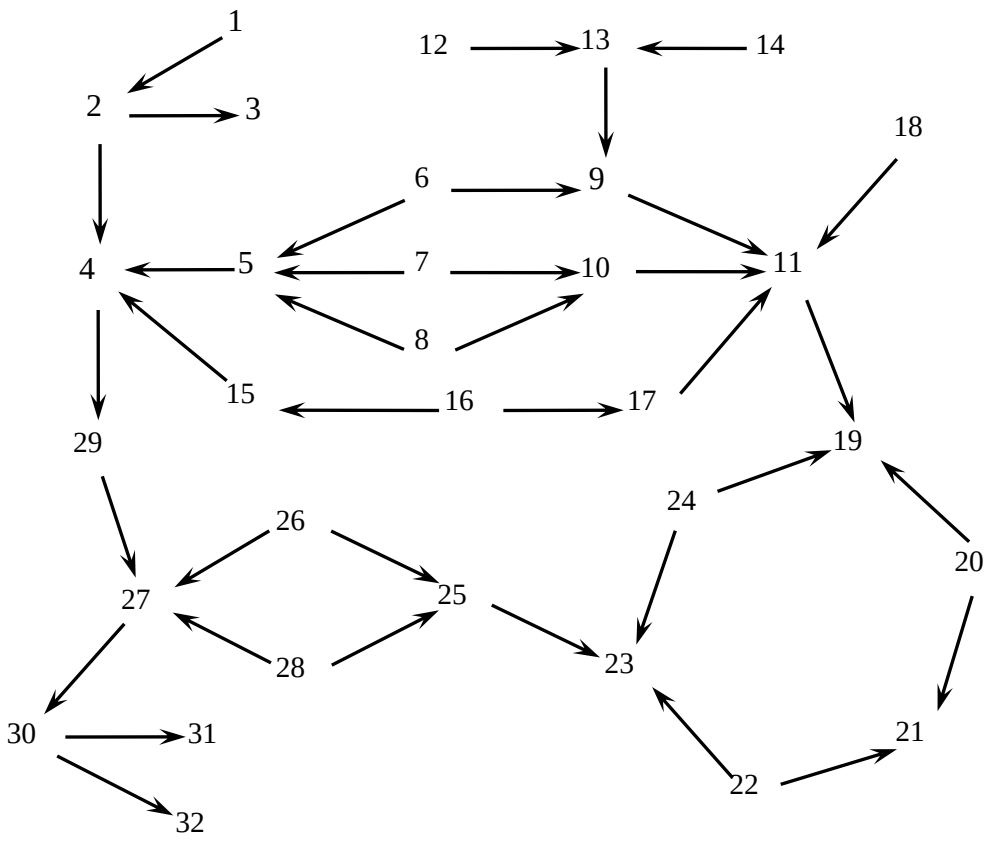


Рис. 2. Структура МПГЗ (‘MF32-37’) для експерименту

Таблиця 1

Емпіричні залежності для ребер моделі

Варіант параметризації	Розмір вибірки	Поріг значущості = 0,01		Поріг значущості = 0,02	
		Кількість втрачених ребер	Кількість помилкових залежностей	Кількість втрачених ребер	Кількість помилкових залежностей
1-й	500	4	12	8	0
	500	5	13	8	0
	500	3	10	8	0
	500	4	9	10	0
	1000	4	0	7	0
	1000	3	0	9	0
	2000	3	0	8	0
	3000	1	0	9	0
	5000	1	0	7	0
2-й	500	3	8	6	0
	500	4	2	9	0
	500	5	14	10	0
	500	5	10	9	0
	1000	4	0	10	0
	1000	5	0	11	0
	2000	4	0	10	0
	3000	3	0	6	0
	5000	2	0	6	0
3-й	500	6	11	13	0
	500	7	18	15	0
	500	6	24	15	1
	500	8	6	15	1
	1000	7	0	13	0
	1000	7	0	13	0
	2000	7	0	15	0
	3000	7	0	14	0
	5000	6	0	13	0

1000 записів ризик втрати ребра становить 13 % (для порогу 0,01) та 28 % (для порогу 0,02).

Для вибірок розміром від 2000 до 5000 записів середній ризик втрати ребра склав 10 % (для порогу 0,01) та 26 % (для порогу 0,02). Той факт, що при зростанні розміру вибірки показники втрат ребер зменшуються слабо, вказує на значну кількість майже маскованих та деградованих ребер.

Відомо, що завдяки співвідношенням параметрів моделі певне ребро генеративної моделі може перетворитися на

масковане [18] або навіть деградувати (тобто абсолютно втратити вплив). На користь таких ситуацій у нашому експерименті свідчить той факт, що для різних вибірок даних повторюються одні й ті самі пропуски ребер. Зокрема, для 2-го варіанта параметризації в результатах виведення з усіх дев’яти вибірок (з обома значеннями порогу значущості) пропущено ребра 7—5 та 24—19. А для третього варіанту параметризації в результатах виведення з усіх дев’яти вибірок даних (коли використано поріг 0,02) пропущено вісім ребер. Не всі зафіксовані пропуски ребра можна вважати справжніми помилками. Пропуск деградованого ребра не є помилкою. А що стосується синдрому маскованих ребер, то відомі методи відтворення моделей (основані на незалежності) також не захищені від нього.

Головний експеримент випробував роботу запропонованих резолюцій. Для цього експерименту використано вибірки розміром 2000 та поріг значущості 0,01. Оскільки деякі вищенаведені резолюції взаємно поглинаються, було реалізовано тільки чотири резолюції. Кожна з обраних резолюцій – єдиного близького, перемички (об’єднана), центральної вершини (єдиного вузла) та кліки асоціацій у деревовидному оточенні, – застосовувалась автономно, як на початку процесу реконструкції. Тривалість застосування чотирьох резолюцій до вибірки даних складала біля третини секунди в середовищі MATLAB. (Потенційно складною задачею міг стати пошук максимальних клік, але вдалося уникнути важкості відомої абстрактної задачі через те, що процедура відразу шукає кліки, які задовольняють всім вимогам резолюції).

Результати головного експерименту наведено в табл. 2. Зафіксовані в таблиці помилки чотирьох резолюцій насправді відображають одну помилку, яка сталася внаслідок того, що транзитна залежність двох-реберного ланцюга перевищила залежність одного з ребер того ланцюга. Якщо розглянути 1-й варіант параметризації, то чотири резолюції у підсумку відтворили 24 ребра (без помилок), що становить дві третини ребер моделі. Для 2-го варіанта параметризації ці резолюції відтворили 18 ребер без помилок. Для 3-го варіанта параметризації правильно відтворено 26 ребер та вставлено одне помилкове ребро.

Таблиця 2

Ефективність застосування емпіричних резолюцій відтворення ребер

Резолюція	Варіант параметризації	Правильно відтворених ребер	Помилкових ребер
Р. єдиного близького	1	16	0
	2	16	0
	3	19	1
Р. перемички (об’єднана)	1	19	0
	2	10	0
	3	19	1
Р. центральної вершини (єдиного вузла)	1	16	0
	2	7	0
	3	18	1
Р. кліки асоціацій у деревовидному оточенні	1	6	0
	2	5	0
	3	16	1

Висновки

Показано, яким чином (апріорі знаючи тільки приналежність моделі до класу монопотокових структур) можна ідентифікувати більшість ймовірнісних зв’язків (безпосередніх залежностей), спираючись виключно на безумовні залежності двох змінних. Ідентифікація ребер (зв’язків) виконується емпіричними резолюціями, які строго обґрунтовані на базі припущення безумовної (маргінальної) реберної не-оманливості. Вже за своєю конструкцією це припущення є одним з найнадійнішим серед простих версій припущень каузальної неоманливості, відомих у галузі каузального виведення. Висока емпірична надійність цього припущення та побудованих на ньому резолюцій підтверджується результатами експерименту. Плануємо розвинути і перенести ідеї та принципи на загальний клас моделей із структурою ациклонних орграфів (байєсівських мереж).

1. *Spirtes P*. Introduction to causal inference. Journal of Machine Learning Research. 2010. Vol. 11, P. 1643−1662.
2. *Kalisch M., Bühlmann P*. Causal structure learning and inference: a selective review. Quality Technology & Quantitative Management. 2014. Vol. 11, N 1. P. 3–21.
3. *The TETRAD* Project: Constraint based aids to causal model specification / R. Scheines, P. Spirtes, C. Glymour et al. Multivariate Behavioral Research. 1998. Vol. 33, N 1. P. 65–118.
4. *Chickering D., Heckerman D., Meek C.* Large-sample learning of Bayesian networks is NP-hard. Journal of Machine Learning Research. 2004. Vol. 5. – P. 1287–1330.
5. *Балабанов О.С.* Відтворення каузальних мереж на основі аналізу марковських властивостей. Математичні машини та системи. 2016. № 1. С. 16–26.
6. *Балабанов О.С.* Каузальні мережі: аналіз, синтез та виведення з статистичних даних: дис. … доктора фіз.-мат. наук (01.05.01– теоретичні основи інформатики та кібернетики) / Балабанов Олександр Степанович. – К.: Ін-т кібернетики ім. В.М. Глушкова НАНУ, 2014. 305 с.
7. *Балабанов О.С.* Системи ймовірнісних залежностей: графові та статистичні властивості. Математичні машини та системи. 2009. № 3. С. 80–97.
8. *Балабанов О.С.* Правила підбору сепараторів в баєсівських мережах. Проблеми програмування. 2007. № 4. С. 33–43.
9. *Chow C.K., Liu C.N.* Approximating discrete probability distributions with dependence trees. IEEE trans. on Information Theory. 1968. Vol. 14, N 3. P.462–467.
10. *Балабанов О.С.* Індуктивне відтворення деревовидних структур систем залежностей. Проблемы программирования. 2001. № 1–2. С. 95–108.
11. *Балабанов О.С.* Прискорення алгоритмів відтворення байєсівських мереж. Адаптація до структур без циклів. Проблеми програмування. 2011. № 1. С. 63–69.
12. *Geiger D., Paz A., Pearl J.* Learning simple causal structures. Internat. Journal of Intelligent Systems. 1993. Vol. 8, N 2. P. 231–247.
13. *de Campos L.M., Huete J.F.* On the use of independence relationships for learning simplified belief networks. Intern. Journal of Intelligent Systems. 1997. Vol. 12. Issue 7. P. 495–522.
14. *Балабанов О.С.* Ефективний метод виявлення структур залежностей в статистичних даних. Проблемы программирования. 2004. № 2–3. С. 312–319.
15. *Балабанов А.С.* К выводу структур моделей вероятностных зависимостей из статистических данных. Кибернетика и системный анализ. 2005. № 5. С. 19–31.
16. *Балабанов А.С.* Индуктивный метод восстановления монопотоковых вероятностных графовых моделей зависимостей. Проблемы управления и информатики. 2003. № 5. С.75–84.
17. *Балабанов А.С.* Реконструкция модели вероятностных зависимостей по статистическим данным. Инструментарий и алгоритм. Проблемы управления и информатики. 2009. № 6. С. 90–103.
18. *Балабанов А.С.* Индуцированная зависимость, взаимодействие факторов и дискриминация каузальных структур. Кибернетика и системный анализ. 2016. № 1. С. 10–22.

References

1. Spirtes P. (2010). Introduction to causal inference. Journal of Machine Learning Research. 11, 1643−1662.
2. Kalisch M., Bühlmann P. (2014). Causal structure learning and inference: a selective review. Quality Technology & Quantitative Management. 11 (1), 3–21.
3. Scheines R., Spirtes P., Glymour C. et al. (1998). The TETRAD Project: Constraint based aids to causal model specification. Multivariate Behavioral Research. 33 (1), 65–118.
4. Chickering D., Heckerman D., Meek C. (2004). Large-sample learning of Bayesian networks is NP-hard. Journal of Machine Learning Research. 5, 1287–1330.
5. Balabanov O.S. (2016). Vidtvorennya kauzalnych merezh na osnovi analizu markovskich vlastyvostej [Reconstruction of causal networks via analysis of Markov properties]. Mathematical Machines and Systems. (1), 16–26. [In Ukrainian].
6. Balabanov O.S. (2014). 'Causal nets: analysis, synthesis and inference from statistical data', Doctor of math. sciences thesis, V.M. Glushkov Institute of Cybernetics, Kyiv, Ukraine. [In Ukrainian].
7. Balabanov O.S. (2009) Probabilistic dependency models: graphical and statistical properties. Mathematical Machines and Systems. (3), 80–97. [In Ukrainian].
8. Balabanov O.S. (2007). Rules for picking up separators in Bayesian networks. Problems in programming. (4), 33–43. [In Ukrainian].
9. Chow C.K., Liu C.N. (1968). Approximating discrete probability distributions with dependence trees. IEEE trans. on Information Theory. 14 (3), 462–467.
10. Balabanov O.S. (2001). Inductive recovery of structures of dependency trees. Problems in programming. (1–2), 95–108. [In Ukrainian].
11. Balabanov O.S. (2011). Accelerating algorithms for Bayesian networks recovery. Adaptation to structures without cycles. Problems in programming. (1), 63–69. [In Ukrainian].
12. Geiger D., Paz A., Pearl J. (1993). Learning simple causal structures. Internat. Journal of Intelligent Systems. 8 (2), 231–247.
13. de Campos L.M., Huete J.F. (1997). On the use of independence relationships for learning simplified belief networks. Intern. Journal of Intelligent Systems. 12 (7), 495–522.
14. Balabanov O.S. (2004). Efficient method for discovery of dependency structures in statistical data. Problems in programming. (2–3), 312–319. [In Russian].
15. Balabanov A.S. (2005). Inference of structures of models of probabilistic dependences from statistical data. Cybernetics and Systems Analysis. 41 (6), 808–817. – Springer, New York.
16. Balabanov A.S. (2003). Inductive reconstruction method for "mono-flow" probabilistic graphical models of Dependencies. Journal of Automation and Information Sciences. 35 (10), 1–8. – Begell House Publishers, Danbury.
17. Balabanov A.S. (2009). Reconstruction of the model of probabilistic dependences by statistical data. Tools and algorithm. J. of Automation and Information Sciences. 41 (12), 32–46.
18. Balabanov O.S. (2016). Induced dependence, factor interaction, and discriminating between causal structures. Cybernetics and Systems Analysis. 52 (1), 8–19.

Одержано 20.12.2016

Про автора:

Балабанов Олександр Степанович,
доктор фізико-математичних наук,
провідний науковий співробітник.
Кількість наукових публікацій в
українських виданнях – 50.
Кількість наукових публікацій в
зарубіжних виданнях – 9.
http://orcid.org/0000-0001-9141-9074.

***Місце роботи автора*:**

Інститут програмних систем
НАН України,
03187, м. Київ-187,
проспект Академіка Глушкова, 40.
Тел.: (044) 5263420.
E-mail: bas@isofts.kiev.ua

УДК 004.896 + 004.932.2

О.Л. Древаль, А.Ю. Дорошенко

ЗАСТОСУВАННЯ АЛГОРИТМУ ФАЗОВОЇ КОРЕЛЯЦІЇ ТА ЙОГО РОЗШИРЕННЯ В ОПТИЧНОМУ СТАБІЛІЗАТОРІ БЕЗПІЛОТНИХ ЛІТАЛЬНИХ АПАРАТІВ

В статті розглядається застосування алгоритму фазової кореляції як основи математичного апарата оптичного стабілізатора безпілотних літальних апаратів (БПЛА) для визначення зміщення корпусу відносно поверхні. Також розглянуто можливість застосування розширених варіантів алгоритму фазової кореляції з метою збільшення точності визначення зміщення та зменшення обчислювальних затрат. Подано код реалізації алгоритму мовою MATLAB та експериментальне дослідження можливості застосування реалізації алгоритму на мові С.

Ключові слова: БПЛА, оптична стабілізація, обробка зображень, фазова кореляція, субпіксельна реєстрація зображень

Вступ

З розвитком науково-технічного прогресу БПЛА набули широкої популярності у різноманітному класі задач. Застосування БПЛА для задач, що вимагають підвищеної точності позиціонування корпусу апарата та забезпечення його стабільного положення у просторі-часі вимагає комбінованого підходу до проблеми стабілізації, що передбачає застосування у комплексі інерційної навігаційної системи, магнітометричного обладнання, супутникових навігаційних систем – GPS, ГЛОНАСС та ін. [1].

Головним недоліком використання інерційної навігаційної системи (ІНС) як основної є неможливість компенсації руху БПЛА з постійною швидкістю, оскільки складові компоненти інерційної навігаційної — гіроскоп та акселерометр системи чутливі лише до зміни значення швидкості (прискорення). За умов ідеальної моделі (похибка інтегрування строго дорівнює 0, початкова швидкість БПЛА у просторі нульова) ІНС забезпечуватиме ідеальну стабілізацію положення апарата у просторі в будь-який момент часу. В реальних умовах при перетворенні «прискорення → швидкість» з часом збільшуватиметься значення усталеної похибки за швидкістю у просторі [2].

Класична модель керування БПЛА передбачає наявність оператора, в задачі якого входить корекція усталеної похибки інерційної системи, що накладає певні обмеження як на можливу область застосування БПЛА, так і на автономність системи в цілому. Задачі, в яких наявні вимоги до такого режиму експлуатації БПЛА, як автоматичне підтримання заданого положення/курсу у просторі без участі оператора протягом тривалого часу (наприклад, оператор в цей час керує підвісом видової телекамери, озброєнням БПЛА, або іншим обладнанням), ІНС виявляється недостатньо.

Логічним наслідком постановки задачі автономного керування БПЛА і найбільш широко розповсюдженим підходом до вирішення зазначеної проблеми є додаткове використання навігаційних систем – сигнал позиціювання з відповідного модуля використовується для компенсації дрейфу нуля головної інерційної навігаційної системи. Однак, використання глобальних супутникових навігаційних систем як додаткових систем стабілізації зазвичай обмежується допоміжною роллю за причиною їх низької точності. Створення автономної системи з використанням глобальних супутникових навігаційних систем можливе при наявності певних обмежень, таких як порівняно велика висота польоту БПЛА та відсутність перешкод, що дозволить фактично нівелювати проблему низької точності абсолютного позиціонування.

Застосування систем глобального позиціонування може додатково накладати

певні обмеження на використання апаратів у незахищеному оточенні, оскільки існують засоби для глушіння або підробки сигналів GPS/ГЛОНАСС та інших супутникових навігаційних систем, приклади наведено у роботі [3].

Оптичний стабілізатор

Для вирішення описаної у попередньому розділі проблеми забезпечення стабільного положення БПЛА з високою точністю відносно поверхні в ситуації, що унеможливлює точне отримання поточних координат у глобальній системі відліку, пропонується використати якісно інший підхід. Для отримання допоміжного зворотного зв'язку для стабілізації усталеного положення БПЛА пропонується застосування системи оптичної стабілізації, деталі конструкції якої описано у [4]. Спрощену схему показано на рис. 1. На підставі аналізу низки послідовних знімків відеокамери, цифровий процесор обробки сигналів оптичного стабілізатора вираховує підсумкові показники, що свідчать про направлення переміщення БПЛА у просторі, і передає результати своєї роботи у модуль ІНС, де вони використовуються для корекції показників швидкості та кутової швидкості (у випадку відсутності модуля магнітометрії).

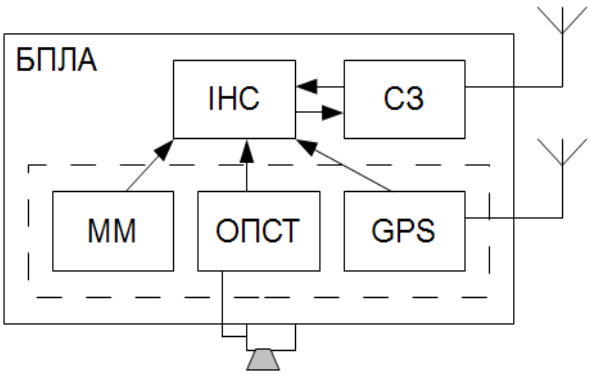


Рис. 1. Модифікована схема стабілізаційного апарата БПЛА

Задача, визначення зсуву зображення у даній системі є технологічно близькою до задачі визначення переміщення комп'ютерного оптичного маніпулятора «миша», де використовується схожий принцип роботи – після аналізу послідовності знімків поверхні (що представляють собою квадратну матрицю з пікселів різної яскравості), вираховуються підсумкові показники, що свідчать про направлення переміщення миші уздовж осей X і Y.

Аналогічний принцип використовується при обробці відео для програмної стабілізації небажаних рухів зображення, що виникають при здійсненні відеозапису з рухомого об'єкта.

Застосування алгоритму крос-кореляції

Показано, що для роботи оптичного стабілізатора необхідним є визначення зміщення зображення вздовж осей X та Y відносно попереднього зображення, необхідне застосування відповідного алгоритму. Швидким та широко розповсюдженим методом, що дозволяє виконати це завдання є фазова кореляція (запропоновано у [5]), що використовує перехід у частотну область для визначення взаємного паралельного зсуву двох однакових зображень. Математичний опис алгоритму знаходження фазової кореляції виглядає наступним чином.

Вихідні дані – зображення $\hat{g}_a$, $\hat{g}_b$, що отримані у попередній та поточній ітераціях відповідно, представлені у вигляді квадратних матриць розмірності $N \times N$.

На першому кроці до зображень застосовується віконна функція (наприклад, Вікно Геммінга, вікно Ханна, вікно Гаусса та ін.) на обох зображеннях, щоб зменшити вплив крайових ефектів [6].

Вибір віконної функції визначається з характеру вихідного зображення та обчислювальних можливостей цільового устаткування.

Для прикладу, коефіцієнти вікна Ханна обраховуються за наступною формулою:

$$w(n) = \frac{1}{2}\left(1 - \cos\left(\frac{2\pi n}{N-1}\right)\right), \qquad (1)$$

де N – розмір матриці зображення.

Для застосування віконної функції до двовимірної матриці необхідно над

результатом (1) виконати перетворення [7]:

$$w_{2D} = w \times w^T . \quad (2)$$

Застосовуючи результат (2) на вихідні зображення, маємо:

$$\hat{g}_a = g_a \circ w_{2D},$$

$$\hat{g}_b = g_b \circ w_{2D}, \quad (3)$$

де $\circ$ – покомпонентний добуток (добуток Адамара).

На другому кроці до зображень $\hat{g}_a$, $\hat{g}_b$ застосовується дискретне двовимірне перетворення Фур'є:

$$G_a = F\{\hat{g}_a\},$$

$$G_b = F\{\hat{g}_b\}. \quad (4)$$

Беручи спряжені числа з (4), перемножуючи перетворення Фур'є один з одним поелементно, і нормалізуючи цей добуток поелементно розраховується взаємна спектральна щільність:

$$R = \frac{G_a \circ G_b^*}{\left| G_a \circ G_b^* \right|}. \quad (5)$$

Нормалізована крос-кореляція отримана застосовуючи зворотне перетворення Фур'є до (5):

$$r = F^{-1}\{R\}. \quad (6)$$

Для визначення зміщення вихідного зображення необхідно знайти положення піку нормалізованої крос-кореляції:

$$(\Delta x, \Delta y) = \arg\max_{(x,y)}\{r\}. \quad (7)$$

Результатом (7) є значення зміщення у дискретних елементах зображення (пікселі) g_b відносно g_a. Знаючи висоту та характер місцевості, неважко представити Δx, Δy у вигляді зміщення на іншу величину (наприклад, метри). У загальному випадку в цьому перетворенні немає потреби, оскільки для здійснення корекції ІНС БПЛА непотрібні точні абсолютні значення зміщення – необхідно звести рух об'єкта до нуля, що свідчитиме про набуття ним сталого положення.

Вищеописаний математичний алгоритм описується наступним кодом на мові MATLAB:

```
wf = 'hann';
g_aw = wfn(g_a, wf);
g_bw = wfn(g_b, wf);
[x1, y1] = size(g_aw);
[x2, y2] = size(g_bw);
G_a = fft2(g_aw);
G_b = fft2(g_bw);
G_b_ = conj(G_b);
R=(G_a .* G_b_) ./ abs(G_a .* G_b_);
r=abs(ifft2(R)).
```

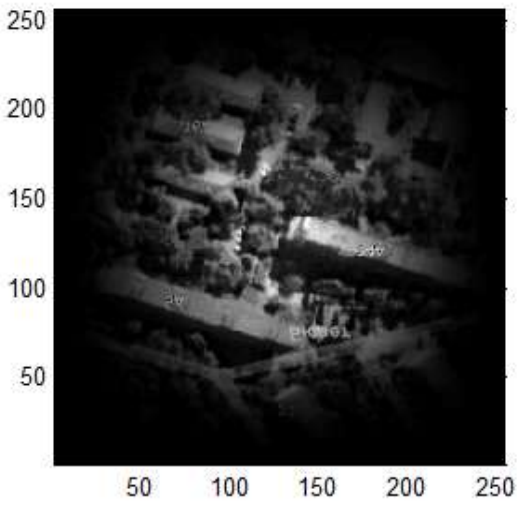
До отриманого результату r застосовується алгоритм пошуку пікового значення, що може бути описаний, наприклад, за допомогою наступного коду:

```
[~, I] = max(r(:));
[dx, dy] = ind2sub(size(r), I).
```

Отримаємо значення зміщення у пікселях зображення g_b відносно g_a.

На рис. 2 показано вихідні зображення та результат роботи алгоритму крос-кореляції – зверху вниз: зображення g_aw, g_bw (після застосування віконної функції, для даного прикладу — вікно Ханна), результат роботи алгоритму r. Після пошуку пікового значення визначено його положення:

dx = 14; dy=13.



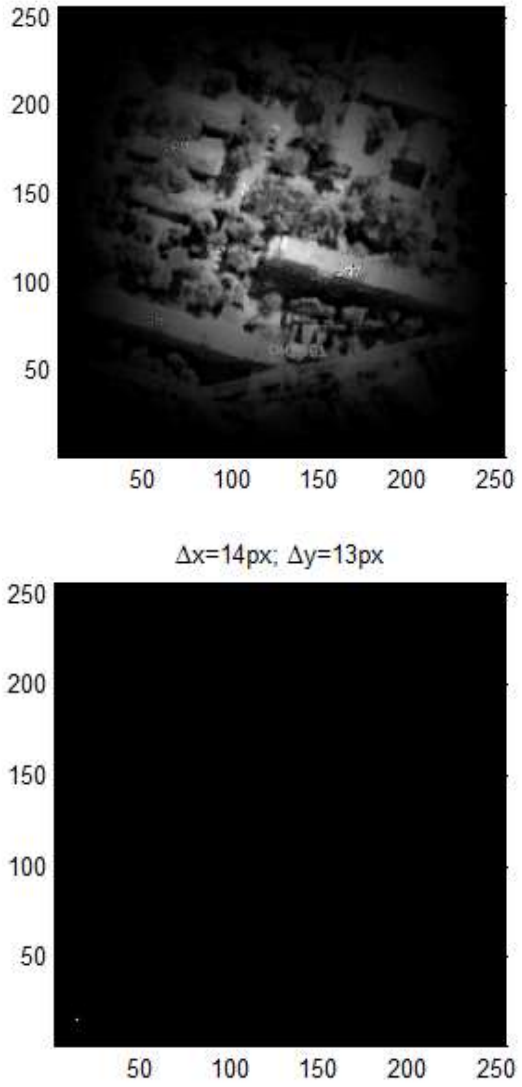


Рис. 2. Результат роботи алгоритму

Розширення алгоритму крос-кореляції для субпіксельної реєстрації зсуву

У роботі [8] показано можливість застосування алгоритму фазової кореляції для знаходження зсуву цифрових зображень з точністю кілька сотих пікселя.

Авторами пропонується наступна послідовність кроків, що заміщує кінець алгоритму, описаний у попередньому розділі.

Після обрахування зворотного дискретного перетворення Фур'є до результату r застосовуються дві маски – вилучаються спектральні компоненти, які лежать поза радіусом R від центрального піку, вилучаються спектральні компоненти, для яких нормована величина G_a або G_b менша за заданий поріг α. Оптимальне значення R, запропоноване у [8] знаходять з виразу:

$$R = \frac{0{,}6 \cdot N}{2},$$

де N – мінімальна кількість семплів вихідного цифрового зображення g_a.

Поріг α визначається з огляду на необхідність вилучення частотних компонент, величини яких менші за $\alpha \cdot p_{RMS}$, де p_{RMS} – середнє квадратичне спектральних величин, що лежать у області 5×5 пікселів від центрального піку кроскореляції R.

Вибір розширеного алгоритму дозволяє збільшити точність знаходження зміщення до субпіксельної, проте при виборі має прийматися до уваги баланс між роздільною здатністю оптичного сенсора, швидкодією обладнання та частотою обробки кадрів.

Автори роботи [8] пропонують застосовувати до вихідних зображень віконну функцію Блекмана – Харріса з метою мінімізації бічних пелюсток спектра.

Дослідження алгоритму на однокристальному мікроконтролері

Для дослідження можливості застосування алгоритму фазової кореляції у польотному комп'ютері БПЛА, проведено дослідження швидкодії роботи запропонованого алгоритму на мікросхемі STM32F407. До складу STM32F407 входить апаратна підсистема обчислень з плаваючою комою одинарної точності (single-precision FPU), що дозволяє збільшити швидкість проведення обчислень та використовувати представлення із плаваючою комою, не обмежуючись арифметикою з фіксованою позицією коми.

При розробці програмного коду використана CMSIS DSP бібліотека, що містить оптимізовані засоби роботи з матрицями комплексних та дійсних значень.

Початкове та зміщене зображення задані статично, у комплексному представленні. Для вимірювання швидкодії використано апаратний лічильник циклів процесора, що входить до блоку Data Watchpoint

and Trace Unit (DWT).

Вихідний код та виміри швидкодії його роботи приведені у таблиці.

Тактова частота роботи процесора 144МГц, компілятор IAR7.80.3, рівень оптимізації -O3 –oSize.

З результатів, приведених у таблиці видно, що найбільш ресурсомісткими операціями є перетворення Фур'є та зворотне перетворення Фур'є, що займають 77 % загального часу виконання алгоритму. Другим за часом виконання є алгоритм ділення, що займає 6.5 % часу виконання – через особливості використаного FPU операція ділення займає у 14 разів більше часу за операцію множення. Застосування віконної функції не потребує великих затрат часу, оскільки віконна функція описана у вигляді матричної константи спільного розміру з зображенням, при такому способі подання застосування віконної функції зводиться до поелементного множення двох матриць. Функція-генератор для подібних віконних функцій може бути описана за допомогою MATLAB:

```
w = blackmanharris(l);
w2 = w*w.';
```

де l – розмір зображення (висота або ширина), w2 – матриця, що містить згенеровану віконну функцію для відповідного розміру зображення.

Загальний час виконання алгоритму становить близько 5мкс для зображення 32x32пікс. та 21мкс для зображення 64x64пікс., що експериментально підтверджує лінійну складність алгоритму.

Таблиця

Результат роботи алгоритму

Операція	Код С	Час виконання, цикли	
		32x32	64x64
/* a = wnd(a) */	arm_cmplx_mult_real_f32(a->vec, (float*)window, a->vec, IMG_SZ*IMG_SZ);	13 088	52 250
/* b = wnd(b) */	arm_cmplx_mult_real_f32(b->vec, (float*)window, b->vec, IMG_SZ*IMG_SZ);	13 069	52 241
/* a = F(a) */	fft2_32(&cfft_instance, a->cplx32, 0);	187 172	752 283
/* b = F(b) */	fft2_32(&cfft_instance, b->cplx32, 0);	187 292	752 245
/* b = conj(b) */	arm_cmplx_conj_f32(b->vec, b->vec, IMG_SZ*IMG_SZ);	7 966	31 782
/* a = a.*b */	arm_cmplx_mult_cmplx_f32(a->vec, b->vec, a->vec, IMG_SZ*IMG_SZ);	18 460	73 751
/* b = abs(a) */	arm_cmplx_mag_f32(a->vec, b->vec, IMG_SZ*IMG_SZ);	32 327	129 100

/* a = a./b */	for(uint32_t i=0; i<IMG_SZ*IMG_SZ; i++){
	a->vec[2*i] = a->vec[2*i]/b->vec[i];
	a->vec[2*i+1] = a->vec[2*i+1]/b->vec[i];
	}
		51 217	204 887

/* a = ~F(a) */	fft2_32(&cfft_instance, a->cplx32, 1);	228 124	913 070
/* a = conj(a) */	arm_cmplx_mag_f32(a->vec, a->vec, IMG_SZ*IMG_SZ);	32 285	129 096
/* find peak */	arm_max_f32(a->vec, IMG_SZ*IMG_SZ, &m, &i); uint32_t dx = i % IMG_SZ; uint32_t dy = i / IMG_SZ;	8 498	33 842
	Загалом	779 511	3 124 566

Особливості та обмеження застосування алгоритму крос-кореляції

Запропонований алгоритм, що дозволяє визначити зміщення зображення, у контексті БПЛА може бути використаний не тільки в основі допоміжної стабілізуючої системи, але і як розширення навігаційної системи. Для цього необхідно додати інтегруючу ланку, яка буде сумувати моментальні значення зміщення за весь час роботи апарата, що дозволить отримати абсолютні значення координат. Знайдені результати можуть слугувати для отримання абсолютного зміщення апарата, проте увага має приділятися випадкам зміни висоти БПЛА над поверхнею рельєфу – при цьому коефіцієнти перетворення зміщення зображення (пікс.) у абсолютне переміщення (м/см/мм і т. п.) мають бути змінені або скинуто лічильник інтегратора.

На рис. 3 показано модель запропонованої системи визначення абсолютних координат. Модуль висотоміра (ВМ) керує коефіцієнтами перетворення величин (k), що перетворюють зсув зображення у дискретних елементах у фізичну величину. Інтегрована величина надходить до модуля ІНС, де використовується для розрахунків.

Також модуль висотоміра може здійснювати скидання (встановлення в початкове положення) інтеграторів за координатами у випадку істотної зміни висоти (в залежності від специфіки польоту БПЛА).

Існує можливість використання описаного алгоритму як основу слідкуючої системи на базі БПЛА. Однак має прийматися до уваги те, що при такому застосуванні розміри цільового об’єкта мають бути істотними (у контексті кута охоплення оптичного сенсора), а також контрастність зображення об’єкта має забезпечувати його чітке виділення серед елементів рельєфу.

Основними недоліками запропонованого підходу є чутливість до змін спостережуваної поверхні (якщо поверхня не є стійкою, наприклад, потік води, то можливості оптичної стабілізації значно скорочуються, оскільки в такому випадку на результат обчислень буде впливати власний рух поверхні), відсутність стабілізації по вертикальній осі, чутливість до змін кута нахилу апарата (при значних кутах нахилу БПЛА зображення істотно спотворюється і зміщення визначається неправильно).

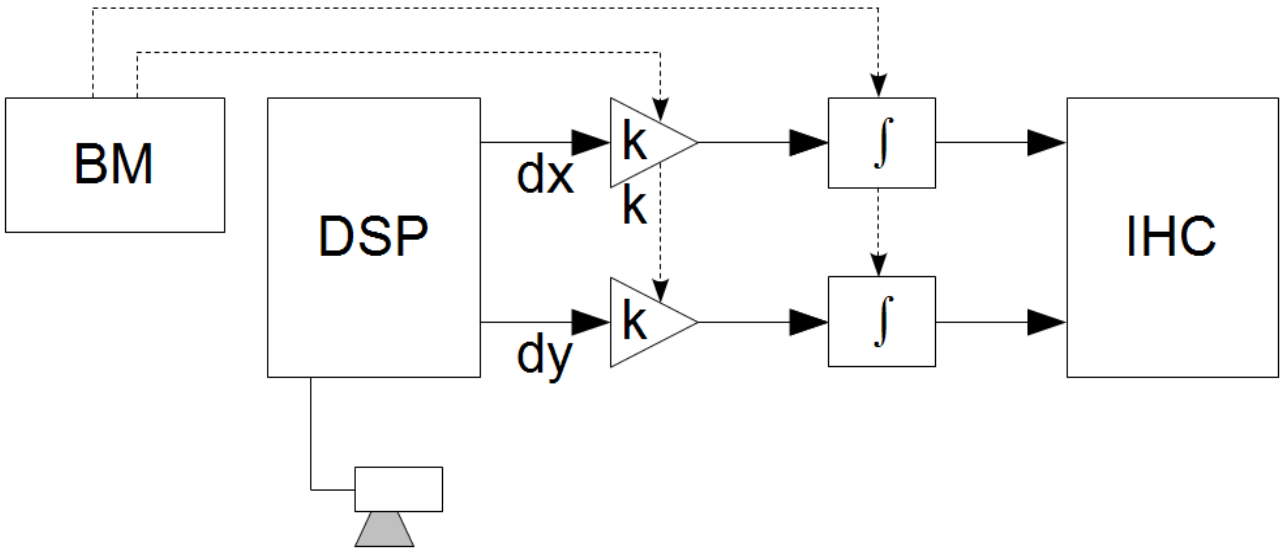


Рис. 3. Використання оптичного стабілізатора разом з висотоміром для отримання абсолютних координат

У разі відсутності достатньої кількості розрізнюваних елементів рельєфу, недостатнього освітлення або виходу з ладу відеоапаратури, результат алгоритму крос-кореляції не є інформативним. Для прикладу, на рис. 4 показано результат роботи алгоритму у випадку однорідної поверхні зі слабким рівнем освітлення.

Зміна кута нахилу створює спотворення, що розцінюється алгоритмом фазової кореляції як зміна координат об'єкта.

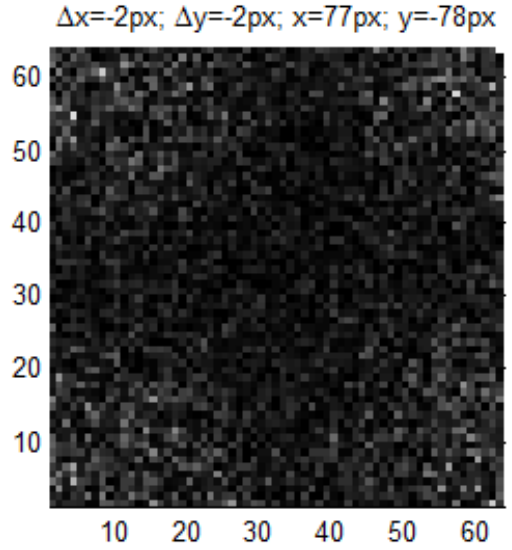


Рис. 4. Результат роботи алгоритму для поверхні з слабко вираженим рельєфом

На рис. 5 показано ідеальний випадок, при якому кут відхилення апарата від нормалі становить 0°.

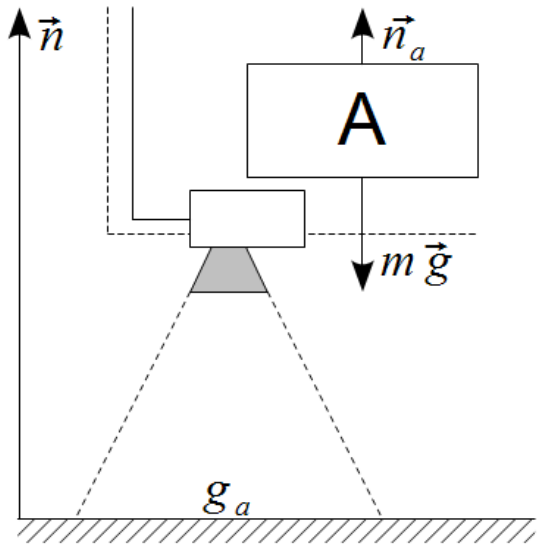


Рис. 5. Ідеальний випадок положення

При цьому кут нахилу апарата (визначається бортовим акселерометром) нульовий, нижня поверхня БПЛА розташована паралельно до поверхні рельєфу. Зображення g_a у цьому випадку поступає до DSP без спотворень, що відповідає випадку розрахунків, показаному на рис. 2.

Одним з підходів до вирішення даної проблеми є застосування акселерометра, як допоміжного пристрою, що блокує обробку спотворених зображень g_a процесором, якщо кут нахилу апарату α до нормалі перевищує певну величину (рис. 6).

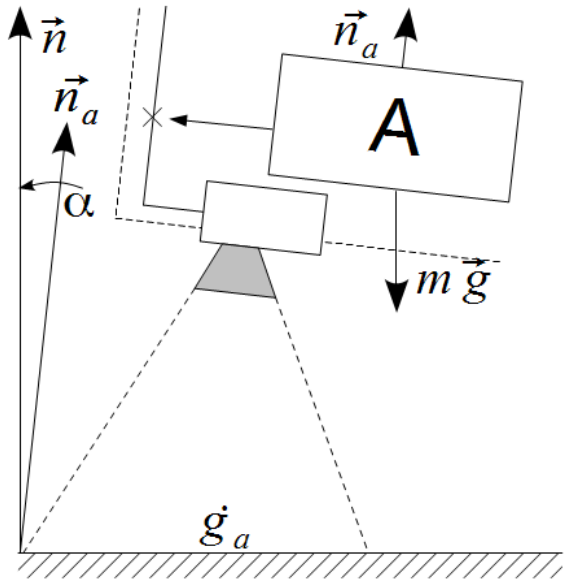


Рис. 6. Перевищення допустимого нахилу

При цьому втрачається певна кількість кадрів, що були зроблені при кутах нахилу апарата, що визначені бортовим обладнанням як недопустимі, що у свою чергу призводить до зменшення реакції положення апарата у просторі.

Допустимі значення кута нахилу визначаються експериментально в залежності від відстані апарата до поверхні рельєфу (за збільшенням відстані, спричинені зміною кута нахилу камери також збільшуються), характеру рельєфу та необхідної точності стабілізації.

Вказану особливість роботи алгоритму можна застосувати також для визначення кута нахилу БПЛА, однак ця можливість потребує додаткового дослідження та

модифікацій програми DSP та алгоритму стабілізації.

З урахуванням вищевказаних особливостей, найбільш ефективним є застосування оптичного стабілізатора для стабілізації положення БПЛА на невисокій висоті польоту, що співрозмірна з лінійними розмірами корпусу апарата.

Висновки

Запропонований алгоритм та його розширення, що, згідно результатів описаного експериментального дослідження, має невеликі вимоги до обчислювальної системи, може бути використано для збільшення точності встановлення стійкого положення мультироторних БПЛА здатних утримувати позицію у просторі (наприклад, три-, квадро-, мультикоптери) на низькій висоті польоту як за участі оператора так і без неї.

Можливе використання модифікації алгоритму для визначення абсолютних координат БПЛА. Перспективною областю застосування оптичної стабілізації є повністю автономні системи, що керуються штучним інтелектом для автономного виконання точних робіт на малій висоті польоту [9].

1. *Глотов В., Гуніна А.* Аналіз можливостей застосування безпілотних літальних апаратів для аерознімальних процесів. Сучасні досягнення геодезичної науки та виробництва. 2014. № 28. С. 65–70.
2. *Пельпор Д.С.* Гироскопические системы ориентации и навигации. Москва: Машиностроение, 1982. 165 с.
3. *Seo S.H., Lee B.H., Im S.H., Jee G.I.* Effect of Spoofing on Unmanned Aerial Vehicle using Counterfeited GPS Signal. Journal of Positioning, Navigation, and Timing. 2015. N 4. С. 57–65.
4. *Древаль О.Л., Дорошенко А.Ю.* Використання алгоритму фазової кореляції для допоміжної системи оптичної стабілізації літальних апаратів. Вісник НТУ України "КПІ": Інформатика, управління та обчислювальна техніка. – 2017. (у друку).
5. *Kuglin C.D., Hines D.C.* The phase correlation image alignment method. Proc. Int. Conf. Cybernetics Society. 1975. С. 163–165.
6. *Попов В.С.* Исследование влияния боковых лепестков спектра окон на погрешности обработки и передачи сигнала [Електронний ресурс]. Информационные системы и телекоммуникации. – 2010. – Режим доступу до ресурсу: windowing-matlab.narod.ru.
7. *Harris F.J.* On the use of windows for harmonic analysis with the discrete Fourier transform. Proceedings of the IEEE. 1978. N 66. С. 51–83.
8. *Harold S.S.* A Fast Direct Fourier-Based Algorithm for Subpixel Registration of Images. IEEE Transactions on Geoscience and Remote Sensing. 2001. N 39. С. 10.
9. *Ростопчин В.В., Дмитриев М.Л.* Применение цифровых оптических систем для беспилотных летательных аппаратов [Електронний ресурс]. Режим доступу до ресурсу: www.uav.ru/articles/opteq_uav.pdf.

References

1. Glotov V., Gunina A. Analysis of the literature. Possibilities of for UAVsfor aerophotographic processes. Modern achievements of geodetic science and industry. 2014. N 28. P. 65–70.
2. Pelpor D.S. Gyroscopic orientation and navigation system. Moscow: Mashinostroenie, 1982. 165 p.
3. Seo S.H., Lee B.H., Im S.H., Jee G.I. Effect of Spoofing on Unmanned Aerial Vehicle using Counterfeited GPS Signal. Journal of Positioning, Navigation, and Timing. 2015. N 4. С. 57–65.
4. Dreval O.L., Doroshenko A.Yu. Application of the phase correlation algorithm for use in the UAV stabilization. Visnyk NTUU “KPI” Informatics, operation and computer science. 2017. (not yet released)
5. Kuglin C.D., Hines D.C.. The phase correlation image alignment method. Proc. Int. Conf. Cybernetics Society. – 1975. – P. 163–165.
6. Popov V. S. Investigation of the effect of the side windows of the spectrum petals on the error signal processing and transmission [Electronic resource]. Information systems and telecommunications. 2010. Access: windowing-matlab.narod.ru.
7. Harris F.J. On the use of windows for harmonic analysis with the discrete Fourier

transform. Proceedings of the IEEE. 1978. N 66. P. 51–83.
8. Harold S.S. A Fast Direct Fourier-Based Algorithm for Subpixel Registration of Images. IEEE Transactions on Geoscience and Remote Sensing. 2001. N 39. P. 10.
9. Rostopchin V.V., Dmitriev M.L. The use of digital optical systems for unmanned aerial vehicles [Electronic resource]. Access: www.uav.ru/articles/opteq_uav.pdf.

Одержано 30.01.2017

***Про авторів*:**

Древаль Олег Леонідович,
бакалавр системної інженерії,
студент кафедри Автоматики
і управління в технічних системах
факультету Інформатики та
обчислювальної техніки
НТУ України «КПІ».
Кількість наукових публікації
в українських виданнях – 5.
http://orcid.org/0000-0002-8944-9837.

Дорошенко Анатолій Юхимович,
доктор фізико-математичних наук,
професор, завідувач відділу
Теорії комп'ютерних обчислень
Інституту програмних систем
НАН України,
професор кафедри Автоматики та
управління в технічних системах
НТУ України "КПІ".
Кількість наукових публікацій в
українських виданнях – понад 200.
Кількість наукових публікацій у
зарубіжних виданнях – понад 50.
Індекс Хірша – 5.
http://orcid.org/0000-0002-8435-1451.

Місце роботи авторів:

ТОВ «Сі Пі Ай – Київ»,
проспект Перемоги, 68/1, оф. 68,
м. Київ, 03113.
Тел.: (044) 207 3392, вн.: 4457.
E-mail: swinemaker@gmail.com

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

УДК 004.424, 004.415

UDC 004.424, 004.415

_DS_-теорія. Наукові аспекти та перспективи розвитку / В.Г. Колесник. – С. 3–20.

_DS_-theory. Scientific aspects and prospects for development / V.G. Kolesnyk. – P. 3–20.

У роботі представлена теорія схем декомпозиції як наукова теорія. Описано її атрибути – парадигма, поле досліджень, мета досліджень, основне завдання, метод рішення, основна теоретична модель – схема декомпозиції. Утилітарна, практична мета теорії – запропонувати механізм генерації прикладних алгоритмів (не машинного коду). Показано, що схема декомпозиції як опис замінює опис алгоритму і при цьому залишається декларативним на противагу опису алгоритму, як імперативному. При цьому схема декомпозиції є опис, вихідне завдання для генерації алгоритмів. Описано види схем декомпозиції та операції над ними. Описано алгоритмічно релевантні фактори, які слід враховувати при генерації алгоритмів для того, щоб алгоритми ставали реальними. У роботі описана робота з формалізації і математичному опису явищ і об'єктів теорії схем декомпозиції. Запропоновано механізм контролю висновків і результатів теорії. У роботі також описано напрям розвитку теорії схем декомпозиції. Пріоритетний напрямок співвідноситься з створенням і розвитком системи містких понять і абстракцій. З погляду практичного застосування теорії схем декомпозиції запропоновано підхід подібний до того, що є в машинобудуванні, – розвиток і просування передових технологій за запитом, по необхідності – як результат цілеспрямованих досліджень.

Ключові слова: алгоритм, аналіз, генерація, дані, декомпозиція, наука, програмна інженерія, програмування, процес, структура, синтез.

The work presents the theory of decomposition schemes as a scientific theory. It's attributes – paradigm, field of research, aim of research, main task, decision method, basic theory model – decomposition scheme were described. Practical aim of the theory is to suggest the mechanism of applied algorithms generation (not machine code). It is shown that the decomposition scheme as a description changes the algorithm description and still remains declarative on the contrary to the algorithm description as imperative. Thereby the decomposition scheme is the description, original for algorithms' generation. The kinds of decomposition schemes and operations on them have been described. Algorithmically relevant factors which should be considered when generating algorithms in order the algorithms turned real have been described. Working over formalization and mathematical description of the phenomena and objects of the theory of the decomposition schemes have been described here. The mechanism for control of conclusions and results of the theory has been suggested. This work also describes the direction of development of the theory of the decomposition schemes. The priority direction relates to the creation and development of the system of capacious notion and abstractions. From the point of view of the practical application of the theory of the decomposition schemes there the approach which is alike to the one used in machine-building – development and promoting of hi-tech when required – as a result of the targeted research has been suggested.

Key words: algorithm, analysis, generation, data, decomposition, science, program engineering, programming, process, structure, synthesis

УДК 004.42:510.69

Отношения логического следствия в логиках монотонных предикатов и логиках антитонных предикатов / О.С. Шкильняк. – С. 21–29.

Исследованы отношения логического следствия в чистых первопорядковых композиционно-номинативных логиках частичных однозначных и частичных неоднозначных квазиарных предикатов. Описаны композиционные предикатные алгебры, языки и классы интерпретаций (семантики) этих логик, выделен ряд отношений логического следствия. Основное внимание акцентировано на изучении таких отношений в логиках монотонных предикатов и логиках антитонных предикатов. Для этих логик определено 20 отношений логического следствия, установлено, что из них лишь 7 различных: $\overset{PE}{\models}_{IR}$, $\overset{PE}{\models}_{T}$, $\overset{PE}{\models}_{F}$, $\overset{PE}{\models}_{TF}$, $\overset{RM}{\models}_{T}$, $\overset{RM}{\models}_{F}$, $\overset{RM}{\models}_{TF}$. Приведены примеры, свидетельствующие о различии рассмотренных отношений логического следствия, установлены соотношения между такими отношениями.

Ключевые слова: логика, предикат, семантика, логическое следствие.

UDC 004.42:510.69

Logical consequence relations in logics of monotone predicates and logics of antitone predicates / O.S. Shkilniak. – P. 21–29.

Logical consequence is one of the fundamental concepts in logic. In this paper we study logical consequence relations for program-oriented logical formalisms: pure first-order composition nominative logics of quasiary predicates. In our research we are giving special attention to different types of logical consequence relations in various semantics of logics of monotone predicates and logics of antitone predicates. For pure first-order logics of quasiary predicates we specify composition algebras of predicates, languages, interpretation classes (sematics) and logical consequence relations. We obtain the pairwise distinct relations: irrefutability consequence $\overset{P}{\models}_{IR}$, consequence on truth $\overset{P}{\models}_{T}$, consequence on falsity $\overset{P}{\models}_{F}$, strong consequence $\overset{P}{\models}_{TF}$ in *P*-semantics of partial single-valued predicates and strong consequence $\overset{R}{\models}_{TF}$ in *R*-semantics of partial multi-valued predicates. Of the total of 20 of defined logical consequence relations in logics of monotone predicates and of antitone predicates, the following ones are pairwise distinct: $\overset{PE}{\models}_{IR}$, $\overset{PE}{\models}_{T}$, $\overset{PE}{\models}_{F}$, $\overset{PE}{\models}_{TF}$, $\overset{RM}{\models}_{T}$, $\overset{RM}{\models}_{F}$, $\overset{RM}{\models}_{TF}$. A number of examples showing the differences between various types of logical consequence relations is given. We summarize the results concerning the existence of a particular logical consequence relation for certain sets of formulas in a table and determine interrelations between different types of logical consequence relations.

Key words: logic, predicate, semantics, logical consequence.

УДК 004.853, 004.55

Классификация средств и методов семантического поиска в Web / Ю.В. Рогушина. – С. 30–50.

Рассмотрены проблемы, связанные с

UDC 004,853, 004.55

Classification of means and methods of the Web semantic retrieval / J. Rogushina. – P. 30–50.

Problems associated with the improve-

усовершенствованием поиска информации в открытой среде, обоснована потребность в ее семантизации. Проанализировано современное состояние и перспективы развития систем семантического поиска, ориентированных на обработку информационных ресурсов Web, рассмотрены критерии классификации таких систем. В этом анализе значительное внимание отводится использованию в семантическом поиске онтологий, которые содержат знания относительно предметной области поиска и относительно пользователя, для которого выполняется поиск. На основе анализа свойств существующих систем семантического поиска с точки зрения этих критериев выделенные области дальнейшего усовершенствования этих систем, предложена их реализация в системе семантического поиска “МАИПС”.

Ключевые слова: семантический поиск, онтология, Semantic Web, персонификация поиска.

ment of information retrieval for open environment are considered and the need for it’s semantization is grounded. The current state and prospects of development of semantic search engines that are focused on the Web information resources processing are analysed, the criteria for the classification of such systems are reviewed. In this analysis the significant attention is paid to the semantic search use of ontologies that contain knowledge about the subject area and the search users. The sources of ontological knowledge and methods of their processing for the improvement of the search procedures are considered. Examples of semantic search systems that use structured query languages (eg, SPARQL), lists of keywords and queries in natural language are proposed. Such criteria for the classification of semantic search engines like architecture, coupling, transparency, user context, modification requests, ontology structure, etc. are considered. Different ways of support of semantic and otology based modification of user queries that improve the completeness and accuracy of the search are analyzed. On base of analysis of the properties of existing semantic search engines in terms of these criteria, the areas for further improvement of these systems are selected: the development of metasearch systems, semantic modification of user requests, the determination of an user-acceptable transparency level of the search procedures, flexibility of domain knowledge management tools, increasing productivity and scalability.
In addition, the development of means of semantic Web search needs in use of some external knowledge base which contains knowledge about the domain of user information needs, and in providing the users with the ability to independent selection of knowledge that is used in the search process. There is necessary to take into account the history of user interaction with the retrieval system and the search context for personalization of the query results and their ordering in accordance with the user information needs. All these aspects were taken into account in the design and implementation of semantic search engine "MAIPS" that is based on an ontological model of users and resources cooperation into the Web.

Key words: semantic search, ontology, Semantic Web, personification of retrieval.

УДК 004.62

Інтеграція аксіоматики дескриптивних логік з реляційною моделлю даних / І.С. Чистякова. – С. 51–58.

Робота є логічним продовженням раніше опублікованих досліджень, які були присвячені опису відображень між дескриптивною логікою та реляційною моделлю даних. За допомогою попередньо створеної бінарної реляційної структури даних здійснюється відображення аксіоматики дескриптивної логіки ALC у реляційну модель даних (RDM). В роботі використовуються результати, що були отримані у минулих дослідженнях, а саме – структура даних RM^2, відображення базових концептів логіки ALC у RDM.

Ключові слова: семантична інтеграція даних, семантичний веб, відображення даних, реляційна модель даних, онтологія, ALC, дескриптивна логіка, аксіоми дескриптивних логік.

UDC 004.62

Integration of the description logics axiomatic into relational data model / I.S. Chystiakova. – P. 51–58.

The paper is a logical continuation of the previously published work, which was dedicated to the creation of mappings from the description logic into binary relational data model. On base of the previously created binary relational data structure we perform mappings of the ALC axiomatic into relational data model (RDM). The results of previous research namely data structure RM^2, mappings of the basic ALC concepts were used in this paper.

Key words: semantic data integration, semantic web, data mapping, relational data model, ontology, ALC, description logic, description logic axioms.

УДК 681.3

Метод распараллеливания циклов сеточных вычислительных задач для графических ускорителей / А.Е. Дорошенко, О.Г. Бекетов. – С. 59–66.

Разработано формальное преобразование гнезда вычислительного цикла, позволяющее осуществить переход от последовательного алгоритма к параллельному, ориентированное на выполнение на устройствах с SIMD архитектурой, в частности, на графическом ускорителе с использованием технологии CUDA и на гетерогенных кластерах. Описана и проиллюстрирована процедура перехода от последовательного к параллельному алгоритму. Для оптимизации обработки больших объемов данных использована процедура сериализации данных. Преимуществом предложенного метода является то, что он позволяет осуществлять преобразование данных, объем которых превышает объем памяти исполняющего устройства. Проведен

UDC 681.3

Method of parallelization of loops for grid calculation problems on GPU accelerators / A.Yu. Doroshenko, O.G. Beketov. – P. 59–66.

The formal parallelizing transformation of a nest of calculation loop for SIMD architecture devices, particularly for graphics processing units applying CUDA technology and heterogeneous clusters is developed. Procedure of transition from sequential to parallel algorithm is described and illustrated. Serialization of data is applied to optimize processing of large volumes of data. The advantage of the suggested method is its applicability for transformation of data which volumes exceed the memory of operating device. The experiment is conducted to demonstrate feasibility of the proposed approach. Technique presented in the provides the basis for further practical implementation of the automated system for parallelizing of nested loops.

эксперимент над задачей метеорологического прогнозирования погоды для демонстрации возможностей разработанного подхода. Методика, предложенная в данной работе, закладывает основу для дальнейшей практической реализации автоматизированной системы распараллеливания вложенных циклов.

Ключевые слова: автоматизированные методы параллелизации, оптимизация циклов, вычисления общего назначения на графических процессорах.

Key words: automatic parallelization, loop optimization, general-purpose computing on graphics processing units.

УДК 004.82

Про технологію використання зовнішніх даних при створенні і редагуванні енциклопедичних текстів / Г.Ю. Проскудіна, К.О. Кудім. – С. 67–82.

У роботі обговорюється проект Вікідані, веб-сервіс запитів і мова запитів. Робота веб-сервісу, мова запитів і форм виведення результатів демонструється на численних прикладах. Розроблено технологію використання Вікіданих сторонніми системами. У зв'язку з цим розглядається розширення ExternalData, розроблене для програмного забезпечення MediaWiki. В ході тестової експлуатації розширення ExternalData було доопрацьовано. Розширення використовується для вставки запитів даних до зовнішніх джерел, в нашому випадку, до бази знань Вікідані і їх результатів у вікі-розмітку, створюваних текстів статей. Розроблена процедура створення сторінки-списку.

Ключові слова: електронна енциклопедія, база Вікідані, сервіс запитів, мова запитів.

UDC 004.82

About technologies of use of external data on creating and editing of encyclopedic texts / G.Yu Proskudina, K.A. Kudim. – P. 67–82.

The paper discusses developing of Wikidata project, the query web service and the query language. The workflow of the web service, query language and result output forms are demonstrated with plenty of examples. Wikidata usage technology by third-party systems is developed. In this concern ExternalData extension which is part of MediaWiki software is considered. Additionally the instruction for installation and configuration of the extension is presented. During test period ExternalData extension was improved. procedure for automatic list generation in wiki page is developed.

Key words: digital encyclopedia, database Wikidata, query service.

УДК 004.942+004.415.2

Архитектура и функциональные возможности мультиагентной системы Навигация / А.Л. Яловец. – С. 83–96.

Рассматриваются особенности проектирования, разработки и функциони-

UDC 004.942+004.415.2

The architecture and functionality of the multi-agent system Navigation / A.L. Yalovets. – P. 83–96.

The features of the design, development and functioning of the multi-agent sys-

рования мультиагентной системы Навигация. Приводится архитектура системы и обосновывается выбор языка реализации системы. Детально анализируются функциональные возможности подсистем мультиагентной системы Навигация. На содержательном примере сравниваются результаты мультиагентного моделирования процессов преследования/убегания агентов средствами мультиагентной системы при разных режимах ее функционирования.

Ключевые слова: архитектура, мультиагентная система, функции подсистем, мультиагентное моделирование.

tem Navigation are investigated. System architecture and substantiate the choice of language implementation of the system are provided. The functionality of the subsystems of multi-agent systems Navigation is analyzed in detail. The results of multi-agent modelling of pursuit/escape processes by means of the multi-agent system in different modes of its functioning are compared on meaningful example.

Key words: architecture, multi-agent system, functions of subsystems, multi-agent modelling.

УДК 004.855:519.216

Принципы и аналитические средства реконструкции структур вероятностных зависимостей в специальном классе / А.С. Балабанов. – С. 97–110.

Предложен и обоснован набор эмпирических резолюций, которые опираются исключительно на безусловные зависимости двух переменных и обеспечивают идентификацию непосредственных связей (ребер) в структурах зависимостей в классе монопотоковых графов. Этот класс структур является подклассом ациклонных орграфов и суперклассом для поли-лесов. Охарактеризованы свойства монопотоковых моделей. Корректность разработанных эмпирических резолюций основывается на эмпирически надежном предположении безусловной (маргинальной) реберной необманчивости.

Ключевые слова: восстановление структуры зависимостей, монопотоковый граф, безусловная зависимость, эмпирическое проявление зависимостей, эмпирические резолюции идентификации ребер.

UDC 004.855:519.216

Principles and analytical tools for reconstruction of probabilistic dependency structures in special class / O.S. Balabanov. – P. 97–110.

We examine a problem of reconstruction of dependency structure from data. It is assumed that model structure belongs to class of “mono-flow” graphs, which is a subclass of acyclonic digraph (known as DAGs) and is super-class relatively to the poly-trees. Properties of the mono-flow dependency models are examined, especially in terms of patterns of unconditional dependencies and mutual information. We characterize the twin-association evolving among two variables. Specialized methods of inference of mono-flow dependency model are briefly reviewed. To justify correctness of model recovery from data we formulate an assumption of unconditional (marginal) edge-wise faithfulness, perhaps the most reliable one among all simple versions of Causal faithfulness assumption. On the basis of the assumption and the properties of mono-flow dependency models we derive several empirical resolutions for edge identification, which make use 2-placed statistics only. A lot of experiments with artificial data have demonstrated efficiency of the resolutions in that they correctly recover many edges and commit low error rate.

Key words: dependence structure recovery, mono-flow graph, unconditional dependence, empirical manifestation of dependence, empirical resolutions of edge identification.

УДК 004.896 + 004.932.2

Применение алгоритма фазовой корреляции и его расширения в оптическом стабилизаторе беспилотных летательных аппаратов /
О.Л. Древаль, А.Е. Дорошенко. – С. 111–119.

В статье рассматривается применение алгоритма фазовой корреляции как основы математического аппарата оптического стабилизатора беспилотных летательных аппаратов (БПЛА) для определения смещения корпуса относительно поверхности. Также рассмотрена возможность применения расширенных вариантов алгоритма фазовой корреляции для увеличения точности определения смещения и уменьшения вычислительных затрат. Приведен пример реализации алгоритма на языке MATLAB и результаты экспериментального исследования возможности применения реализации алгоритма на языке C в контексте встраиваемых систем.

Ключевые слова: БПЛА, оптическая стабилизация, обработка изображений, фазовая корреляция, субпиксельная регистрация изображений.

UDC 004.896 + 004.932.2

Application of the phase correlation algorithm and its extension in the optical stabilizer of unmanned aerial vehicles / O.L. Dreval, A.Yu. Doroshenko. – P. 111–119.

The use of the phase correlation algorithm as the basis of unmanned aerial vehicles (UAV) Optical Stabilizer on purpose to determine the displacement of the housing relative to the surface is described. Also, the possibility of application of the phase correlation algorithm extensions to increase accuracy and reduce computing costs is considered. The algorithm implementation in bot MATLAB language and C is provided. The computational complexity of the algorithm implementation in C was experimentally analyzed in the scope of embedded systems. The proposed algorithm gives not only an auxiliary possibility to stabilize the aircraft position but also to calculate absolute position of the aircraft in space.

Key words: UAV, optical stabilization, image processing, phase correlation, subpixel image registration.

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**.