НАЦІОНАЛЬНА АКАДЕМІЯ НАУК УКРАЇНИ
ІНСТИТУТ ПРОГРАМНИХ СИСТЕМ

ISSN 1727-4907

ПРОБЛЕМИ ПРОГРАМУВАННЯ

НАУКОВИЙ ЖУРНАЛ

PROBLEMS OF PROGRAMMING

SCIENTIFIC JOURNAL

2021
№ 4

ТЕМИ ВИПУСКУ:

- ***Моделі та засоби систем баз даних і знань***
- ***Інформаційні системи***
- ***Інструментальні засоби та середовища програмування***
- ***Моделі та методи машинного навчання***
- ***Історія програмування***

НАЦІОНАЛЬНА АКАДЕМІЯ НАУК УКРАЇНИ
ІНСТИТУТ ПРОГРАМНИХ СИСТЕМ

ПРОБЛЕМИ 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

Редакційна колегія

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

Затверджено до друку вченою радою Інституту програмних систем НАН України.
Протокол № 10 від 18.11.2021 р.

Редактор З.В. Єгорова
Комп'ютерна верстка В.П. Бумажний

Підписано до друку 07.12.2021. Формат 60x84/8. Папір офс. Ум. друк. арк. ___
Обл.-вид. арк.___ Тираж 120 прим. Ціна договірна. Замовл.

Віддруковано ТОВ «Про формат»
вул. Маршала Жукова, 43 б, м. Київ, 02166

НАЦІОНАЛЬНА
АКАДЕМІЯ НАУК УКРАЇНИ
ІНСТИТУТ ПРОГРАМНИХ СИСТЕМ

ПРОБЛЕМИ ПРОГРАМУВАННЯ

науковий журнал

№ 4 ***жовтень - грудень*** ***2021***

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 публі-куватися основні результати дисертаційних робіт.

ISSN 1727-4907

NATIONAL ACADEMY
OF SCIENCES OF UKRAINE
INSTITUTE OF SOFTWARE SYSTEMS

PROBLEMS OF PROGRAMMING

scientific journal

№ 4 ***October – December*** ***2021***

Founded in March, 1999

CONTENTS

УДК 519.164:004.14 https://doi.org/10.15407/pp2021.04.003

П.І. Андон, О.О. Слабоспицька

ЗАСОБИ ЗАБЕЗПЕЧЕННЯ ТА ЗАСВІДЧЕННЯ ЯКОСТІ КОНТЕКСТНО-ЗАЛЕЖНОЇ КОМПОЗИЦІЇ СЕМАНТИЧНИХ ВЕБ-СЕРВІСІВ

Розроблений авторами метод динамічної композиції адаптивного семантичного Веб-сервісу розвинуто алгоритмами узгодженого застосування спеціальної OWL-S-специфікації контексту його виконання на етапах життєвого циклу та опрацювання циклічних залежностей на функціональному рівні композиції за допомогою формалізмів Сервісу-вузла, Сервісу-посередника й Спрощення – для сталого забезпечення якості формованого Веб-сервісу. Запропоновано засвідчення рівня якості для причетних сторін шляхом динамічної верифікації, зокрема, перевірки відповідності запитаному контексту та живості, з використанням процесного числення контекстно-залежних амбієнтів. Надані алгоритми підвищують відповідність формованого Веб-сервісу очікуванням запитувачів та уможливлюють його стале контекстно-залежне уточнення для уніфікованої підтримки як змінних розподілених ділових процесів сучасних організацій, так і потреб споживачів певної галузі.

Ключові слова: OWL-S, семантичний Веб-сервіс, динамічна контекстно-залежна композиція Веб-сервісів, циклічна залежність, засвідчення якості, граф композиції, числення амбієнтів

Постановка проблеми

Для ефективної автоматизованої підтримки (не)передбачено змінних розподілених ділових процесів сучасних організацій в актуальних наразі слабко формалізованих предметних областях (ПрО), де ресурси її створення дедалі обмеженіші, вимоги до якості – жорсткіші, а очікування суб'єктів і параметри інформаційної інфраструктури істотно різнорідні, – все більш запитаними стають (само)адаптивні сервіс-орієнтовані програмні системи (СоПС). Вони являють собою on-line композиції (семантичних) Веб-сервісів, здатні до автоматизованого змінення поведінки для задоволення нових вимог і пристосування до нових (не)пе-редбачених ситуацій під час виконання [1].

Призначення СоПС і властива їх життєвому циклу (ЖЦ) глибока невизначеність [2] зумовлюють актуальний виклик їх конструювання – розроблення засобів не лише сталого забезпечення, але й *засвідчення* рівня якості, прийнятного для всіх причетних сторін, тобто збирання, аналізу, синтезу й адресного поширення вірогідних свідчень, що СоПС задовольняє поточні функційні й нефункційні вимоги в її ЖЦ [2, 3]. Аналіз актуального доробку інженерії (само)адаптивних СоПС [3] висвітлює перспективний підхід до опрацювання цього виклику – адекватну онтологічно базовану формалізацію контексту виконання СоПС та її складників й узгоджене застосування формалізованого контексту на стадіях ЖЦ для підвищення рівня якості їх продуктів.

Стаття підсумовує результати досліджень проекту ДР 0112U002764 під керівництвом академіка НАН України, д.ф.-м.н. П.І.Андона з розвитку, в руслі зазначеного підходу, нового методу побудови адаптивного композитного семантичного Веб-сервісу (АКС) [4,5], запропонованого авторами на підставі аналізу обмежень показових підходів до динамічної композиції семантичних Веб-сервісів для уніфікованої підтримки як розподілених і змінних процесів сучасних організацій, так і різнорідних змінних потреб окремих споживачів певної ПрО.

Формальні засоби розвитку методу

На підставі зіставлення перспективних методів контекстно-залежної композиції семантичних Веб-сервісів, поданих у табл.1, для розвитку методу [4,5] вибрано:

а) доповнення моделі АКС [4] поповнюваним ядром індустріально апробованих методів on-line відшукування й компонування (семантичних) Веб-сервісів на функціональному рівні (на грунті методів

Таблиця 1.
Перспективні методи контекстно-залежної композиції семантичних Веб-сервісів

№	Метод (назва, джерело)	Урахування контексту			Засоби	
		опис	етапи	техніки	специфікації	верифікації
1	Динамічна покрокова композиція А.Буччіароне [13]	Доменні об'єкти	Визначення композиції	Агрегування систем з переходом між станами	Агрегування систем з переходом між станами	Планування шляхом перевірки моделей
2	Узагальнена композиція семантичних Веб-сервісів С.Бансал [7]	Перед- і пост-умови, побічні впливи з їх об'єктами	Визначення композиції	OWL-S	Відсутні	Відсутні
3	Верифіковна графоподібна композиція семантичних Веб-сервісів для запитаного контексту виконання Р.Бен Ламін [8-10]	Онтологія контексту [9]	Всі етапи ЖЦ	Розширення мови OWL-S Відбір компонентних сервісів ССА [9]	Числення контекстно-залежних амбієнтів (ССА) [8,9]	Середовище виконання ССА [8]
4	Ціле-кероване планування Л.Ю [14]	Онтологія контексту	Опис контексту Визначення композиції	Долучення концептів контексту Відбирання Веб-сервісів	Відсутні	Відсутні
5	Аспектне програмування Л.Лі [15]	Онтологія контексту	Виконання композиції	Зв'язування аспектів	Відсутні	Відсутні
6	MDA/Notify, Decide, Configure Х.Байдурі [16]	UML-модель	Визначення композиції	Погляди на кон-текст	Відсутні	Відсутні
7	Планування/семантичне зіставлення вхідних і вихідних концептів Х.Мчейк [17]	Розширення мови OWL-S	Опис контексту Відшукування Веб-сервісів Виконання композиції	Розширення OWL-S Відбір Веб-сервісів Зв'язування аспектів	Відсутні	Відсутні
8	Планування/ застосування мови PDDL А.Фурно [18]	Онтологія OWL-SC	Опис контексту Визначення композиції	Розширення OWL-S Адаптаційні правила	Відсутні	Відсутні
9	Графо-орієнтоване планування Ю.Ян [19]	Показники QoS	Опис контексту Визначення композиції	Долучення QoS Відбирання Веб-сервісів	Відсутні	Відсутні
10	Застосування високорівневих мереж Петрі К.Букаді [20]	Мова XML	Визначення композиції	Погляди на контекст	Кольорові мережі Петрі	Планування шляхом перевірки моделей
11	Г.Алферез, В.Пелечано [11]	Онтології Семантичного Вебу	Те ж саме	Моделювання за допомогою RDF і даних спеціальних типів	BPMN, динамічна модель властивостей	Те ж саме
12	Трифазний підхід з використанням нечітких множин М.Мадкур [21]	Нечіткі множини	-"-	Правила адаптування політик застосування сервісів	Відсутні	Відсутні

М.Родрігеса [6], С.Бансал [7], Р.Бен-Ламіна [8-10], Г.Алфереза [11]) та на процесному рівні (М.Пісторе [12]);

б) удосконалення процедури композиції АКС алгоритмами опрацювання циклічної залежності на функціональному рівні за допомогою побудови сервіса-вузла [22], а в разі його неефективності – *сервіса-посередника* й *спрощення* згідно із запропонованими визначеннями;

в) засвідчення якості АКС за допомогою процесного числення контекстно-залежних амбієнтів (Ф.Сів) [23].

Долучення до ядра методів Р.Бен-Ламіна та Г.Алфереза для запитаного контексту виконання зумовлено їх перевагами для забезпечення й засвідчення якості АКС. Як показано в табл.1, серед схарактеризованих методів лише вони передбачають онтологічно базовану формалізацію контексту виконання композитного Веб-сервісу та його компонентів, відокремлену від опису виконуваних функцій, та її узгоджене застосування на етапах композиції. Завдяки цьому постачальник може зазначати в опублікованому описі (композитного) Веб-сервісу найдоцільніший, на його думку, контекст виконання, а споживач – компонувати й верифікувати цей сервіс для бажаного контексту.

Графо-подібна композиція семантичних Веб-сервісів для заданого контексту виконання

Засади методу. Відповідність формованого композитного Веб-сервісу заданому контексту його виконання забезпечують п'ять взаємопов'язаних тез [9,10]:

а) онтологічно-базована формалізація контексту виконання семантичних Веб-сервісів та їх композиції, уніфікована щодо етапів їх ЖЦ і ролі суб'єкта контексту (постачальник, брокер, споживач тощо);

б) узгоджене використання формалізованого контексту на всіх етапах ЖЦ композитного сервісу: від формування вимог до його публікування і/або виконання;

в) кількісне врахування, в планувальному графі композиції, трьох можливих ступенів семантичної відповідності вхідних і вихідних концептів сервісів у його складі (Exact, Plugin, Subsume) й доповнення графа таблицею індексів сумісності контекстів цих сервісів із запитаним контекстом;

г) вилучення з отриманого планувального графа всіх підграфів, відповідних запиту на композицію, та їх упорядкування за спаданням оцінок задовільності для запитувача – суми оцінок семантичної узгодженості й контекстної сумісності, зважених згідно з його перевагами;

д) використання спеціального процесного числення – Контекстно-залежних Амбієнтів (Context-aware Ambients, CCA) [23, 24] для формальної специфікації й верифікації композитного Веб-сервісу, насамперед перевірки його властивостей відповідності контексту запитувача та уточненої живості – «композитний сервіс завжди надає запитаний користувачем результат».

Наведені тези свідчать, що розглядуваний метод являє собою природний розвиток базового методу композиції на функціональному рівні М.Родрігеса [6] і є доцільним для застосування після нього в процесі покрокової композиції АКС [5].

Моделі та алгоритми композиції. Для реалізації тез а) – д) передбачено дві групи формалізмів [9,10]:

а) моделі та відповідні їм машинно-читані описи:

контексту виконання семантичного Веб-сервісу [25,26];

семантичного Веб-сервісу в заданому контексті [27];

запиту споживача щодо композиції семантичних Веб-сервісів для заданого контексту;

композитного семантичного сервісу, формованого за запитом;

б) алгоритми формування й застосування моделей:

прямого пошуку для побудови повного прошарованого графа компонентних Веб-сервісів згідно із запитом на їхню композицію;

зворотного пошуку для вилучення всіх нескорочуваних підграфів, відповідних запиту, які й являють собою моделі пошукуваних композитних сервісів;

впорядкування отриманих композитних сервісів за спаданням рейтингів їх прийнятності для запитувача, обчислених як сума індексів семантичної відпо-

відності між компонентними сервісами та відповідності їх контекстів запитаному контексту, зважених згідно з перевагами запитувача [25].

У межах методу контекст охоплює «довільні відомості, які можна використати для опису ситуації щодо сутності...– особи, місця або об'єкта, пов'язаного із взаємодією користувача й застосунку, включно із самими користувачем і застосунком» [8]. *Модель контексту* являє собою підмножину концептів спеціальної *онтології контекстів* [25,26] із їх взаємозв'язками. Ця онтологія є багаторівневим розширенням онтології OWL-S для уніфікованого опису множин контекстів у довільних ПрО мовою OWL. На її верхньому рівні відображено основні відомості щодо контексту за допомогою фіксованої структури концептів, наведеної на рис.1. Докладніші відомості визначено на (поповнюваних) нижчих рівнях, залежних від ПрО виконання композитного Веб-сервісу.

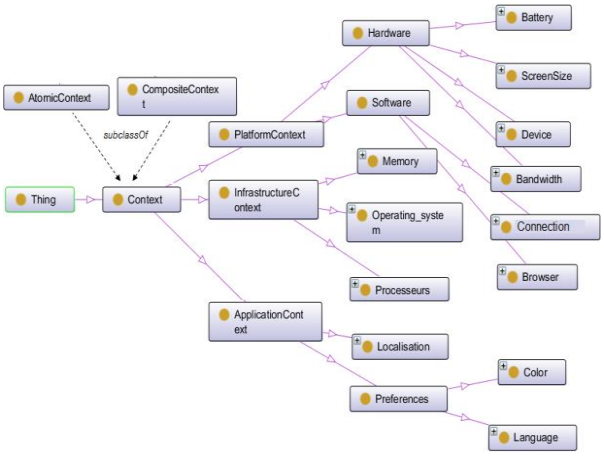


Рис. 1. Структура верхнього рівня онтології контекстів

Як показано на рис. 1, концептами верхнього рівня є «Контекст застосунку», «Платформний контекст», «Інфраструктурний контекст», а також «Атомарнй контекст» і «Складений контекст». Запровадження останнього концепту уможливлює подання контексту композитного сервісу як перетину атомарних контекстів його складників для точнішого відображення очікувань запитувачів і ситуацій у середовищі виконання.

Опис семантичного Веб-сервісу, виконуваного в заданому контексті, являє собою де-факто стандартну трійку Д.Мартіна – профіль, модель та обґрунтування сервісу [27], де, однак, згідно з пропозиціями [28] профіль сервісу доповнено атрибутом «adaptedTO» («пристосований_ДО»). Значенням цього атрибута є URL зовнішнього файлу з описом динамічного контексту, найдоцільнішого для виконання сервісу, на думку його постачальника. Приклад такого опису з атрибутом «adaptedTO» для профілю Веб-сервісу огляду ресурсів Інтернет подано на рис. 2.

Для зберігання доповнених описів передбачено відповідний розвиток реєстру UDDI [29], ефективність якого засвідчує його макетна реалізація з використанням JUDDIv3 і бази даних MySQL [8].

Запропоновані описи контексту та семантичного Веб-сервісу в заданому кон тексті дозволяють постачальнику сервісу легко оновлювати нефункціональні відомості щодо контексту в зовнішньому файлі без змін самого опису. Вони також уможливлюють формальну постановку проблеми композиції семантичних Веб-сервісів для заданого контексту за допомогою

Визначення 1 [9]. Модель семантичного Веб-сервісу із заданим контекстом – це трійка

$$w = \langle I, O, C \rangle, \qquad (1)$$

де I й O – множини його вхідних і вихідних параметрів (концептів онтології певної ПрО);

C – множина параметрів контексту (концептів онтології контекстів).

Визначення 3.2 [9]. Запит на композицію семантичних Веб-сервісів для заданого контексту – це п'ятірка

$$rc = \langle In, Out, Cr; ws,wc \rangle, ws, wc \rangle, \qquad (2)$$

де In і Out – множини вхідних і вихідних запитаних параметрів,

Cr – множина параметрів контексту користувача;

ws і wc – вагові коефіцієнти відносної важливості для користувача семантичної узгодженості формованого композитного сервісу та його відповідності запитаному контексту.

Визначення 3.3 [9]. Постановка проблеми композиції семантичних Веб-сервісів для заданого контексту – це структурований кортеж

```
<profile:Profile rdf:ID="__DESTINATION_PROFILE">
<service:isPresentedBy rdf:resource="#__DESTINATION_SERVICE"/>
<profile:serviceName xml:lang="en">
A_MyDESTINATIONService
</profile:serviceName>
<profile:textDescription xml:lang="en">
This service returns DESTINATION of the organization responsible for providing surfing activities.
</profile:textDescription>
<profile:hasInput  rdf:resource="#_ORGANIZATION"/>
<profile:hasInput  rdf:resource="#_SURFING"/>
<profile:hasOutput rdf:resource="#_DESTINATION"/>
<profile:has_process rdf:resource="__DESTINATION_PROCESS" />
<profile:adaptedTo
rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI">
http://localhost/services/AOWLS/contexts/Context__destination_MyOfficeservice.xml
</profile:adaptedTo>
</profile:Profile>
```

Рис. 2. Приклад опису Веб-сервісу огляду ресурсів Інтернет

$cp=\langle O,R,CO,rc;(\langle cs_i(rc), s_i, ss_i, sc_i\rangle, i\geq 1)\rangle$, (3)

$s_i = (ws\times ss_i + wc\times sc_i)/(ws + wc)$,

де O – задана онтологія ПрО, для якої запитано композицію;

R – множина семантичних Веб-сервісів для заданого контексту з моделлю (1) (об'єднання вмісту доступних реєстрів Інтернет);

CO – онтологія контекстів;

rc – запит на композицію (2);

$cs_i(rc)$ – i-й композитний семантичний сервіс для запиту rc в їх множині, впорядкованій за спаданням оцінок їх задовільності для споживача s_i;

ss_i й sc_i – оцінки семантичної узгодженості формованого композитного сервісу та його відповідності запитаному контексту [25], розглянуті далі.

Алгоритми вирішення поставленої проблеми (3) передбачають побудову на множині R повного планувального графа в його традиційному форматі [10], який, однак, стає контекстно-залежним, оскільки його вершинами є сервіси з певними контекстами (1). Під час його формування враховують усі можливі ступені семантичної подібності між вихідним (out) і вхідним (in) концептами послідовно відшукуваних проміжних сервісів, яким додатково зіставлені кількісні оцінки $e(out,in)$ [10]:

– 1, якщо об'єми відповідних концептів однакові, тобто ступінь подібності – Exact;

– 0.9, якщо вихідний концепт одного сервісу є вкладеним концептом вхідного концепту іншого, тобто ступінь подібності – Plugin;

– 0.5, якщо для вихідного концепту одного сервісу вхідний концепт іншого є вкладеним концептом, тобто ступінь подібності – Subsume;

– 0, якщо семантична подібність між вихідним і вхідним концептами відсутня (ступінь Fail).

На підставі цих оцінок формується допоміжна матриця семантичних зв'язків, вигляд якої фіксує технічне

Визначення 4 [10] Матриця семантичних зв'язків для постановки проблеми композиції (3) – квадратна матриця з елементами $sl_{ij} = 0$, якщо серед вихідних концептів i-го сервісу $w_i \in R$ і вхідних концептів j-го сервісу $w_j \in R$ немає семантично подібних;

$sl_{ij} = (V_{ij}; sim_{ij}), V_{ij} = \{(out,in), out\in O(w_i), in\in I(w_j)\}$; (4)

– в іншому випадку, причому sim_{ij} – середнє оцінок $e(out,in)$ для множини V_{ij}.

Процедура вирішення проблеми композиції в постановці (3) передбачає три обов'язкові кроки:

а) побудова повного контекстно-залежного планувального графа та відповідних йому матриць індексів контекстної подібності (CS) й семантичних зв'язків (SLM) за допомогою алгоритму прямого пошуку, поданого на рис. 3;

б) вилучення з побудованого графа відповідних запиту підграфів $cs_i(rc)$ з (3) та обчислення оцінок задовільності відповідних композитних семантичних Веб-сервісів для споживача s_i із складниками ss_i й sc_i за допомогою алгоритму зворотного пошуку;

в) впорядкування отриманих композитних сервісів за спаданням оцінок s_i за допомогою алгоритму сортування.

Четвертим необов'язковим кроком є публікування композитних сервісів з

```
Input  : ReqComp < In_Req,Out_Req,C_Req >: User's Request
         R : Repository of Adaptable Semantic Web Services
         O : Domain Ontology
         CO : Context Ontology
Output: G : Context Aware Semantic Plannin Graph
1  A_0=∅; P_0={In_Req}; G=(A_0,P_0)
2  i=1;
3  Init_SLM(SLM) ;
4  repeat
5  |  S=Select_SimSC_Services(P_{i-1}, R, O, CO);
6  |  P_i=P_{i-1};
7  |  for each s in S do
8  |  |  if (! Exist(s)) then
9  |  |  |  SLM=Update_SLM;
10 |  |  |  Add(P_i, s.out);
11 |  |  |  CS=Build_CS(CS,s);
12 |  |  |  A_i=A_i ∪ {s};
13 |  |  end
14 |  end
15 |  G=G ∪ {(A_i,P_i)};
16 |  i=i+1;
17 until Out_Req ⊆ Last_ P_i(G) ∨ Fixed_Point(G);
```

Рис. 3. Псевдокод алгоритму прямого пошуку найкращою оцінкою s_i у розширенні реєстру UDDI [29].

Як показано на рис. 3, застосування всіх можливих ступенів семантичної подібності вхідних і вихідних концептів для формування ярусів планувального графу підвищує кількість його підграфів, відповідних запиту, оскільки повний збіг таких концептів може не досягатися.

На початку виконання алгоритму планувальний граф містить шар 0, де множина дій A_0 порожня, а множина висловлювань P_0 подає початковий стан проблеми планування й містить вхідні параметри запиту. В досліджуваному випадку композиції семантичних Веб-сервісів кожна дія в G подає атомарний Веб-сервіс. Вхідні параметри Веб-сервісу стають передумовами дії в графі G, а вихідні параметри відображаються в позитивні наслідки дії.

Планувальний граф G ітеративно поповнюють, доки не буде виконано одну з умов завершення алгоритму:

– досягнуто рівня, для якого множина пропозицій містить усі вихідні параметри з запиту на композицію (2);

– досягнуто фіксованого рівня i, де $A_{i+1} = A_i$, $P_{i+1} = P_i$.

Для кожного шару $i > 0$ нові Веб-сервіси відшукують в репозиторії R на підставі семантичного зіставлення їхніх вхідних параметрів з висловлюваннями (i-1)-го шару та сумісності контекстів, виявлених Веб-сервісів із запитаним контекстом користувача (*Select_SimSC_Services* на рис. 3). Для кожного нового відшуканого Веб-сервісу, ще не долученого до графу G, додають нові елементи до матриць семантичних зв'язків (*Update_SLM*) та індексів контекстної подібності (*Build_CS*), а його вихідні параметри долучають до множини висловлювань P_i.

Побудова планувального графа з матрицями *CS* і *SLM* уможливлює формування множини найкращих рішень проблеми композиції (3) (*ACSWS_List*) за допомогою алгоритму зворотного пошуку, поданого на рис. 4.

Згідно з рис. 4, пошук починають з останнього шару висловлювань у планувальному графі G.

Нехай g – множина цільових висловлювань, яких необхідно «досягти» для заданого шару висловлювань P_i. Алгоритм знайде підмножину дій з A_i, які разом «досягають» g, і відбере ті з них, що мають найвище значення семантичної узгодженості, на підставі матриці семантичних зв'язків з оцінками (4). Об'єднання передумов цих дій являє собою нову множину цілей, яких треба досягти для попереднього шару висловлювань P_{i-1}. Виконання цього процесу за алгоритмом, наведеним на рис. 5, продовжують, поки не буде досягнуто першого шару висловлювань графа G, а саме P_0.

На кроці впорядкування алгоритм сортування (див. рис. 6) приймає множину найкращих композитних Веб-сервісів, сформовану на попередньому кроці, й надає їх перелік *Orders_Sols*, впорядкований за спаданням оцінок (4).

```
Input  : ReqComp < In_Req, Out_Req, C_Req >: User's Request
         G : Contewt Aware Semantic Plannin Graph
         SLM : Semantic Link Matrix
         CS : Context Similarity Table
Output: ACSWS_List :
         List of best adaptable composed Semantic Web Services
1 if (! Out_Req ⊆ Last_P_i(G)) then
2 |  return ∅;
3 else
4 |  ACSWS_List :=extract_best_sols(G, SLM, CS)
5 end
```

Рис.4. Псевдокод алгоритму зворотного пошуку

```
Input  : goals : set of goals to be achieved
         i: a level in the Graph G
         G: Context Aware Semantic Plannin Graph
         SLM:Semantic Link Matrix
         CS:Context Similarity Table
Output: Sols: set of best solutions
1  if ( i = 0) then
2  |  return Sols;
3  else
4  |  goals' = ∅;
5  |  for each proposition p in goals do
6  |  |  sources= Get_Sources(p)
7  |  |  /*return the set of services which semantically produce p */
8  |  |  Choose an action a such that ∃ source in sources ,
   |  |    SLM(a, source) = max_j(SLM(a, source_j)), source_j ∈ sources
   |  |    goals'=goals' ∪ {a.Ins }
9  |  |  sol=sol ∪ {a}
10 |  end
11 |  Sols=Sols ∪ {sol}
12 |  extract_best_Solutions(G, SLM, goals', i − 1, CS)
13 end
```

Рис. 5. Псевдокод алгоритму вилучення найкращих композитних сервісів

```
Input  : ACSWS_List : List of valid adaptable composed Semantic Web
         Services
Output: Ordered_Sols: ranked composition solutions
1 for each sol in ACSWS_List do
2 |  Score_sol=[ w_sem * score_sem(sol)+w_context * score_context(sol)] /( w_sem+w_context)
3 end
4 Ordered_Sols= Sort(ACSWS_List, Score_sol)
```

Рис. 6. Псевдокод алгоритму сортування

Специфікація й верифікація АКС із контекстом за допомогою числення контекстно-залежних амбієнтів. Універсальний формальний апарат специфікації АКС із контекстом (2) і запиту щодо нього (3) надають темпоральні логіки лінійного й розгалуженого часу, а верифікації відповідних описів – методи й інструментальні засоби перевірки моделей за допомогою цих логік [30]. Однак вони зорієнтовані переважно на опис ділового процесу, виконуваного АКС, і залишають акцентований методом контекст його перебігу поза увагою. Тому для засвідчення якості формованого АКС з контекстом пропонується його специфікація та верифікації насамперед за допомогою числення контекстно-залежних амбієнтів (calculus of context-aware ambients, CCA) [23,24,31].

Числення CCA розвинуто Ф.Сівом на ґрунті відомого числення *мобільних амбієнтів* Л.Карделлі–А.Гордона для моделювання контекстно-залежних динамічно взаємодіючих систем й виведення логічних висновків щодо їхньої поведінки. Базове поняття *амбієнта* [23,24] являє собою концептуальне подання довільної сутності (фізичної, віртуальної, мобільної або непорушної), яка визначає обмежене місце, де має місце обчислення, та:

– має ім'я, межі й може бути вкладена до іншого амбієнта;

– може бути мобільною;

– може обмінюватися повідомленнями з собою та з іншими амбієнтами;

– може усвідомлювати присутність інших амбієнтів у її середовищі.

Хоча з теоретичної точки зору CCA можна природним чином «занурити» до системи інсерційного моделювання О.А.Летичевського, перспективність його застосування для побудови контекстно-залежного АКС [4,5] зумовлена наявністю відповідної йому мови програмування ccaPL і вільно доступного середовища виконання програм цією мовою [23,31].

У CCA передбачено чотири основні синтаксичні категорії [23,24], зіставлені на рис. 7:

1) *процеси*, які подають обчислення, тобто поведінку модельованої системи;

2) *спроможності* (capabilities) – елементарні дії, які можуть виконувати процеси;

3) *розміщення* (locations) – позиції в ієрархії амбієнтів;

4) *контекстні вирази* – формули запровадженої просторової логіки, що подають властивості контексту процесів.

Для CCA-верифікації АКС з контекстом у [29] виокремлено вісім патернів структур керування в сервісній моделі зі складу його опису мовою AOWL-S, яким зіставлено CCA-подання.

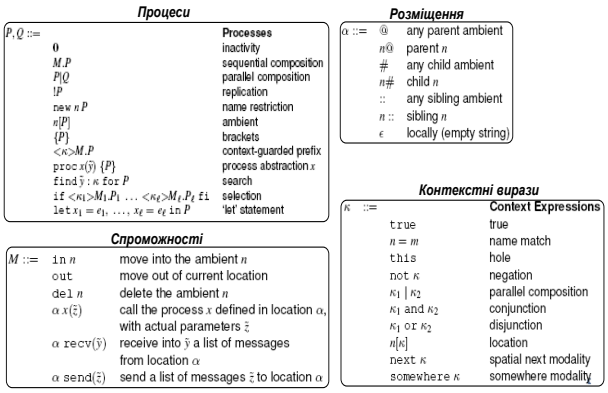


Рис. 7. Синтаксис числення ССА

1. *Атомарному процесу*, що реалізується атомарним семантичним сервісом з контекстом $S(C,P)$ (де С – найдоцільніший контекст виконання S, а P – атомарний процес, який слід виконати), в ССА відповідає амбієнт з іменем S: $S[C|P]$ (нотація $C|P$ вказує, що процес P виконується в контексті C).

2. Щоб моделювати *послідовність* композитних семантичних Веб-сервісів з контекстом, дії, охоплені послідовністю, подають амбієнтами, а відношення передування – повідомленнями, які надсилають ці амбієнти. Враховуючи ієрархічну структуру амбієнтів ССА, послідовність можна моделювати амбієнтом-батьком, що подає композитний Веб-сервіс, з амбієнтами– нащадками одного рівня A,B. У синтаксисі ССА подання послідовності має вигляд системи формальних висловлювань:

$$P_A = S_{Seq} \uparrow (x).\tau_A.B ::< y > .0$$

$$P_B = A :: (y).\tau_B.S_{Seq} \uparrow (z).0$$

$$P_{Seq} = EC(C_A, C_B)?P_A.P_B \quad .$$

або фрагмента коду мовою ccaPL:

```
<Sequence>
<Process> C_A, A</Process>
<Process>C_B, B</Process>
</Sequence>
```

.

Процес P_A отримує повідомлення x від амбієнта-батька (позначеного $S_{seq}\uparrow$), виконує внутрішню дію τ_A, передає результат y своєму однорівневому амбієнту B (позначеному B::) і завершується. Процес P_B отримує повідомлення y від свого однорівневого амбієнта A (позначеного A::), виконує внутрішню дію τ_B, надсилає результат z амбієнту-батьку S_{Seq} і завершується. Результуючий процес P_{Seq} є контекстно-захищеним, де $EC(C_A, C_B)$ є контекстним виразом щодо найприйнятніших контекстів C_A і C_B для процесів P_A і P_B.

3. Дії, охоплені структурою керування «*Розгалуження*» мають виконуватися одночасно. Щоб моделювати розгалуження композитних семантичних Веб-сервісів з контекстом, попередні дії й дії, охоплені розгалуженням, слід перетворити в амбієнти ССА й застосувати оператор паралельної композиції «|» для подання відношення одночасного виконання.

У синтаксисі ССА розгалуження з діями B,C, яким передує дія A, подають таким чином:

$$P_A = S_{Split} \uparrow (x).CE(C_A)?\tau_A.CE(C_A, C_B)?(B ::< y > | D ::< z >).0$$

$$P_B = P_A \uparrow (y).CE(C_A, C_B)?\tau_B.S_{Split} \uparrow < y_1 > .0$$

$$P_D = P_A \uparrow (z).CE(C_A, C_D)?\tau_D.S_{Split} \uparrow < z_1 > .0 \quad ,$$

а мовою ccaPL – так:

```
<Process> C_A, A</Process>
<Split>
   <Process>C_B, B</Process>
  <Process> C_D, D</Process>
</Split>
```

.

Процес P_A отримує повідомлення x від свого амбієнта-батька (позначеного $S_{seq}\uparrow$). Якщо контекст виконання підтверджує контекстний вираз $CE(C_A)$, процес P_A виконує внутрішню дію τ_A, одночасно надсилає повідомлення y і z своїм однорівневим амбієнтам B і D (позначеним B:: і D::) й завершується.

Процеси P_B і P_D отримують повідомлення y і z від свого однорівневого амбієнта A (позначеного A::). Якщо контекст виконання підтверджує контекстний вираз $CE(C_A, C_B)$, процес P_B виконує свою внутрішню дію τ_B, надсилає результат y_1 амбієнту-батьку S_{Split} і завершується.

Одночасно, якщо контекст виконання підтверджує контекстний вираз $CE(C_A,C_D)$, процес P_D виконує внутрішню дію τ_D, надсилає результат z_1 амбієнту-батьку S_{Split} і завершується.

4. Структура керування «*Розгалуження з об'єднанням*» (Split+Join) охоплює одночасні дії, які необхідно синхронізувати. Щоб моделювати розгалуження з об'єднанням композитних семантичних Веб-сервісів з контекстом, дії, охоплені послідовністю, й наступні дії слід перетворити в амбієнти ССА. Одночасно виконання внутрішніх дій наступного процесу роз-

почнеться лише після отримання повідомлень-тригерів, надісланих процесами, що охоплені Розгалуженням з об'єднанням, якщо відповідний контекстний вираз підтверджений контекстом виконання.

У синтаксисі ССА розгалуження з об'єднанням, де послідовність містить дії *A*,*B* і наступну дію *D*, подають так:

$$P_A = S_{Split-Join} \uparrow (x).\tau_A.CE(C_A, C_B, C_D)?(B ::< y > | D ::< z >).0$$

$$P_B = CE(C_A, C_B)?A ::(y).\tau_B.S_{Split-Join} \uparrow < t_1 > .0$$

$$P_D = CE(C_A, C_D)?A ::(y).\tau_D.S_{Split-Join} \uparrow < t_2 > .0$$

,

а мовою ссаPL – так:

```
<Split-Join>
   <Process>C_A, A</Process>
   <Process> C_B, B</Process>
</Split-Join>
<Process> C_D, D</Process>
```

.

Процес P_A отримує повідомлення x від амбієнта-батька (позначеного $S_{Split-Join}\uparrow$) й виконує внутрішню дію τ_A. Якщо контекстний вираз $CE(C_A,C_D)$ підтверджено контекстом виконання, процес P_A надсилає повідомлення z_1 своєму однорівневому амбієнту D (позначеному D::) і завершується. Одночасно процес P_B отримує повідомлення y від свого амбієнта-батька ($S_{Split-Join}\uparrow$) й виконує внутрішню дію τ_B. Якщо контекстний вираз $CE(CB,CD)$ підтверджено контекстом виконання, процесс P_B надсилає повідомлення z_2 своєму однорівневому амбієнту D (D::) і завершується.

P_D одночасно отримує повідомлення z_1 і z_2 від своїх однорівневих амбієнтів A і B. Тоді процес P_D виконує внутрішню дію τ_D, посилає результат t амбієнту-батьку $S_{Split-Join}$ і завершується.

5. Структуру керування «*Вибір*» застосовують, коли треба випадковим чинном вибрати для виконання точно один процес з їх множини. В синтаксисі ССА вибір між процесами B і D після виконання процесу A подають таким чином:

$$P_A = S_{Choice} \uparrow (x).\tau_A.(CE(C_A, C_B)?B ::< y > .0 + CE(C_A, C_D)?D ::< z > .0)$$

$$P_B = CE(C_A, C_B)?A ::(y).\tau_B.S_{Choice} \uparrow < t_1 > .0$$

$$P_D = CE(C_A, C_D)?A ::(z).\tau_D.S_{Choice} \uparrow < t_2 > .0$$

,

а мовою ссаPL – так:

```
<Process> C_A, A</Process>
  <Choice>
    <Process>C_B, B</Process>
    <Process> C_D, D</Process>
 </Choice>
```

.

де S_{Choice} – амбієнт-батько.

Амбієнти A, B і D є однорівневими нащадками амбієнта S_{Choice}. Амбієнт A отримує повідомлення x від амбієнта-батька S_{Choice}, виконує внутрішню дію τ_A й далі недетерміновано вибере одного зі своїх онорівневих амбієнтів B або D для виконання. Якщо вибрано B, і контекстний вираз $CE(CA,CB)$ підтверджено його контекстом C_B, то амбієнт A надсилає повідомлення y амбієнту B й завершується. Таким же чином, якщо вибрано D, і контекстний вираз $CE(C_A, C_D)$ підтверджено його контекстом C_D, амбієнт A надсилає повідомлення y амбієнту D й завершується.

Якщо контекстний вираз $CE(C_A,C_B)$ (відповідно $CE(C_A,C_D)$) виконано, то процес P_B (відповідно, P_D) отримує повідомлення x (відповідно, y) від однорівневого амбієнта A, виконує внутрішню дію τ_B (τ_D), надсилає результат t_1 (відповідно, t_2) амбієнту-батьку S_{Choice} і завершується.

6. Структуру «*Якщо - То - Інакше*» застосовують, коли потрібно вибрати для виконання одну з двох дій згідно з умовою. Щоб моделювати цю структуру для АКС з контекстом, охоплені нею дії слід перетворити в амбієнти ССА, а умові зіставити захисні вирази, які застосовують, щоб вибрати дію для виконання. Коли структура «Якщо - То – Інакше» охоплює дії A і B, її ССА-подання має вигляд:

$$P_A = [Cond = True]CE(C_A)?S_{If-Then-Else} \uparrow (x).\tau_A.S_{If-Then-Else} \uparrow < y > .0$$

$$P_B = [Cond = False]CE(C_B)?S_{If-Then-Else} \uparrow (x).\tau_B S_{If-Then-Else} \uparrow < y > .0$$

де $S_{If\text{-}Then\text{-}Else}$ – амбієнт-батько, амбієнти A і B – його нащадки, PA і PB – процеси, виконувані A і B.

Цю структуру описує ссаPL-код:

```
<If-Then-Else>
 <If-Condition> Cond </If-Condition>
 <Then>
  <Process> C_A, A</Process>
 </Then>
 <Else>
  <Process>C_B, B</Process>
 </Else>
</If-Then-Else>
```

.

Якщо умову *Cond* задоволено, процес P_A запускається, коли контекстний вираз $CE(C_A)$ підтверджено його контекстом C_A. P_A отримує повідомлення x від амбієнта-батька $S_{If\text{-}Then\text{-}Else}$, виконує внутрішню дію τ_A, надсилає результуюче повідомлення y

$S_{If\text{-}Then\text{-}Else}$ й завершується. Коли умову *Cond* не виконано, процес P_B запускається, якщо контекстний вираз $CE(C_B)$ підтверджено контекстом C_B.

7. Структура керування «*Повторювати-Поки*» підтримує повторення дії, доки умову підтверджено. Щоб моделювати цю структуру для АКС з контекстом, підтримувану нею дію слід перетворити в амбієнт і застосовувати умову, щоб визначити, чи продовжувати цикл. Оператор реплікації «!» застосовують для позначення повторення тіла циклу.

Якщо ця структура охоплює одну дію *A*, її ССА-подання має вигляд:

$$P_A = !\left(\left[Cond = True\right]CE(C_A)\,?\,S_{Repeat\_While}\uparrow(x).\tau_A\right).S_{Repeat\_While}\uparrow<y>.0,$$

де $S_{Repeatf\text{-}While}$ – амбієнт-батько з нащадком – амбієнтом *A, PA* – процес, виконуваний *A*.

Наведеному поданню відповідає фрагмент коду мовою ccaPL:

```
<Repeat-While>
 <While-Condition>
    Cond
 </While-Condition>
<While-Process>
 <Process> C_A, A</Process>
 </While-Process>
</Repeat-While>
```

8. На відміну від попередньої структури, ССА-подання останньої структури «*Повторювати до*» («Repeat-Until для АКС з контекстом має вигляд:

$$P_A = \left(CE(C_A)\,?\,S_{Repeat\_Until}\uparrow(x).\tau_A.P_A.\left[Cond = True\right]\right).S_{Repeat\_Until}\uparrow<y>.0,$$

де збережено попередні познаки.

Якщо контекстний вираз $CE(C_A)$ підтверджено, P_A обробляє повідомлення *x* від амбієнта-батька $S_{Repeatf\text{-}Until}$ і виконує свою внутрішню дію. Процес P_A рекурсивно повторюється, доки умова *Cond* виконана. Коли *Cond* порушено, процес P_A надсилає результуюче повідомлення *y* до $S_{Repeatf\text{-}While}$ й завершується.

Подання цієї структури мовою ccaPL має вигляд

```
<Repeat-Until>
<Until-Process>
     <Process> C_A, A</Process>
</Until-Process>
<Until-Condition>
    Cond
</Until-Condition>
</Repeat-Until>
```

Сервіс-посередник і спрощення для усунення циклічних залежностей в АКС з контекстом

Зіставлення змістовних описів алгоритмів усунення циклічної залежності в композиції семантичних Веб-сервісів за допомогою *сервісу-вузла* [1,22], розроблених на попередньому етапі робіт за проєктом ДР 0112U002764, і найпростішого формалізованого опису композитного семантичного Веб-сервісу на функціональному рівні [1,6] дозволяє формально описати сервіс-вузол за допомогою

Визначення 5. Нехай $CD=\{sc_i,\ i\geq 1\}$, $sc_i = (In(sc_i);Out(sc_i))$ – множина описів, у сенсі [1,6], семантичних Веб-сервісів, які утворюють циклічну залежність. Опис на функціональному рівні семантичного Веб-сервісу-вузла для її усунення – це пара множин його вхідних і вихідних концептів

$ck = \langle INK;\ OUK\rangle$,

які є об'єднаннями подібних множин для Веб-сервісів зі складу залежності:

$$INK = \cup\{In(sc_i),\ sc_i\in CD\};\qquad (5)$$
$$OUK = \cup\{Out(sc_i),\ sc_i\in CD\}.$$

Вираз (5) висвітлює ситуації доцільного й недоцільного застосування *сервіса-вузла* для опрацювання циклічної залежності. Саме, він ефективний в ідеальній ситуації, коли «непродуктивні» (тобто без внеску до вихідних концептів запиту) сервіси $sc_i\in CD$ високонадійні, з привабливими показниками якості й низькою трудомісткістю/вартістю використання, їх кількість відносно невелика порівняно з кількістю всіх компонентів композитного Веб-сервісу, а взаємозв'язки в межах залежності відносно прості. Тоді непродуктивні сервіси надають резервування функціональних сервісів – їх своєрідне «кріплення» , підвищуючи якість композитного Веб-сервісу. Однак у реальній «неідеальній» ситуації складна внутрішня структура сервісу-вузла і/або незадовільна якість сервісів у її складі можуть неприйнятно знизити якість композитного сервісу аж до його недоступності.

«Неідеальність» штатних ситуацій функціонування СоПС визначає потребу в альтернативних формалізмах без обмежень сервісу-вузла. Запропоновано їх визначення в руслі підходів до розв'язання циклічних залежностей за допомогою компонента-по-

середника (так званого «proxy») в розподілених компонентно орієнтованих застосунках, зокрема відомому каркасі Spring [32].

Визначення 6. Опис, на функціональному рівні, семантичного Веб-сервісу-*посередника* для усунення циклічної залежності *CD* – це пара множин його вхідних і вихідних концептів

$cp = \langle INP; OUP \rangle$,

які є, відповідно, об'єднаннями підмножин:

а) вхідних концептів сервісів зі складу залежності *CD*, семантично зіставлених вихідним концептам компонентних сервісів *sb*, які безпосередньо передують *CD* в композиції, або вхідним концептам запиту щодо неї (*INP*):

$$INP = \cup\{In^*(sc_i),\ sc_i \in CD\};$$

$$In^*(sc_i) = \{ic \in In(sc_i) \mid \exists sb \notin CD,\ x \in Out(sb) \cup IN;\ \text{Exact}(x,ic)\ !\ \text{Plugin}(x,ic)\}; \quad (6)$$

б) вихідних концептів сервісів зі складу залежності *CD*, семантично зіставлених вхідним концептам компонентних сервісів *sa* – безпосередніх наступників залежності *CD* у композиції, або вихідним концептам запиту щодо неї (*OUP*):

$$OUP = \cup\{Out^*(sc_i),\ sc_i \in CD\};$$

$$Out^*(sc_i) = \{ic \in Out(sc_i) \mid \exists sa \notin CD,\ x \in In(sa) \cup Out;\ \text{Exact}(x,ic)\ !\ \text{Subsume}(x,ic)\}. \quad (7)$$

Зауваження 1. Якщо в множині *OUK* з (5) жодний концепт не є семантично зіставленим з певним концептом компонентного сервісу поза циклічною залежністю *CD* або в (6),(7) $OUP=\varnothing$, гілку(и) з *CD* слід вилучити з графа композиції.

Згідно з виразами (6),(7), сервіс-посередник долає зазначені обмеження сервіса-вузла за допомогою двох основних відмінностей від нього:

а) відсутності «надлишкової» внутрішньої структури, тобто тих компонентних Веб-сервісів, вихідні концепти яких (з (1)) не є семантично зіставленими з деяким вхідним концептом певного компонентного сервісу поза циклом на гілці без тупиків;

б) урахування в його поданні функціонального рівня лише вхідних концептів тих сервісів у циклі, вхідні концепти яких безпосередньо пов'язані з вихідними концептами певних сервісів у шарі графа композиції, що передує шару з циклом, або з вхідними концептами запиту щодо композиції, а також вихідних концептів, семантично зіставлених вхідним концептам довільних компонентних сервісів поза циклом, розміщеним на гілках без тупиків.

Визначення 7. *Спрощення* циклічної залежності *CD* – її під-граф, отриманий вилученням Веб-сервісів, вихідні концепти яких не є семантично зіставленими з деякими сервісами – наступниками *CD*, а також вилученням з множин вхідних і вихідних концептів залишених Веб-сервісів тих концептів, які були семантично зіставлені з вхідними й вихідними концептами вилучених сервісів.

Зауваження 2. На відміну від сервісу-вузла, сервіс-посередник і сервіси зі складу спрощення потребують додаткового відшукування або компонування, насамперед базовими методами з їхнього ядра, запропонованого для розвитку моделі АКС [4] на початку статті.

Співвідношення трьох запроваджених формалізмів зручно проілюструвати на де-факто стандартному прикладі композитного сервісу з залежностями для онлайн-магазину [22]. Склад його компонентних сервісів описано в табл. 2 (вхідним концептам у дужках зіставлено сервіси, що їх продукують).

Структуру композитного сервісу показано на рис. 8. Згідно з ним, у композитному сервісі наявні дві циклічні залежності (позначені сірим кольором):

– авто-цикл S_3 (S_3 потребує вихідні концепти від себе й S_1);

– цикл з п'яти сервісів (S_7; S_8; S_{10}; S_{11} і S_{12}), де S_7 потребує вихідні концепти від S_{10}, S_{10} залежить від S_{11}, який, у свою чергу, залежить від S_8, залежного від S_{12}, тоді як S_{12} залежить від S_7.

Результати опрацювання цих залежностей за допомогою сервісу-вузла, сервісу-посередника та спрощення зіставлені на рис. 9. Ліворуч показано сервіси-вузли CN_1, CN_2 і сервіси-посередники *SP*1,*SP*2 (для авто-циклу вузол *CN*1 і посередник *SP*1 збігаються), а праворуч – чотири сервіси *SS*3, *SS*7, *SS*8, *SS*10 зі складу спрощення циклу (S_7; S_8; S_{10}; S_{11}; S_{12}), отримані вилученням з

Таблиця 2. Веб-сервіси для ілюстративного сценарію онлайн-магазину

№	Веб-сервіс	Входи	Виходи
S_1	Онлайн каталог	Вибір елемента (запит користувача)	Деталі={Вибір категорії, код товару, Вибір кількості, Ціна товару, Вартість, IP адреса}
S_2	Розміщення клієнта	IP адреса (S_1)	Розміщення клієнта
S_3	Перевірка позицій	Вибір (категорії, код товару, кількість) (S_1), Номер товару (S_3)	Наявність, кількість
S_4	Податки	Вибір категорії, вибір ціни (S_1)	Сума податку
S_5	Страхування	Вибір категорії, вибір ціни (S_1), Вибір кількості, Вартість (S_1)	Довідка про страхування, вартість
S_6	Отримання	Наявність (S_3)	Розташування
S_7	Розрахунок ціни	Ціна товару (S_1), Сума податку (S_4), Вартість страхування (S_4), Вартість доставки (S_{10})	Загальна ціна
S_8	Формування замовлення	Узгодження деталей (S_1), Загальна ціна (грн.) (S_{12})	Замовлення на придбання
S_9	Прогноз погоди	Розміщення предмету (S_6), Місцезнаходження клієнта (S_2)	Погода
S_{10}	Доставка	Погода(S_9), Узгодження деталей (S_1), Квитанція (S_{11}),	Код доставки, вартість, номер відстеження
S_{11}	Оплата	Замовлення на придбання (S_8)	Квитанція
S_{12}	Конвертер валют	Загальна ціна (S_7)	Загальна ціна в національній валюті

їх множин вхідних і вихідних концептів тих концептів, які були семантично зіставлені з вхідними й вихідними концептами «непродуктивних» вилучених сервісів S_{11}, S_{12}. Хоча структури композитного сервісу-вузла та сервісу-посередника однакові, множини їх вхідних і вихідних концептів відмінні між собою в сенсі визначень 5, 6.

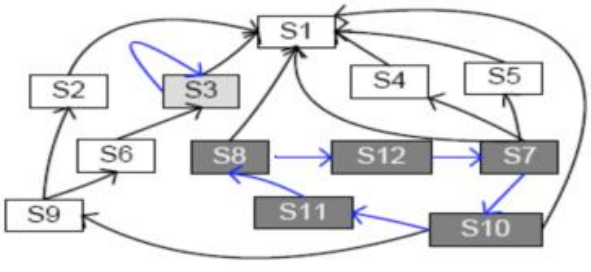


Рис. 8. Два цикли в композитному сервісі

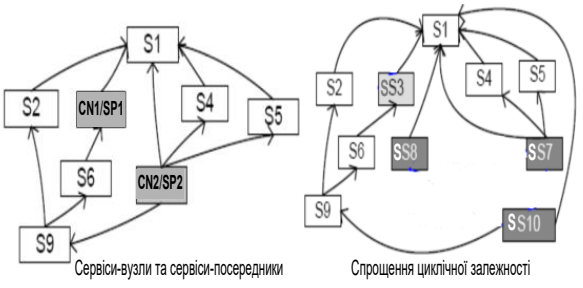


Рис. 9. Опрацювання циклічної залежності: вузол, посередник, спрощення

Як показано на рис. 9, сервіс-вузол замінює циклічну залежність, інкапсулюючи її внутрішню структуру; сервіс-посередник «вирізує» цю структуру без порушення цілісності; спрощення «розриває» залежність, залишаючи лише продуктивні компонентні сервіси з їхніми функціонально необхідними зв'язками.

Висновки

Запропоновано стале забезпечення якості адаптивного композитного семантичного Веб-сервісу за допомогою OWL-S-специфікації контексту виконання його самого і компонентних Веб-сервісів та узгодженого застосування специфікованого контексту на всіх етапах компонування, а також формалізмів сервісу-посередника й спрощення для усунення циклічних залежностей у композиції в ситуаціях неефективності попередньо запропонованого формалізму сервісу-вузла. Обґрунтовано застосування спеціального процесного числення контекстно-залежних амбієнтів (Ф.Сів) для засвідчення якості формованого адаптивного композитного семантичного Веб-сервісу за допомогою його динамічної верифікації, насамперед перевірки властивостей відповідності контексту запитувача та живості – «композитний сервіс завжди надає запитаний користувачем результат».

Застосування розроблених формалізмів сприяє підвищенню рівня відповідності композиції очікуванням запитувачів та уможливлює багаторазове уточнення базової композиції функціонального рівня, формованої на початку конструювання адаптивного композитного Веб-сервісу, на підтримку (не) передбачено змінних розподілених ділових процесів сучасних організацій.

Література

1. Андон П.І. Збіркове програмування компонентних і сервіс-орієнтованих прикладних програмних систем / П.І. Андон, О.О. Слабоспицька // Проблеми програмування. – 2017. – № 3 – С. 31–51.
2. Weyns D. et al. Perpetual Assurances for Self-Adaptive Systems. [Electronic resourse]. Mode of access: https://arxiv.org/ ftp/arxiv/ papers/1903/1903.04771.pdf.
3. De Lemos R. Software engineering for self-adaptive systems: Research challenges in the provision of assurances / R De Lemos, D Garlan, C Ghezzi et al. (Eds.) // Self-Adaptive Systems III LNCS 9640. – 2017. – P. 3-30.
4. Слабоспицька О.О. Рамкова модель адаптивного композитного сервісу в семантичному Веб-середовищі / О.О.Слабоспицька // Проблеми програмування. – 2017. – №4. – С. 51-65.
5. Слабоспицька О.О. Уніфікований процес композиції адаптивного сервісу в семантичному Веб-середовищі / О.О.Слабопицька // Проблеми програмування. – 2018. – № 1. – С. 65–76.
6. Rodriguez Mier P. An Integrated Semantic Web Service Discovery and Composition Framework / P Rodriguez Mier, C Pedrinaci, M Lama et al. //IEEE Trans. on Services Comp. – 2015. – V.9. – Is.4. – P. 537–555.
7. Bansal S. Generalized semantic Web service composition / S.Bansal, A.Bansal, G.Gupta, M.Brian Blake// Service Oriented Computing and Applications. – 2016. – V.10. – Is. 2. – P. 111–133.
8. Ben Lamine R. A Framework for the composition and formal verification of adaptable semantic Web services / R.Ben Lamine, R.Ben Jemaa, I.Amous // Proc. of ACM MoMM conf. (MoMM'18). – ACM, Yogyakarta, Indonesia, 2018. – 9 p.
9. Ben Lamine R. Formal Specification of Adaptable Semantic Web Services Composition // Int. J. of Information Technology and Web Engineering. 2018. – N 13(4). – P. 14–34.
10. Ben Lamine R. Graph Planning Based Composition For Adaptable Semantic Web Services / R.Ben Lamine, R.Ben Jemaa, I.Amous // Proc. of the 2017 Int. Conf. on Knowledge Based and Intelligent Information and Engineering Systems, KES2017. – 2017, Marseille, France. – P. 358–368.
11. Alférez G.H. Achieving autonomic Web service compositions with models at runtime / G.H.Alférez, V.Pelechano / Computers & Electrical Engineering. – 2017. – V. 63. – P./ 332-352.
12. Pistore M. A Minimalist Approach to Semantic Annotations for Web Processes Compositions / M.Pistore, L.Spalazzi, P.Traverso // Proc. European Semantic Web Conf. ESWC 2006. – 2006. – P. 620-634.
13. Bucchiarone A. et al. Domain Objects for Dynamic and Incremental Service Composition / A.Bucchiarone, M. De Sanctis, M. Pistore // In: Villari M.et al. (Eds.) Proc. ESOCC 2014, Manchester, UK – LNCS 8745, 2014. – P. 62–80.
14. Yu L. Goal-driven context-aware service composition / L.Yu, A.Glenstrup, Y.Zhang et al. – Pervasive Computing and Applications, ICPCA 2010. – 2010. – P. 342–347.
15. Li L. Semantic based aspect-oriented programming for context-aware Web service composi-

tion / L.Li, D.Liu, A.Bouguettaya // Information Systems – 2011. – V. 36(3). – P. 551–564.
16. Baidouri H. Towards a Context-Aware Composition of Services / H.Baidouri, H.Hafiddi, M.Nassar et al. // (2012). . Int.J. of Computer Science and Network Security. – 2012. – V. 12(3). – P. 133–140.
17. Mcheick H. Modele de composition des services Web semantiques par orchestration a la demande / H.Mcheick // Proc. of the 2012 25th IEEE Canadian Conference on Electrical and Computer Engineering (CCECE. 2012), Montreal, Canada. [Electronic resourse]. Mode of access: https://www.researchgate.net/publication/257527267_Modele_de_Composition_Des_Services_Web_Semantiques_par_Orchestration_a_la_demande.
18. Furno A. Context-aware composition of semantic web services / A.Furno, E.Zimeo // Mobile Networks and Applications. – 2014. – V. 19(2) – P. 235–248.
19. Yan Y. Anytime QoS-aware service composition over the GraphPlan / Y. Yan, M.Chen // Service Oriented Computing and Applications. – 2015. – V. 9(1). –, P. 1–19.
20. Boukadi K. Specification and Verification of Views over Composite Web Services Using High Level Petri-Nets / K.Boukadi, C.Ghedira, Z.Maamar et al. // Proc. of the 9th International Conference on Enterprise Information System, ICEIS 2007. – P. 107–112.
21. Madkour M. Context-Aware Service Adaptation: An Approach Based on Fuzzy Sets and Service Composition / M.Madkour, I.El Ghanami, A.Maach // J. of Inform. Sci. and Eng. – 2013. – V. 29. – P. 116–126.
22. Omer A.M. Automatic management of cyclic dependency among web services / A.M.Omer, A.Schill // Proc. of IEEE Int. Conf. on Computational Science and Engineering. – 2011. – P. 44–51.
23. Siewe F. The Calculus of Context-aware Ambients (CCA): How to program: Capabilities and Processes / F.Siewe – 2017. [Electronic resourse]. Mode of access: http://www.cse.dmu.ac.uk/~fsiewe/ccaPL/ccaPL_processes.pdf.
24. Сторінка «Calculus of Context-aware Ambients (CCA)» Дослідницької лабораторії технології програмного забезпечення Університету ДеМонфору (Лейчестер, Велика Британія). Електронний ресурс. Режим доступу: http://www.cse.dmu.ac.uk/~fsiewe/fscca.html.
25. Najar S. Service Discovery Mechanism for an Intentional Pervasive Information System / S.Najar, M.Kirsch-Pinheiro, C.Souveyet et al. // IEEE 19th Int. Conference on Web Services (ICWS). – 2012. – P. 520–527.
26. Cherif S. An Integrated context-aware Planning Approach to Self-Adaptation Web Service Composition / S.Cherif, R.Ben Djemaa, I.Amous // Proc. of the 14th Int. Conf. on Advances in Mobile Computing and Multi Media (MoMM ‘16), 2016. – P. 3–11.
27. Martin D. Bringing semantics to web services: The OWL-S approach / D. Martin, M.Paolucci et al. // Proc First Int Work Semant Web Serv Web Process Compos SWSWPC 2004. – LNCS, V. 3387, 2005. – P. 26-42.
28. Kirsch-Pinheiro M. (2008). Context-aware service selection using graph matching / M. Kirsch-Pinheiro // 2nd Non Functional Properties and Service Level Agreements in Service Oriented Computing Workshop. – 2008. [Electronic resourse]. Mode of access: http://ftp.informatik.rwth-aachen.de/Publications/CEUR-WS/Vol-411/paper7.pdf.
29. Cherif S. Adaptable Web Service Registry for Publishing Context Aware Service Composition / S.Cherif, R.Ben Djemaa, I.Amous. I. // Proc. of the 17th Int. Conf. on Information Integration and Web-based Applications & Services (IIWAS’15), Brussels, Belgium. – 2015. [Electronic resourse]. Mode of access: https://dl.acm.org/doi/pdf/10.1145/2837185.2837189?download=true.
30. Карпов Ю.Г. Model Checking. Верификация параллельных и распределенных программных систем / Ю.Г.Карпов – СПб.:БХВ-Петербург, 2010. – 560 с.
31. Siewe F. ccaPL: a Programming Language for the Calculus of Context-aware Ambients / F.Siewe – 2012. [Electronic resourse]. Mode of access: citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.725.7085&rep=rep1&type=pdf.
32. Динеш Р. Spring. Все паттерны проектирования / Р.Динеш – Издательский дом «Питер», 2018. – 320 с.

References

1. Andon P.I. Assembly programming component and service-oriented applied software systems / P.I Andon, O.O. Slabospitska // Problems in Programming. – 2017. – № 3 – P. 31–51.
2. Weyns D. et al. Perpetual Assurances for Self-Adaptive Systems. [Electronic resourse].

Mode of access: https://arxiv.org/ ftp/arxiv/papers/1903/1903.04771.pdf.
3. De Lemos R. Software engineering for self-adaptive systems: Research challenges in the provision of assurances / R De Lemos, D Garlan, C Ghezzi et al. (Eds.) // Self-Adaptive Systems III LNCS 9640. – 2017. – P. 3-30.
4. Slabospitska O.O. Reference model for Semantic Web adaptive composite service / O.O Slabospitska // Problems in Programming. – 2017. – № 4 – P. 51–65.
5. Slabospitska O.O. Unified process for adaptive Semantic Web service composition / O.O Slabospitska // Problems in Programming. – 2018. – № 1. – P. 65–76.
6. Rodriguez Mier P. An Integrated Semantic Web Service Discovery and Composition Framework / P Rodriguez Mier, C Pedrinaci, M Lama et al. //IEEE Trans. on Services Comp. – 2015. – V.9. – Is.4. – P. 537–555.
7. Bansal S. Generalized semantic Web service composition / S.Bansal, A.Bansal, G.Gupta, M.Brian Blake// Service Oriented Computing and Applications. – 2016. – V.10. – Is. 2. – P. 111–133.
8. Ben Lamine R. A Framework for the composition and formal verification of adaptable semantic Web services / R.Ben Lamine, R.Ben Jemaa, I.Amous // Proc. of ACM MoMM conf. (MoMM'18). – ACM, Yogyakarta, Indonesia, 2018. – 9 p.
9. Ben Lamine R. Formal Specification of Adaptable Semantic Web Services Composition // Int. J. of Information Technology and Web Engineering. 2018. – N 13(4). – P. 14–34.
10. Ben Lamine R. Graph Planning Based Composition For Adaptable Semantic Web Services / R.Ben Lamine, R.Ben Jemaa, I.Amous // Proc. of the 2017 Int. Conf. on Knowledge Based and Intelligent Information and Engineering Systems, KES2017. – 2017, Marseille, France. – P. 358–368.
11. Alférez G.H. Achieving autonomic Web service compositions with models at runtime / G.H.Alférez, V.Pelechano / Computers & Electrical Engineering. – 2017. – V. 63. – P./ 332-352.
12. Pistore M. A Minimalist Approach to Semantic Annotations for Web Processes Compositions / M.Pistore, L.Spalazzi, P.Traverso // Proc. European Semantic Web Conf. ESWC 2006. – 2006. – P. 620-634.
13. Bucchiarone A. et al. Domain Objects for Dynamic and Incremental Service Composition / A.Bucchiarone, M. De Sanctis, M. Pistore // In: Villari M.et al. (Eds.) Proc. ESOCC 2014, Manchester, UK – LNCS 8745, 2014. – P. 62–80.
14. Yu L. Goal-driven context-aware service composition / L.Yu, A.Glenstrup, Y.Zhang et al. – Pervasive Computing and Applications, ICPCA 2010. – 2010. – P. 342–347.
15. Li L. Semantic based aspect-oriented programming for context-aware Web service composition / L.Li, D.Liu, A.Bouguettaya // Information Systems – 2011. – V. 36(3). – P. 551–564.
16. Baidouri H. Towards a Context-Aware Composition of Services / H.Baidouri, H.Hafiddi, M.Nassar et al. // (2012). . Int.J. of Computer Science and Network Security. – 2012. – V. 12(3). – P. 133–140.
17. Mcheick H. Modele de composition des services Web semantiques par orchestration a la demande / H.Mcheick // Proc. of the 2012 25th IEEE Canadian Conference on Electrical and Computer Engineering (CCECE. 2012), Montreal, Canada. [Electronic resourse]. Mode of access: https://www.researchgate.net/publication/257527267_Modele_de_Composition_Des_Services_Web_Semantiques_par_Orchestration_a_la_demande.
18. Furno A. Context-aware composition of semantic web services / A.Furno, E.Zimeo // Mobile Networks and Applications. – 2014. – V. 19(2) – P. 235–248.
19. Yan Y. Anytime QoS-aware service composition over the GraphPlan / Y. Yan, M.Chen // Service Oriented Computing and Applications. – 2015. – V. 9(1). –, P. 1–19.
20. Boukadi K. Specification and Verification of Views over Composite Web Services Using High Level Petri-Nets / K.Boukadi, C.Ghedira, Z.Maamar et al. // Proc. of the 9th International Conference on Enterprise Information System, ICEIS 2007. – P. 107–112.
21. Madkour M. Context-Aware Service Adaptation: An Approach Based on Fuzzy Sets and Service Composition / M.Madkour, I.El Ghanami, A.Maach // J. of Inform. Sci. and Eng. – 2013. – V. 29. – P. 116–126.
22. Omer A.M. Automatic management of cyclic dependency among web services / A.M.Omer, A.Schill // Proc. of IEEE Int. Conf. on Computational Science and Engineering. – 2011. – P. 44–51.

23. Siewe F. The Calculus of Context-aware Ambients (CCA): How to program: Capabilities and Processes / F.Siewe – 2017. [Electronic resourse]. Mode of access: http://www.cse.dmu.ac.uk/~fsiewe/ccaPL/ccaPL_processes.pdf.
24. Сторінка «Calculus of Context-aware Ambients (CCA)» Дослідницької лабораторії технології програмного забезпечення Університету ДеМонфору (Лейчестер, Велика Британія). Електронний ресурс. Режим доступу: http://www.cse.dmu.ac.uk/~fsiewe/fscca.html.
25. Najar S. Service Discovery Mechanism for an Intentional Pervasive Information System / S.Najar, M.Kirsch-Pinheiro, C.Souveyet et al. // IEEE 19th Int. Conference on Web Services (ICWS). – 2012. – P. 520–527.
26. Cherif S. An Integrated context-aware Planning Approach to Self-Adaptation Web Service Composition / S.Cherif, R.Ben Djemaa, I.Amous // Proc. of the 14th Int. Conf. on Advances in Mobile Computing and Multi Media (MoMM ‘16), 2016. – P. 3–11.
27. Martin D. Bringing semantics to web services: The OWL-S approach / D. Martin, M.Paolucci et al. // Proc First Int Work Semant Web Serv Web Process Compos SWSWPC 2004. – LNCS, V. 3387, 2005. – P. 26-42.
28. Kirsch-Pinheiro M. (2008). Context-aware service selection using graph matching / M. Kirsch-Pinheiro // 2nd Non Functional Properties and Service Level Agreements in Service Oriented Computing Workshop. – 2008. [Electronic resourse]. Mode of access: http://ftp.informatik.rwth-aachen.de/Publications/CEUR-WS/Vol-411/paper7.pdf.
29. Cherif S. Adaptable Web Service Registry for Publishing Context Aware Service Composition / S.Cherif, R.Ben Djemaa, I.Amous. I. // Proc. of the 17th Int. Conf. on Information Integration and Web-based Applications & Services (IIWAS’15), Brussels, Belgium. – 2015. [Electronic resourse]. Mode of access: https://dl.acm.org/doi/pdf/10.1145/2837185.2837189?download=true.
30. Карпов Ю.Г. Model Checking. Верификация параллельных и распределенных программных систем / Ю.Г.Карпов – СПб.:БХВ-Петербург, 2010. – 560 с.
31. Siewe F. ccaPL: a Programming Language for the Calculus of Context-aware Ambients / F.Siewe – 2012. [Electronic resourse]. Mode of access: citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.725.7085&rep=rep1&type=pdf.
32. Dinesh R. Spring 5 Design Patterns. / R. Dinesh – Publishing house «Piter», 2018. – 320 p.

Отримана 28.11.2021

Про авторів:

Андон Пилип Іларіонович,
академік НАН України, д-р ф.-м. н.,
директор,
кількість публікацій в українських виданнях – більше 400,
кількість публікацій у закордонних виданнях – 10,
номер ORCID –_0000-0002-2204-1554

Слабоспицька Ольга Олександрівна,
кандидат фізико-математичних наук,
с.н.с., ст. наук. співроб.,
кількість публікацій в українських виданнях – більше 50,
кількість публікацій у закордонних виданнях – 7,
номер ORCID – 0000-0001-6556-0947

Місце роботи авторів:

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

УДК004.6 http://doi.org/10.15407/pp2021.04.019

O. Novytskyi

METHODS AND TECHNIQUES FOR MANAGEMENT OF ONTOLOGY-BASED KNOWLEDGE REPRESENTATION MODELS IN THE CONTEXT OF BIG DATA

Ontology-based knowledge representation models in the context of big data are one way to reduce complexity for data processing across methods of semantic description. This research paper aims to provide an overview of the methods and techniques for efficient management of the ontology-based models that improve big data systems. For this case, the shapes constraint language (SHACL) for information validation was reviewed as the key method. The knowledge representation systems and reasoners are studied and reviewed in the paper as well. The author describes approaches based on ontologies in the context of big data. The proper management of ontology-based knowledge representation models through offered methods and techniques brings improved data integration, big data quality, and business process integration.

Key words: ontology-based model, ontologies, big data, reasoners, representation system, ontologies, shapes constraint language, information validation, ontology-based knowledge representation models.

Introduction

Big data means complex data sets that are unable to process adequately via traditional data applications. Big data management is handled by special-purpose resource planning systems called enterprise information systems. These systems represent business processes adequately and force the overall cost-effectiveness [1]. Modern enterprises are focused on the enterprise-wide centralized information system to validate and integrate large amounts of complex data. To capture and represent complex and big data, ontology-based knowledge representation models are used. One of the factors which impact big data processing is the complexity of understanding the data. Semantic technology is allowed to automatically recognize data. This article is to explain the approach of using ontologies for big data to ensure a common understanding of information. The ontology-based knowledge representation models make explicit domain assumptions [2].

Querying information in the context of big data becomes accessible for large enterprises. Ontologies bring detailed and meaningful distinctions between relationships, classes, and properties. The paper is devoted to ontology-based modeling and its management in semantic graph databases. Big data quality is improved with the help of ontology-based knowledge representation models and the reasoners that enable consistency and satisfiability checks [3].

The research paper also reviews an alternative using ontologies to model data. SHACL (shapes constraint language) is overviewed to demonstrate the benefits of this method for information validation in the triplestore and for validating RDF graphs against a set of constraints. The overview of the OWL reasoners and RDF graph capture systems is used as the guide for big data players (large enterprises, structure that is in the stage of developing a large-scale centralized database) on how to manage ontology-based models and improve the data quality with the help of automated reasoning of the information in the semantic graph database.

OWL Reasoners for ontologies.

The research paper reviews the main two reasoners with the wide range of optimizations that benefit big data improvements. They contain updated algorithms and tableaux algorithms that are native to the ontology-based knowledge representation models.

FaCT++ is one of the newest reasoners that is designed to implement tableaux algorithms and updated heuristic optimization techniques. The table of characteristics of the FaCT++ reasoner is given below.

Description	A new highly-optimized reasoner with tableaux-based SROIQ algorithms
License	LGPL v2
Semantics	OWL DL Classification OWL EL Classification OWL DL Consistency OWL EL Consistency OWL DL Realization OWL EL Realization

Table 1. FaCT++ Characteristics. Source: ORE

The FaCT++ reasoner implementation starts with the preprocessing stage. It is applied to the knowledgebase and can be transformed according to the internal representation requirements. FaCT++ performs classification. With the help of applied optimizations, the FaCT++ reasoner is used to reduce the quantity of subsumption tests to be performed [4].

The main application of the FaCT++ optimizations is to transform concepts into SNF. The simplified normal form lets users implement negation, conjunction, universal restriction, at-most restrictions. The main FaCT++ features for big data optimization are:

(a) Absorption – is suitable for rewriting optimization. There are concept and role absorption techniques to take into consideration. The concept absorption is responsible for GCIs elimination via concept definition axioms. The role absorption eliminates GCIs in the concept-free mode.

(b) TCE (Told Cycle Elimination) – the technique for text optimization. This cycle is often eliminated together with definitional cycles. The user can undertake TCE and definitional cycles with the help of axiom transformations.

(c) Synonym Replacement – this FaCT++ technique aims at extending simplification properties. Synonym Replacement improves clash detection in the early stage. The knowledgebase is transformed in the context of synonym elimination with the help of axioms.

The FaCT++ reasoner is used for satisfiability checking optimizations. New ordering heuristics are available for the implementation of new optimization methods. There is a special-purpose To-Do list. The user can force entry assortment with the help of the FaCT++ To-Do algorithm. It is worth noting that the reasoner provides the Backjumping optimization. The tree label matters when the dependency set of information items is formed. Boolean optimization that is available with the help of the FaCT++ reasoner allows users to implement constant propagation (BCP) [5].

HermiT is the reasoner for ontology-based knowledge representative models. It is used for the identification of subsumption relationships (between classes and other specifications). This reasoner is public and available for the users without restrictions. The notable feature of the HermiT reasoner is its new versions with the updated reasoning algorithms [5], [7].

The main HermiT characteristics are given in the table below [8]

Description	The conformant reasoner for the ontology-based knowledge representation models. The HermiT uses direct semantics. It is based on the hyper-tableaux algorithms.
License	LGPL 3.0
Semantics	OWL DL Classification, OWL EL Classification OWL DL Consistency OWL EL Consistency OWL DL Realization OWL EL Realization

Table 2. HermiT Characteristics

The HermiT reasoner allows users to classify the ontology-based knowledge representation models faster. The manual classification often takes hours. The reasoner makes it possible to classify even big data knowledgebase and complex information for minutes.

The HermiT reasoner uses direct semantics for optimization processes and hypertableaux algorithm implementation. The last version of the reasoner is called HermiT 1.3.8. Besides the main function of DL Safe rule handling, the new version of reasoner allows big data players to add new rules directly to the ontology-based models [6].

The number of optimization techniques of the HermiT reasoner is similar to the FaCT++ one described in the research paper above. The

significant feature of the HermiT reasoner is the high-level DL Safety rules compliance. DL Safety rules will be considered incomplete if:

a) the knowledgebase contains property chains in the rule bodies;

b) the KB includes transitivity axioms in the rule bodies of the knowledgebase;

c) the complex properties are used in the rule bodies of the ontology-based model.

The HermiT reasoner is one of the newest reasoners that is recommended for ontology-based knowledge representation model management in the context of big data. The direct semantics use and hyper-tableaux algorithm approach improve the quality of data and simplify business processes related to ontologies and semantics.

RacerPRO is the improved version of the former Racer knowledge representation system. As above-described reasoners and other programs suitable for ontology-based knowledge representation model management, RacerPRO is used for optimized tableaux algorithm implementation. The description logic of $\mathcal{SHIQ}(\mathcal{D}^{-})$ is used for this knowledge representation system.

Description	Racer is a knowledge representation system that implements a highly optimized tableau calculus for a very expressive description logic. It provides the reasoning for T-boxes and A-boxes as well.
License	-
Semantics	OWL DL Classification, OWL EL Classification OWL DL Consistency OWL EL Consistency OWL DL Realization OWL EL Realization

Table 3. HermiT Characteristics

The RacerPRO license is BSD 3-clause. This system is required for big data projects because it is the separate-standing knowledge representation system for solving main reasoning problems [9].

The reasoning procedure takes place in the streaming model that is suitable for complex data proceedings. Both T-boxes and A-boxes often include issues to solve when it comes to knowledge representation. RacerPRO solves these reasoning problems with the help of standard tableaux algorithms and unique interference services (e.g. logical abduction). The architecture of the latest version of the RacerPRO system is presented in Figure 1.

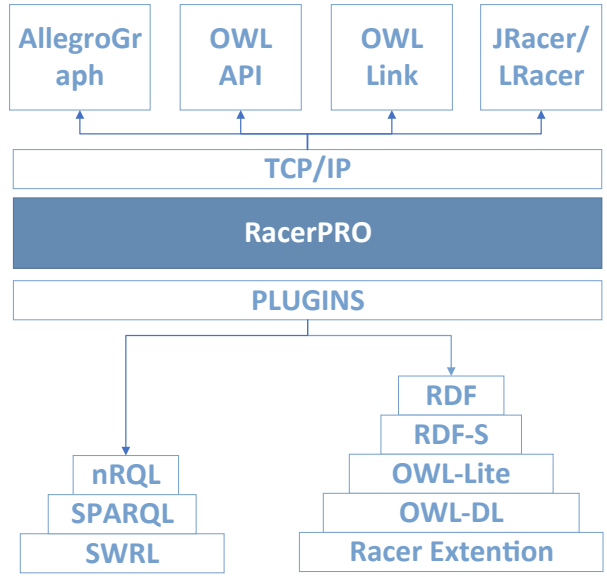


Figure 1. RacerPRO Architecture.

The additional benefit of the RacerPRO reasoning and knowledge representation system is its query language called nRQL. Using the new Racer Query Language means supplementary assistance when it comes to ontology-based model management:

attribute values of different individuals;

improved propertied for string attributes;

negation-as-failure support.

Reasoning over ontology as a rule is a complex task. In real tasks for reasoning, we have to store facts in RDF. Therefore, in the next part, we make a short review semantic reasoner with RDF storage.

Snorocket. This is the special-purpose algorithm based on the healthcare terminology classifiers. Snorocket will be suitable for big data projects related to the clinical, medical, healthcare, science directions [10]. Snorocket is not a multifunctional solution for ontology-based knowledge representation model management. This is suitable for working with ontology related to medical data only.

Snorocket is available for users in the extension format. The classifiers of the algorithm allow healthcare representatives to manage semantic data related to medi-

cal terminology. Big data projects based on healthcare or medical content, imagery, and other information can benefit from using Snorocket. Nevertheless, this extension with the implementation of the unique Dresden algorithm is not suitable for any other knowledgebase. The limited ways of the application make Snorocket the last RDF store system in the list of top ones overviewed in the research paper.

Methods for RDF graphs validation. Shapes Constraint Language (SHACL)

RDF is a main part of the Semantic Web. Its simple data model provides powerful expressiveness which can be applied to represent information in any scope. Practical Semantic Web applications require some technology to describe and validate the RDF data [11]. One of such technology for RDF is SHACL [12], [13], which has developed to model some restrictions in the form of constraints on data.

The shapes constraint language (SHACL) is considered as the alternative to traditional ontologies that are used for data modeling. SHACL is used for RDF graphs validation. There is a set of constraints that are applicable to the validation process. SHACL includes shapes that specify metadata according to its resource. The big data knowledgebase is compliable with the shapes constraint language. The special-purpose shape specifies the resource in the context of big data as well. This resource can be the principle of data use, the reason for data use, and the frequency of data use. The SHACL data validation process is applicable for both unavailable and available data in the triplestore. The shape constraint language conditions are called shapes expressed in the RDF graph format. The main purpose of the SHACL data validation is to check information according to the range of conditions. Those pieces of data that meet the shape constraints can be viewed as a description of data graphs. It is worth noting that SHACL-generated descriptions based on the shape constraint language validation of graphs can be used out of the validation process [12].

It makes SHACL the key method for ontology-based knowledge representation model management. The ready-done descriptions with the help of shape constraint language validation algorithms can be implemented in the context of big data:

- for code generation;
- for data generation.

These descriptions are suitable for code building that is one more technique out of the validation process. The separate-standing aspect to take into account is the relationship between SHACL and RDFs inferencing. The shape constraint language includes the property entailment to identify the interference specifications. To protect the knowledgebase items and bring a smooth validation process, it is recommended to use only verified RDF resources to proceed in SHACL RDF-based technologies [13].

The SHACL validation is recommended for big data because, in comparison with the standard ontologies and semantic techniques, this is the efficient way to avoid ontology limitations (limited set of property constructs). The RDF resource validation is suitable for ontology-based knowledge representation models in the context of big data for its shape-generated failure determination and data improvement properties.

Reasoners with built-in RDF store features

The range of special-purpose databases for graphs that store triples is called the RDF database. It is worth noting that triples or RDF databases are considered as data points. These points are represented in the SPO relationship (subject-predicative-object relationship). All the data items are stored in the same format – a triple format. The database receives and uses information and stores it in triple form. The RDF database is suitable for ontology-based knowledge representation management in the context of big data because all the complex information is well-organized with the help of triple sets.

One more reason to use the RDF database as the ontology-based model management when it comes to big data is the convenience to display information in graphs provided by this type of database. To carve the graphs from the triple database, any query language is used. The functionality and flexibility of the RDF database benefit enterprise-centric knowledgebase and big data projects.

Not all the databases can be included in the category of triple ones. There is a range of requirements for the digital product to be named as the RDF database. The main features of the triple database to potential-to-inclusion products are:

a) sufficient data storage is provided;

b) data as recorded as triples;

c) users are allowed to retrieve the data with the help of query language.

These are features of average RDF store. For the purpose to determine the most efficient triple databases for ontology-based knowledge representation model management.

HyLAR is the special-purpose reasoner for ontology-based knowledge representation models that contains RDF-based libraries. These libraries obtain a wide range of functionality for ontology-based model management. The HyLAR reasoner can be considered as the supplementary reasoning engine for big data. Its rdfstore.js, SPARQLs, and RDF-ext libraries are used as the triple databases [14].

The HyLAR reasoner is available in three versions to implement for the knowledgebase:

a) NPM module

b) A server-based solution

c) Browser version.

The HyLAR reasoner with its RFD libraries supports business database rules. This is one more reason to use the HyLAR-based database for big data projects. The database processing generated by the HyLAR reasoner and its RDF-based libraries is presented in the infographics below [14].

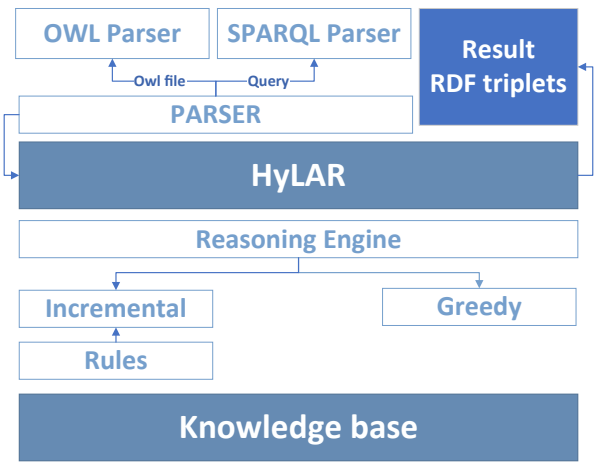


Fig 2. HyLAR Architecture.

HyLAR is used as the reasoning engine combined with the OWL and SPARQL parsers. The reasoner brings results in the format of triples that is well seen on Fig 2. The knowledgebase with ready-done available triples can be applied together with the HyLAR reasoning engine for conversion and creation of enterprise-centralized big data projects with qualitative and checked information.

Apache Jena. The Jena open-source Java framework includes a special-purpose RDF API. Jena contains information only in the format of RDF triples. The collection of RDF triples forms the general database and is included in the Jena data structure called *Model*. Jena is optimal for ontology-based knowledge representation model management because the data structure model of this framework easily determines PDF graphs and provides them relations. The database is well-structural and easy-to-navigate which is required for big and complex data [15].

The way on how the relationships go in the one-direction mode through the triple coding exemplified in Fig 3 [16].

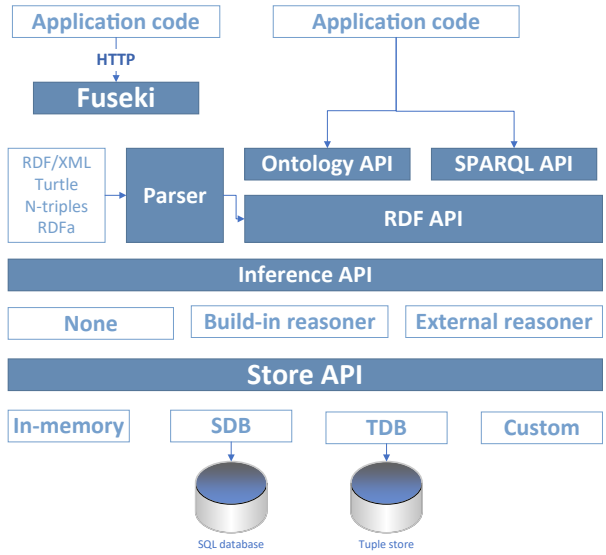


Fig 3. Jena Architecture

Another benefit of Jena in the context of big data is the availability of both RDF and Ontology APIs. The distinct concepts of the framework with the RDF-based triple collection are its opportunity to build up direct relations between graphs (nodes) in the structure, rich-in-functions APIs that provide sufficient management tools for the ontology-based knowledge representation models, and big data

orientation with Jena's in-memory structures in combination with intended methods of the complex data simplification.

RDFox is the semantic reasoning engine with the functionality of the RDF triple store. This one of the core systems for big data with its unique conception of shared memory parallel reasoning [17].

RDFox is notable with memory-economical properties that are suitable for enterprise-centric knowledgebase and big data projects. About 1.5 billion triples can be stored in 50 GB of the RDFox RDF store. The following table presents the main characteristics of the RDFox reasoning engine.

Description	The latest version of the former RDFox semantic reasoning engine. The latest version was launched in 2021. It contains a triple store that is suitable for knowledge representation purposes.
Additional Features	Rule reasoner OWL reasoner RDFS Reasoner
Semantics	RDF OWL SPARQL

Table 4. RDFox Characteristic.

The ontology-based knowledge representation model management in the context of big data can be undertaken through the RDFox semantic reasoning engine. The key benefit of the system is its triple store with memory-efficient stock. Additionally, big data projects can benefit from RDFox named graphs, Data-log extensions, and incremental update & aggregation.

Conclusions

The ontology-based knowledge representation models bring strong benefits to the digital world. The big data sphere is not the exception. Essential relationships between concepts automated data reasoning, and semantic advantages are key benefits of the ontology-based models for big data projects. But ontologies require proper management when it comes to accurate knowledge representation. The current research paper faces the problematics of the poor ontology-based knowledge representation model management in the context of big data.

The most top-ranking systems, reasoners, and other digital solutions were overview by the author. The best ones in the article are described in the research paper. Besides theoretical information given in the general paragraphs about reasoners, shape constraint language, and triple database, there is an analytical background for each reviewed solution. Pros & cons are presented under the description of the reasoners and reasoning engines. According to the undertaken research, the ontology-based knowledge representation models in the context of big data can be easily managed by the reasoning engines, reasoner extensions, query languages, hyper-tableaux algorithms, SHACL implementation, RDF database usage. The future prospects of digital transformation and new technique and method development with a focus on big data are real. The ontology-based knowledge representation models are successfully managed by the digital solutions now. The huge progress in the big data-driven direction is predicted over the coming decade.

Reference

[1] B. Mouad, «An Evaluation and Comparative study of massive RDF Data management approaches based on Big Data Technologies,» *International Journal of Emerging Trends in Engineering Research,* т. 7, pp. 48-53, 2019.

[2] Y. Sure-Vetter, S. Staab та R. Studer, «Methodology for Development and Employment of Ontology Based Knowledge Management Applications.,» *ACM SIGMOD Record 3,* т. 4, № 31, pp. 18-23, 2002.

[3] P. Haase та L. Stojanovic, «Consistent Evolution of OWL Ontologies,» *The Semantic Web: Research and Applications,* pp. 182-197, 2005.

[4] T. Dmitry , «Incremental and Persistent Reasoning in FaCT++,» в *ORE*, 2014.

[5] T. Dmitry та H. Ian , «FaCT++ description logic reasoner: System description,» в *International joint conference on automated reasoning*, Berlin, 2006.

[6] R. Shearer, B. Motik та I. Horrocks, «HermiT: A Highly-Efficient OWL Reasoner,» *Owled,* т. 432, p. 91, 2008.

[7] Data and knowlege group. University of Oxford., «HermiT OWL Reasoner,» [online]. Available: http://www.hermit-reasoner.com/. [Date: 05 2021].

[8] D. Michel , G. Birte , G. Rafael , H. Matthew, J.-R. Ermesto, N. Matentzoglu та P. Bijan , «ORE Live Competition,» 05 2021. [online]. Available: http://dl.kr.org/ore2015/vip.cs.man.ac.uk_8008/reasoners.html.

[9] V. Haarslev, «The RacerPro knowledge representation and reasoning system.,» *Semantic Web,* т. 3, № 3, pp. 267-277, 2012.

[10] M. J. Lawley та C. Bousquet, «Fast classification in Protégé: Snorocket as an OWL 2 EL reasoner.,» в *6th Australasian Ontology Workshop (IAOA'10). Conferences in Research and Practice in Information Technology*, 2010.

[11] G. Jose Emilio Labra, «Validating and Describing Linked Data Portals using RDF Shape Expressions.,» в *LDQ@ SEMANTICS*, 2014.

[12] J. Corman, J. L. Reutter та O. Savković, «Semantics and validation of recursive SHACL,» в *International Semantic Web Conference*, Cham, 2018.

[13] W3C, «Shapes Constraint Language (SHACL),» 20 July 2017. [online]. Available: https://www.w3.org/TR/shacl/. [Date: 05 2021].

[14] M. Terdjimi, M. Lionel та M. Mrissa, «Hylar: Hybrid location-agnostic reasoning.,» в *ESWC Developers Workshop 2015*, 2015.

[15] A. Ameen, K. Ur Rahman Khan та R. B. Padmaja, «Reasoning in semantic web using Jena.,» *Computer Engineering and Intelligent Systems,* т. 5, № 4, pp. 39-47, 2014.

[16] Apache Software Foundation, «Jena architecture overview,» [online]. Available: https://jena.apache.org/about_jena/architecture.html. [Date: 05 2021].

[17] Oxford Semantic Technologies, «RDFox,» [online]. Available: https://www.oxfordsemantic.tech/product. [Date: 05 2021].

Received: 27.10.2021

About author:

Oleksandr Novytskyi,
PhD, Researcher.
Number of scientific publications
in Ukrainian journals – 13.
https://orcid.org/0000-0002-9955-7882.

Місце роботи автора:

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

УДК 681.3 http://doi.org/10.15407/pp2021.04.026

Ю.В. Рогушина, І.Ю. Гришанова

ДОСЛІДЖЕННЯ ПРИНЦИПІВ, МОДЕЛЕЙ ТА МЕТОДІВ ПАРАДИГМИ МЕНЕДЖМЕНТУ НАУКОВИХ ДАНИХ FAIR ДЛЯ АНАЛІЗУ МЕТАДАНИХ BIG DATA

Розглянуто базові принципи, моделі та методи парадигми менеджменту наукових даних FAIR (Findable, Accessible, Interoperable, Reusable як окремого випадку великих даних (Big Data), яка орієнтована на повторне використання результатів наукових досліджень. Проаналізовано, як властивості даних FAIR сприяють уніфікації й об'єднанню наукової інфраструктури у парадигмі відкритої науки. Запропоновано методи та програмні засоби, за допомогою яких властивості даних FAIR можуть відтворюватися у семантично розмічених Wiki-ресурсах, що побудовані на основі Semantic MediaWiki.
Ключові слова: метадані, Big Data, семантичні Wiki-ресурси.

Вступ

Цифрові технології усе ширше проникають у різні галузі, загострюючи проблему керування великими даними, вимагаючи оптимізації методів і підходів до обробки даних, а також ефективних способів збору даних. Одним з окремих випадків Big Data [1] є наукові дані великого обсягу. Незважаючи на те, що термін «великі дані» частіше пов'язують із соціальними мережами та фінансовою індустрією, спочатку великі дані генерувалися в рамках широкомасштабних наукових проєктів – зокрема, проєкту Великого адронного колайдера, що вимагало створення принципово нових засобів і методів обробки екстремально великих обсягів відомостей, які генерувалися у ході експериментів.

Тому, здійснюючи дослідження щодо застосування сучасних методів керування знаннями для інтелектуальної обробки Big Data та їх метаданих [2], доцільно проаналізувати існуючі підходи до подання та обробки великих наукових даних.

У поєднанні з величезною кількістю даних, які сьогодні необхідно обробляти в наукових і медичних дослідженнях, важлива системна вимога полягає в тому, щоб дані не губилися. Поширеним підходом для запобігання втраті даних є збереження їх у каталозі даних. Каталоги даних допомагають організовувати, структурувати та відстежувати метадані та згенеровані дані, щоб інформацію можна було зберігати та обмінюватися в межах організації. Використання каталогів даних може навіть привести до того, що вчені отримають більше цитат, оскільки вони створюють можливості для розробки або повторного використання попередніх досліджень.

Наприклад, каталог даних значно полегшує пошук відповідних даних. Щоб дозволити всій науковій спільноті отримати якнайбільшу користь від даних досліджень, повторне використання даних слід покращити надійним способом, захищаючи як виробника даних, так і зовнішніх повторних користувачів.

Підвищуючи якість і порівнянність даних досліджень, колеги-вчені повинні мати можливість повторно використовувати певний набір даних. Встановлення довіри між виробниками даних і зовнішніми повторними користувачами даних є проблемою, яка вимагає більш серйозних змін у поведінці вчених, ніж просто збільшення додаткових метаданих у наборі даних.

Щоб сприяти необхідним змінам, сьогодні кілька фінансових агенцій вимагають, аби одержувачі гранту надали план управління даними або план управління даними, з описом того, яким чином дані будуть доступні для колег-дослідників. Аналіз публікацій показує, що сьогодні не існує єдиної форми представлення великих наукових даних, доступної для комерціалізації, що ускладнює одержання вигоди від інвестицій у дослідницькі інфраструктури.

Саме в науці довелося вперше розбиратися зі збереженням і передачею великих масивів даних, з питаннями дотримання прав їхніх власників, створення безпечного інформаційного і правового середовища для користувачів наукового устаткування, обліку соціальних наслідків упровадження нових технологій тощо. В інших областях при роботі з Big Data акцент ставиться на ефективності використання конкретних методів і їхньої максимальної універсальності, а не на забезпеченні відкритості і доступності наявних даних. Детальний огляд таких досліджень наведено в [3].

Тому значний інтерес викликають FAIR – принципи керування даними без втручання користувача, що можна розглядати як один із перших кроків до формування цифрової інфраструктури для трансферу наукових результатів у форму, зрозумілу інвесторам, чиновникам, суспільству і придатну для контролю за обсягами наукових даних.

FAIR дані (FAIR_data) – це дані, які відповідають принципам знаходжуваності, доступності, інтероперабельності та повторного використання [4].

У березні 2016 консорціум науковців і організацій визначив базові принципи «FAIR Guiding Principles for scientific data management and stewardship", де був введений відповідний акронім FAIR (Findable, Accessible, Interoperable, Reusable) для зручності ведення дискусії.

Властивості FAIR даних

Дані FAIR мають наступні властивості:

1. *Findable*. Щоб використовувати дані, їх необхідно спочатку знайти там, де вони зберігаються. Метадані та дані повинні бути легко доступними як для людей, так і для комп'ютерів. Можливість машинної обробки метаданих є важливим для автоматичного виявлення наборів даних і служб, тому це важливий компонент процесу FAIRification.

F1. (Мета)даним призначається глобально унікальний і постійний ідентифікатор.

F2. Дані описуються докладними метаданими (визначені в R1 нижче).

F3. Метадані чітко і явно містять ідентифікатор даних, які вони описують.

F4. (Мета)дані реєструються або індексуються в пошуковому ресурсі.

Тож, для вирішення задачі пошуку даних, такі дані і додаткові матеріали мають мати достатньо повні метадані, мета-опис і унікальний постійний ідентифікатор.

2. *Accessible*. Коли користувач знаходить необхідні дані, він/вона повинен знати, як до них отримати доступ (можливо, включаючи аутентифікацію та авторизацію).

A1. (Мета)дані можна отримати за їхнім ідентифікатором за допомогою стандартизованого протоколу зв'язку.

A1.1 Протокол є відкритим, безкоштовним і універсальним.

A1.2 Протокол припускає процедуру аутентифікації та авторизації, якщо це необхідно.

A2. Метадані доступні, навіть якщо дані не доступні.

Задача доступності формулюється так, що метадані (з метаописами) і самі дані мають бути зрозумілі для людини та придатні для програмної обробки. Дані повинні зберігатися в надійному репозиторії.

3. *Interoperable*Зазвичай дані потрібно інтегрувати з іншими даними. Крім того, дані мають взаємодіяти із застосунками або робочими процесами для аналізу, зберігання та обробки.

I1. (Мета)дані використовують офіційну, доступну, спільну та широко застосовану мову для представлення знань.

I2. (Мета)дані використовують словники, які відповідають принципам FAIR.

I3. (Мета)дані включають кваліфіковані посилання на інші (мета)дані. Таким чином, задача інтероперабельності (сумісності) може бути вирішена за умови, коли для метаданих використовується формальна, доступна та широко вживана мова подання знань.

4. *Reusable*. Кінцевою метою FAIR є оптимізація повторного використання даних. Щоб досягти цього, метадані та дані повинні бути добре описані, щоб їх можна було відтворювати та/або комбінувати в різних налаштуваннях.

R1. Мета(дані) повинні бути детально описані набором точних і відповідних атрибутів.

R1.1. (Мета)дані видаються з чіткою та доступною ліцензією на використання даних.

R1.2. (Мета)дані пов'язані з їх походженням.

R1.3. (Мета)дані відповідають стандартам спільноти, що стосуються домену.

Отже, вимога повторного використання говорить про те, що дані і колекції мають однозначні ліцензії, які описують їх використання та чітку інформацію про джерело даних та їхнє походження.

Наприклад, агентство DARPA, що традиційно працює над проблемами інтеграції фундаментальних наукових праць із прикладними рішеннями, реалізує проєкт Automating Scientific Knowledge Extraction, що спрямований на автоматизацію процесів здобуття наукових знань з визначенням місцезнаходження нових інформаційних ресурсів, а також їхнього аналізу з метою отримання нових знань і генерації нових моделей.

У рамковій програмі ЄС із розвитку наукових досліджень і технологій «Горизонт — 2020» продемонстрована необхідність у створенні нових методів і підходів до обробки даних, таких як персоналізація та деперсоналізація даних, миттєвий збір даних тощо, рішення яких неможливе без ефективної організації керування потоками даних.

Для подолання різнорідності БД, для уніфікації й об'єднання наукової інфраструктури 2016 року було створено портал EOSC (European Open Science Cloud) [5] – віртуальне середовище із вільним доступом для збереження, керування, аналізу і передачі даних із усіх сфер знань в усі країни ЄС. Наукова цифрова інфраструктура ЄС містить множину регламентованих, відкритих, але спеціалізованих БД і репозиторіїв: BioMA, Global Marine Information System (GMIS), Central Core DNA Sequence Information System (CCSIS) і ін. Подібні ресурси постійно актуалізуються, мають чіткі регламенти представлення даних наукових досліджень, надають інструменти і механізми для керування контентом, однак тематика даних обмежена, а правила представлення метаданих не погоджені (різнорідні). Спроби створення універсальних сховищ даних, незалежних від тематики досліджень, призводять до розбалансування системи збереження, тому що репозиторії не мають обмежень щодо формату представлення даних і дескрипторів метаданих. Внаслідок цього інформаційна система ускладнюється, втрачає гнучкість і не забезпечує ефективного пошуку даних і їхнього повторного використання.

На рішення подібних проблем націлена ініціатива Go FAIR, що містить базові принципи поліпшення можливостей пошуку, забезпечення доступу до даних, їхньої сумісності і, що особливо важливо, повторного використання [6].

Згідно FAIR, функції пошуку, здобуття і представлення даних реалізують не користувачі, а інформаційна система. При цьому мова йде не тільки про власне дані і метадані, а й про алгоритми та інструменти керування ними. Крім того, до розробки підходів щодо керування науковими даними залучаються всі зацікавлені сторони: науково-дослідні організації й окремі вчені; оператори баз даних і видання, що публікують наукові статті і результати експериментів; організації, що фінансують ці наукові дослідження; виробники програмного забезпечення й інструментів обробки даних; компанії, що надають послуги з аналізу й інтерпретації даних. Важливо, що в коло зацікавлених сторін також включаються самі обчислювальні системи (алгоритми обробки даних) як самостійний об'єкт — залежно від їхнього рейтингу приймається рішення про включення обчислювального методу до конфігурації [7].

Для підтримки пошукових функцій (серед даних і метаданих) інформаційному блоку надається унікальний постійний глобальний ідентифікатор, а самі дані описуються розширеною множиною метаданих, які однозначно і явно включають ідентифікатор описуваних даних. Дані (та метадані) реєструються чи індексуються в доступному для пошуку ресурсі.

Для оптимізації доступу до даних потрібно керуватися наступними засадами: дані (метадані) можуть бути отримані за їхнім ідентифікатором за стандартизованими протоколами зв'язку; протокол до-

ступу до даних – відкритий і передбачає використання уніфікованого протоколу доступу — за необхідності для доступу до даних використовується процедура аутентифікації й авторизації, а метадані можуть бути доступні навіть за відсутності доступу до самих даних.

Має бути забезпечена сумісність даних не тільки з іншими даними, а й із застосунками та інструментами для їх аналізу, збереження й обробки: дані (метадані) використовують формальну, доступну і поширену мову опису даних; дані (метадані) використовують словники, що реалізовані відповідно до керівних принципів FAIR; дані (метадані) містять у собі повні посилання на інші дані (метадані).

Кінцева мета FAIR — оптимізація повторного використання даних та їх об'єднання в різних задачах: дані (метадані) докладно описують із застосуванням набору однозначних і релевантних атрибутів; дані (метадані) супроводжуються чіткою і доступною ліцензією на їхнє використання; дані (метадані) мають детальну історію їхнього походження; дані (метадані) подаються у відповідності зі стандартами тематичного наукового співтовариства.

Представлені елементи даних і метаданих взаємопов'язані, але водночас незалежні й відокремлені. Кожен з них визначає сукупність метрик (характеристик) – вимог, які передаються ресурсам, інструментам, словникам обробки даних для забезпечення їх повторного використання третіми сторонами, у тому числі коли вони не мають прямого відношення до науки. Водночас існує можливість керування рівнем входження в озера даних FAIR тих чи інших користувачів за рахунок градації у процесі визначення характеристик наданих ресурсів. Варіюючи і комбінуючи метрики опису об'єктів, можна досягти високого ступеня адаптивності представлення даних і метаданих в інформаційній системі. .

Керівні принципи FAIR не потребують будь - якої стандартизації чи конкретної технології підтримки. Принципи виступають як керівництво для створення даних для озер даних з урахуванням функціональності їх пошуку, доступності, сумісності і повторного використання.

Наведені принципи стосуються трьох типів сутностей: дані (або будь-який цифровий об'єкт), метадані (інформація про цей цифровий об'єкт) та інфраструктура. Наприклад, принцип F4 визначає, що і метадані, і дані реєструються або індексуються в ресурсі з можливістю пошуку (компонент інфраструктури).

У цілому FAIR подібний до open data, але існує ключова відмінність. Відкриті дані доступні кожному без будь-яких ліцензійних обмежень, угод, авторських прав чи патентів, тоді як FAIR допускає можливість доступу до даних (метаданих) у певний час і за певних умов. Інакше кажучи, FAIR-дані можуть бути як відкритими, так і частками, якщо вони доступні лише визначеній групі користувачів. Такий підхід є більш гнучким і дозволяє характеризувати дані на кожному етапі їхнього життєвого циклу.

Наприклад, у процесі фізичного експерименту дані доступні тільки групі експериментаторів, потім — науковому співтовариству з метою їхньої інтерпретації, а після обробки переходять у загальний доступ (open data) як результат експерименту. На практиці наукові дані неодноразово переходять через такі стадії «відкритості». У переважній більшості випадків персональні і комерційні дані не можуть бути загальнодоступними, це суперечить ідеям open data, але допустимо в FAIR.

Зараз багато європейських дослідницьких інфраструктур (DTU Library, International Neuroinformatics Coordinating Facility, TU Dublin, Biobanking and Biomolecular Resources Research Infrastructure of Czech Republic, Radboud University тощо) використовують концепцію FAIR для надання доступу до своїх наукових даних. Створено і розвиваються методичні рекомендації та інструкції з представлення даних відповідно до FAIR.

У рамках програми «Горизонт-2020» ініційовано проєкт PaNOSC, що поєднує шість великих європейських дослідницьких інфраструктур (ESRF, European XFEL, CERIC-ERIC, ELI Delivery Consortium, ESS, ILL) для розвитку Європейської хмари відкритої науки (European Open Science Cloud) — універсального міждисциплінарного репозиторію наукових даних із відкритим

доступом для дослідників у всіх галузях. У рамках PaNOSC дослідникам з таких галузей, як хімія, біологія, матеріалознавство тощо надаються сервіси й інструменти для збереження, пошуку й аналізу даних, отриманих на нейтронній і фотонній дослідницьких інфраструктурах.

За рахунок використання постійних унікальних ідентифікаторів реалізується можливість передачі метаданих між сервісами. Це дозволяє збільшити на порядок можливість повторного використання результатів наукового дослідження в масштабах прямо не зв'язаних тематичних галузей наукових досліджень. У перспективі мова йтиме про забезпечення для усього світового наукового співтовариства, незалежно від тематики досліджень, доступу через EOSC до експериментальних даних від європейських дослідницьких інфраструктур.

На поточному етапі досліджень мова йде не стільки про об'єкт цифрової наукової інфраструктури (база даних, озеро даних), скільки про послугу керування великими даними: реалізується механізм керування множиною даних, доступним різним типам користувачів — науковим співтовариствам, державним структурам тощо.

Парадигма Відкритої Науки

Парадигма Відкритої Науки є спробою світової наукової спільноти розв'язати проблему наукової невідтворюваності (scientific irreproducibility) [8, 9]. «Наукова невідтворюваність – неспроможність повторити чужі експерименти та дійти того ж висновку – [10]. Для цього запропоновано базові принципи, на яких повинні ґрунтуватися наукові дослідження:

- *Відкритий доступ*. Тобто результати досліджень, наукові публікації, які поширюються онлайн і без затрат або інших перешкод, повинні мати вільний доступ.
- *Відкрита наука*. Дослідники діляться своїми методами, програмним кодом та даними досліджень через централізовані спеціалізовані репозиторії.
- *Відкриті дані*. Дані повинні бути вільно доступні кожному для використання, повторного аналізу і публікації на свій розсуд, без обмежень з боку авторського права, патентів або інших механізмів контролю.

Виходячи з цих принципів, дослідники мають не тільки публікувати свої дані в Web, а й надавати до них доступ у такому вигляді (і форматі), щоб забезпечити їх сумісність із поширеними стандартами, а також можливість їх повторного використання. Проблема ускладнюється тим, що йдеться про дані великого обсягу, які швидко змінюються та слабо структуровані, тобто їх подання та збереження базується на технологіях Big Data.

Впровадження FAIR

Із початку 2018 року спільнота GO FAIR працює над впровадженням Керівних принципів FAIR. Результатом цих спільних зусиль є структура з трьох пунктів, яка формулює основні кроки до кінцевої мети – глобального Інтернету даних і послуг FAIR, де дані є знаходжуваними, доступними, інтероперабельними та повторно використовуваними (FAIR) для машин.

На сайті https://www.go-fair.org/fair-principles/ надано докладне роз'яснення принципів і практичне керівництво щодо того, як розробляти та використовувати FAIR дані, де їх шукати [11].

Структура FAIRification дає практичні вказівки «як це зробити» для зацікавлених сторін, які прагнуть бути FAIR.

Крім того, дотримуючись цієї структури, зацікавлені сторони можуть бути впевнені, що їхні зусилля щодо FAIRification будуть оптимально скоординовані із зусиллями інших зацікавлених сторін у спільноті GO FAIR. Структура з трьох пунктів максимізує повторне використання існуючих ресурсів, максимізує взаємодію та прискорює зближення стандартів і технологій, що підтримують дані та послуги FAIR.

Як правило, процес FAIRification починається, коли спільнота практиків розглядає свої вимоги до метаданих, що стосуються домену ПрО, та інші міркування політики, і формулює ці міркування як компоненти метаданих, що використовуються машиною. Для складання цих міркувань можна керуватися розділом Метаданих для машин (M4M) Workshops.

Схеми метаданих для повторного використання, створені в M4M, складають частину більшого профілю впровадження FAIR (FIP).

Профіль впровадження FAIR, у свою чергу, керує вибором і конфігурацією інфраструктури FAIR. Наприклад, використання точок даних FAIR (FDP) або FAIR Digital Objects (FDO), які сприяють створенню глобального Інтернету даних і послуг FAIR.

Розроблений підхід допомагає широкому колу зацікавлених сторін побачити, що для них означає «справедливий процес» на практиці, і ввійти в новий ландшафт FAIR. Це не тільки зберігає пріоритет практичних елементів FAIRification, а й дозволяє розподілити підхід до координації громади, який необхідний для швидкого масштабування та конвергенції.

З квітня 2020 року функціонують робочі групи, які розробляють методи, інструменти та документацію навколо платформи процесу FAIRification:

• Робоча група Metadata 4 Machines

• Робоча група FAIR Implementation Profile

• Робоча група FAIR Data Point

У зв'язку з викликами, пов'язаними з пандемією COVID-19, 3-кроковий FAIRification Framework активно розробляється в кількох проєктах. Безпосередньою метою цих трьох робочих груп є створення посібника, який об'єднає методи та ресурси для проведення семінарів M4M, для створення профілів впровадження FAIR та для встановлення точок даних FAIR.

Проаналізувавши розробки, що пов'язані зі створенням та використанням FAIR для наукових Big Data, можна відмітити доцільність застосування онтологій ПрО, що відповідають окремим галузям наук або є основою для інтеграції інформації з різних галузей та з різних країн. Такі онтології можуть бути використані як джерело знань для метаданих таких Big Data. Але це викликає потребу в автоматизованій побудові відповідних онтології – за пертинентними інформаційними ресурсами різного ступеня структурованості та з використанням уже існуючих онтологічних структур.

Створення джерела даних FAIR на базі Semantic MediaWiki.

Семантизовані Wiki-ресурси, такі як Semantic MediaWiki, які дозволяють створювати семантичні дані та базуються на використанні стандартів Semantic Web, надають потужне рішення для спільного редагування даних та їхніх метаописів, створення різних довільних наборів властивостей у шаблонах цих метаописів, з одночасним поданням їх як в машинно-оброблюваній формі, так і формі, придатній для розуміння людиною, що в результаті дає можливість оперувати цими даними, автоматизовано керувати, проводити аналіз, публікувати.

Однак Semantic MediaWiki не містить адекватних і ефективних вбудованих функцій імпорту та експорту між інтероперабельними форматами Semantic Web (таких, як RDF або OWL) і внутрішнім Wiki-форматом. Для вирішення цієї задачі розробляються проєкти, як наприклад, RDFIO (pharmb.io/project/rdfio.) – набір інструментів для імпорту RDF-даних в галузі біомедичних досліджень в Semantic MediaWiki з метаданими, які необхідні для експорту цих даних у формат RDF або OWL [12] .

Семантизація програмних засобів, що використовують онтології для керування метаданими наукових даних, є перспективним напрямком, який дозволить незалежно від галузі наукових досліджень, прийнятих у цій галузі стандартів, типів даних, розробити інформаційний ресурс (IP), який спільно створюється і спільно використовується відповідно до усіх принципів FAIR.

Використання для цього семантичного розширення Wiki-технології [13, 14] Semantic MediaWiki [15], що історично себе показало досить потужним інструментом, має широке використання, відкритий код та відкриті принципи розробки і підтримки, постійно розвивається, – є найбільш виправданим вибором.

Вбудовані можливості Semantic MediaWiki з завантаження файлів різного формату і додавання до них метаданих з різним набором атрибутів, які можливо змінювати, доповнювати, та які мають можливість обробки програмно, водночас зрозумілі звичайній людині. Простий і доступний інтерфейс, проста мова розмітки, інтуїтивний інтерфейс, можливість спільної роботи, відкрите розміщення в Web, роблять MediaWiki найкращим рішенням.

Розглянемо відповідність IP, що створений у середовищі Semantic MediaWiki, основним вимогам FAIR.

Findable

Першим кроком для використання даних є їх пошук.

Semantic MediaWiki надає можливості семантичного пошуку – на основі семантичних властивостей та категорій окремих Wiki-сторінок, які можуть розглядатися як гнучкий набір метаданих. Забезпечується можливість машинної обробки таких метаданих. .

F1. Таким (Мета)даним в Semantic MediaWiki надається глобальний унікальний і постійний ідентифікатор – кожній семантичній властивості або категорії в IP відповідає окрема Wiki-сторінка з унікальним ідентифікатором, яка описує характеристики та сферу застосування відповідного фрагменту метаданих. Крім того, забезпечується можливість перегляду того, де саме (на яких сторінках) використовується цей фрагмент метаданих та яких значень він набуває.

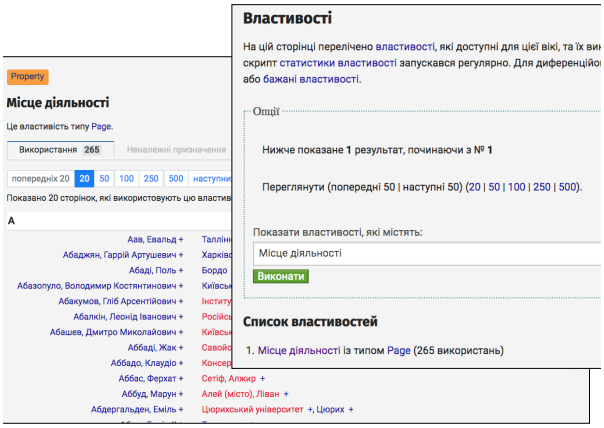


Рис.1. Пошук метаданих в середовищі Semantic MediaWiki

Певні набори метаданих, які характеризують типові для IP інформаційні об'єкти, можна об'єднувати за допомогою стандартного механізму «Форми» та «Шаблони» MediaWiki.

F2. Механізми MediaWiki мають можливість додавати докладні метадані до даних, які завантажуються до стандартного сховища MediaWiki, або вказувати гіперпосилання на зовнішнє сховище. Для цього також використовуються семантичні властивості та категорії.

F3. Метадані чітко і експліцитно містять ідентифікатор даних, які вони описують. Ідентифікатор даних в Semantic MediaWiki подається окремим значенням властивості, що додається до набору властивостей метаданих (рис.2).

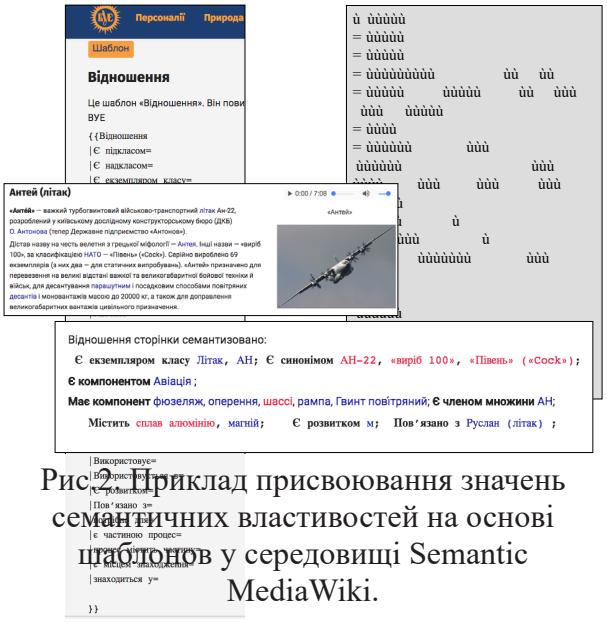


Рис.2. Приклад присвоювання значень семантичних властивостей на основі шаблонів у середовищі Semantic MediaWiki.

F4. Метадані реєструються або індексуються в пошуковому ресурсі. Оскільки Semantic MediaWiki є Web-застосуванням, вона містить файл налаштувань, де є можливість вказати тип відкритості – відкрита чи закрита система. Цей тип вказується лише раз у момент першого запуску й записується в файл налаштування LocalSettings.php. Для відповідності цьому принципу, треба встановити у цьому файлі певні права:

```
# Enable/disable reading by anonymous users
$wgGroupPermissions['*']['read'] = true;
# Enable/disable anonymous editing
$wgGroupPermissions['*']['edit'] = true;
# Allow new user registrations
$wgGroupPermissions['*']['createaccount'] = true;
```

Для ефективного знаходження даних такі дані і додаткові матеріали до них повинні мати достатньо повні метадані, а також метаопис і унікальний постійний ідентифікатор. Дані, розміщені на сторінках Semantic MediaWiki з вказаними нала-

штуваннями, подаються до Web відкрито і добре індексуються глобальними пошуковими системами, такими як Google та Bing.

Accessible

Коли користувач знайде необхідні дані, він повинен знати, як до них можна отримати доступ. Можливо, включаючи аутентифікацію та авторизацію. Таку інформацію в Semantic MediaWiki можна доповнювати окремими атрибутами на сторінках із метаописами.

A1. (Мета)дані можна отримати за їхнім ідентифікатором за допомогою стандартизованого протоколу зв'язку. Зазвичай Web-системи розміщуються на серверах і мають стандартний http або захищений https протокол доступу.

A1.1 Протоколи http та https є відкритими, безкоштовними і універсальними. Вони є базовими протоколами Інтернет і Web.

A1.2 Протокол https за необхідності припускає процедуру аутентифікації та авторизації.

A2. Метадані доступні, навіть якщо дані більше не доступні. Оскільки метадані розміщуються окремо на сторінках MediaWiki (фізично – в БД MediaWiki), цей принцип виконується.

Отже, метадані (з метаописами) і самі дані в Semantic MediaWiki (з відповідними налаштуваннями серверів та інфраструктури, заданими наборами метаданих із властивостями унікальних ідентифікаторів) зрозумілі як для людини, так і для програмної обробки й зберігаються в надійному репозиторії, тобто відповідають зазначеним вище вимогам.

Interoperable

Дані, що представлені за допомогою Semantic MediaWiki, можуть бути інтегровані з іншими даними. Крім того, дані можуть взаємодіяти з іншими застосуваннями або робочими процесами для аналізу, зберігання та обробки.

I1. (Мета)дані в Semantic MediaWiki використовує базові стандарти Semantic Web, експорт даних в XML, RDF, а за можливості встановлення інших додаткових плагінів, то і в OWL або інші спеціалізовані стандарти. PDF, CSV, LaTeX, тощо. Експорт результатів семантичного пошуку в формат RDF входить до функціоналу Semantic MediaWiki.

I2. MediaWiki є відкритою системою, тобто за умови відкритого типу встановлення (як вказано в п.F4 і A1), система публікації даних і метаданих, базована на Semantic MediaWiki, може використовувати, інтегрувати, імпортувати (стандартна функція імпорту або додатковий спеціальний плагін) будь-які словники, онтології, які подані стандартною мовою Semantic Web, що відповідає принципам FAIR.

I3. (Мета)дані в Semantic MediaWiki включають посилання на інші метадані. Стандартний функціонал MediaWiki дозволяє додавати необмежену кількість посилань на інші джерела, а можливість додавати різні додаткові властивості дозволяє розширювати набори властивостей до даних в залежності від задачі.

Отож, задача інтероперабельності (сумісності) в Semantic MediaWiki вирішується простою публікацією додаткових властивостей, які для людини показуються простими описами, а для програмного запиту надаються в одному із загальновживаних форматів XML, JSON, RDF.

Reusable

Повторне використання даних у Semantic MediaWiki забезпечується механізмами семантичних властивостей та категорій Semantic MediaWiki, які підтримують гнучке внесення змін до структури метаданих ресурсу та автоматизацію деяких елементів цього процесу. Такі властивості однозначно описані на окремих Wiki-сторінках, тому їх можна відтворювати та комбінувати в різних налаштуваннях.

R1. Можливість Semantic MediaWiki додавати необмежену кількість атрибутів та категорій кожній сторінці забезпечує вимогу FAIR щодо того, що метадані повинні бути детально описані множиною точних і відповідних атрибутів.

R1.1. Механізми MediaWiki розроблялась для використання у відкритому середовищі з різними правами власності, тому вона містить певні механізми додавання інформування про різні типи ліцензій до різних об'єктів.

R1.2. Походження даних та метаданих досягається в Semantic MediaWiki шля-

хом додавання додаткового метаопису про джерело та умови походження даних.

R1.3. (Мета)дані в Semantic MediaWiki відповідають стандартам спільноти, які стосуються домену ПрО, що може бути формалізована за допомогою онтології цієї ПрО, яка використовується як основа для семантичної розмітки Wiki-сторінок і визначає набори даних, їхні імена та можливі значення.

Висновки

Наведений вище аналіз парадигми менеджменту наукових даних FAIR та порівняння основних вимог FAIR до подання даних та метаданих з виразними властивостями середовища Semantic MediaWiki свідчить про те, що інформаційні ресурси, які створюються в цьому середовищі, відповідають сучасним вимогам до відкритих даних великого обсягу. Це уможливлює використання таких Wiki-ресурсів як основи для побудови та семантичного аналізу метаданих у релевантних предметних областях. Одним із можливих напрямків застосування цього підходу є використання структури метаданих Wiki-ресурсу для аналізу метаданих Big Data.

Даний підхід апробовано в процесі створення бази знань портальної версії Великої української енциклопедії (vue.gov.ua) [16], яка є джерелом інтероперабельних знань, що придатні для повторного використання в інших інтелектуальних застосуваннях.

Література

1. Hurwitz, J., Nugent, A., Halper, F., Kaufman, M., 2013,. Big Data. New York.
2. Rogushina J., Gladun A., Pryima S. Use of Ontologies for Metadata Records Analysis in Big Data. Selected Papers of the XVIII International Scientific and Practical Conference “Information Technologies and Security” (ITS 2018). CEUR Vol-2318. http://ceur-ws.org/Vol-2318/paper5.pdf.
3. Балякин А., Малышев А. Управление большими данными в исследовательских инфраструктурах // Открытые системы. СУБД, 2020, № 03. – https://www.osp.ru/os/2020/03/13055606.
4. FAIR_data. https://en.wikipedia.org/wiki/FAIR_data.
5. Gomez-Diaz, T., Recio, T. (2021). Open comments on the Task Force SIRS report: Scholarly Infrastructures for Research Software (EOSC Executive Board, EOSCArchitecture).
6. The FAIR Guiding Principles for scientific data management and stewardship. Available from: https://www.nature.com/articles/sdata201618.
7. The FAIR data principles. Available from: : https://www.force11.org/group/fairgroup/fairprinciples (дата обращения: 29.08.2020).
8. The Irreproducibility Crisis of Modern Science – CUSES, Consequences and the Road to Reform, National Association of Scholars, 2018, Available from: https://www.nas.org/reports/the-irreproducibility-crisis-of-modern-science.
9. Challenges in irreproducible research, Nature, 18-10-2018, Available from: https://www.nature.com/collections/prbfkwmwvz/.
10. Baker, 1,500 scientists lift the lid on reproducibility. Nature, 533(7604): 452-454. (2016) doi:10.1038/533452a, Available from: https://www.nature.com/articles/533452a.
11. Three-point FAIRification Framework. Available from: https://www.go-fair.org/how-to-go-fair/.
12. Lampa, S., Willighagen, E., Kohonen, P., King, A., Vrandečić, D., Grafström, R., Spjuth, O. 2017. RDFIO: extending Semantic MediaWiki for interoperable biomedical data management. Journal of biomedical semantics, 8(1), 2017, P.1-13.
13. Manual:What is MediaWiki?. Available from: https://www.mediawiki.org/wiki/Manual:What_is_MediaWiki%3F.
14. MediaWiki. Available from: https://www.mediawiki.org/wiki/MediaWiki.
15. *Krötzsch M., Vrandečić D., Völkel M.* Semantic mediawiki. International semantic web conference, 2006, pp. 935-942. Available from: https://link.springer.com/content/pdf/10.1007/11926078_68.pdf.
16. Rogushina J.V., Grishanova I.J. Ontological methods and tools for semantic extension of the media WIKI. Проблеми програмування, № 2-3, 2020. С.-61-73. Available from: http://pp.isofts.kiev.ua/ojs1/article/download/398/437

References

1. Hurwitz, J., Nugent, A., Halper, F., Kaufman, M. (2013). Big Data. New York.

2. Rogushina J., Gladun A., Pryima S. Use of Ontologies for Metadata Records Analysis in Big Data. Selected Papers of the XVIII International Scientific and Practical Conference "*Information Technologies and Security*" (ITS 2018). CEUR Vol-2318. Available from: http://ceur-ws.org/Vol-2318/paper5.pdf [Accessed 18/11/2021]
3. Baliakin A., Malyshev A. (2020) Management of Big Data in research infrastructures. Open systems, 03. Available from: https://www.osp.ru/os/2020/03/13055606 [Accessed 18/11/2021]
4. FAIR_data. Available from: https://en.wikipedia.org/wiki/FAIR_data [Accessed 18/11/2021]
5. Gomez-Diaz, T., Recio, T. (2021). Open comments on the Task Force SIRS report: Scholarly Infrastructures for Research Software (EOSC Executive Board, EOSCArchitecture). arXiv preprint arXiv:2108.06127.
6. The FAIR Guiding Principles for scientific data management and stewardship. Available from: https://www.nature.com/articles/sdata201618 [Accessed 18/11/2021]
7. The FAIR data principles. Available from: https://www.force11.org/group/fairgroup/fairprinciples [Accessed 18/11/2021]
8. The Irreproducibility Crisis of Modern Science – CUSES, Consequences and the Road to Reform, National Association of Scholars, (2018), Available from: https://www.nas.org/reports/the-irreproducibility-crisis-of-modern-science [Accessed 18/11/2021]
9. Challenges in irreproducible research, Nature, 18-10-2018, Available from: https://www.nature.com/collections/prbfkwmwvz/ [Accessed 18/11/2021]
10. Baker, 1,500 scientists lift the lid on reproducibility. Nature, 533(7604): 452-454. (2016) doi:10.1038/533452a, Available from: https://www.nature.com/articles/533452a.
11. Three-point FAIRification Framework https://www.go-fair.org/how-to-go-fair/ [Accessed 18/11/2021]
12. Lampa, S., Willighagen, E., Kohonen, P., King, A., Vrandečić, D., Grafström, R., & Spjuth, O. (2017). RDFIO: extending Semantic MediaWiki for interoperable biomedical data management. *Journal of biomedical semantics,* 8(1), 1-13 [Accessed 18/11/2021]
13. Manual:What is MediaWiki? Available from: https://www.mediawiki.org/wiki/Manual:What_is_MediaWiki%3F [Accessed 18/11/2021]
14. MediaWiki. Available from: https://www.mediawiki.org/wiki/MediaWiki.
15. Krötzsch M., Vrandečić D., Völkel M. (2006) Semantic mediawiki. *International semantic web conference,* pp. 935-942. Available from: https://link.springer.com/content/pdf/10.1007/11926078_68.pdf [Accessed 18/11/2021]
16. Rogushina J.V., Grishanova I.J. (2020) Ontological methods and tools for semantic extension of the media WIKI. *Problems in Programming,* 2-3, P.-61-73. Available from: http://pp.isofts.kiev.ua/ojs1/article/download/398/437 [Accessed 18/11/2021].

Отримано: 20.11.2021

Об авторах:

Рогушина Юлія Віталіївна,
Канд.фіз.-мат.наук, старший науковий співробітник Інституту програмних систем НАН України, публікації в українських виданнях – 170, публікації в іноземних журналах – 35, індекс Хірша (Scopus) – 5,
ORCID http://orcid.org/0000-0001-7958-2557.
e-mail: ladamndraka2010@gmail.com

Гришанова Ірина Юріївна,
науковий співробітник Інституту програмних систем НАН України, публікації в українських виданнях – 19, публікації в іноземних журналах – 3, індекс Хірша (Scopus) – 1,
ORCID http://orcid.org/0000-0003-4999-6294.

Місце роботи авторів:

Інститут програмних систем
НАН України, 03181, Київ-187,
проспект Академіка Глушкова, 40,
e-mail: ladamandraka2010@gmail.com,
i26031966@gmail.com
066 550 1999.

УДК 004.94 http://doi.org/10.15407/pp2021.04.038

В.А. Резніченко

60 РОКІВ БАЗАМ ДАНИХ (частина друга)

Наводиться огляд досліджень і розробок баз даних з моменту їх виникнення в 60-х роках минулого століття і по сьогодні. Виділяються наступні етапи: виникнення і становлення, бурхливий розвиток, епоха реляційних баз даних, розширені реляційні бази даних, постреляційні бази даних і великі дані. На етапі становлення описуються системи IDS, IMS, Total і Adabas. На етапі бурхливого розвитку висвітлені питання архітектури баз даних ANSI/X3/SPARC, пропозицій КОДАСИЛ, концепції і мов концептуального моделювання. На етапі епохи реляційних баз даних розкриваються результати наукової діяльності Е. Кодда, теорія залежностей і нормальних форм, мови запитів, експериментальні дослідження і розробки, оптимізація та стандартизація, управління транзакціями. Етап розширених реляційних баз даних присвячений опису темпоральних, просторових, дедуктивних, активних, об'єктних, розподілених та статистичних баз даних, баз даних масивів, машин баз даних і сховищ даних. На наступному етапі розкрита проблематика постреляціонних баз даних, а саме NOSQL-, NewSQL- і онтологічних баз даних. Шостий етап присвячений розкриттю причин виникнення, характеристичних властивостей, класифікації, принципів роботи, методів і технологій великих даних. Нарешті в останньому розділі дається короткий огляд досліджень і розробок із баз даних у Радянському Союзі.

Ключові слова. Типи баз даних: ієрархічна, мережева, реляційна, навігаційна, темпоральна, просторова, просторово-темпоральна, просторово-мережева, об'єктів, що переміщуються, дедуктивна, активна, об'єктно-орієнтована, об'єктно-реляційна, розподілена, паралельна, масивів, статистична, багатовимірна, машина баз даних, сховища даних, NoSQL, ключ-значення, стовпчикова, документно-орієнтована, графова, мультімодельна, хмарна, наукова, багатозначна, XML, NewSQL, онтологічна, великі дані.

Етап 4. Розширені реляційні бази даних (1980 – 2000 +)

Із моменту виникнення реляційна модель піддавалася критиці в зв'язку з простотою її структури даних. Відтак пропонувалися розвинутіші моделі, що дозволяли адекватніше представляти інформаційні моделі різних предметних областей. Однак їхньою особливістю було те, що всі вони створювалися на базі реляційної моделі й отримали назву розширених реляційних баз даних. Переважна більшість наведених у наступному абзаці БД або створювалися на основі реляційних БД, або мали варіанти такої реалізації.

У цей період активізувалися дослідження із взаємопроникнення технологій штучного інтелекту (ШІ) і БД. 1988 року відбулися два різних симпозіуми з інтеграції ШІ і БД. Відповідно до потреб у БД виникли два напрямки представлення в них правил (що породжують нові дані з існуючих). В результаті виникли БД двох типів: активні і дедуктивні. Крім того, потреба включення в БД часу і простору привела до появи темпоральних і просторових БД, а потреба застосування об'єктної технології до БД привела до появи об'єктних БД. Прагнення суттєво підвищити продуктивність баз даних для роботи з великими обсягами даних викликало необхідність дослідження і розробки машин баз даних. А успіхи у створенні комп'ютерних мереж привели до появи розподільних і паралельних баз даних. Зрештою в цей же період було визнано, що БД мають використовуватись не лише для «рутинної» роботи зі збору, зберігання та пошуку ретельно відібраних і перевірених даних, а й для їх систематизації, узагальнення, статистичної та аналітичної обробки. Так з'явилися статистичні БД, БД для роботи з масивами, багатовимірні БД і, зрештою, сховища даних.

Темпоральні бази даних.

Темпоральні бази даних – це бази, що зберігають дані, дотичні до часу. Час як окремий тип даних присутній в усіх СУБД. Однак це не є підставою вважати їх темпоральними, адже інтерпретація часу і семантика взаємозв'язку між часом і даними залишається за розробником. У темпораль-

них базах даних мають існувати правила інтерпретації часу й можливості розкриття семантики взаємозв′язку даних із часом.

Дослідження з використання поняття часу в інформаційних системах були здійснені вже в 60-х роках минулого століття. Вважається [212], що вперше ідеї фіксації змінюваної в часі інформації в базах даних з′явилися 1976 року в праці Яніса А. Бубенка молодшого (Janis A. Bubenko, Jr) [213]. Згодом були проведені активні дослідження із розкриття семантики часу на концептуальному рівні [213 - 218], створення залежних від часу моделей даних для статистичних реляційних баз даних [219 - 222] і розробки темпоральних мов запитів [223 - 227].

Яніс Бубенко мол.

Основні поняття. Вже до середини 80-х років склалися основні положення темпоральних БД (ТБД). Їхня суть полягала в наступному. Темпоральний домен (temporal domain), загалом визначається як безліч темпоральних індивідів (temporal individuals), на яких задані темпоральні відношення (temporal relations). Темпоральний домен характеризується такими аспектами: структурним (лінійний час, розгалужений час), дискретним (безперервний час, дискретний час), граничним (обмежений час, безкінечний час).

Темпоральними індивідами можуть виступати моменти часу (часові точки) або часові інтервали. Для моментів часу задане відношення лінійного порядку (linear order), а для інтервалів – відношення Аллена [228].

Асоціація темпоральних індивідів із даними бази даних проводиться з допомогою часових позначок (timestamps).

Лінія часу – це часова вісь, задана на конкретному часовому домені й призначена для асоціації часових позначок із даними. Вирізняють дві лінії часу:

Лінія дійсного часу. Дійсний час (valid time) визначає період часу, протягом якого відбувається той чи інший факт моделювання реальності.

Лінія транзакційного часу. Транзакційний час – це період часу, протягом якого інформація про факт зберігається в базі даних.

Бази даних, які підтримують обидві лінії часу, називаються бітемпоральними.

Крім того, вірогідне існування користувального часу (user -defined time) – часу (інтервалу часу), що прив′язується в базі даних до факту самим користувачем.

На лінії часу присутній спеціальний момент часу, котрий називається ЗАРАЗ, і має специфічні особливості [229].

У темпоральній реляційній моделі час може прив′язуватись або до атрибутів, або до кортежів.

Темпоральні моделі даних. Темпоральна модель даних – це модель, в якій існує можливість прив′язувати дані до часу. Ці моделі відрізняються залежно від того, які лінії часу вони підтримують (модель дійсного часу, модель транзакційного часу, бітемпоральна модель), які темпоральні індивіди використовують (моменти часу – точкова модель або інтервали) і до яких даних прив′язується час (до значень атрибутів чи кортежів).

Пік досліджень із темпоральних моделей даних припадає на 80-і роки. Наведемо список праць, присвячених темпоральним моделям даних:

Яків бен – Зві (Jacob Ben-Zvi) [224, 225] – бітемпоральна інтервальна модель;

Джонс і Мейсон (Jones and Mason) [226] – інтервальна модель дійсного часу;

Річард Снодграс (Richard Snodgrass) [227] – біотемпоральна точкова й інтервальна модель;

Лоренцос і Джонсон (Lorentzos and Johnson) [230, 231] – точкова й інтервальна модель дійсного часу.

Річард Снодграс

Всі ці моделі передбачали прив′язку часу до кортежів. Також пропонувалися темпоральні моделі з прив′язкою часу до атрибутів [232 - 237].

Із оглядом досліджень темпоральних моделей даних можна ознайомитись у статтях [238 - 240].

Темпоральні залежності.

Теорія проєктування баз даних базується на понятті залежностей. Серед них фундаментальним поняттям є функціональна залежність.

Динамічний варіант функціональної залежності (DFD) вперше був висунутий Віану (Vianu) [241]. Передбачалося, що така FD має виконуватися в даному кортежі і в його оновленому варіанті. В цій же праці був досліджений взаємозв′язок між DFD і статичним FD. Ще один вид темпоральної FD був запропонований Війсеном (Wijsen) [242, 243]. В цьому випадку необхідно, аби FD виконувалась у поєднанні старого й нового відношення. Війсен визначив також тренд – залежність (trend dependency) [244], що є узагальненням визначеної ним же темпоральної FD.

Ще один різновид TFD був визначений у статті [245]. В ній пропонувалося розширення теорії нормалізації з урахуванням надлишковості, яка породжується TFD. У праці [246] було визначено залежність, викликану обмеженнями (constraint generating dependency - CGD). CGD означає, що кожен атрибут відношення приймає значення з домену, визначеного обмеженням. Це характерно для атрибутів часових позначок темпоральних моделей даних. Нарешті, в статті [247] були досліджені темпоральні розширення відношень спеціалізації та узагальнення.

Мови. У проведених дослідженнях було запропоновано темпоральні реляційні алгебри [230 – 232, 236, 237, 248 - 250] і обчислення [233, 236] для різних темпоральних моделей даних. Була також визначена вкладена бітемпоральна модель даних та відповідна їй алгебра [234].

Тансел і Аркун розробили мову HQUEL [251], яка є розширенням мови QUEL «історичними» даними. Вони ж запропонували мову TBE (Time-By-Example) [252], в якій скористалися ідеєю графічної реляційної мови QBE. Снодграс також запропонував темпоральний варіант QUEL, який було названо TQUEL [227, 253].

Пропонувалися різні варіанти темпорального розширення SQL [235, 254 - 258].

TSQL2. Одним із ключових періодів у галузі досліджень темпоральних баз даних, часом їхнього «офіційного» представлення можна вважати 1992 – 1995 роки. Спершу Річард Снодграс (Richard T. Snodgrass) висунув ідею про можливе темпоральне розширення стандарту SQL-92, а згодом, 1993 року, був проведений семінар [259], який продемонстрував зацікавленість наукового співтовариства в розробці темпорального розширення стандарту SQL-92. В результаті було організовано комітет зі створення такої мови, названої Temporal Structured Query Language TSQL2. Провідну роль у роботі комітету відіграв Снодграс. Уже у вересні 1993 року було випущено перший чорновий варіант мови, а в грудні – другий. В результаті плідної роботи в березні 1994 року з′явилася перша попередня версія специфікації мови [260], а у вересні навчальний посібник [261]. Нарешті 1995 року була опублікована остаточна специфікація мови запитів TSQL2 [262].

Подальша діяльність була пов′язана із включенням і розширенням основних ідей TSQL2 в SQL3. Ця мова була названа SQL/Temporal. Були опрацьовані питання

підтримки в SQL/Temporal дійсного і транзакційного часу [263, 264]. Остаточні пропозиції переходу від TSQL2 у SQL3 було сформульовано в [265].

Темпоральний SQL: 2011. 1995 року в ANSI/ISO було прийнято рішення про розгортання робіт зі створення нового стандарту SQL, який би включав темпоральні властивості. В зв′язку з цим США внесли пропозицію розширити відповідні можливості SQL, що базувалися на піонерських дослідженнях колективу під керівництвом Снодграса.

Ці пропозиції базувалися на детально на той час пропрацьованих групою Снодграса специфікаціях мови TSQL2, яка була темпоральним розширенням SQL-92, а також на пропозиціях переносу TSQL2 в SQL3. Однак деякі члени ISO піддали сумніву ці пропозиції США в зв′язку із наявними в них серйозними проблемами і протиріччями. Зі свого боку Великобританія внесла пропозицію, сформульовану на основі досліджень Нікоса Лоренцоса (Nikos Lorentzos) з університету Афін, Греція. Однак США не погодилися з позицією ISO щодо їхньої пропозиції і не підтримали Великобританію. Через це ANSI і ISO вирішили відкласти подальшу роботу по темпоральному SQL до офіційної публікації версії SQL-99.

Після публікації SQL-99 ані США, ані Великобританія не внесли жодних нових пропозицій, котрі б усунули виниклі раніше протиріччя. В зв′язку з цим 2001 року ANSI і ISO вирішили припинити діяльність зі створення стандарту темпорального SQL. Друга спроба додавання темпоральних властивостей в SQL була зроблена 2008 року. Вона почалася з обговорення, введення і прийняття пропозицій двох комітетів INCITS DM32.2 и ISO/IEC JTC1 SC32 WG3 із «системно – версійних таблиць» (systemversioned tables).

Ще одна темпоральна риса була додана в SQL 2010 року у вигляді «таблиць із прикладними періодами» (application-time period tables). Обидва ці поняття й розроблені для них відповідні мовні засоби були включені до стандарту SQL:2011. Із темпоральними особливостями SQL:2011 можна ознайомитися в [266, 267].

Просторові бази даних.

Просторова база даних (ПБД) – це база даних, призначена для зберігання, маніпулювання і виконання записів до даних про просторові об′єкти, представлені певними абстракціями. Тоді як традиційні БД призначені для зберігання й обробки числової і символьної інформації, ПБД дають можливість працювати з цілісними просторовими об′єктами, що об′єднують як традиційні види даних (описова частина або атрибутивна), так і геометричні (дані про розміри й розташування об′єктів у просторі).

Ще на початку 70-х років поняття обробки просторових даних використовували для позначення діяльності, пов′язаної з електронною обробкою даних з метою підвищення продуктивності під час складання і редагування карт, картографічних вимірів та аналізу просторових даних.

Моделі просторових даних.

Двома видами моделей просторових даних є: польова й об′єктна.

Польова модель. Використовується для представлення безперервних або аморфних явищ, зокрема, температури або хмарності. Ця модель підтримує функціональну точку зору, коли базисна система відліку (просторова система координат, як от, широта і довгота) функціонально відображаються в задану сферу знань. Наприклад, у градуси для температури. Комп′ютерною реалізацією польової моделі є растрова структура даних – рівномірна решітка, накладена на базисний простір. Іншими популярними структурами даних для представлення полів є триангульована нерегулярна мережа (triangulated irregular network TIN), лінії контуру, точкові решітки.

Операції польової моделі поділяються на три типи [268]:

локальні операції – значення функції в даній точці залежить лише від значення аргументу в цій точці (поточкова сума, різниця, максимум, середнє значення);

фокальні операції – значення функції в даній точці залежить від значень малого оточення цієї точки (нахил, середньозважене значення в довкіллі);

зональні операції – польова функція задається не на точках або їхньому малому оточенні, а на областях (багатокутниках) вцілому (сума, середнє, мінімальне польове значення кожної зони).

Об′єктна модель. Пояснює інформаційний простір як сукупність дискретних, ідентифікованих просторових сутностей. Вона представляється в комп′ютерах так званою векторною структурою даних. Основне питання об′єктної моделі – вибір базової множини типів просторових даних. Було проведено чимало досліджень, результатом яких став стандарт OGC [271], що визначив наступні типи (з певним спрощенням):

прості об′єкти – точки, криві, поверхні;

набори об′єктів – набір точок, набір кривих, набір поверхонь.

Операції. Здійснена велика кількість досліджень із визначення просторових операцій. Стаття [273] була однією з перших спроб загального опису операцій над картами (алгебра карт) з позицій растрового аналізу і стала базовою мовою для роботи з польовими моделями. З часом ця алгебра була уточнена в [274]. У [275] запропонована розширена мова запитів для географічних баз даних. У статті [276] вперше пропонувалося розширення алгебри реляційної моделі шляхом введення просторових об′єктів і операцій. У статті [277] пропонується мова SpatialSQL, де до мови SQL влючені просторові операції і відношення. Алгебра ROSE (**RO**bust Spatial **E**xtension) [278] базується на реляційній моделі, використовує типи даних для представлення точок, ліній, областей і пропонує вичерпний набір операцій; семантика типів і операцій визначена формально. В праці [279] представлена просторова логіка, яка може використовуватись для міркувань щодо топологічних і просторових взаємозв’язків між об′єктами. Перевага даного підходу – строго визначена семантика й використання механізму логічного висновку.

Для об′єктної моделі були визначені наступні операції [269]:

операції на множинах (однаково, не однаково, дорівнює, не дорівнює, є членом, є пустим, об′єднання, різниця, кардинальність…);

топологічні операції (є границею, внутрішня частина, зовнішня частина, замикання, торкаються, пересікаються, знаходиться всередині, знаходиться зовні, охоплює…);

метричні операції (відстань, кут, довжина, площа периметр…);

операції напряму (на північ, на схід, праворуч, зверху, спереду, між…);

мережеві операції (попередник, послідовник, з′єднані, шлях…);

динамічні операції (повернути, масштабувати, зсунути, розділити, злити…).

У термінах об′єктно – реляційних баз даних просторова модель даних реалізується визначенням просторових типів даних і операцій над об′єктами цих типів.

У зв′язку з цим чимало робіт із ПБД було спрямовано на розробку абстрактних типів даних (АТД) і їх впровадження в мови запитів. Завдяки створенню консорціуму Open Geospatial Consortium Inc. (OGC) вдалося серйозно просунутись у створенні стандартів із геопросторових технологій [270, 272]. Зокрема, OGC представив специфікацію [272] вмонтування в SQL двомірних просторових АТД на основі об′єктної моделі, а також запропонував вичерпний список операцій. У монографії [269] подано глибокий аналіз проблематики ПБД.

Просторові типи даних. Просторові типи даних уможливлюють моделювання об′єктів у просторі, а також їхні взаємозв′язки, властивості й операції. Вони становлять особливий інтерес у ПБД [269, 280, 281]. Більшість найпопулярніших абстракцій просторових об′єктів належить до класу структурних просторових типів даних. Ці типи даних представляють простір у вигляді точок, ліній, областей, поверхонь, об′ємів, просторових розбиттів (spatial partitions), просторових мереж та інших подібних об′єктів. Тобто просторові об′єкти розглядаються з точки зору їхньої структурної форми і просторових розмірів. Просторові типи даних для точок, ліній і областей розглядаються в [276, 278, 282 - 286], для поверхонь і об′ємів у [287], для просторових розбиттів у [288] і для просторових

мереж у [289]. Оригінальний підхід для визначення просторових типів даних пропонується в [289], названий Realm. Realm – це кінцева множина точок і лінійних відрізків, що не пересікаються. Вони можуть знаходитися у вузлах рівномірно розділеної сітки. На основі цих примітивних понять визначаються складніші структури й операції над ними.

Відношення головного напрямку (Cardinal direction relationships). Поняття «напрямок» є однією з важливих характеристик просторових систем. На алгоритмах обчислення просторових напрямків базуються ПБД і ГІС.

Відношення головного напрямку (ВГН) – це просторове відношення, що вказує знаходження одного об′єкту відносно іншого. Воно має як кількісне, так і якісне значення. Було запропоновано низку моделей ВГН. Спершу пропонувалися моделі, які просторові об′єкти представляли точками, а напрямок визначався відповідно до нанесеної сітки [292, 293]. В проєкційних моделях сітка наносилась паралельно вісям координат, а в конусній моделі – під кутом. У наступних моделях об′єкти апроксимувалися так званими «представницькими» областями, серед яких найчастіше пропонувалися обмежувальні прямокутники [298, 299]. Однак цей метод давав невірний напрямок, коли об′єкти накладені один на одний, переплетені або підковоподібні [293]. Згодом були запропоновані більш точні моделі ВГН, де висхідні об′єкти представлені своїми точними фігурами, а посилальні об′єкти апроксимуються обмежувальним прямокутником. Знову таки, залежно від нанесеної сітки розрізняють моделі] та конусного відношення напрямку (cone-based directional relationships) [269]. Була також пропозиція моделювати ВГН тернарними відношеннями [291]. Щодо згаданих вище моделей ВГН досліджувалися наступні завдання:

ефективне визначення відношень, які є між множинами об′єктів [291, 299, 300, 303];

обчислення інверсних відношень [290, 291, 295, 299, 304];

обчислення композиції двох або більше відношень [295, 299, 301, 304];

перевірка узгодженості множин відношень [295, 296, 297, 299, 302].

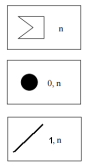
Оригінальне вирішення з моделювання напрямку у вигляді просторового об′єкту подане в [305].

Концептуальне моделювання. Загальновизнаним засобом концептуального моделювання є ER-мова. Пропонувалося чимало розширень цієї мови для представлення в ній просторових об′єктів із просторовими характеристиками [306]. Одним з таких розширень є ER-схема з піктограмами [307]. Піктограми в ній застосовуються для зазначення типів просторових сутностей і просторових зв′язків. Базовими типами геометричних фігур є: точка, (ламана) лінія, й багатокутник. Вони мають такі піктограми об′єктів:

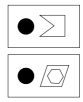

точка


лінія


багатокутник


Мульти-фігури – це множини базових фігур. Надається можливість вказати кількість елементів у мульти – множині за аналогією із зазначенням потужності закінчень зв′язків у звичайній ER-мові (M : N - не менше M і не більше N).


Похідні фігури – фігури об′єкту є похідною від фігур інших об′єктів. Як от, багатокутник країни є похідним від багатокутника областей.


Альтернативні фігури - об′єкт може бути представлений кількома фігурами. Наприклад, залежно від масштабу, місто може бути представлене точкою або багатокутником.


Будь-яка можлива фігура - об′єкт може бути представлений будь-якою допустимою фігурою.

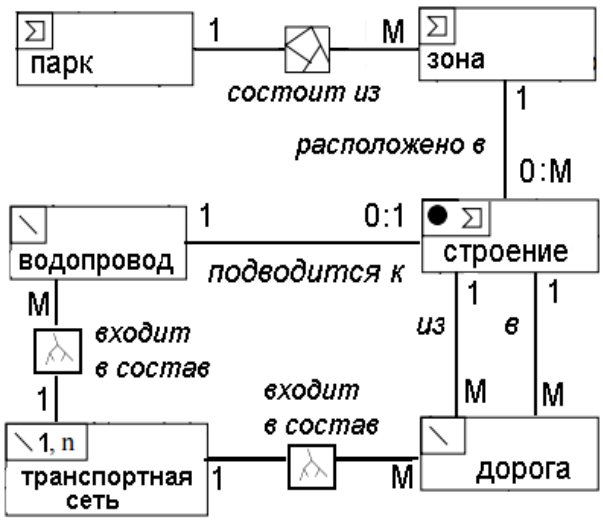

Фігура - об'єкт, що визначається користувачем, являє собою не стандартну фігуру, а визначену користувачем.

Піктограми зв′язків:

Є частиною в сенсі ієрархічної структури.

*Є частиною в сенсі відношення розбиття (*partition*).*

На рисунку нижче наведено приклад просторової ER-схеми.



Детальний опис цього розширення ER-мови наведено в працях [268, 308]. Були зроблені пропозиції щодо використання геопросторових онтологій для концептуального моделювання геоінформаційних систем [309], які б уможливлювали представлення у моделі ПО глибшу просторову семантику. Крім того, пропонується розширити геопросторові онтології темпоральними характеристиками з використанням OWL-Time [310].

Ці дослідження із просторових онтологій поділяються на дві категорії:

- онтології для інтеграції структур збереження просторових даних. В такому разі вирішення задачі інтеграції різних ПБД з метою вирішення задачі обміну даними між ними [311];

- онтології для більш точного представлення семантики даних. У цьому випадку просторова онтологія створюється або для відображення семантики конкретної предметної області з характерним саме для неї набором об′єктів і понять [312], або ж створюється певна універсальна онтологія, що охоплює максимально широке коло понять і характеристик геопросторових об′єктів [313 - 315].

Величезний обсяг геоінформаційних ресурсів у Інтернеті ініціював дослідження зі створення геопросторового семантичного веба [316, 317]. Так, приміром, була висунута пропозиція геопросторового розширення RDF (GeoRDF) [318].

Просторово – мережеві бази даних (ПМБД). Вони призначені для підтримки просторових мереж через надання необхідних моделі даних, мови запитів, структури зберігання і методів індексування. Модель просторової мережі може бути у вигляді графа, вершинами якого є точки у просторі. Характерними задачами, які вирішуються в ПМБД, є знаходження шляху між двома вершинами, який відповідає вказаним обмеженням.

Так само, як і в звичайних БД, в ПМБД виділяють три рівня моделювання: концептуальний, логічний та фізичний.

Завдання концептуальної моделі – адекватно представити численні об′єкти, їхні зв′язки, властивості й обмеження. Для цього пропонується використовувати піктографічну ER-модель [307] із завантаженими необхідною інформацією вершинами і ребрами. Досконалішою є транспортна модель даних UNETRANS [319].

Логічна модель даних передбачає використання моделі конкретної комерційної СУБД. Зазвичай застосовують об′єктно – реляційну модель. У праці [269] описано спеціальні операції на графах, які використовуються в ПСБД. Ще одна логічна модель системи GraphDB описана в роботі [320].

Фізична модель даних пов′язана із конкретною реалізацією ПМБД. На цьому рівні вирішуються задачі структур зберігання, методів індексування і доступу, управління пам′яттю тощо. Тут застосовують такі структури, як мережевий граф, матриця суміжності, список суміжності [269]. У праці [321] пропонується метод доступу CCAM. У статтях [322, 323] обговорюються питання виконання таких стандартних запитів у ПМБД, як «найкоротший шлях», «найближчі пари», «найближчий сусід». Важливим транспортним обмеженням для просторової мережі є так звані «обмеження на виконання поворотів». Якщо їх не враховувати, то можлива

побудова нездійсненних шляхів. Питанню побудови шляхів із урахуванням обмежень на повороти присвячені статті [324 - 327].

Просторово – часові мережеві БД.

Практично всі транспортні задачі на мережах залежать від часу доби, коли вони вирішуються. Іншими словами, просторово – мержеві моделі залежать від часу. Тому для вирішення транспортних задач необхідно застосовувати просторово – часові моделі мереж. У цьому напрямку також проводяться дослідження. Так, зокрема, для моделювання просторово – часових мереж у статті [328] пропонується робити копії всієї просторової мережі для кожного необхідного моменту часу, а в працях [329, 330] - зв′язувати з усіма вершинами і ребрами мінливі у часі атрибути.

Просторово – часові бази даних (ПЧБД). Вони представляють еволюцію в часі просторових об′єктів. Така еволюція може бути дискретною або безперервною в часі. У разі безперервного часу говорять про рухливі об′єкти, тож у зв′язку з цим вводять поняття рухливих точок, ліній, багатокутників.

Об′єкти, які переміщуються, мають свої операції, функції і предикати. Було висунуто декілька підходів для моделювання дискретних змін просторових об′єктів. Один з них – впровадження в темпоральні БД просторових типів даних [331]. Інший підхід [332] – залишити просторові об′єкти без змін, але доповнити кожну компоненту об′єкту (до прикладу, точку чи сегмент) темпоральною характеристикою. Просторово – часові типи даних дозволяють описувати динамічну поведінку просторових об′єктів у часі [333].

Було запропоновано просторово – часову мову запитів STQL [334] на базі SQL. У [277] також подається варіант просторового SQL. У [334 - 336] пропонуються просторово – часові предикати. У праці [337] подається огляд сучасного стану досліджень ПЧБД.

БД переміщуваних об′єктів (БДПО). Це просторово – часова база даних, призначена для фіксації та відстеження місцезнаходження об′єктів, що рухаються. Дослідження БДПО були ініційовані в кінці 90-х років минулого стріччя [333, 338, 339, 340, 341]. Зазвичай БДПО послуговуються пласкою просторово – мережевою моделлю даних [342, 343].

Два дослідження практично дали життя цьому напрямку. По-перше, була запропонована модель MOST (Moving Objects Spatio-Temporal) [338, 339], яка дозволила відслідковувати в БД набір залежних від часу місцезнаходжень, наприклад, рух транспортного засобу. Було введене поняття динамічного атрибуту й визначена мова запитів FTL (Future Temporal Logic), яка дозволяла специфікувати змінні в часі взаємозв′язки між очікуваними місцезнаходженнями рухомих об′єктів. Нарешті були висунуті рішення щодо обліку невизначеності під час обробки запитів.

Другою важливою подією цього часу було відкриття 1996 року Європейського проєкту Chorochronos [340], де була здійснена спроба інтегрувати концепції просторових і часових баз даних. У цьому проєкті для представлення переміщуваних об′єктів була запропонована так звана модель обмежень (constraint model) [344, 345] і розроблений прототип DEDALE [341].

Розрізняють два види БДПО: в першому випадку БДПО моделює, представляє і дає можливість формулювати запити до передісторії переміщення для проведення наступного просторово – часового аналізу [346, 347]. Другий різновид дозволяє моделювати, прогнозувати і робити запити на поточне і майбутнє переміщення [338, 348]. В цьому випадку доводиться обирати між неточністю прогнозних результатів і витратами на оновлення БД [339], що дає рішення задачі управління невизначеністю [349].

Поширеним підходом у дослідженнях із БДПО є створення спеціальних типів даних (точки, області й багатокутники, що переміщуються), а також спеціальних операцій і предикатів. До прикладу, в [333] пропонуються типи даних, які дозволяють задавати залежні від часу просторові об′єкти та операції над ними. Система типів даних для рухомих об′єктів була строго визначена в праці [350].

На завершення підкреслимо, що в монографіях [335, 351, 352] детально висвітлені практично всі питання, дотичні до рухомих об′єктів.

Просторові СУБД.

Чимало розповсюджених комерційних СУБД підтримують роботу з просторовими даними.

Серед реляційних СУБД до них належать: Oracle Database Spatial, MS SQL Server 2008, DB2 Spatial Extender, Informix Spatial Blade, MySQL Spatial, Spatial Query Server корпорації Boeing, розширення PostGIS СУБД PostgreSQL, розширення SpatiaLite для SQLite.

Серед NoSQL-систем підтримка просторових даних реалізована в MongoDB, RethinkDB, Cassandra.

Дедуктивні бази даних.

Із зростанням обсягів інформаційних ресурсів загострюється проблема їх розуміння й інтерпретації, особливо, якщо це стосується складних предметних областей. Для вирішення цієї проблеми необхідно мати механізми підтримки міркувань, аби робити складні висновки. Тож для цього почали залучати математичну логіку.

В кінці 70-х років почалося формування підходів до використання апарату логіки в базах даних [5, 6]. 1982 року Чандра і Харел [353] опублікували статтю, яка вважається першою працею в галузі теорії дедуктивних баз даних (ДБД). ДБД як один із напрямків теорії баз даних, почали стрімко розвиватися в середині 80-х років минулого століття. Висхідним пунктом появи і становлення ДБД була теорія логічного програмування, зокрема, Prolog.

ДБД – це результат об′єднання логічного програмування із реляційними базами даних. ДБД виразніші за реляційні бази даних, але менш виразні ніж системи логічного програмування. ДБД – це система баз даних, здатна робити висновки на основі правил і фактів, які зберігаються в базі даних. ДБД являє собою базу фактів і базу правил, де перша в теорії ДБД має назву екстенсійна база даних (ЕБД), а друга називається інтенсійною базою даних (ІБД). ЕБД подається у вигляді реляційних відношень, а ІБД являє собою підмножину Prolog: без функціональних символів і спеціальних предикатів типу cut і var.

ІБД – це сукупність правил, котрі з логічної точки зору виглядають як хорновські диз'юнкти. Такі правила мають форму «якщо А, то В», де А називається «тілом», В – «головою». Тіло складається з кон′юнкції літералів (підцілей). Літераль – це атом або його заперечення. Атом – це предикат, який містить змінюваності або константи.

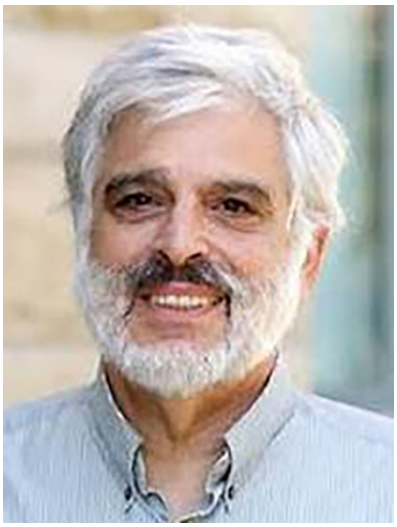
В багатьох статтях та енциклопедіях поняття ДБД і мова Datalog розглядаються як синоніми. Вважається, що найперша опублікована згадка терміну «Datalog» була у 1985 році в рукопису [354]. Пізніше термін «Datalog» згадувався у працях [355, 356]. Однак усе ж заведено вважати, що офіційно мова Datalog уперше була досліджена в книзі Майєра (David Maier) і Уоррена (David S. Warren) [7] як спрощений варіант Prolog без функціональних символів. Автори цю назву пояснювали тим, що предикати без функціональних символів нагадують відношення бази даних.

Давід Майер

Зауважимо, що 1985 року термін Datalog також був вжитий як мова запитів до бази даних в системі, що підтримує природну мову [357]. Цей термін являв собою скорочення від “database dialogue» і ніякого відношення до Prolog не мав.

У будь – якому разі до середини 80-х років термін Datalog ствердився як мова дедуктивних даних, а не як мова логічного

програмування. 1989 року була опублікована чудова монографія Джеффрі Ульмана (Jeffrey D. Ullman) [6], де окремий розділ був присвячений детальному викладу суті Datalog як логічної моделі даних. Багато чого з цього розділу використовується згодом у переліку принципів Datalog.



Джеффрі Ульман

Безпечне правило. Щоб Datalog -правила інтерпретувалися операціями над кінцевими відношеннями, змінні правила мають бути обмежені, тобто мають знаходитися принаймні в одному атомі (літералі без заперечення). Правило безпечне за умови, якщо всі його змінні обмежені. На цей аспект було звернуто увагу в працях [358, 359].

Розрізняють три види правил:

прості – без рекурсії та заперечень;

рекурсивні – які містять рекурсивні визначення;

із запереченнями – які містять атомарні формули із запереченнями.

Прості правила. Для простих правил існує спосіб їх перетворення у вирази реляційної алгебри (див., наприклад, [6]). Отримані вирази визначають відношення для предикатів ІБД і являють собою єдину мінімальну модель.

Рекурсивні правила. Для рекурсивних Datalog-програм, що не містять заперечу вальних під цілей, існує єдина мінімальна модель, яка містить задані ЕБД-відношення, і ця модель є єдиною мінімальною нерухомою точкою відносно ЕБД-відношень відповідних Datalog-правил. Відомо, що у разі, якщо функція монотонна, то її рекурсивне обчислення приводить до нерухомої точки. В реляційній алгебрі всі операції, за винятком різниці, є монотонними. Тому в реляційній алгебрі, розширеній циклами, можливо побудувати алгоритм, що обчислює найменшу нерухому точку рекурсивної Datalog-програми. В [6, 360] подані огляди з оптимізації рекурсивних запитів у ДБД.

Правила із запереченнями. Правила можуть мати в тілі заперчувальні підцілі. В такому разі виникають дві проблеми. По-перше, заперечення можуть викликати нескінченні інтерпретації. Тому було розширено вимогу існування безпекових правил із запереченнями – змінні, що зустрічаються в заперечуваних підцілях, обов′язково мають бути в під цілях без заперечень. Друга проблема пов′язана з тим, що за наявності заперечень Datalog-програма може мати безліч мінімальних моделей (мінімальних нерухомих точок). Тоді зміст програми із запереченнями задається вибором певної «переважаючої» моделі [361 - 368]. Заперечення викликають також проблеми в рекурсії. Через це було введено поняття стратифікованого заперечення [361, 362, 369, 370] і стратифікованих програм, що мають інтуїтивно зрозумілу семантику [371 - 374]. Стратифікація не гарантує існування найменшої нерухомої точки. Але це обмеження уможливлює вибір серед численних мінімальних нерухомих точок такої, яка буде інтерпретацією смислу Datalog-програми. Були також визначені більш специфічні класи стратифікованих програм, зокрема, локально стратифіковані програми [375], модульно стратифіковані програми [376].

Оптимізація. Однією з найскладніших проблем створення ДБД є оптимізація. Для нерекурсивних правил проблема оптимізації аналогічна традиційній реляційній оптимізації, а за наявності рекурсії і заперечення виникають додаткові проблеми і можливості. У цьому напрямку було здійснено численні дослідження. Серед них виділимо метод магічних множин (magic-sets) [6], а також низку методів, що на ньому базуються [377], алгоритм підрахунку (counting algorithm) [378], факторингову оптимізацію (factoring optimization) [379], метод видалення зайвих правил і літералів [380], метод оптимізації екзистенційних запитів [381],

метод «конвертів» (envelopes) [382] тощо. У праці [383] здійснено аналітичний огляд різних стратегій оптимізації і порівняльний аналіз їхньої продуктивності.

Дедуктивні системи баз даних. Щодо створення дедуктивних систем баз даних, то тут можна виділити два напрямки. З одного боку проводилися дослідження й розроблялися проєкти зі створення самостійних систем. Із їх оглядом можна ознайомитися в [384, 385]. З іншого боку, 1999 року в чергову версію SQL (SQL99) було закладено можливість формулювати й виконувати рекурсивні запити. Як уже було вище згадано, прості правила й спеціальні правила із запереченнями повністю виражаються в реляційній алгебрі, а, отже, й в стандартному (без рекурсії) SQL. Введення рекурсії в SQL привело до того, що всі можливості дедуктивних Datalog-програм почали виявлятися в SQL99. Додатково виникали можливості вказувати напрямок пошуку (вшир, вглиб), фіксації та запобігання безкінечних циклів, створювати рекурсивні представлення, визначати пряму і взаємну рекурсію, лінійну і нелінійну рекурсію. З робочим варіантом стандарту рекурсивного SQL можна ознайомитись в [386], а з коротким списком практичного використання – в [387].

Слід зазначити, що до кінця 80-х років сформувався напрямок об′єктно-дедуктивних баз даних. Для них були розроблені спеціальні мови, як от O-Logic, F-Logic, ROL, IQL [388, 389]. У статті [390] подається аналітичний огляд об′єктно-дедуктивних баз даних.

Активні бази даних.

Традиційні бази даних пасивні. Дані розміщуються, оновлюються, переносяться і вибираються їхніми БД під впливом зовнішніх чинників (людина або програма). Бізнес-правила, що застосовуються до вмісту бази даних, також, керуються як правило, зовнішніми джерелами. Коротше кажучи, традиційні бази даних не є активними учасниками функціонування інформаційної системи і забезпечують лише функцію зберігання даних. Для подолання цього недоліку було введено концепцію активних баз даних. Активна база даних (АБД) – це база даних, стосовно якої СУБД виконує не лише дії, що їх явно вказує користувач, а й додаткові дії, згідно із правилами, закладеними в саму БД.

Зародження ідей АБД пов′язують із появою концепції тригеру-механізму, який уперше був запропонований у дослідницькому проєкті System R компанії IBM. Підтримка концепції тригеру передбачалася в мові цієї системи SEQUEL. Одначе, варто зазначити, що ідея тригеру раніше була втілена в мові визначення даних CODASYL [33, 34], (хоча сам термін «тригер» ще не вживався). В мові передбачалася підтримка концепції процедури бази даних, яка може асоціюватися з різними об′єктами бази даних у специфікації схеми. Процедура бази даних запускається автоматично в разі, якщо над об′єктом, з яким вона асоційована, виконується одна з даних у специфікації операцій. Водночас виконання процедури може передувати виконанню заданої операції, відбуватися після неї або мати місце у разі виникнення помилки.

АБД має передбачати підтримку таких можливостей:

- містити логіку обробки даних (бізнес-правила) власне в базі даних, щоб вона управлялася через СУБД, а не прикладними програмами чи користувачами;
- забезпечити моніторинг подій і умов, які впливають на дані й можуть ініціювати обробку даних, що ними управляє СУБД;
- містити спосіб, із яким ці події й умови могли б запускати логіку обробки даних всередині бази даних.

ECA – правило. Для підтримки згаданих вище можливостей в активну базу даних було введено поняття ECA-правила – це конструкція із трьох складників: подія, умова й дія (Event-Condition-Action). Вперше воно було визначено в проєкті HiPAC (High Performance ACtive database system) [392]. Семантика правила проста: коли відбувається подія, перевіряється умова, і, в разі її істинності, відбувається дія.

Умовою ECA-правила можуть бути: запит до бази даних, логічне вираження, виклик підпрограми (процедури або функції, що повертає логічне значення).

Дія ECA-правила – довільний код, викликаний появою події та за істинності

умови. Дія виконується або у вигляді складової частини транзакції ECA-правила, або у вигляді самостійної транзакції залежно від режиму зв′язування. В межах одного ECA-правила може виконуватись кілька дій одночасно, тому варто відслідковувати конфліктні ситуації. Тіло дії може ініціювати події, котрі викликають виконання іншого правила тощо. І, зрештою, ланцюжок послідовно ініційованих вкладених правил може бути рекурсивним.

Моделі ECA-правил. Для опису ECA-правил були висунуті дві моделі: модель знань (knowledge model) і модель виконання (execution model) [391, 392]. Модель знань описує, чим є активні правила, а модель виконання специфікує, яким чином інтерпретуються ECA-правила в процесі їх виконання.

Модель знань ECA-правил. У праці [393] було узагальнено і класифіковано характеристики моделі знань, що раніше були представлені в літературі [394 - 396]. Ці характеристики відносяться до всіх складових ECA-правил і наводяться далі.

Характеристики моделі знань події. До них належать наступні:

- *Джерела подій* (event sources): операції над структурними елементами бази даних (structure operations), поведінка зовнішнього середовища (behaviour invocation – дії користувачів або прикладних програм), команди транзакцій (begin, abort, rollback commit), часові характеристики;

- *Виявлення подій* – це процес аналізу потоку подій для знаходження тих, що відповідають заданому зразку. Зазвичай виявлення подій включає процедури фільтрації й агрегації. Основоположні дослідження із виявлення подій були здійснені під час виконання проєктів HiPAC [392, 397], Snoop [398, 399], ODE [400], SAMOS [401]. В [402] подано огляд досліджень із виявлення подій, а в статті [403] – специфікацій подій.

- *Гранулярність подій* (event granularity) – вказує, чи визначається подія для кожного об′єкту з множини, або з даної підмножини, або конкретних об′єктів множини.

- *Прості й складові події.* Подія, що об′єднує кілька подій, називається складовою. Піонерською працею в сфері складових подій вважається проєкт HiPAC [392]. Для опису складових подій у межах розроблених систем були визначені різні алгебри, як-от [395, 398, 399, 401, 404]. У праці [398] для складових подій введена характеристика «політика споживання» (consumption policy). Вона визначає ситуацію, де , як вважається, складова подія відбулася. В праці [405] подається загальний метод і мова EPL специфікації семантики складових подій.

Характеристики моделей знань умови. До них належать:

– *факультативність – чи є умова обов'язковою в ECA-правилі, чи ні. Вважається, що в правилі має бути присутня або подія, або умова;*

– *контекст* – загалом як контекст ECA-правила пропонуються такі варіанти:: стан бази в період – запуску транзакції (DBT), -ініціювання події (DBE), - перевірки умови (DBC), - виконання дії (DBA), а також прив′язка умови до події (BindE), а також дії до умови (BindC). За контекст умови беруться DBT, DBE, DBC и BindE.

Характеристики моделей знань дії. До них належать:

– *види дій.* Пропонуються такі: робота зі структурою бази даних, ініціювання зовнішнього середовища, інформування, аварійне завершення, виконання іншої дії чимось, що ініціювало подію («do-instead» Стоунбрейкера [394]);

– *контекст* – аналогічно контексту умови специфікує, які саме дані доступні дії. Допустимими значеннями є DBT, DBE, DBC, DBA и BindC.

Модель виконання ECA-правил. Модель виконання (execution model) специфікує, яким чином трактуються ECA-правила в процесі їх виконання [391]. Вона стосується подій, умов і дій ECA-правил.

Модель виконання подій. Якщо активна база даних підтримує виявлення складових подій, то необхідні правила їх виявлення й відпрацювання. Для вирішення цієї ситуації були запропоновані так звані «режими споживання подій» (event consumption modes) [398, 404, 406]. Далі подаються характеристики моделі виконання умов і дій ECA-правил.

Правила перевірки умов і виконання дій. Режими зв′язування (Coupling modes). Вони визначають, як ініціюється

перевірка умови у відповідь на подію, що відбулася, і як планується, контролюється і виконується дія ECA-правила у разі позитивного результату перевірки умови. Вперше вони були досліджені у проєкті HiPAC [397]. Визначаються вони для пар подія-умова і умова-дія. Зв'язування подія-умова визначає, коли слід перевірити умову відносно події, а умова-дія – коли слід виконати дію щодо умови. Для обох видів зв'язування було запропоновано однакові варіанти зв'язування:

– *негайно* (immediately) – умова/дія перевіряється/виконується одразу ж після події/умови;

– *відстрочено* (deferred) – перевірка/виконання умови/дії відтерміновується до завершення транзакції (до виконання Commit), де ініційований тригер. Були також запропоновані варіанти, коли відстрочка задавалася визначеним користувачем часом відстрочки [407] або ж виконанням спеціальних команд [396];

– *окремо* (detached) – перевірка/виконання умови/дії здійснюється в іншій транзакції, ніж подія/умова, причому виконання дії може залежати або бути незалежним від завершення транзакції, де відбулася подія чи було перевірено умову.

У джерелі [397] було встановлено, що в тригері не всі варіанти пар значень режимів зв'язування є допустимими. Зазначимо також, що в дослідницькому проєкті REACH (REal-time ACtive Heterogeneous System) [408] було висунуто ще два варіанти режиму зв'язування для підтримки побічних ефектів незворотніх дій ECA-правил.

Запуск подією кількох правил. Можлива ситуація, коли подія ініціює запуск кількох правил. У такому разі було запропоновано механізми планування порядку виконання правил [391, 394, 409].

Політика підсумкового ефекту (net effect policy). Для випадку, якщо в межах одного правила виконується кілька дій відносно одних і тих самих даних, пропонується політика підсумкового ефекту [391], коли виконується лише одна дія або навіть не виконується жодна.

Виклик правилом іншого правила. Правило може ініціювати виклик іншого правила тощо. При цьому виникають ситуації, коли виконання внутрішнього правила суперечить виконанню зовнішнього правила. А також можливі цикли, коли правило ініціює виконання самого себе. Ці ситуації також розглядаються в літературі [391].

Системи активних баз даних. Розроблено системи активних реляційних баз даних (РБД) і об'єктно-орієнтованих баз даних (ООБД).

Активні РБД. Включення активних механізмів до РБД не є чимось новим. Переважна більшість комерційних систем підтримують механізми тригерів. Окрім того, досліджуються розробки розвинутіших засобів підтримки активних правил. Пропозиції із включення активної поведінки в реляційні системи, зазвичай, обмежуються можливостями традиційних пасивних реляційних систем. Приміром, подіями, які ініціюють правила, є лише операції над даними (вставка, видалення, заміна). Як правило, в реляційних системах не розглядаються складові події. Вони не мають розвинутих режимів зв'язування, і мова опису правил вбудовується в мову запитів. Серед «ранніх» реляційних систем, що мали механізми активації, можна виділити Starburst [396, 410], Postgress [411], Ariel [412]. Прикладами активного розширення реляційної моделі даних можуть бути праці [413 - 419]. Серед них особливий інтерес становлять ті, що досліджують взаємозв'язок активних і дедуктивних баз даних [415 - 419].

У стандарт SQL 1999 року були включені тригери [420]. Відтоді всі промислові реляційні СУБД як механізм активних правил включають щонайменше тригери SQL.

Активні ООБД. Щодо ООБД, то на відміну від реляційних, вони завжди підтримували тісний зв'язок між поведінкою користувачів і даними БД. Така поведінка представляється методами, приписуваними класам даних БД. Цей факт, а також інкапсуляція структури об'єкту, вказують, що деякі аспекти, котрі можуть представлятися в РБД за допомогою активної поведінки, в ООБД підтримуються за допомогою методів. Однак дослідження з активного розширення ООБД почалися практично одночасно з появою самих ООБД. А їхня суттєва відмінність полягає в тому, що в активних ООБД примітивні події часто асоціюють-

ся з викликами методів, а не з операціями над структурними елементами БД. Розроблена чимала кількість систем активних ООБД, серед яких можна відзначити HiPAC [392, 395, 421], EXACT [407], NAOS [422], Chimera [423], Ode [424], SAMOS [401], Sentinel [406], REACH [408]. Зі стислим описом цих систем можна ознайомитись у [391], там же є посилання на інші праці з активного розширення ООБД.

Система HiPAC. На завершення зазначимо, що HiPAC стала однією з передових систем активних баз даних свого часу і єдиною, орієнтованою на потреби додатків реального часу. Це привело до створення інноваційної моделі тригеру. Модуль ECA-правил, поданий в HiPAC, нині широко застосовується в активних обчислювальних системах, у системах обробки складних подій і в розподілених системах. Керівник проєкту HiPAC Умешвар Дайал (Umeshwar Dayal) 2010 року став лауреатом інноваційної премії SIGMOD імені Едгара Ф. Кодда за піонерські праці і суттєвий внесок у розподільчі гетерогенні бази даних, високопродуктивні активні бази даних, моделі довгочасних транзакцій та дослідження в галузі бізнес-процесів.

Умешвар Дайал

Об′єктні бази даних.

Поява напрямку об′єктних баз даних (ОБД) визначалася, передовсім, потребами практики: необхідністю розробки складних прикладних систем, для яких технологія попередніх систем баз даних не була цілком задовільною. Дослідження в галузі ОБД почалися в зв′язку з необхідністю розробки ефективного механізму, що дозволяв би об′єктно-орієнтованим додаткам зберігати об′єкти по закінченні своєї роботи і користуватися ними під час наступного запуску. Тобто необхідно було об′єктно-орієнтованому середовищу надати прозорий механізм збереження й вибірки об′єктних даних з баз даних.

ОБД виникли не на порожньому місці. Відповідний базис забезпечувався працями в галузі баз даних, напрямками мов програмування з абстрактними типами даних і об′єктно-орієнтованих мов програ-мування.

Перші об′єктні СУБД. На початку 80-х років чимало дослідницьких груп з університетів, наукових інститутів, провідних комп′ютерних компаній і невеликих фірм – початківців взялися за створення ООСУБД. Було випущено перші промислові ООСУБД G-Base (1986 г), Gemstone (1987 г.), IRIS (1987), Statice (1988 г.), Vbase (1988 г.), ObjectStore (1988 г.), Versant (1988 г.), O2 (1988 г.), ORION (1989 г.).

Два напрямки в ОБД. До кінця 80-х років визначилися два напрямки у створенні об′єктних баз даних: об′єктно-орієнтовані бази даних (ООБД) і об′єктно-реляційні бази даних (ОРБД). ООБД користуються об′єктно-орієнтованою мовою програмування як мовою бази даних і забезпечує цілісність об′єктів із наданням усіх функціональних можливостей, властивих традиційним базам даних. ООБД передбачають створення самостійних об′єктно-орієнтованих систем управління базами даних (ООСУБД). ООСУБД реалізує гнучку модель, базовану на тій же парадигмі, що й об′єктно-орієнтовані мови програмування. ООСУБД забезпечують глибшу інтеграцію з об′єктно- орієнтованими додатками, ніж реляційна база даних.

У свою чергу ОРБД розширює можливості реляційних баз даних засобами підтримки об′єктів.

Маніфест об′єктно-орієнтованих систем баз даних. 1989 року група провідних спеціалістів і дослідників баз даних написали «Маніфест об′єктно-орієнтованих систем баз даних» [425] (так званий Перший маніфест). Це був перший документ, де була зроблена спроба дати визначення системам об′єктно-орієнтованих баз даних.

Були описані основні властивості і характеристики, що їх мала мати технологія ООБД.

У ньому також зазначається, що поточний стан справ у проблематиці ООБД характеризується: відсутністю загальноприйнятої моделі даних, відсутністю формальної теорії і активної експериментальної діяльності.

Загальноприйнятна об′єктно-орієнтована модель даних була відсутня не тому, що не було жодної розробленої повної моделі, а через відсутність спільного узгодження щодо прийняття будь-якої моделі. Що ж до формальної теорії, то для ООБД мало б бути щось подібне до того, що створив Ковальський для логічного програмування. Необхідність такої теорії очевидна: формальна семантика основних понять ООБД визначена недостатньо. ЇЇ відсутність практично унеможливлювала досягнення консенсусу відносно моделі даних.

Дослідження у сфері об′єктних баз даних особливо активно відбувалися у 80-і роки. Це привело в кінці 80-х до виникнення промислових компаній і ринку систем управління об′єктними базами даних (СУОБД). Водночас ринок об′єктних баз даних гостро потребував стандарту. Вирішальне слово в цій, як, власне, і в інших проблемах, пов′язаних із об′єктними базами даних, було сформульоване в стандарті ODMG.

Стандарт на зберігання об′єктів ODMG 3.0. Влітку 1991 року в США була створена Object Data Management Group (ODMG) – Група Управління Об′єктними даними – як консорціум виробників СУОБД. ODMG очолив Рік Кеттелл (Rick Cattell).

Рік Кеттелл

Завданням групи була розробка стандарту на збереження об′єктів у базах даних. У період із 1993 по 2001 рік ODMG опублікувала п′ять версій своєї специфікації. Останньою з них була версія 3.0 [434], після чого група завершила свою роботу.

Стандарт на зберігання об′єктів ODMG – 3.0 розроблено на основі трьох існуючих стандартів: управління базами даних (SQL), стандарти OMG - Object Management Group і стандарти на об′єктно-орієнтовані мови програмування (C++, Smalltalk, Java). ODMG сприяє можливості взаємодії з базами даних в об′єктно-орієнтовані мови програмування. Стандарт складається з таких частин:

- *Об′єктна модель* – уніфікована основа всього стандарту. Вона розширює об′єктну модель консорціуму OMG.

- *Мова визначення об′єктів* (ODL - Object Definition Language) – засіб визначення типів об′єктів, які відповідають об′єктній моделі даних ODMG. ODL використовується для підтримки перенесення об′єктних схем між відповідними системами управління об′єктними даними (СУОД).

- *Мова об′єктних запитів* (OQL - Object Query Language) - SQL –подібна декларативна мова, яка надає ефективні засоби для отримання об′єктів із бази даних.

- *Формат обміну об′єктами* (OIF - Object Interchange Format) – мова опису завантаження і розвантаження поточного стану СУОД в/з файлів. Використовується для обміну збереженими об′єктами між СУОД.

- *Зв′язування з ОО-мовами.* Стандарт зв′язування з C++, Smalltalk і Java визначає Object Manipulation Language (OML) – мова маніпулювання об′єктами, яка розширює базові ОО-мови шляхом маніпулювання і збереження об′єктів.

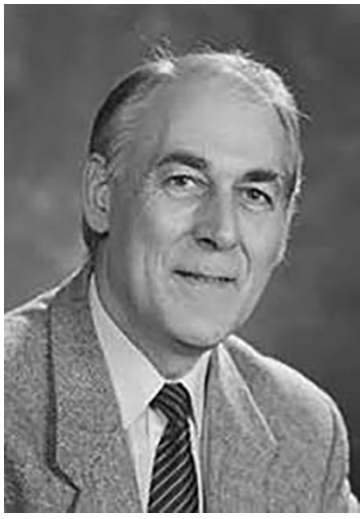
Другий маніфест. Товариство дослідників реляційних баз даних відповіло на активність у сфері OODB своїм маніфестом на підтримку ORBD. 1990 року М. Стоунбрейкер та його колеги з комітету перспективних систем БД опублікували «Маніфест систем баз даних третього покоління» [435] (так званий Другий маніфест), де стверджувалось, що СУБД третього покоління, тобто ті, що прийдуть за реляційними, мають бути створені на основі реляційних техно-

логій. Прихильники цього напрямку дотримуються принципу еволюційного розвитку можливостей СУБД без докорінної відмови від попередніх підходів та збереження наступності з системами попереднього покоління. Цей принцип був підтриманий у процесі створення дедуктивних і темпоральних баз даних. Так само розвивалося створення об′єктно- реляційних баз даних.

На основі цієї ідеї під керівництвом М. Стоунбрейкера в університеті Берклі (Каліфорнія, США) було розроблено СУБД Postgres [436], Це була перша практично реалізована об′єктно-реляційна система баз даних, де був продемонстрований підхід з інтеграції об′єктних і реляційних концепцій.

Варто також згадати Вона Кіма (Won Kim), який 1991 року випустив систему UniSQL [437], що також вважається однією з перших об′єктно-реляційних СУБД.

Третій маніфест. У березні 1995 року була опублікована стаття Хью Дарвена (Hugh Darwen) і Крістофера Дж. Дейта (Christopher J. Date), названа авторами «Третім маніфестом» [438].



Хью Дарвен

В ній висловлювалася думка щодо майбутніх систем управління базами даних і підхід з інтеграції реляційної та об′єктної технологій. У маніфесті піднята проблема щодо вирішення задачі невідповідності між об′єктно-орієнтованими мовами програмування і системами управління реляційними базами даних. Автори пропонують взяти за основу реляційну базу даних у вигляді визначених користувачем типів.

Крістофер Дж. Дейт

Схеми реалізації ОРБД. Пропонувалися різні схеми реалізації ОРБД. Як – от:

Об'єктно – реляційний шлюз (Object-Relational Gateway). Об′єктно – орієнтований додаток працює як звичайний користувач із об′єктною мовою, а шлюз виділяє і замінює всі об′єктно – орієнтовані елементи цієї мови на їх реляційні компоненти. Попри зниження продуктивності, такий варіант дозволяє програмістам цілком сконцентруватися на об′єктно – орієнтованій розробці.

Об'єктно – реляційний інтерфейс (Object-Relational Interface). Між ООБД і ОРБД знаходиться проміжний інтерфейс, що відображає об′єктні конструкції в реляційні і навпаки. Об′єктно – орієнтований додаток працює з ООСУБД, яка через інтерфейс взаємодіє з реляційною СУБД.

Уніфіковані СУБД (unified DBMS). Ще одним вирішенням є створення гібридних об′єктно – реляційних СУБД, здатних зберігати і традиційні табличні дані, й об′єкти.

Об′єктно – реляційні СУБД. У другій половині 90-х років провідні компанії почали випускати СУБД, які підтримували об′єктно – реляційну модель даних. Першою 1996 року з′явилася на ринку СУБД Informix, створена на основі системи Illustra Стоунбрейкера. 1997 року була випущена об′єктно – реляційна версія СУБД DB2 компанії IBM, в основі якої був дослідницький прототип Starburst IBM Almaden. Цього ж року компанія Oracle випустила продукт такого ж класу Oracle 8. Сьогодні практично всі сучасні реляційні СУБД є об′єктно – реляційними. Всі вони розширюють реля-

ційну базу даних засобами представлення об′єктів.

Позиції об′єктно – реляційного підходу зміцнилися завдяки прийнятій 1999 року версії стандарту SQL-3, де було введено підтримку об′єктно – орієнтованої концепції (структурні типи даних, типізовані таблиці, об′єкти, методи, які визначаються користувачем).

(Далі буде)

References

212. Snodgrass R.T., Ahn I. A taxonomy of time databases. ACM SIGMOD Record, 1985,Vo. 14, No 4, pp. 236-246
213. Bubenko J.A, Jr. The temporal dimension in information modeling. Technical Report RC 6187 #26479, IBM Thomal J. Watson Research Center, Nov. 1976
214. Bubenko J.A. Jr. The Temporal Dimension in Information Processing. In: Proceedings of IFIP WG 2.6 Working Conference on Architecture and Models in Data Base Management Systems, G M Nijssen, Ed, North Holland, 1977, pp. 93-118
215. Breutmann B., Falkenberg E., Mauer R. "CSL: a language for defining conceptual schemas". in Proceedings of the Database Architecture Conference, Venice, June 1979, pp. 237-256
216. Hammer M., McLeod D. Database Descitption with SDM A Semantic Database Model ACM Transactions on Database Systems, 6, No 3, Sep 1981, pp 351-386
217. Klopprogge M.R. TERM: An Approach to Include the Time Dimension in the Entity- Relationship Model. In: Proceedings of the Second International Conference on the Entity Relationship Approach, Washington, DC, pp. 477–512 (October 1981)
218. Anderson,T.L. Modeling Time at the Conceptual Level. In Improving Database Usability and Responsiveness, Ed. P. Scheuermann Jerusalem, Israel Academic Press, 1982, pp. 273-297
219. Codd, E.F. Extending the database relational model to capture more meaning. ACM Transactions on Database Systems, Vol. 4, No. 4, Dec 1979, pp 397-434
220. Sernadas A Temporal aspects of logical procedure definition. Information Systems, 1980, vol. 5, No 3, pp. 167-187
221. Clifford, J. and Warren D.S. Formal semantics for time in databases. ACM Transactions on Database Systems, vol. 8, No 2, June 1983, pp. 214-254
222. Ariav G. A temporally oriented data model. ACM Transactions on Database Systems (TODS), 1986 vol. 11, No 4, pp. 499-527
223. Ariav G., Morgan H.L., Zisman M.D. MDM: Embedding the time dimension in information systems, Technical Report 82-03-01 Department of Decision Sciences, Wharton School, University of Pennsylvania, 1982
224. Ben-Zvi J. "The Time Relational Model," PhD thesis, Computer Science Dept., UCLA, 1982
225. Gadia S. Ben-Zvi's Pioneering Work in Relational Temporal Databases. In: Tansel A. et al. Temporal Databases: Theory, Design, and Implementation (Redwood City, CA: The Benjamin/Cumming s Publishing Company, 1993). pp. 202-207
226. Jones S., Mason P.J. Handling the Time Dimension in a Data Base. In Proceedings of the Inlernational Conference on Data Bases, Eds. S.M. Deen and P Hammersley British Computer Society University of Aberdeen, Heyden, July 1980 pp 65-83
227. Snodgrass R. The temporal query language TQuel. In PODS '84: Proceedings of the 3rd ACM SIGACT-SIGMOD symposium on Principles of database systems,
228. Allen J.F. "Maintaining knowledge about temporal intervals". Communications of the ACM, Nov. 1983, 26(11. pp.832-843
229. Clifford J., Dyreson C.E., Isakowitz T., Jensen C.S., Snodgrass R.T. On the semantics of "now.". ACM Trans Database Syst. 1997; 22(2), pp.171–214.
230. Lorentzos N.A., Johnson R.G. TRA a model for a temporal relational algebra. In: Rolland C, Bodart F, Leonard M, editors. Temporal aspects in information systems. North-Holland; 1988. p. 203–215.
231. Lorentzos N.A., Johnson R.G. Extending relational algebra to manipulate temporal data. Inf Syst. 1988;13(3):289–296.
232. Tansel A.U. Adding time dimension to relational model and extending relational algebra. Information Systems. 1986;11(4):343–355.
233. Tansel A.U. Temporal relational data model. IEEE Transactions on Knowledge and Data Engineering, 1997, 9(3): 464-479

234. Tansel A.U., Atay C.E. Nested bitemporal relational algebra. Conference: Computer and Information Sciences - ISCIS 2006, 21th International Symposium, Istanbul, Turkey, November 1-3, 2006, Proceedings, pp. 622-633
235. Navathe S.B., Ahmed R. A temporal relational model and a query language. Information Sciences: an International Journal. 1989;49(1-3):147–175.
236. Gadia S.K. A homogeneous relational model and query languages for temporal databases. ACM Trans Database Syst. 1988;13(4):418–448.
237. Clifford J, Croker A. The historical relational data model (HRDM. and algebra based on lifespans. In: Proceedings of the 3rd International Conference on Data Engineering; 1987. p. 528–537.
238. Gregersen H., Jense C.S. Temporal Entity-Relationship Models—a Survey. IEEE Transactions on Knowledge and Data Engineering, 1999, Vol. 11, No. 3, pp. 464 - 497
239. Arora S. A comparative study on temporal database models: A survey, 2015 International Symposium on Advanced Computing and Communication (ISACC), 2015, pp. 161-167,
240. Gandhi L. Literature survey of temporal data models. International Journal of Latest Trends in Engineering and Technology. 2017, Vol. 8 No. 4-1, pp.294-300
241. Vianu V. Dynamic functional dependencies and database aging. J ACM. 1987;34(1):28–59.
242. Wijsen J. Design of temporal relational databases based on dynamic and temporal functional dependencies. In: Clifford J, Tuzhilin A, editors. Temporal databases. Workshops in computing. Berlin/Heidelberg/New York: Springer; 1995. p. 61–76.
243. Wijsen J. Temporal FDs on complex objects. ACM Trans Database Syst. 1999;24(1):127–176.
244. Wijsen J. Reasoning about qualitative trends in databases. IInformation Systems. 1998;23(7):463–487.
245. Wang X.S., Bettini C., Brodsky A., Jajodia S. Logical design for temporal databases with multiple granularities. ACM Trans Database Syst. 1997;22(2):115–170.
246. Baudinet M., Chomicki J., Wolper P. Constraint generating dependencies. J Comput Syst Sci. 1999;59(1):94–115.
247. Jensen C.S., Snodgrass R.T. Temporal specialization and generalization. IEEE Trans Knowl Data Eng. 1994;6(6):954–974
248. Sarda N.L. Algebra and query language for a historical data model. The Computer Journal. 1990;33(1):11–18
249. Lorentzos N.A., Johnson R.G. Extending relational algebra to manipulate temporal data. Information Systems. 1988;13(3):289–296.
250. Tuzhilin A, Clifford J. A temporal relational algebra as basis for temporal relational completeness. In: Proceedings of the 16th International Conference on Very Large Data Bases; 1990. p. 13–23.
251. Tansel A.U., Arkun M.E. HQUEL, a Query Language for Historical Relational Databases. SSDBM'86: Proceedings of the 3rd international workshop on Statistical and scientific database management,1986 pp. 135-142
252. Tansel A.U., Arkun M.E, Ozsoyoglu G. Time-by-example query language for historical databases. IEEE Trans Softw Eng. 1989;15(4):464–478.
253. Snodgrass S. The temporal query language TQUEL. ACM Trans Database Syst. 1987;12(2): 247–298.
254. Lorentzos N.A., Mitsopoulos Y.G. SQL extension for interval data. IEEE Trans Knowl Data Eng. 1997;9(3):480–99.
255. Sarda N.L. Extensions to SQL for historical databases. IEEE Trans Knowl Data Eng. 1990;2(2):220–230.
256. Navathe S.B., Ahmed R. TSQL: a language interface for history databases. In: Rolland C, Bodart F, Leonard M, editors. Temporal aspects in information systems. North-Holland; 1988. p. 109–122.
257. Toman D. Point-based temporal extensions of SQL and their efficient implementation. In: Etzion O, Jajodia S, Sripada S, editors. Temporal databases: research and practice. Springer,; 1997, p. 211–237.
258. Böhlen M.H., Jensen C.S., Snodgrass R.T. Temporal statement modifiers. ACM Trans Database Syst. 2000;25(4):407–456.
259. Snodgrass R.T. editor. In: Proceedings of the ARPA/NSF International Workshop on an Infrastructure for Temporal Databases, 1993.
260. Snodgrass R.T., Ahn I., Ariav G., Batory D.S., Clifford J., Dyreson C.E., Elmasri R., Grandi F., Jensen C.S., Käfer W., Kline N., Kulkarni K., Leung T.Y.C., Lorentzos N., Roddick J.F.,

Segev A., Soo M.D., Sripada S.M. TSQL2 language specification. ACM SIGMOD Rec. 1994;23(1):65–86.

261. Snodgrass R.T., Ahn I., Ariav G., Batory D.S., Clifford J., Dyreson C.E., Elmasri R., Grandi F., Jensen C.S., Käfer W., Kline N., Kulkarni K., Leung T.Y.C., Lorentzos N., Roddick J.F., Segev A., Soo M.D., Sripada S.M. A TSQL2 tutorial. ACM SIGMOD Rec. 1994;23(3):27–33
262. Snodgrass R.T. Editor. The TSQL2 temporal query language. Kluwer Academic; 1995.
263. Snodgrass R.T., Böhlen M.H., Jensen C.S., Steiner A. Adding valid time to SQL/temporal. Change proposal, ANSI X3H2-96-501r2, ISO/IEC JTC1/SC21/WG3 DBL MAD-146r2, Nov 1996.
264. Snodgrass R.T., Böhlen M.H., Jensen C.S., Steiner A. Adding transaction time to SQL/temporal. Change proposal, ANSI X3H2-96-502r2, ISO/IEC JTC1/SC21/ WG3 DBL MAD-147r2, Nov 1996.
265. Snodgrass R.T., Böhlen M.H., Jensen C.S., Steiner A. Transitioning temporal support in TSQL2 to SQL3. In: Ezion O, Jajodia S, Sripada SM, editors. Temporal databases: research and practice. Berlin: Springer; 1998. p. 150–194.
266. Kulkarni K, Michels J-E. Temporal features in SQL:2011. ACM SIGMOD Rec. 2012;41(3):34–43.
267. Reznichenko V.A. Temporal SQL:2011 (Rus). Software Engineering, 2013, vol. 15, No 3-4, pp. 48-65
268. Worboys MF, Duckham M. GIS: a computing perspective. Boca Raton: CRC press; 2004.
269. Shekar S, Chawla S. Spatial databases: a tour. Englewood Cliffs: Prentice-Hall; 2003
270. The Open Geospatial Consortium Date: 2011-12-19 OGC Reference Model. 44 p.
271. Open Geospatial Consortium Inc. Date: 2011-05-28 Editor: John R. Herring OpenGIS® Implementation Standard for Geographic information - Simple feature access - Part 1: Common architecture. 93 p.
272. Open Geospatial Consortium Inc. Date: 2010-08-04 Editor: John R. Herring OpenGIS® Implementation Standard for Geographic information - Simple feature access - Part 2: SQL option. 111 p.
273. Tomlin C.D. A map algebra. In: Proceedings of the Harvard Computer Graphic Conference; 1983
274. Chan K.K.L., Tomlin C.D. Map Algebra as a Spatial Language. In D. M. Mark and A. U. Frank, editors, Cognitive and Linguistic Aspects ofGeographic Space, pp. 351–360.Kluwer Academic Publishers, Dordrecht, 1991
275. Scholl M., Voisard A. Thematic map modeling. In: Proceedings of the 1st International Symposium on Advances in Spatial Databases; 1989. p. 167–190.
276. Güting R.H. Geo-relational algebra: a model and query language for geometric database systems. In: Advances in Database Technology, Proceedings of the 1st International Conference on Extending Database Technology; 1988. p. 506–527.
277. Egenhofer M.J. Spatial SQL: a query and presentation language. IEEE Trans Knowl Data Eng. 1994;6(1): 86–95.
278. Güting R.H, Schneider M. Realm-based spatial data types: the ROSE algebra. VLDB J. 1995;4(2):243–286.
279. Cui Z., A.G. Cohn & D.A. Randell, Qualitative and Topological Relationships in Spatial Databases. 3rd Int. Symp. on Advances in Spatial Databases (SSD'93), LNCS 692, 296-315, 1993.
280. Güting RH. An introduction to spatial database systems. VLDB J. 1994;3(4):357–99.
281. Rigaux P, Scholl M, Voisard A. Spatial databases - with applications to GIS. San Francisco: Morgan Kaufmann Publishers; 2002.
282. Clementini E., Di Felice P. A model for representing topological relationships between complex geometric features in spatial databases. Inf Sci. 1996; 90(1–4):121–136.
283. Schneider M. Spatial data types for database systems - finite resolution geometry for geographic information systems, vol. LNCS 1288. Berlin/New York: Springer; 1997.
284. Schneider M, Behr T. Topological relationships between complex spatial objects. ACM Trans Database Syst. 2006;31(1):39–81
285. Worboys M.F, Bofakos P. A canonical model for a class of areal spatial objects. In: Proceedings of the 3rd International Symposium on Advances in Spatial Databases; 1993. p. 36–52.
286. Egenhofer M.J. & R.D. Franzosa, Point-Set Topological Spatial Relations. Int. Journal of Geographical Information Systems, 5(2), 161-174, 1991.
287. Schneider M.,Weinrich B. An abstract model of three dimensional spatial data types. In:

Proceedings of the 12th ACM International Symposium on Geographic Information Systems; 2004. p. 67–72.
288. Erwig M., Schneider M. Partition and conquer. In: Proceedings of the third international conference on spatial information theory; 1997. p. 389–408.
289. Güting R.H., Schneider M. Realms: A Foundation for Spatial Data Types in Database Systems. 3rd Int. Symp. on Advances in Spatial Databases, LNCS 692, 14-35, 1993.
290. Cicerone S., Di Felice P. Cardinal directions between spatial objects: the pairwise-consistency problem. Inf Sci. 2004;164(1–4):165–88.
291. Clementini E., Billen R. Modeling and computing ternary projective relations between regions. IEEE Trans Knowl Data Eng. 2006;18(6):799–814.
292. Freksa C. Using orientation information for qualitative spatial reasoning. In: Proceedings of the International Conference on Spatial Information Theory; 1992. p. 162–78.
293. Goyal R. Similarity assessment for cardinal directions between extended spatial objects. PhD Thesis, Department of Spatial Information Science and Engineering, University of Maine; 2000.
294. Hernández D. Qualitative representation of spatial knowledge. LNCS, vol. 804. Berlin: Springer; 1994.
295. Ligozat G. Reasoning about cardinal directions. J Visual Lang Comput. 1998;9(1):23–44.
296. Liu W., Li S. Reasoning about cardinal directions between extended objects: the NP-hardness result. Artif Intell. 2011;175(18): 2155–2169.
297. Liu W., Zhang X, Li S., Ying M. Reasoning about cardinal directions between extended objects. Artif Intell. 2010;174(12–13):951–983
298. Mukerjee A, Joe G. A qualitative model for space. In: Proceedings of 7th National Conference on AI; 1990. p. 721–727.
299. Papadias D. Relation-based representation of spatial knowledge. PhD Thesis, Department of Electrical and Computer Engineering, National Technical University of Athens; 1994.
300. Peuquet D.J., Ci-Xiang Z. An algorithm to determine the directional relationship between arbitrarilyshaped polygons in the plane. Pattern Recognit. 1987;20(1):65–74.
301. Skiadopoulos S, Koubarakis M. Composing cardinal direction relations. Artif Intell. 2004;152(2):143–71
302. Skiadopoulos S, Koubarakis M. On the consistency of cardinal directions constraints. Artif Intell. 2005;163(1):91–135.
303. Skiadopoulos S, Giannoukos C, Sarkas N, Vassiliadis P, Sellis T, Koubarakis M. Computing and managing cardinal direction relations. IEEE Trans Knowl Data Eng. 2005;17(12):1610–23.
304. Skiadopoulos S, Sarkas N, Sellis T, Koubarakis M. A family of directional relation models for extended objects. IEEE Transactions on Knowledge and Data Engineering, vol.17, No. 12, 2005, pp 1610–1623
305. Shekhar S., Liu X. Direction as a Spatial Object: A Summary of Results. In R. Laurini, K. Makki, and N. Pissinou, editors, ACM-GIS '98, Proceedings of the 6th international symposium on Advances in Geographic Information Systems, November 6-7, 1998, Washington, DC, USA, pp. 69–75. ACM, 1998.
306. Thanasis Hadzilacos, Nectaria Tryfona. An Extended Entity-Relationship Model for Geographic Applications. ACM SIGMOD Record, Vol. 26, No. 3, 1997, pp. 24–29
307. Shekhar S., Vatsavai R.R., Chawla S., Burke T.E. Spatial pictogram enhanced conceptual data models and their translation to logical data models. In: ISD '99: Selected Papers from the International Workshop on Integrated Spatial Databases, Digital Inages and GIS, 1999 pp. 77–104
308. Gandhi V., Kang J., Shekhar S. Spatial Databases. Technical Report; 07-020, 2007. Retrieved from the University of Minnesota Digital Conservancy, https://hdl.handle.net/11299/215734
309. Frederico T. Fonseca, Max J. Egenhofer. Ontology-driven geographic information systems. In Claudia Bauzer Medeiros, editor, ACM-GIS '99, Proceedings of the 7th International Symposium on Advances in Geographic Information Systems, November 2-6, 1999, Kansas City, USA, pages 14–19. ACM, 1999.
310. Simon Jonathan David Cox, Chris Little. Time Ontology in OWL. Technical Report • July 2016. - https://www.researchgate.net/publication/305810003_Time_Ontology_in_Owl
311. Bennacer N., Aufaure M.A., Cullot N., Sotnykova A., Vangenot C. (2004). Representing

and reasoning for spatiotemporal ontology integration. In R. Meersman, Z. Tari, & A. Corsaro (Eds.), OTM int. conf. on the move to meaningful internet systems (pp. 30–31). Springer.
312. Baglioni M., Masserotti M.V., Renso C., Spinsanti L. (2007). Building geospatial ontologies from geographical databases. In F. Fonseca, M. A. Rodríguez, & S. Levashkin (Eds.), International conference on geospatial semantics (pp. 195–209). Springer.
313. Hogenboom F., Borgman B., Frasincar, F. & Kaymak U. (2010). Spatial knowledge representation on the semantic web. Proceedings of the IEEE 4th International Conference on Semantic Computing (ICSC 2010), pp. 252-259, September 2010.
314. Parent C., Spaccapietra S., Zimányi E. (2006). Conceptual modeling for traditional and spatio-temporal applications: The MADS approach. Springer.
315. Spaccapietra S., Cullot N., Parent C., Vangenot, C (2004). On spatial ontologies. Database Laboratory, Swiss Federal Institute of Technology. 9 p.
316. Egenhofer M.J. Toward the semantic geospatial web. Proceedings of the Tenth ACM International Symposium on Advances in Geographic Information Systems, 2002, pp. 1–4.
317. Fonseca F., Rodriguez M.A. From geo-pragmatics to derivation ontologies: New directions for the geospatial semantic web. Transactions in GIS, 2007, vol. 11, No. 3, pp. 313-316.
318. Subbiah G., Alam A., Khan L. Thuraisingham B. An integrated platform for secure geospatial information exchange through the semantic web. Proceedings of ACM Workshop on Secure Web Services (SWS), 20066 George Mason University, Fairfax, VA, USA
319. Curtin K., Noronha V., Goodchild M., Grise S. ARCGIS transportation model (UNETRANS), UNETRANS data model reference, 2003.
320. Guting R.H. GraphDB: modeling and querying graphs in databases. In: VLDB '94: Proceedings of the 20th International Conference on Very Large Data Bases, 1994, pp.297–308
321. Shekhar S., Liu D.R. CCAM: a connectivity-clustered access method for networks and network computations. IEEE Trans Knowl Data Eng. 1997;9(1): 102–119.
322. Jensen C.S., Kolar J., Pederson T.B., Timko I. Nearest neighbor queries in road networks. In: GIS '03: Proceedings of the 11th ACM International Symposium on Advances in Geographic Information System; 2003. pp. 1–8
323. Papadias D, Zhang J, Mamoulis N, Tao Y. Query processing in spatial network databases. In: VLDB '03: Proceedings of the 29th international conference on Very large data bases - Vol. 29, 2003, pp. 802–813
324. Miller H.J., Shaw S.L. GIS-T data models, geographic information systems for transportation: principles and applications. Oxford: Oxford University Press; 2001.
325. Anez J., de la Barra T., Perez B. Dual graph representation of transport networks. Transp Res. 1996;30(3):209–216.
326. Winter S. Modeling costs of turns in route planning. GeoInformatica. 2002; 6(4):345–361.
327. Hoel E.G., Heng W.L., Honeycutt D. High performance multimodal networks. In: Proceedings of the 9th International Symposium on Advances in Spatial and Temporal Databases; 2005, pp. 308-327.
328. Kohler E., Langtau K., Skutella M. Time-expanded graphs for flow-dependent transit times. In: Proceedings of the 10th Annual European Symposium on Algorithms; 2002, pp. 599–611
329. George B., Shekhar S. Spatio-temporal network databases and routing algorithms: a summary of results. In: Proceedings of the 10th International Symposium on Advances in Spatial and Temporal Databases; 2007. p. 460–477.
330. George B., Shekhar S. Time-aggregated graphs for modeling spatio-temporal networks - an extended abstract. In: Proceedings of the 25th International Conference on Conceptual Modeling; 2006 p. 85–99.
331. Tansel A.U, Clifford J., Gadia S., Jajodia S., Segev A., Snodgrass R.T, editors. Temporal databases: theory, design, and implementation. Benjamin-Cummings Publishing Co., 1993, 633 p.
332. Worboys M.F. A unified model for spatial and temporal information. Comput J. 1994;37(1): 25–34.
333. Erwig M., Güting R.H., Schneider M., Vazirgiannis M. Spatio-temporal data types: an approach to modeling and querying moving objects in databases. Geoinformatica. 1999;3(3):265–291.

334. Erwig M., Schneider M. Developments in spatiotemporal query languages. In: Proceedings of the IEEE International Workshop on Spatio-Temporal Data Models and Languages; 1999. p. 441–449.
335. Güting R.H., Schneider M. Moving objects databases. San Francisco: Morgan Kaufmann; 2005
336. Erwig M., Schneider M. Spatio-temporal predicates. IEEE Trans Knowl Data Eng. 2002; 14(4):1–42.
337. Jitkajornwanich K., Pant N., Fouladgar M., Elmasri R. A survey on spatial, temporal, and spatio-temporal database research and an original example of relevant applications using SQL ecosystem and deep learning, Journal of Information and Telecommunication, 2020, 4:4, 524-559,
338. Sistla A. P., Wolfson O., Chamberlain S., Dao S. Modeling and Querying Moving Objects. ICDE '97: Proceedings of the Thirteenth International Conference on Data Engineering, 1997, pp. 422–432
339. Wolfson O., Chamberlain S., Dao S., Jiang L., Mendez G. Cost and imprecision in modeling the position of moving objects. In: Proceedings of the 14th International Conference on Data Engineering; 1998. p. 588–596.
340. Frank A., Grumbach S., Güting R.H., Jensen C.S., Koubarakis M., Lorentzos N., Manolopoulos Y. Chorochronos: a research network for spatiotemporal database systems. ACM SIGMOD Record, Vol. 28I, No. 3,. 1999, pp 12–21
341. Grumbach S., Rigaux Ph., Segoufin L. The DEDALE system for complex spatial queries SIGMOD '98: Proceedings of the 1998 ACM SIGMOD international conference on Management of data, 1998 pp. 213–224
342. Vazirgiannis M., Wolfson O. A Spatiotemporal Model and Language for Moving Objects on Road Networks. SSTD '01: Proceedings of the 7th International Symposium on Advances in Spatial and Temporal Databases, 2001, pp. 20–35
343. Güting R.H., Victor Teixeira de Almeida, Zhiming Ding. Modeling and querying moving objects in networks The International Journal on Very Large Data Bases, 2006, vol. 15, No. 2, pp 165–190
344. Belussi A., E. Bertino & B. Catania, Manipulating Spatial Data in Constraint Databases. 5th Int. Symp. on Advances in Spatial Databases (SSD'97), LNCS 1262, 115-141, 1997.
345. Rigaux P., Scholl M., Segoufin L., Grumbach S. Building a constraint-based spatial database system: model, languages, and implementation. Inf Syst. 2003;28(6):563–595.
346. Erwig M., Güting R.H., Schneider M., Vazirgiannis M. Spatio-temporal data types: an approach to modeling and querying moving objects in databases. Geoinformatica. 1999;3(3):265–291.
347. Güting R.H., Böhlen M.H., Erwig M., Jensen C.S., Lorentzos N.A., Schneider M., Vazirgiannis M. A foundation for representing and querying moving objects. ACM Trans Database Syst. 2000;25(1): pp. 1–42.
348. Sistla A.P., Wolfson O., Chamberlain S., Dao S. Querying the uncertain position of moving objects. In: Etzion O, Jajodia S, Sripada S, editors. Temporal databases: research and practice, LNCS, vol. 1399. Berlin: Springer; 1998. p. 310–37.
349. Trajcevski G., Wolfson O., Hinrichs K., Chamberlain S. Managing uncertainty in moving objects databases. ACM Trans Database Syst. 2004;29(3): 463–507.
350. Güting R.H., Böhlen M.H., Erwig M., Jensen C.S., Lorentzos N.A., Schneider M., Vazirgiannis M. A foundation for representing and querying moving objects in databases. ACM Trans Database Syst. 2000;25(1):1–42.
351. Pelekis N, Theodoridis Y. Mobility data management and exploration. New York: Springer; 2014.
352. Renso C, Spaccapietra S, Zimányi E. Mobility data: modeling, management, and understanding. Cambridge, UK: Cambridge University Press; 2013.
353. Chandra A.K., Harel D. Horn clauses and the fixpoint hierarchy Proc. ACM Symp. on the Principles of Database Systems (PODS) (1982), pp. 158-163
354. Porter H.H., Oct. 1985. Optimizations to Earley deduction for DATALOG programs. Available at: http://www.cs.pdx.edu/~harry/earley/datalog.pdf
355. Afrati C.H. Papadimitriou Ch. Papageorgiou G. Roussou A. Sagiv Y, Ullman J.D. 1986. Convergence of sideways query evaluation. In ACM Symposium on Principles of Database Systems, pp. 24–30
356. Bancilhon, R. Ramakrishnan. 1986. An amateur's introduction to recursive query process-

ing strategies. In Proc. of the 1986 ACM SIGMOD International Conference on Management of Data, SIGMOD '86, pp. 16–52.
357. Hafner C.D., Godden K. Portability of syntax and semantics in DATALOG. ACM Trans. on Information Systems, 1985, 3(2):141–164.
358. Zaniolo, C. [1986]. "Safety and compilation of nonrecursive Horn clauses," Proc. First Intl. Conf. on Expert Database Systems, pp. 167-178, Benjamin-Cummings, Menlo Park, CA.
359. Ramakrishnan R., Bancilhon F., Silberschatz A. [1987]. "Safety of recur sive Horn clauses with infinite relations," Proc. 5ixth ACM Symp. on Principles of Database Systems, pp. 328-339.
360. Bancilhon F., Ramakrishnan R. An amateur's introduction to recursive query processing strategies. SIGMOD Record, v. 15, no.2, 1986, pp. 16-52
361. Apt K.R., Blair H., Walker A., Towards a Theory of Declarative Knowledge, in: J. Minker (ed.), Foundations of Deductive Databases and Logic Programming, Morgan Kaufmann, San Mateo, CA, 1988, pp. 89-148.
362. Chandra A.K., Harel D. Horn Clause Queries and Generalizations, J. Logic Programming 2(1):1-15 (Apr. 1985).
363. Gelfond M., Lifschitz V. The Stable Model Semantics for Logic Programming, in: Proceedings of the Fifth International Conference and Symposium on Logic Programming, 1988.
364. Przymusinska H., Przymusinski T.C. Weakly Perfect Model Semantics for Logic Programs, in: Proceedings of the Fifth International Conference/Symposium on Logic Programming, 1988.
365. Przymusinski, T.C. On the Declarative Semantics of Stratified Deductive Databases in: J. Minker (ed.), Foundations of Deductive Databases and Logic Programming, 1988, pp. 193-216.
366. Przymusinski T.C. Extended Stable-Semantics for Normal and Disjunctive Programs, in: Seventh International Conference on Logic Programming, 1990, pp. 459-477.
367. Ross K. Modular Stratification and Magic Sets for DATALOG Programs with Negation, in: Proceedings of the ACM Symposium on Principles of Database Systerns, 1990, pp. 161-171.
368. Van Gelder A., Ross K., Schlipf J.S. The Well-Founded Semantics for General Logic Programs, Journal of he ACM 38(3):620- 650 (1991)
369. Naqvi S. A Logic for Negation in Database Systems, in: J. Minker (ed.), Proceedings of the Workshop on Foundations of Deductive Databases and Logic Programming, 1986, pp. 378-387.
370. Van Gelder A. Negation as Failure Using Tight Derivations for General Logic Programs, Journal of Logic Programming 6(1):109-133 (1989).
371. Balbin I., Port G.S., Ramamohanarao K., Meenakshi K. Efficient Bottom-Up Computation of Queries of Stratified Databases, Journal of Logic Programming 11:295-345 (1991).
372. Bayer R. Query Evaluation and Recursion in Deductive Database Systems, Technical Report 18503, Teehnisehe Universitaet Muenchen, Feb. 1985.
373. Beeri C., Naqvi S., Ramakrishnan R., Shmueli O., Tsur S. Sets and Negation in a Logic Database Language, in: Proceedings of the ACM Symposium on Principles
374. Kerisit J.M., Pugin J.M. Efficient Query Answering on Stratified Databases, in: Proceedings of the International Conference on Fifth Generation Computer Systems, Tokyo, Japan, Nov. 1988, pp. 719-725.
375. Przymusinski T. On the Declarative Semantics of Stratified Deductive Databases, in J. Minker (ed.), Foundations of Deductive Databases and Logic Programming, 193-216, Morgan-Kaufmann, Los Altos, 1988.
376. Ross K.A. Modular Stratification and Magic Sets for Datalog Programs with Negation. Proceedings of the ACM Symposium on Principles of Database Systems, 161-171, Nashville, 1990.
377. Warren D.S. Memoing for Logic Programs, Communzcations of the ACM 35 (3): 93-111 (Mar. 1992)
378. Sacca D., Zaniolo C. The Generalized Counting Methods for Recursive Logic Queries, in: Proceedings of the First International Conference on Database Theory, 1986.
379. Naughton J.F., Ramakrishnan R., Sagiv Y., Ullman J.D. Argument Reduction Through Factoring, in: Proceedings of the Fifteenth International Conference on Very Large Databases, Amsterdam, The Netherlands, Aug. 1989, pp. 173-182.

380. Sagiv Y., Optimizing Datalog Programs, in: J. Minker (ed.), Foundations of Deductive Databases and Logic Programming, Los Altos, CA, Morgan Kaufmann, 1988, pp. 659-698.
381. Ramakrishnan R., Beeri C., Krishnamurthy R. Optimizing Existential Datalog Queries, in: Proceedings of the ACM Symposium on Principles of Database Systems, Austin~ TX, Mar. 1988, pp. 89-102.
382. Sippu S., Soisalon-Soinen E. An Optimization Strategy for Recursive Queries in Logic Databases, in: Proceedings of the Fourth International Conference on Data Engineering, Los Angeles, CA, 1988.
383. Bancilhon F., Ramakrishnan R. An amateur's introduction to recursive query processing strategies. SIGMOD Record, Vol. 15, No.2, 1986, pp. 16-52
384. Gallaire H., Minker J. and Nikolas J.M. Logic and databases: a deductive approach. Computing Surveys, 16:1, 1984, pp. 154-185
385. Ramakrishnan R., Ullman J.D. A survey of deductive database systems. The Journal of Logic Programming, 1995, Vol. 23, No 2, pp. 125-149
386. Finkelstein S.J., Mattos N., Mumick I., Pirahesh H. Expressing Recursive Queries in SQL SO/IEC JTC1/SC21 WG3 DBL MCI-X3H2-96-075 Tech. Rep., March, 1996
387. Reznichenko V.A. Recursive SQL (Rus). Software Engineering, 2010, vol. 4, No 4, pp. 48-65.
388. Kifer M., Lausen G. F-Logic: A Higher Order Language for Reasoning about Objects, Inheritance, and Schema. SIGMOD Record, v. 18, no.2,1989, pp. 139- 146.
389. Liu M. Deductive Database Languages: Problems and Solutions. ACM Computing Surveys, v. 31, no. 1, 1999. pp. 27-62
390. Falcone Sampaio P.R., Paton N.W. (1997) Deductive object-oriented database systems: A survey. In: Geppert A., Berndtsson M. (eds) Rules in Database Systems. RIDS 1997. Lecture Notes in Computer Science, vol 1312. Springer, Berlin, Heidelberg. pp 1-19
391. Paton N.W., Díaz O. Active database systems. ACM Computing Surveys. 1999, vol. 31, No 1, pp. 63–103
392. Chakravarthy S, Blaustein B, Buchmann A.P, Carey M, Dayal U, Goldhirsch D, Hsu M, Jauhuri R. Ladin R, Livny M, McCarthy D, McKee R, Rosenthal A. HiPAC: a research project in active, time-constrained database management. Technical report. CCA-88-02. Cambridge, MA: Xerox Advanced Information Technology; 1988
393. Paton N., Diaz O., Williams M., Campin J., Dinn A., Jaime A. Dimensions of active behaviour. In N. Paton and M. Williams Eds., Proc. 1st Int. Workshop on Rules In Database Systems, Springer- Verlag., 1994, pp. 40-57.
394. Stonebraker M., Jhingran A., Goh J., Potamianos S. On rules, procedures, caching and views in database systems. In Proc. ACM SIGMOD 1990, pp. 281-290
395. Dayal U., Buchmann A., McCarthy D. Rules are objects too: A knowledge model for an active object oriented database system. In K. Dittrich Ed., Proc. 2nd Inti Workshop on OODBS, Volume 334, 1988, pp. 129-143. Springer-Verlag. Lecture Notes in Computer Science
396. Widom J., Finkelstein S. Set-Oriented Production Rules in Relational Database Systems. In Proceedings of the ACM SIGMOD International Conference on Management of Data,1990, pp. 259-270.
397. Dayal U, Blaustein B, Buchmann A, Chakravarthy S, Hsu M, Ladin R, McCarty D, Rosenthal A, Sarin S, Carey M.J, Livny M, Jauhari R. The HiPAC project: combining active databases and timing constraints. ACM SIGMOD Rec. 1988;17(1):51–70.
398. Chakravarthy S, Krishnaprasad V, Anwar E, Kim S.K Composite events for active database: semantics, contexts, and detection. In: Proceedings of the 20th International Conference on Very Large Data Bases; 1994. p. 606–617.
399. Chakravarthy S, Mishra D. Snoop: an expressive event specification language for active databases. Data Knowl Eng. 1994;14(1):1–26.
400. Gehani N.H., Jagadish H.V., Schmueli O. Gehani N., Jagadish H.V., Shmueli O. COMPOSE: A system for composite specification and detection. In: Adam N.R., Bhargava B.K. (eds) Advanced Database Systems. 1993, pp. 3-15. Lecture Notes in Computer Science, vol 759. Springer, Berlin.
401. Gatziu S., Dittrich K. Events in an active object-oriented database. In N. Paton and M. Williams Eds., Proc. 1st Int. Workshop on Rules in Database Systems, 1994, pp. 23-39. Springer-Verlag

402. Mellin J., Berndtsson M. Event Detection. In Encyclopedia of Database Systems, Ling Liu, M. Tamer Özsu Editors, pp. 1361-1366
403. Mellin J., Berndtsson M. Event Specification. In Encyclopedia of Database Systems, Ling Liu, M. Tamer Özsu Editors, pp. 1389-1393
404. Gehani N., Jagadish H.V., Smueli O. Event specification in an active object-oriented database. In: Proceedings of the ACM SIGMOD International Conference on Management of Data; 1992. p. 81–90
405. Motakis I., Zaniolo C. Composite Temporal Events in Active Databases: A Formal Semantics. In: Clifford J., Tuzhilin A. (eds) Recent Advances in Temporal Databases. Workshops in Computing. Springer, London, 1995), 332-352
406. Chakravarthy S., Anwar E., Maugis L., Mishra D. Design of Sentinel: an object-oriented DBMS with event-based rules. Information and Software Technology, 1994, 36, 9, 555-568.
407. Diaz O., Jaime, A. EXACT: an EXtensible approach to ACTive object-oriented databases. VLDB Journal 1997,.6, 4, 282-295
408. Branding H, Buchmann A, Kudrass T, Zimmermann J. Rules in an open system: the REACH rule system. In: Proceedings of the 1st International Workshop on Rules in Database Systems, Workshops in Computing; 1994. p. 111–126
409. Agrawal R., Cochrane R., Lindsay B. On maintaining priorities in a production rule language. In G. Lohman, A. Sernadas, and R. Camps Eds., Proc. 17th VLDB, 1991, pp. 479-487. Morgan-Kaufmann.
410. Widom J. The Starburst Rule System: Language Design, Implementation, and Applications. In: IEEE Data Engineering Bul letin, Special Issue on Active Databases, 1992, 15(4): 15-18
411. Stonebraker M., Kemnitz, G. The POSTGRES Next-generation Database Management System. Communications of the ACM 1991, Vol. 34, No.10, pp. 78 92
412. Hanson E.N. The Design and Implementation of the Ariel Active Database Rule System. IEEE Trans. Knowl. Data Eng. 1996, 8(1): 157-172
413. Kotz A., Dittrich K., Mulle J. Supporting semantic rules by a generalized event/trigger mechanism. In Advance in Database Technology, EDDT, Venice6 1988, pp. 76 91.
414. Reddi S., Poulovassilis A., Small C. Extending a Functional DBPL With ECA-Rules. In T. Sellis Ed., Proc. 2nd Int. Wshp. on Rules in Database Systems 1995, pp. 101-115. Springer-Verlag.
415. Kiernan G., de Maindreville C., Simon E. Making Deductive Databases a Practical Technology: a step forward. In II. Garcia-Molina and II. Jagadish Eds., Proc. ACM SIGMOD Conf. 1990., pp. 237-246
416. Zaniolo C. A Unified Semantics for Active and Deductive Databases. In: Paton N.W., Williams M.H. (eds) Rules in Database Systems. Workshops in Computing. Springer, London. 1994, pp 271-287
417. Harrison J., Dietrich. S. Integrating active and deductive rules. In N. Paton and M. Williams Eds., Proc. 1st Int. Workshop on Rules In Database Systems, 1994, pp. 288 305. Springer-Verlag.
418. Widom J. Deductive and Active Databases: Two Paradigms or Ends of a Spectrum? In N. Paton and M. Williams Eds., Proc. 1st Int. Workshop on Rules In Database Systems 1994, pp. 306 315. Springer-Verlag.
419. Bayer P., Jonker W. A framework for supporting triggers in deductive databases. In N. Paton and M. Williams Eds., Proc. 1st Int. Workshop on Rules In Database Systems 1994, pp. 316 330. Springer-Verlag.
420. Kulkarni K., Mattos N., Cochrane R. Active Database Features in SQL3. In: Paton N.W. (eds) Active Rules in Database Systems. 1999, pp. 197-219 Monographs in Computer Science. Springer, New York, NY.
421. Chakravarthy S. Rule management and evaluation: an active DBMS perspective. SIGMOD RECORD 1989, 18, 3, 20-28.
422. Collet C., Coupaye T. and Svensen T. NAOS: Efficient and modular reactive capabilities in an object-oriented database system. In J. Bocca, M. Jarke, and C. Zaniolo Eds., Proc. 20th VLDD Conf, 1994, pp. 132 -143. Morgan-Kaufmann.
423. Ceri S., Fraternali P., Paraboscіii S., Tanca, L. Active Rule Management in Chimera. In J. Widom and S. Ceri Eds., Active Database Systems: Triggers and Rules for Active Database Processing, 1996, pp. 151-175. Morgan Kaufmann.

424. Gehani N. and Jagadish H. ODE as an Active Database: Constraints and Triggers. In R. C. G.M. Loiiman. A. Sernadas Ed., 17th Intl. Conf. on Very Large Data Bases, Barcelona,1991, pp. 327-336. Morgan Kaufmann
425. Atkinson M., Bancilhon F., DeWitt D., Dittrich K., Maier D., Zdonik S. The object-oriented database system manifesto. In: Proceedings of the 1st International Conference on Deductive and Object-Oriented Databases; 1989. p. 223–240.
426. Hull R., Tanaka K., Yoshikawa M. Behavior Analysis of Object-Oriented Databases: Method Structure, Execution Trees, and Reachibility // Lect. Notes Comput. Sci.- 367.- 1989.- 372-388
427. Mozaffari M., Tanaka Y. ODM: An Object-Oriented Data Model // New. Generat. Comp.- 7, N 1.- 1989.- 4-35
428. Beeri C. A Formal Approach to Object-Oriented Databases // Data and Knowledge Eng.- 5.- 1990.- 353-382
429. Zicari R. Incomplete Information in Object-Oriented Databases // ACM SIGMOD Record.- 19, N 3.- 1990.- 5-16
430. Hong Sh., Maryanski F. Using a Meta Model to Represent Object-Oriented Data Models // 6th Int. Conf. Data Eng., Los Angeles, Calif., USA, Febr. 5-9, 1990.- 11-19
431. Cornelio, Shamkant B. Navathe, Keith L. Doty. Extending Object-Oriented Concepts to Support Engineering Applications // 6th Int. Conf. Data Eng., Los Angeles, Calif., USA, Febr. 5-9, 1990.- 220-227
432. Gunter Saake. Descriptive Specification of Database Object Behaviour // Data and Knowledge Eng.- 6, N 1. 1991.- 47-73
433. Lellani S.K., Spiratos N. Towards a Categorical Data Model Supporting Structured Objects and Inheritance // Proc. 1st Int. East/West Database Workshop, Kiev, Oct. 1990, Lect. Notes Comput. Sci.- 540.- 1991
434. Cattell R.G.G., Barry D.K.(eds.). The Object Data Standard: ODMG 3.0. — San Francisco, Calif.: Morgan Kaufmann, 2000.
435. Stonebraker M., Rowe L.A., Lindsay B., Gray, Carey M., Brodie M., Bernstein Ph., Beech D. Third-Generation Database System Manifesto. SIGMOD Record 19(3), September, 1990. pp 31-43
436. Rowe L, Stonebraker M. The Postgres data model. In: Proceedings of the 13th International Conference on Very Large Data Bases; 1987. p. 83–96.
437. Won Kim. UniSQL/X unified relational and object-oriented database system. SIGMOD ‘94: Proceedings of the 1994 ACM SIGMOD international conference on Management of dataMay 1994, p. 481
438. Darwen H., Date C.J. (March 1995). “The third manifesto”. ACM SIGMOD Record. New York, NY, USA: ACM Press. 24 (1): 39–49.

Одержано 27.07.2021

Про автора:

Резніченко Валерий Анатолієвич,
кандидат фізико-математичних наук,
заступник завідувача відділом.
Кількість публікацій в українських виданнях – 61.
Кількість зарубіжнихих публікацій – 4.
Індекс Хірша – 12.
http://orcid.org/0000-0002-4451-8931.

Місце роботи автора:

Інститут програмних систем
НАН України,
03187, м. Київ-187,
проспект Академіка Глушкова, 40.
Тел.: (044) 526 3559.
E-mail: reznich@isofts.kiev.ua

УДК 004.942:004.021:004.942 http://doi.org/10.15407/pp2021.04.062

В.Я. Петрівський, В.Л. Шевченко, О.С. Бичков, І.П. Сініцин

ІНФОРМАЦІЙНА ТЕХНОЛОГІЯ ЗАБЕЗПЕЧЕННЯ ЖИВУЧОСТІ СЕНСОРНИХ МЕРЕЖ

У сучасному технологічному світі сенсори та сенсорні мережі широко використовуються у переважній більшості сфер людської діяльності. Одним із ключових інженерних завдань під час проєктування сенсорних мереж є питання забезпечення живучості мережі. У статті представлено алгоритми забезпечення живучості сенсорних мереж на основі попередньої оцінки даної властивості. Оцінка живучості залежить від топології мережі. Підвищення показника живучості сенсорної мережі досягається шляхом включення до мережі додаткових датчиків. Запропоновано алгоритм знаходження позиції додаткових датчиків, що враховує радіус покриття сенсорів та необхідність наявності перетину зон покриття датчиків для забезпечення обміну інформацією. Представлено ітераційний алгоритм забезпечення живучості мережі за наявності та врахування динамічних датчиків. Результати обчислювальних експериментів, що представлені у роботі, підтверджують ефективність запропонованих підходів.
Ключові слова: сенсори, сенсорні мережі, живучість сенсорних мереж, інформаційна технологія.

Вступ

Зважаючи на велику кількість переваг та можливостей, датчики широко застосовуються в усіх видах діяльності людини. Основними перевагами датчиків є їхній розмір, низьке споживання енергії та зручність використання [1]. Датчики використовуються для збору та вимірювання різних величин (температури, тиску, витрат тощо) [2, 3], задач аналізу [4], сканування [5] та виявлення [6]. Також датчики є частинами та широко використовуються в концепціях Інтернету речей та Всеохоплюючого Інтернету [7]. Відповідно до [8] Інтернет речей - це мережа фізичних об'єктів, доступ до яких здійснюється через Інтернет. Вони містять вбудовану технологію взаємодії з внутрішніми станами або зовнішнім середовищем. Сферами застосування сенсорних мереж є моніторинг навколишнього середовища, домашній інтелект, сфера охорони здоров'я, контроль промислових процесів, сфера сільського господарства, тощо [9]. Основними загрозами для сенсорних мереж є втрата з›єднання, різні типи атак [10], ненадійні та несанкціоновані датчики, бездротові канали зв›язку [11]. Моделювання втрати каналу зв›язку у бездротових сенсорних мережах представлене у дослідженні [12]. У статті [13] автори запропонували модель активної захисту сенсорних мереж, засновану на теорії ігор. Метод динамічної кластеризації для максимізації стабільності бездротових сенсорних мереж представлений у роботі [14]. Також у статті [15] автори розрізняють дальність зондування та дальність зв'язку датчика, але в нашому випадку це єдиний діапазон, який називається «радіус покриття». Проблема оптимального розміщення датчиків описана в статті [16]. Згідно з [17], живучість – це властивість, що характеризує здатність системи ефективно функціонувати за наявності ушкоджень або відновлювати цю здатність за визначений проміжок часу. Фахівці виділяють функціональну та структурну живучість [18]. Слід також врахувати, що датчики можуть бути статичними та рухомими. У дослідженні живучість визначається як наявність зв›язку між усіма елементами мережі у будь-який момент часу, що забезпечується наявністю перетину зон покриття датчиків на величину, що є більшою або рівною мінімально допустимою величиною зон покриття. У випадку динамічних датчиків вжито поняття «часу тиші», що характеризує період, коли зв›язок між елементами мережі відсутній.

Основна частина

Розглянемо певну територію, позначимо її A, та сенсорну мережу, що покриває дану територію. В свою чергу, мережа складається з n датчиків, кожен з яких має наступні характеристики:

$$s_i = s_i(x_i, y_i, r_i), \quad (1)$$

де x_i, y_i – координати датчику, r_i – радіус покриття.

У нашому випадку під живучістю мережі будемо розуміти наявність зв›язку між усіма сенсорами. Для наявності зв›язку та передачі даних між датчиками необхідною умовою є перетин зон покриття сенсорів. Позначимо величину перетину зон покриття як *c*. Схематично перетин зон покриття датчиків з величиною перетину *c* можна зобразити наступним чином:

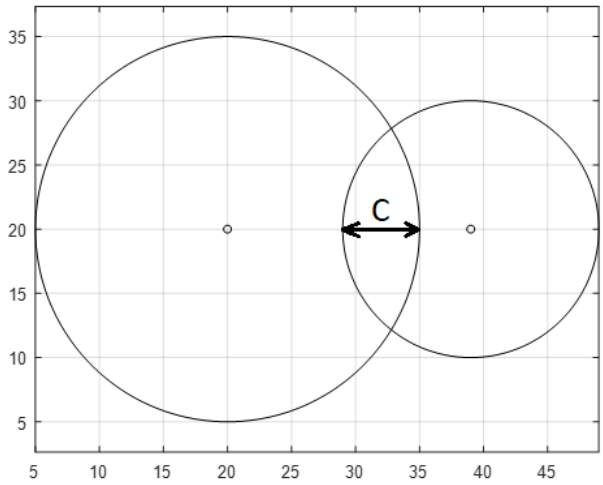


Рис. 1. Зміст величини перетину зон покриття датчиків *c*

Для знаходження датчиків, між якими відсутній зв›язок, пронумеруємо сенсори в довільному порядку. Наступним етапом є побудова матриці зв'язків *H*, елементи якої будуть рівними:

$$h_{i,j} = \begin{cases} 1, \text{зв}'\text{язок присутній} \\ 0, \text{зв}'\text{язок відсутній} \end{cases}. \qquad (2)$$

Використовуючи матрицю зв'язків *H* та алгоритми пошуку вшир чи вглиб [19], знайдемо компоненти зв'язності неорієнтованого графа, що може бути побудований із використанням матриці *H*. У разі наявності понад однієї компоненти зв'язності, мережа не відповідає умові живучості. Також у даному випадку мережу можна умовно поділити на множини сенсорів, зв′язаних між собою. Позначимо дані отримані множини як S_1, S_2, ... , S_k, де k – кількість компонент зв'язності мережі. Для пари множин S_1 та S_2 знайдемо датчики $s_i \in S_1$ та $s_j \in S_2$ відстань між якими є мінімальною. У роботі [20] представлено підхід забезпечення живучості мережі у разі втрати сенсорів. Даний підхід може бути застосований для забезпечення зв'язності мережі сенсорів. Знайдемо позицію «з'єднуючого» сенсору з максимальним радіусом покриття r_a та рівнем перетину зон покриття *c* шляхом розв'язання наступної системи рівнянь:

$$\begin{cases} \sqrt{(x-x_i)^2+(y-y_i)^2} = r_i + r_a - c \\ \sqrt{(x-x_j)^2+(y-y_j)^2} = r_j + r_a - c \end{cases}, \qquad (3)$$

де x_i, y_i, r_i – координати та радіус покриття сенсору s_i, x_j, y_j, r_j – координати та радіус покриття сенсору s_j, r_a – радіус покриття додаткового сенсору.

Розв'язком рівняння (3) будуть дві пари координат. Серед отриманих розв'язків необхідно обрати таку пару координат, коли зона покриття сенсору з даними координатами буде менше перетинатися із зонами покриття інших датчиків та менше виходити за межі загальної території *A*. Коли обидва результати задовольняють даний критерій, обираємо координати довільно.

Розглянемо випадок відсутності розв›язку рівняння (3). Дана ситуація буде означати неможливість знаходження позицій додаткового сенсору із заданими параметрами. У такому випадку знайдемо мінімально допустимий радіус покриття та координати додаткового сенсору. Очевидно, що даний сенсор буде мати мінімально допустимий радіус при координатах, які лежать на умовній прямій між датчиками s_i та s_j. Координати (x_a, y_a) та радіус додаткового датчику r_a будуть розв'язком наступного рівняння:

$$\begin{cases} \sqrt{(x_a-x_i)^2+(y_a-y_i)^2} = r_i + r_a - c \\ \sqrt{(x_a-x_j)^2+(y_a-y_j)^2} = r_j + r_a - c \\ \sqrt{(x_i-x_j)^2+(y_i-y_j)^2} = \sqrt{(x_a-x_j)^2+(y_a-y_j)^2} + \sqrt{(x_a-x_j)^2+(y_a-y_j)^2} \end{cases}, \qquad (4)$$

де x_i, y_i, r_i – координати та радіус покриття сенсору s_i, x_j, y_j, r_j – координати та радіус покриття сенсору s_j.

Отже, для забезпечення живучості мережі необхідно додати сенсор з радіусом покриття r_a з координатами (x_a, y_a). У разі відсутності датчика з відповідним радіусом покриття r_a, необхідно використати декілька існуючих датчиків, що будуть розташовані вздовж умовної прямої між датчиками s_i та s_j та мати однакові радіуси покриття, котрі будуть задовольняти умову:

$$\begin{cases} r_a \; mod \; (r - c) = 0 \\ r \le r^{max} \end{cases}, \quad (5)$$

де r^{max} – максимальний радіус покриття доступного сенсору.

Розглянемо випадок рухомих сенсорів. Нехай маємо сенсорну мережу, що складається з стаціонарних та рухомих сенсорів. Дана мережа може бути описана наступним чином:

$$S = \{s_1, s_2, \dots, s_n, s_1^m, s_2^m, \dots, s_k^m\}, \quad (6)$$

де s_i – стаціонарні сенсори (1), $i = \overline{1,n}$, n – кількість стаціонарних сенсорів, s_j^m– динамічні сенсори, $j = \overline{1,k}$, k – кількість динамічних сенсорів.

У свою чергу, динамічні сенсори представимо так:

$$s_j^m = s_j^m(f_j(x_j^m, y_j^m, t), r_j^m), \quad (7)$$

де $f_j(x_j^m, y_j^m, t)$ – рівняння руху датчику, x_j^m, y_j^m – координати датчику, r_j^m – радіус покриття, $j = \overline{1,k}$, k – кількість динамічних сенсорів.

Забезпечення живучості даної системи полягає у мінімізації «часу тиші» через забезпечення зв'язності мережі. Під «часом тиші» будемо розуміти період часу, протягом якого відсутній зв›язок усіх датчиків між собою. Забезпечення живучості досягатимемо шляхом додавання до мережі додаткових сенсорів.

Покроково ітераційна процедура забезпечення живучості сенсорної мережі у випадку динамічних датчиків може бути описана таким чином:

1. Серед сенсорів мережі S обираємо датчик, найближчий до динамічного датчика, позначимо його $s_{nearest}(x_{nearest}, y_{nearest}, r_{nearest})$ та формуємо систему рівнянь (8);

2. Шукаємо розв›язок системи A;

3. У разі існування розв›язку додаємо до системи рівнянь Q рівняння, що містить координати рухомого датчика в момент часу t_i^*, переходимо до наступного моменту часу $t_{i+1}^* = t_i^* + \Delta t$, виконуємо перевірку умови зупинки алгоритму та перехід до кроку 2, у разі негативного результату перевірки умови зупинки. Загалом рівняння Q буде мати такий вигляд (9;)

4. У разі відсутності розв›язку рівняння Q додаємо сенсор із характеристиками, які є розв'язками рівняння Q у момент часу t_i^*, $s(x_{solution}(t_i^*), r_{solution}(t_i^*), r_{additional})$ до статичних датчиків мережі S, переходимо до наступного моменту часу $t_{i+1}^* = t_i^* + \Delta t$, виконуємо перевірку умови зупинки алгоритму та перехід до кроку 1 у разі негативного результату перевірки умови зупинки.

Умовою зупинки алгоритму буде вихід динамічного сенсору за межі території або досягнення сенсором початкової точки у разі циклічного руху датчику.

$$Q = \begin{cases} \sqrt{(x - x_{nearest})^2 + (y - y_{nearest})^2} = r_{neatest} + r_{additional} - c \\ \sqrt{(x - x(t_i^*))^2 + (y - y(t_i^*))^2} = r(t_i^*) + r_{additional} - c \end{cases}. \quad (8)$$

$$Q = \begin{cases} \sqrt{(x - x_{nearest})^2 + (y - y_{nearest})^2} = r_{neatest} + r_{additional} - c \\ \sqrt{(x - x(t_{i+1}^*))^2 + (y - y(t_{i+1}^*))^2} = r(t_{i+1}^*) + r_{additional} - c \\ \sqrt{(x - x(t_i^*))^2 + (y - y(t_i^*))^2} = r(t_i^*) + r_{additional} - c \\ \sqrt{(x - x(t_{i-1}^*))^2 + (y - y(t_{i-1}^*))^2} = r(t_{i-1}^*) + r_{additional} - c \\ \dots \\ \sqrt{(x - x(t_0^*))^2 + (y - y(t_0^*))^2} = r(t_0^*) + r_{additional} - c \end{cases}. \quad (9)$$

Схематично описаний вище алгоритм можна зобразити у такий спосіб (Рис. 2):

Рис. 2. Алгоритм пошуку позиції додаткових сенсорів

Результати обчислювальних експериментів

Розглянемо сенсорну мережу, координати та радіуси покриття сенсорів якої представлені у таблиці (Табл. 1).

Таблиця 1

Вхідні дані першого обчислювального експерименту

	(x, y)	r
s_1	(20, 70)	19
s_2	(35, 55)	7
s_3	(45, 85)	13
s_4	(80, 80)	15
s_5	(30, 16)	10
s_6	(47, 10)	10
s_7	(67, 17)	13

Схематично дана мережа може бути представлена так:

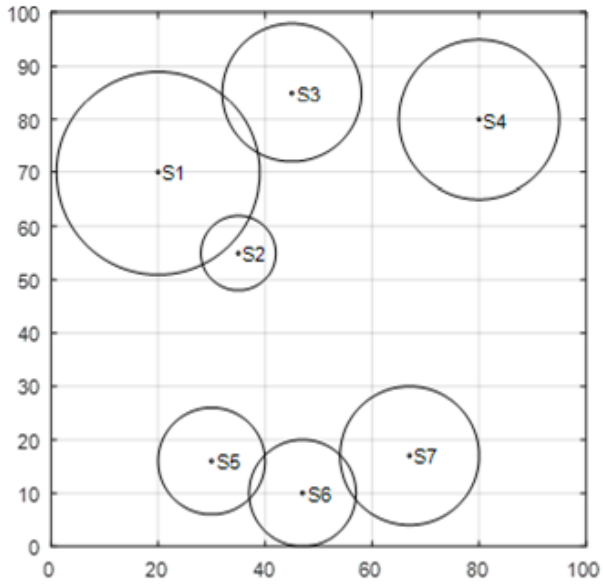


Рис. 3. Сенсорна мережа у початковому стані

Представлена мережа має 3 компоненти зв'язності та не задовольняє умову живучості. Використовуючи запропонований підхід, забезпечимо живучість мережі шляхом додаткових сенсорів із максимальним радіусом покриття $r^{max} = 10$. Після першої ітерації запропонованого підходу (3) кількість компонент зв'язності буде рівною 2, а отримана мережа з додатковим сенсором буде мати вигляд:

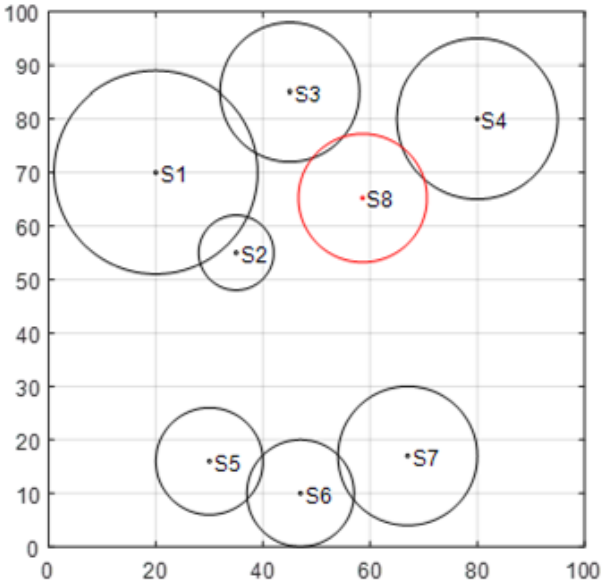
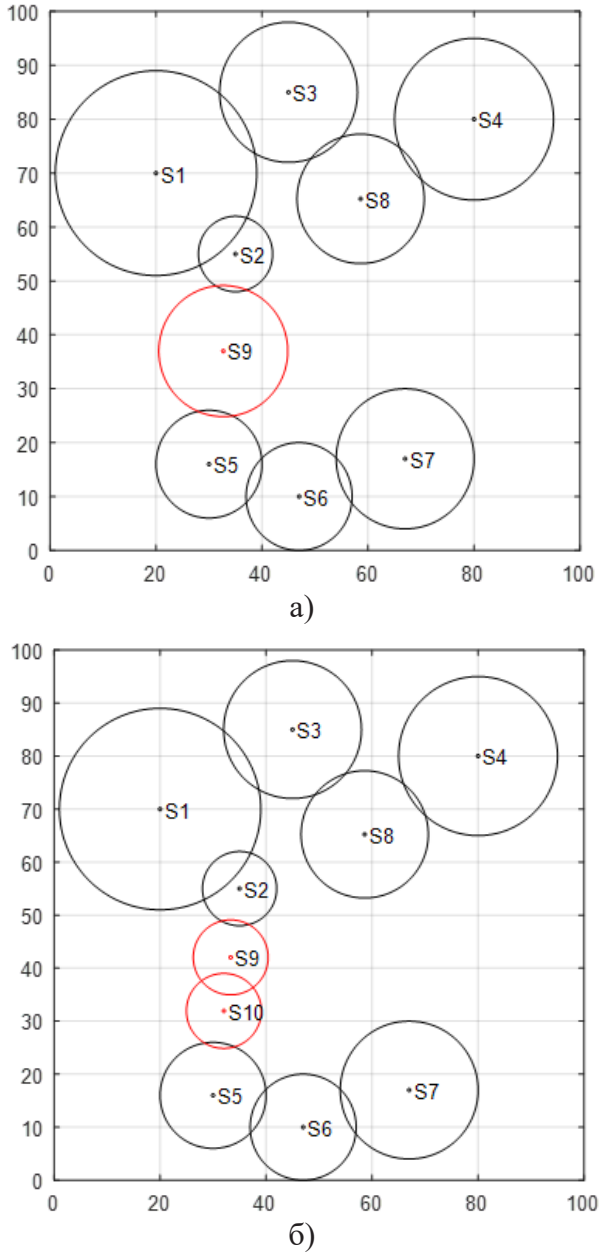


Рис. 4. Сенсорна мережа з додатковим сенсором (s_8)

Зв'язок між сенсорами s_2 та s_5 неможливо забезпечити сенсором із радіусом покриття $r^{max} = 10$. Використавши (4) додат-

ковий сенсор s_9 має бути у точці з координатами (32.69, 36.98) і мати мінімальний радіус покриття 12.15 (Рис. 5а). Із врахуванням наявності сенсорів із максимальним радіусом покриття $r^{max} = 10$ та формули (5), живучість мережі можна забезпечити шляхом включення до мережі двох сенсорів s_9, s_{10} з радіусами покриття $r = 7{,}07$ та координатами (33.33, 42.03) і (32.04, 31.93) відповідно (Рис. 5б).



а) б)

Рис. 5. Забезпечення живучості з використанням а) сенсору з мінімально допустимим радіусом; б) декількох сенсорів з радіусом не більше $r^{max} = 10$

Розглянемо випадок наявності динамічного сенсору. Характеристики датчиків представлені у наступній таблиці (Табл. 2):

Таблиця 2

Вхідні дані другого обчислювального експерименту

	(x, y)	*r*
s_1	(15, 17)	15
s_2	(17, 65)	15
s_3	(20, 41)	15
s_4	(47, 50)	15
s_5	(40, 20)	15
$s_{moveable}$	(95, 35)	10

Схематично описана вище мережа, динамічний датчик у початковій точці та траєкторія руху датчика можуть бути представлені наступним чином (Рис. 6):

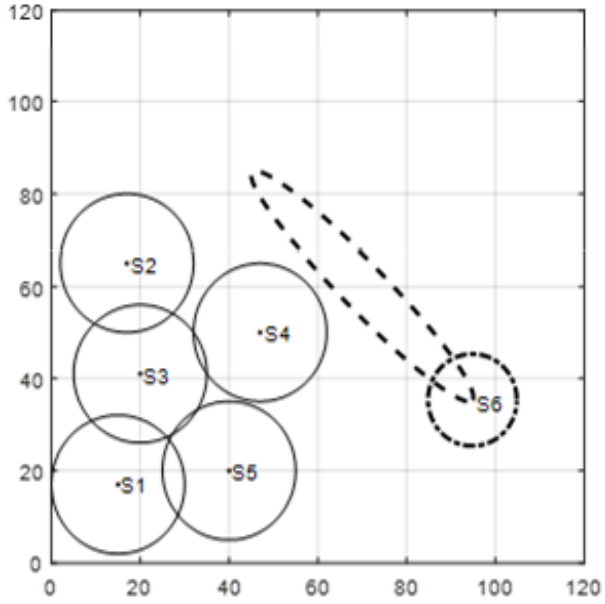
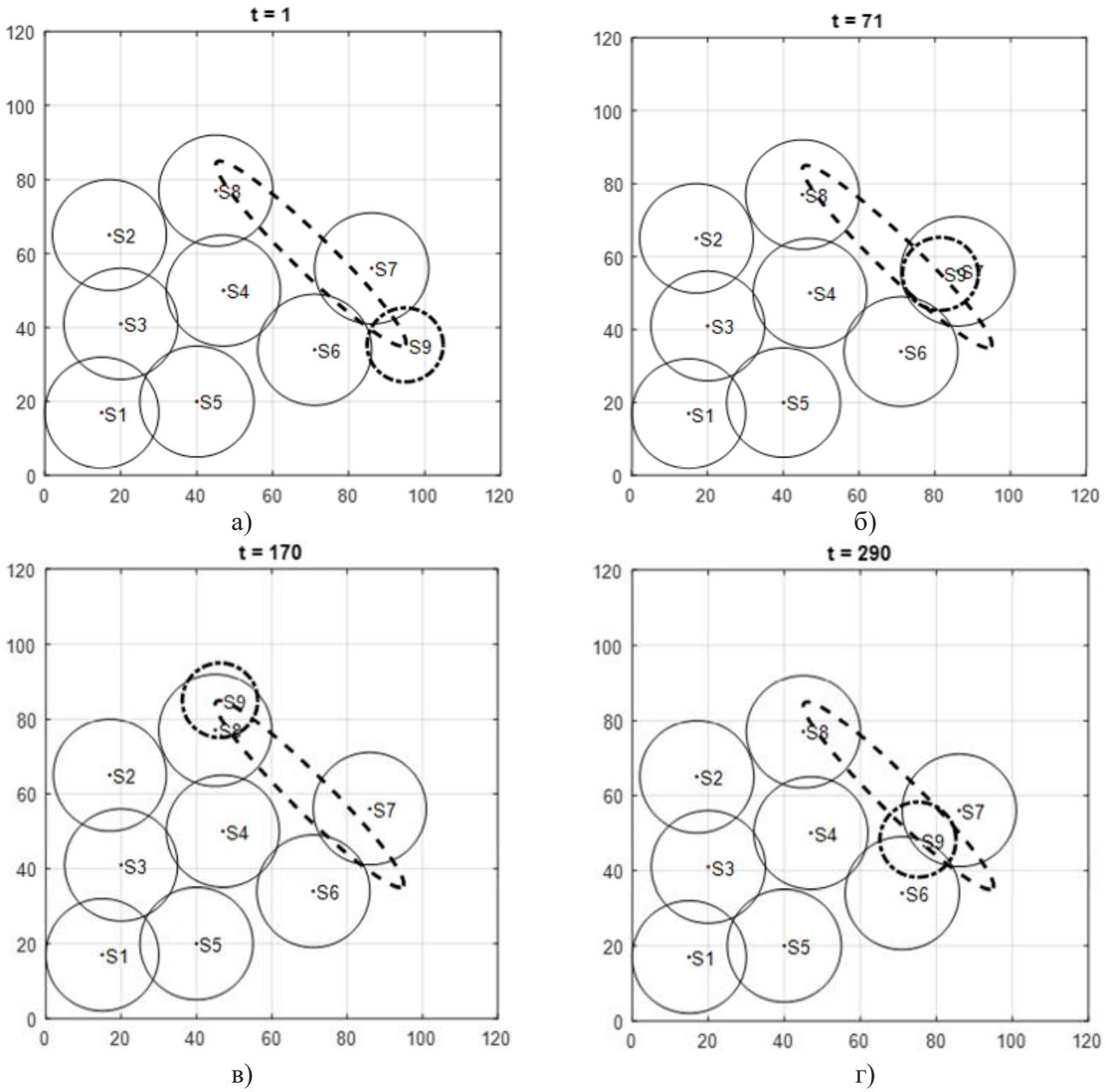


Рис. 6. Сенсорна мережа, динамічний датчик у початковій точці й траєкторія руху датчика

При значенні $\Delta t = 1c$ величина «часу тиші» буде рівною 298с. Використовуючи описаний вище алгоритм, забезпечимо живучість мережі шляхом включення до неї додаткових датчиків. Результат роботи алгоритму схематично може бути представлений так (Рис. 7):

Результат роботи запропонованого ітераційного алгоритму, що полягає у зменшенні часу тиші, можна представити у такому вигляді (Рис. 8):



а) б) в) г)

Рис. 7. Сенсорна мережа, динамічний датчик у різні моменти часу

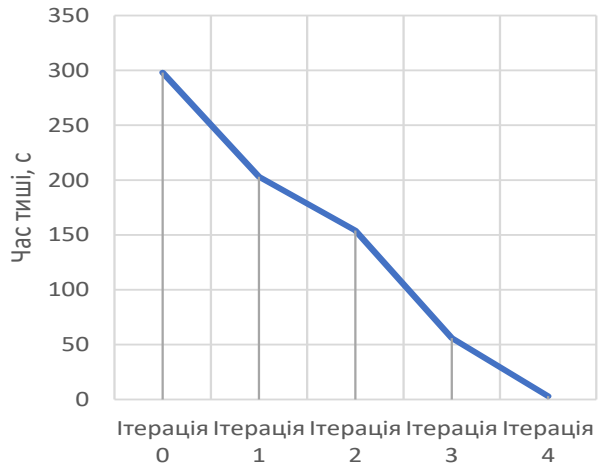


Рис. 8. Значення величини часу тиші на різних ітераціях роботи алгоритму

Висновки

У статті представлено інформаційну технологію забезпечення живучості сенсорних мереж. Під поняттям живучості сенсорної мережі розуміється наявність зв'язку між усіма сенсорами в мережі. Зв'язок між сенсорами забезпечується завдяки перетину зон покриття на мінімально допустиму величину перетину, яка є сталою величиною. Розглядаються випадки статичних та динамічних датчиків. У випадку динамічних сенсорів запропоновано ітераційний алгоритм забезпечення живучості мережі шляхом додавання до мережі додаткових статичних сенсорів із

визначеним радіусом та рівнем перетину зон покриття. Представлені результати обчислювальних експериментів підтверджують ефективність запропонованих підходів.

Література

1. French P.J. Smart Sensors: Advantages and Pitfalls. Technologies for Micro/Nano-Devices, Sensors and Actuators. 2018.
2. Rahman A. Assignment on Tempera ture Sensors. 2018. DOI: 10.13140/RG.2.2.16747.23844.
3. Sparks D. MEMS pressure and flow sensors for automotive engine management and aerospace applications. MEMS for Automotive and Aerospace Applications. 2013. P. 78–105.
4. Panayotova G., Dimitrov G. and Dimitrov D. Wireless Sensors for analysis transport systems. 10th International Conference on Computer Science and Information Technology. 2017.
5. Lercari N. Terrestrial Laser Scanning in the Age of Sensing. Digital Methods and Remote Sensing in Archaeology. 2016. P. 3–33.
6. Teixeira T., Gershon D. A Survey of Human-Sensing: Methods for Detecting Presence, Count, Location, Track, and Identity. ACM Computing Surveys. 2010. №5. P. 59.
7. Abdul-Qawy A., Magesh E. and Tadisetty S. The Internet of Things (IoT): An Overview. Int. Journal of Engineering Research and Applications. 2015. P. 71–82.
8. Evans D. (2011). The Internet of Things How the Next Evolution of the Internet Is Changing Everything. Cisco Internet Business Solutions Group. Available from: https://www.cisco.com/c/dam/en_us/about/ac79/docs/innov/IoT_IBSG_0411FINAL.pdf.
9. Prasanna S., Rao S. An Overview of Wireless Sensor Networks Applications and Security. International Journal of Soft Computing and Engineering. 2012. P. 538–540.
10. Panayotova G., Dimitrov G. Modeling and dataprocessing of information systems. Artificial Intelligence and Pattern Recognition (AIPR). 2016.
11. Zhu L., Zhang Z. and Xu C. Secure Data Aggregation in Wireless Sensor Networks. Secure and Privacy-Preserving Data Communication in Internet of Things. 2017.
12. Kurt S., Tavli B. Path Loss Modeling for Wireless Sensor Networks: Review of Models and Comparative Evaluations. IEEE Antennas and Propagation Magazin. 2016.
13. Alim Al Islam A., Hyder C., Kabir H. at all. (2014). Stable Sensor Network (SSN): A dynamic clustering technique for maximizing stability in wireless sensor networks. Wireless Sensor Network. №7. P. 538–554.
14. Farsi M., Elhosseini M. and Badawy M. Deployment Techniques in Wireless Sensor Networks, Coverage and Connectivity: A Survey. 2019.
15. Maksimovic M., Milosevic V. Evaluating the optimal sensor placement for smoke detection. Yugoslav J. Oper. Res. 2016. №26. P. 33–50.
16. Gupta K., Kulia P. and Jana P. Genetic algorithm for k-connected relay node placement in wireless sensor networks. Proc. 2nd Int. Conf. Comput. Commun. Technol. 2016. P. 721–729.
17. Додонов, О. Г., 2011. Живучість складних систем: аналіз та моделювання: навч.посіб. у 2-х ч. / О. Г. Додонов, М. Г. Кузнецова, О. С. Горбачик. – Київ: НТУУ «КПІ», 264 с.
18. Додонов А. Г., 2011. Живучесть информационных систем / А. Г. Додонов, Д. В. Ландэ. – Киев: Наук. думка, 2011. – 256 с.
19. Cormen T., Leiserson C., Rivest R. at all. (2009). Introduction to Algorithms.
20. Petrivskyi V., Dimitrov G., Shevchenko V. at all. (2020). Information Technology for Big Data Sensor Networks Stability Estimation. Information & Security: An International Journal 47. №. 1. P. 141–154.

Одержано: 18.11.2021

Про авторів:

Петрівський Володимир Ярославович, аспірант.
Кількість публікацій в українських виданнях – 22.
Кількість зарубіжнихих публікацій – 6.
Індекс Хірша – 1.
https://orcid.org/0000-0001-9298-8244.

Шевченко Віктор Леонідович,
доктор технічних наук, професор.
Кількість публікацій в українських виданнях – понад 300.
Кількість зарубіжних публікацій – 17.
Індекс Хірша – 3.
https://orcid.org/0000-0002-9457-7454.

Бичков Олексій Сергійович,
доктор технічних наук, професор.
Кількість публікацій в українських виданнях – понад 300.
Кількість зарубіжних публікацій – 37.
Індекс Хірша – 5.
https://orcid.org/0000-0002-9378-9535.

Сініцин Ігор Петрович
доктор технічних наук, старший науковий співробітник.
Кількість публікацій в українських виданнях – 80.
Індекс Хірша – 1.
http://orcid.org/0000-0002-4120-0784

***Місце роботи авторів*:**

Київський національний університет імені Тараса Шевченка,
Факультет інформаційних технологій,
Кафедра програмних систем і технологій,
вул. Богдана Гаврилишина, 24,
Київ, Україна , 02000.
E-mail: vovapetrivskyi@gmail.com,
gii2014@ukr.net,
bos.knu@gmail.com.

Інститут програмних систем НАН України
пр. Академіка Глушкова, 40, корпус 5,
м. Київ, Україна, 03187.
E-mail: ipsinitsyn@gmail.com

УДК 004.942 http://doi.org/10.15407/pp2021.04.070

В.В. Шевченко, Д.С. Берестов, І.П. Сініцин, В.Я. Петрівський

ОСОБЛИВОСТІ КОМП'ЮТЕРНОГО МОДЕЛЮВАННЯ РОЗПОВСЮДЖЕННЯ ДУМОК У СОЦІУМІ НА ПРИКЛАДІ МОДЕЛІ СТУДЕНТСЬКОЇ СПІЛЬНОТИ

У статті розглядаються особливості комп'ютерного моделювання, засновані на використанні теорії клітинних автоматів як основного інструменту для моделювання процесів формування суспільної думки. За об'єкт моделювання було обрано соціальну поведінку в контексті прийняття навчального матеріалу в рамках освітнього процесу. Як інструмент моделювання була обрана теорія клітинних автоматів, оскільки вона є відносно простим і водночас ефективним методом моделювання взаємодії однотипних об'єктів. Розроблена базова клітинна автоматна модель має розширені правила визначення стану клітини і визначення близькості клітини. Також була виведена залежність коефіцієнта сприйняття інформації від різниці між станом клітини і її оточенням. Ці удосконалення дозволяють використовувати модель для прогнозування змін в уподобаннях соціальних груп.
Ключові слова: клітинні автомати, навчальний процес, моделювання.

Вступ

В умовах світу, що постійно змінюється, для кожної людини важливим є забезпечення своєї затребуваності, як фахівця у сфері її діяльності. Для забезпечення цього необхідно мати навички швидкого навчання та об'єктивну картину світу. Вища освіта дозволяє це отримати. Якісне навчання в університеті складається з багатьох аспектів: вивчення освітньої програми, суспільної активності та наукової діяльності. В цій роботі під наукоцентричним підходом до освіти будемо вважати підхід, за якого наука виступає рушійною силою, що спонукає студента приділяти більше уваги вивченню предметів програми та спонукає його виходити за межі програми для поглиблення знань в обраній спеціальності.. Правильно прищеплений студентові науковий підхід дозволяє в подальшому розробляти нестандартні інноваційні рішення. Але формування саме наукового підходу в студента є складним процесом, оскільки є багато чинників, що заважають його формувати. Примус до науки - марна річ. Тому до заняття наукою студентів залучають через переконання, що це корисно для їхнього фахового зростання. Тобто певною мірою має місце агітація.

Тому **актуальним** є забезпечення ефективного формування у студента розуміння необхідності участі в науковій діяльності через вплив з боку викладачів. Відповідно до цього також **актуальною** є розробка моделі навчального процесу при наукоцентричному підході для прогнозування наслідків керівничих рішень з приводу агітаційної роботи зі студентами.

За допомогою моделей на основі клітинних автоматів порівняно легко моделювати процеси, в яких задіяні однотипні об'єкти реального світу. Тому було запропоновано використовувати теорію клітинних автоматів для моделювання розповсюдження інформації в соціумі.

Мета роботи. Розробити методи прогнозування зміни суспільної думки за рахунок розробки моделі соціальної поведінки щодо акцепції навчального матеріалу при наукоцентричному підході в освіті.

Аналіз існуючих розробок та постановка задачі

Дослідження теорії клітинних автоматів (КлА) ведуться з кінця 40-х років XX століття, коли ідея КлА була «на поверхні». Тому створення цієї теорії приписується чотирьом науковцям, що працювали над її ідеями незалежно та одночасно. Джон фон Нейман працював над теорією самовідтворюваних машин, а його колега з лабораторії Станіслав Улам розробляв математичну модель росту кристалів. Обмін ідеями привів до створення клітино-автоматної моделі еволюції систем. Майже одночасно Норберт Вінер та Артуро Розерблют розроби-

ли клітинно-автоматну модель збудливого середовища для опису розповсюдження імпульсів у нейронних мережах. У подальшому теорія КлА набула популярності в 70-х роках XX століття, коли нею зацікавилися звичайні любителі та популяризатори науки. Зокрема, важливий внесок зробив Джон Конвей, коли розробив клітинний автомат «Гра Життя». Конвей зацікавився проблемою, запропонованою Нейманом моделі гіпотетичної самовідтворюваної машини зі складними правилами поведінки. Конвей спростив ідеї Неймана і створив правила «Гра Життя». Даний автомат та його модифікації вплинули на деякі розділи математики, інформатики та фізики. Про це свідчать багато різних комп'ютерних реалізацій.

Особливістю теорії клітинних автоматів є те, що за її допомогою може бути змодельована та описана велика кількість явищ [1-5]. Наприклад, для моделювання розповсюдження лісових пожеж [6], автомобільного трафіку однією смугою руху [7], опису процесів квантового світу [8]. Але існує досить мало досліджень, напряму пов'язаних з використанням клітинних автоматів для поведінки соціуму в контексті навчальних процесів, зокрема, в університеті. Це породжує **протиріччя** між особливостями постановки задач щодо моделювання з урахуванням великої кількості параметрів та вузьким набором керуючих параметрів існуючих моделей на кшталт «Гри Життя» Джона Конвея.

Загальні поняття теорії клітинних автоматів.

Усі клітинні автомати, що будуть розглядатися далі, керуються наступними правилами:

1) Для роботи клітинного автомату задається початкове значення клітин клітинного поля та правила визначення стану клітин на наступній ітерації.

2) Клітинне поле – це прямокутне поле клітин, при цьому є замкненим та геометрично набуває форми Тору. Кількість клітин задається користувачем через кількості рядків та стовпчиків клітин.

3) Клітина може мати дискретний (наприклад, 0 або 1) чи недискретний стан (наприклад, значення у межах від 0 до 1).

4) Усі клітини поля мають попередньо визначену *околицю* – множину клітин «сусідів», від значень яких залежить стан досліджуваної клітини.

5) На кожній ітерації клітинного автомату за допомогою правил переходу та станів клітин поля визначається новий стан кожної з клітин.

Автомат завершує свою роботу, якщо на полі не залишається жодної «живої» клітини, якщо клітинне поле на черговій ітерації співпадає з будь-яким іншим полем попередніх ітераціях або якщо при черговому кроці жодна з клітин не змінює свого стану.

Розглянутий клітинний автомат без подальшого покращення являє собою досить абстрактну модель, для якої життєву ситуацію знайти досить складно, тому актуальним є його вдосконалення за напрямком введення необхідного набору правил розвитку клітинного автомату.

Вдосконалення загальних правил клітинного автомату
Вдосконалення правила можливих станів клітини.

В більшості клітинних автоматів клітини можуть приймати лише чітко зазначені дискретні значення, що не є доречним у ситуації поширення думок у соціумі. В даному випадку доречнішим є використання недискретних значень. Вважаймо, що надалі думка клітини $-1 \le s \le 1$ може мати значення в проміжку $-1 \le s \le 1$, де крайні межі відповідають діаметрально протилежним точкам зору на деякі питання. В такому разі клітинне поле буде представлено як кольорове клітинне поле (Рис. 1), на якому кожна клітина має свій колір. Чим світліше клітина, тим ближче її значення до 1, чим темніше, тим ближче значення до -1.

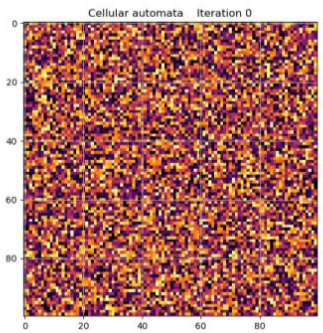


Рис. 1. Клітинне поле з вдосконаленим правилом можливих станів клітини.

Вдосконалення правила визначення стану клітини.

Оскільки об'єкт нашого дослідження (соціум) так чи інакше є замкненою системою і складно (в деяких випадках і неможливо) виділити об'єкти соціуму, що знаходяться біля межі, вважаймо клітинне поле замкненим, що топологічно утворює форму тору.

У цьому розділі ми удосконалимо правила визначення стану клітини. Для обрахунку кількості сусідів використовуймо правило, виведене Муром [7], тобто кожна клітина матиме 8 сусідів у своїй околиці.

Оскільки на ставлення людини певною мірою впливають думки людей, що з ним спілкуються, стан клітини на ітерації можна представити у вигляді функції 9 змінних, де перша зміна відповідає стану піддослідної клітини, а інші 8 змінних є станами сусідніх клітин. Ми пропонуємо такий вигляд функції:

$$s_{t+1} = s_t + \left(\frac{\sum_{i=1}^{n} n_i}{n} - s_t \right) * k \qquad (1)$$

де n – кількість сусідніх клітин,

n_i – стан i-ї клітини з оточення клітини s_t,

s_t – стан клітини на ітерації t,

s_{t+1} – стан клітини на ітерації $t+1$,

k – коефіцієнт сприйнятливості чужої думки.

Коефіцієнт сприйнятливості k може задаватися попередньо користувачем або залежати від попередньо заданих параметрів. У даному випадку було вирішено встановити значення на рівні. Реалізація КлА з впровадженим правилом визначення стану клітини зображено на Рис. 2. Загалом КлА прямує до стану рівноваги (прихід соціуму до спільної думки). Це пов'язано з відсутністю зовнішніх чинників, що могли би кардинально вплинути на процес формування думки.

Введення базового правила визначення околиці клітини.

У клітинному автоматі Джона Конвея «Гра Життя» стан клітини залежить від станів клітин в околиці, а саме від 8 сусідніх клітин. Це унеможливлює моделювання систем, де об'єкти можуть контактувати з різною кількістю об'єктів. Тому було запропоновано ввести базове правило визначення околиці клітини, що дозволяє при моделюванні встановлювати необхідну околицю клітини (радіус взаємодії). Наприклад, радіус $R = 2$ дозволяє мати 24 сусідні клітини. Кількості клітин в околиці при довільному радіусі визначається за формулою:

$$n = (2R+1)^2 - 1,$$

де R – радіус внутрішньопопуляційної взаємодії.

Перевіримо працездатність моделі зі зміненим радіусом взаємодії на прикладі КлА з радіусом взаємодії $R = 2$ (Рис. 3). Зміна радіусу дала очікуваний результат: у широкому колі спілкування соціум прийшов до практично спільної думки за проміжок удвічі менший, аніж з $R = 1$.

Ускладнене правило визначення околиці клітини.

Базове правило визначення околиці клітини може бути ускладнене шляхом введення «мапи місцевості», що є набором значень радіусів взаємодії для кожної окремої клітини поля автомату.

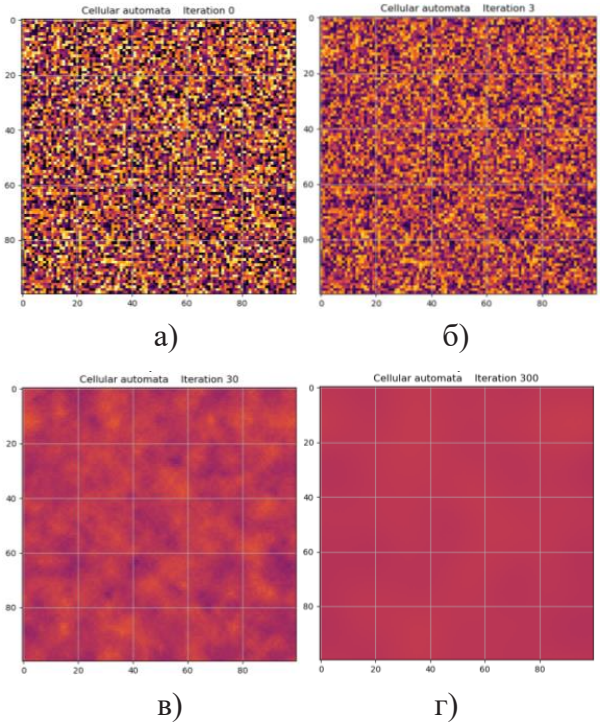
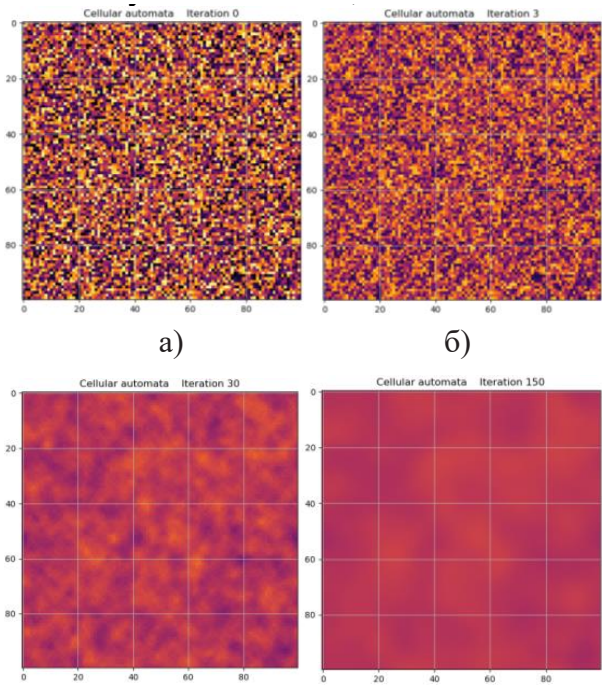


Рис. 2. Клітинний автомат з вдосконаленим правилом визначення стану клітини.
а) початкове поле, б) ітерація 3,
в) ітерація 30, г) ітерація 300.



а) б)

в) г)

Рис. 3. Клітинний автомат з радіусом $R = 2$.
а) початкове поле, б) ітерація 3,
в) ітерація 30, г) ітерація 150.

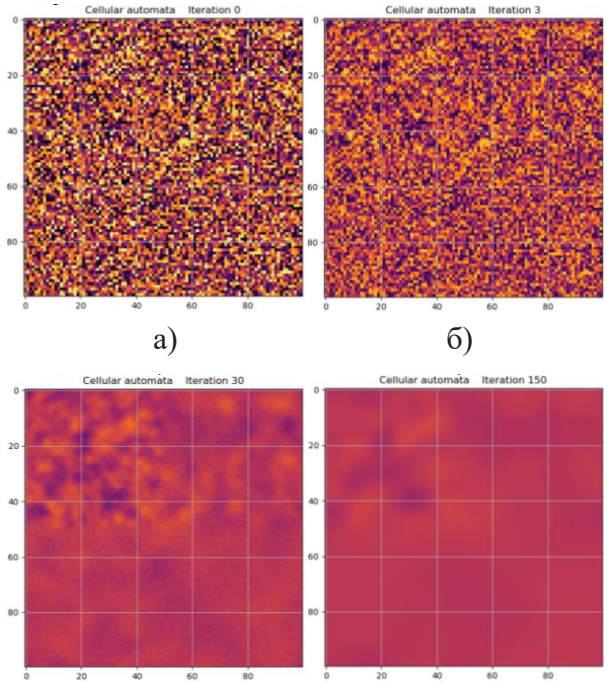
Значення для кожної клітини можуть бути задані довільно або користувачем. Це дозволить змоделювати випадки із соціумом, в якому різні об'єкти взаємодіють із різною кількістю об'єктів.

Для перевірки дієвості ускладненого правила визначення околиці клітини клітинне поле було поділено на чотири сектори, в кожному з яких клітини мали спільний однаковий радіус взаємодії відповідно. Поділ поля зображено на Рис. 4. Приклад роботи клітинного автомату з ускладненим правилом визначення околиці клітини зображено на Рис. 5.

$R = 1$	$R = 2$
$R = 3$	$R = 4$

Рис. 4. Поділ поля на сектори.

На ітерації 30 автомату (Рис. 5. б) можна побачити чіткі межі встановлених секторів (Рис. 4). Залежно від радіусу взаємодії клітини секторів будуть по різному змінювати свої значення, одночасно сектори з $R = 3$ та $R = 4$ прямують до стану рівноваги значно швидше (прихід соціуму до спільної думки), ніж сектори з радіусом $R = 2$ та $R = 1$. Із цього можна зробити проміжний висновок, що широта кола спілкування сильно впливає на формування думки в соціумі.



а) б)

в) г)

Рис. 5. Клітинний автомат з ускладненим правилом визначення околиці клітини.
а) початкове клітинне поле, б) ітерація 3,
в) ітерація 30, г) ітерація 150.

Вдосконалення клітинного автомату за напрямком моделювання поведінки соціуму

Правило залежності коефіцієнту сприйнятливості від різниці станів клітини та її оточення.

Людське життя великою мірою складається з обрання певних варіантів розвитку подій: піти, чи не піти кудись; купити, чи не купити щось; брати, чи не брати участь у якомусь процесі. Коли ж вибір не ставить під загрозу життя, людина схильна обирати те, що і більшість її оточення. Якщо думка, якої дотримується людина, не співпадає з чужою думкою, то вона неохоче змінює власну під впливом чужої. Водночас, коли думка подібна до думки оточення, вона підсилює існуючу думку. Отож, вимальовується чітка залежність сприйнятливості чужої думки від різниці з власною, що для збільшення точності моделі робить доцільним врахування цієї залежності.

В минулих розділах було встановлено межі для можливих значень думки об'єкту: $-1 \le s \le 1$, де крайні значення відповідають протилежним баченням чи виборам в ситуації, а нуль відповідно нейтральна позиція. Також при визначенні стану кожної клітини на наступній ітерації КлА використовується коефіцієнт сприйнятливості k (формула (1)). Значення коефіцієнту може встановлюватися вручну або за якимсь правилом.

Тому було запропоновано ввести правило визначення коефіцієнту сприйнятливості, що залежить від станів клітини та її оточення. Значення k можна розрахувати за допомогою модифікованої логістичної функції:

$$k(\Delta) = \frac{L}{1 + e^{-h*(|\Delta| - x_0)}} \qquad (2)$$

де Δ – різниці між значенням стану клітини та оточення,

L – максимально можливе значення коефіцієнту k,

x_0 – точка середини логістичної кривої,

h – швидкість росту логістичної кривої.

Значення різниці між значенням стану клітини та оточенням розраховується за формулою:

$$\Delta = \frac{\sum_{i=1}^{n} n_j}{n} - s_t \qquad (3)$$

де n – кількість сусідніх клітин,

n_i – стан i-ї клітини з оточення клітини s_t,

s_t – стан клітини на ітерації t.

Графік залежності коефіцієнту сприйнятливості k на базі логістичної функції зображено на Рис. 6. Параметри запропонованої функції (Рис. 6) набувають значення $L = 1$, $x_0 = 0.5$, $h = -15$. У всякому разі, зі зміною постановки задачі коефіцієнти можуть бути змінені під особливості об'єкту моделювання.

Отож, формула (1) в правилі визначення стану клітини може бути об'єднана із запропонованим правилом залежності коефіцієнту сприйнятливості від стану клітини в формулі (2) та (3):

$$s_{t+1} = s_t + \Delta * \frac{L}{1 + e^{-h*(|\Delta| - x_0)}} \qquad (4)$$

Приклад роботи КлА з введеним правилом залежності коефіцієнту сприйнятливості від різниці станів клітини та її оточення зображено на Рис. 7. Варто зазначити, що на відміну від попередніх моделей (Рис. 5), в даній (Рис. 7) можна спостерігати утворення окремих парних епіцентрів клітин з великим значенням (від 0.75 до 1) та протилежним значенням (від -1 до 0.75). Це пов'язано з появою у клітин здатності зберігати власну думку деякий час незважаючи на «вороже» оточення. Даний прояв відповідає конкретним випадкам реального життя, тому нововведене правило є доречним у моделюванні навчального процесу.

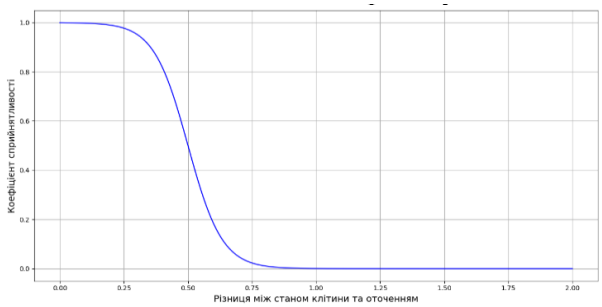
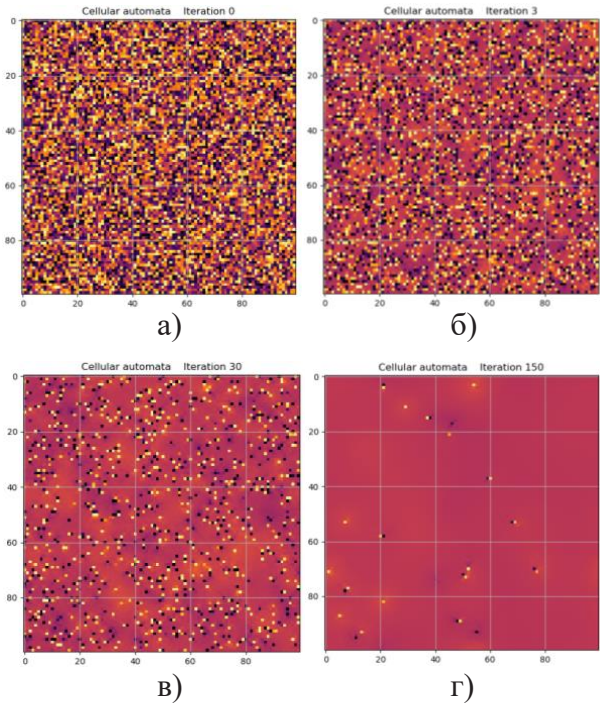


Рис. 6. Графік залежності коефіцієнту сприйнятливості від стану клітини та її оточення.

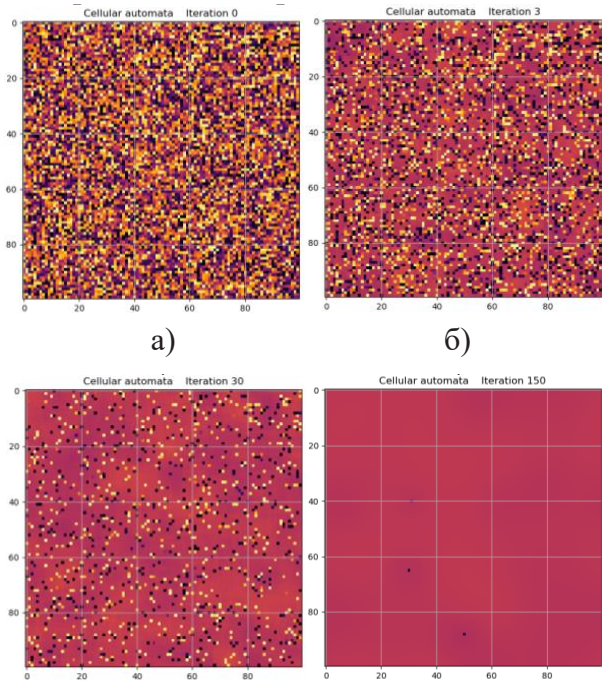


а) б) в) г)

Рис. 7. КлА з введеним правилом залежності коефіцієнту сприйнятливості від різниці станів клітини та її оточення. а) початкове клітинне поле, б) ітерація 3, в) ітерація 30, г) ітерація 150.

Правило циклічної зміни радіусів взаємодії клітин поля.

В реальному житті люди постійно знаходяться в русі, щодня людина знаходиться мінімум у двох різних колах спілкування: домашнє та робоче. Аналогічні процеси відбуваються і під час навчального процесу: проводяться спільні лекції, семінари та практичні заняття для окремих груп. Тож студенти постійно змінюють коло спілкування. Водночас різниця в успішності окремих груп часто зумовлюється різницею у початковій підготовці студентів до навчання, власною активністю студентів та загальною сприятливою атмосферою в групі. Перші два показники в моделі враховуються під час формування початкового клітинного поля, а останній раніше не брався до уваги.



а) б)

в) г)

Рис. 8. Реалізація правила циклічної зміни радіусів взаємодії клітин поля. а) початкове клітинне поле, б) ітерація 1, в) ітерація 2, г) ітерація 4.

Тому було запропоновано ввести правило циклічної зміни радіусів взаємодії клітин поля, що дозволяє задавати декілька «мап місцевості» радіусів взаємодії для циклічної її зміни протягом роботи автомату. Для перевірки роботи правила встановимо циклічність зміни радіусів взаємодії з зсувом на 2 значення, тобто клітина з радіусом взаємодії $R = 4$ змінює його на $R = 2$. Один цикл триватиме 2 ітерації автомату.

Приклад роботи КлА з введеним правилом циклічної зміни радіусів взаємодії клітин поля зображено на Рис. 8. Після введення правила модель змінила свою поведінку: на полі утворюється більше окремих парних епіцентрів клітин з великими (від 0.75 до 1) та (від -1 до -0.75) малими значеннями. Це пов'язано саме з тим, що у циклічній зміні радіусів взаємодії клітини спілкуються з різною кількістю оточуючих клітин. В умовах окремих груп це означає, що бачення питання іншої групи менше впливає на досліджувану групу і в ній з меншою ймовірністю з'являються люди з думкою, що кардинально відрізняється від інших.

Апробація модифікованого клітинного автомату при моделюванні навчального процесу при наукоцентричному підході.

Для моделювання процесів утворення думок в соціумі в роботі була використана математична та алгоритмічна формалізація, яка була відпрацьована вище на простіших моделях зі змінними умовами зміни стану клітин, визначення околиці клітин, залежності коефіцієнту сприйнятливості від різниці станів клітини та її оточення й циклічної зміни радіусів взаємодії клітин поля. Тож абстрактний КлА було пристосовано до задач моделювання процесів утворення думок у соціумі. Для подальшої апробації уточнимо постановку задачі, термінологію та зміст станів та процесів.

Постановка задачі моделювання життєвого циклу навчального процесу на базі модифікованого клітинного автомату.

Стани клітин. Кожна клітина – це індивідуум, свідомість якого може сприймати певну думку з приводу певного питання. Думка індивідуума може змінюватися в межах недискретної шкали $-1 \le s \le 1$, де ліва межа (-1) – одне ставлення до питання, середина (0) – нейтральна позиція до питання, а права межа (1) – протилежна до лівої межі думка. В подальшому будемо вважати, що ідеально-правильною (с точки зору користувачів моделі) думкою є та, що відповідає значенню 1.

Зміна думки індивідуума відбувається під впливом спілкування з оточуючими індивідуумами. На динаміку розвитку соціуму (клітинного поля) загалом дуже сильно впливають початкові значення думок клітин поля та радіус взаємодії кожної з клітин. При цьому без зовнішнього втручання середня думка соціуму прямує до нейтрального значення в проміжку від -0.1 до 0.1.

Радіус взаємодії в інформаційному полі завжди різний і залежить від виду каналу комунікації: особисте, групові заходи, мас-медіа тощо. Тому врахування радіусу, що змінюється – це принципова позиція. Тут треба вказати на відмінність радіусу від розглянутих вище моделей. У моделі навчального процесу радіус взаємодії кожної окремої клітини не може бути для всіх однаковим. Він для всіх різний. І це робить ускладнене правило визначення околиці клітини обов'язковою частиною моделі навчального процесу.

Апробація моделі на прикладі інформаційної операції щодо відвідуваності занять студентами.

Розглянута постановка задачі була апробована на моделі навчального процесу при наукоцентричному підході [9] в освіті, що орієнтується на навчанні студентів через заохочення їх до наукової діяльності та творчості під час навчання.

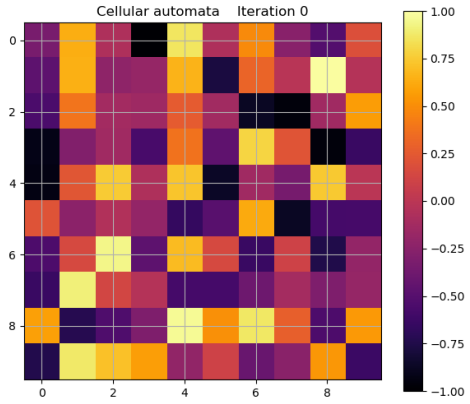
За **позитивну думку** в моделі вважається думка, що «студенту займатися наукою корисно для навчання та саморозвитку як спеціаліста». Протилежною **негативною думкою** є така, що «наукова діяльність студента не має сенсу, бо це ніяк не знадобиться в майбутньому».

Водночас, варто зазначити, що результати моделювання насправді залежать не тільки від початкового стану клітин в соціумі, а й від керуючих впливів зовнішнього оточення.

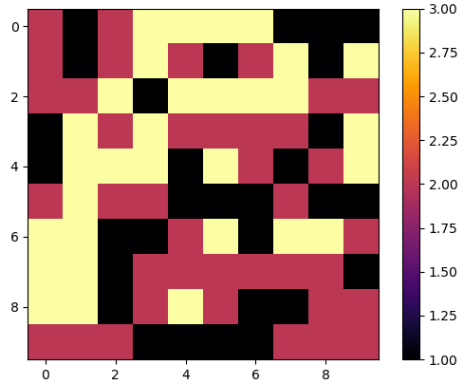
Постановка задачі.

Клітини – студенти одного потоку. При апробації вважаймо, що один потік складається зі 100 студентів і відповідно поле має розмір 10 на 10 клітин.

Поле для спрощення представлене у двовимірному вигляді та для збільшення контактів згорнуте у тор. Початкові значення клітин задаються випадково в межах зазначених чисел. У випадку моделювання реального потоку якоїсь спеціальності для визначення станів клітин поля достатньо провести соціологічне опитування щодо ставлення до наукової діяльності та активності студента. Це також дозволить встановити значення радіусів взаємодії студентів. Для моделі використовується однакове початкове клітинне поле (Рис. 9 а) та поле радіусів взаємодії (Рис. 9 б). Зміна умов буде змінювати клітинне поле також.



а)



б)

Рис. 9. Початкові поля автомату.
а) початкове клітинне поле,
б) поле радіусів взаємодії.

Розглянемо 6 варіантів розвитку подій:

Із випадково розподіленим (за думкою та здібностями) потоком студентів, на думки якого не впливають ззовні.

Із розділеним на групи потоком та без впливу ззовні.

Із розділеним на групи потоком та з впливом на групу з 4х студентів потоку.

Із розділеним на групи потоком та з впливом на групу з 16 студентів потоку.

Із розділеним на групи потоком та з впливом на групу з 36 студентів потоку.

Із розділеним на групи потоком та з впливом на окремих 36 студентів потоку.

Варіант 1. Випадковий розподіл та відсутність впливу

Для аналізу зміни загальної думки потоку будемо використовувати графік залежності значення думки потоку від ітерації автомату (часу). У випадку, коли клітинне поле задається випадково без поділу на підгрупи та без впливу, думка потоку передбачувано приходить на 155 ітерації (Рис. 10) до значення близького до нуля (0.011), що відповідає нейтральному ставленню студента до участі в науковій діяльності. Звісно, деякий час загальна думка коливається через те, що спочатку кожен студент має свою незалежну думку, але в результаті думка кожного прийде до нейтрального загального значення.

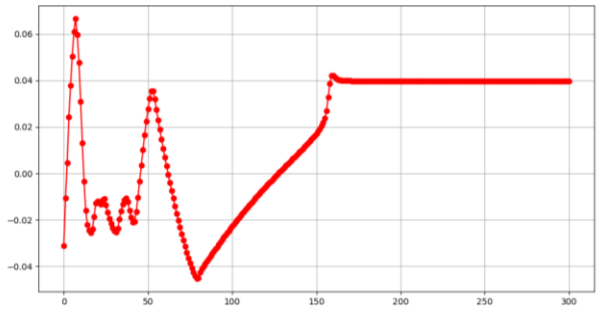


Рис. 10. Графік залежності середньої думки студентів потоку від часу.

Варіант 2. Поділ на групи та відсутність впливу

Оскільки зазвичай групи потоків формуються на початку першого курсу за рейтинговим балом або за балами вступних екзаменів, доречним є використання саме моделі з поділом на підгрупи. Для цього сформуємо нове клітинне поле потоку, який буде складатися з 4 груп, у кожній з яких студенти будуть мати певне ставлення до наукової діяльності в межах: від -1 до -0.5 (лівий верхній край поля – студенти нігілісти, які бояться наукової діяльності та не бачать у цьому сенсу), від -0.5 до 0 (лівий нижній край – студенти, які бояться наукової діяльності та є пасивними в навчальному процесі), від 0 до 0.5 (правий верхній край – студенти, які не бояться науки, але в них не вистачає на це загальної мотивації) та від 0.5 до 1 (правий нижній край – наукоорієнтовані студенти, здатні до наукової роботи за певного зовнішнього втручання). Створене клітинне поле (Рис. 11) буде використано і в подальших моделях.

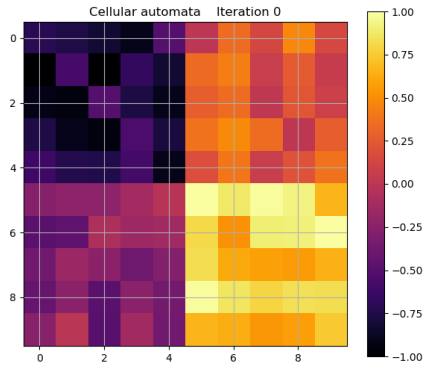


Рис. 11. Клітинне поле потоку з поділом на групи.

У разі поділу потоку на групи та невтручання в процес формування думки до нейтрального стану середня думка потоку (Рис. 12) приходить в 4.4 рази швидше, ніж у варіанті 1 (Рис. 10).

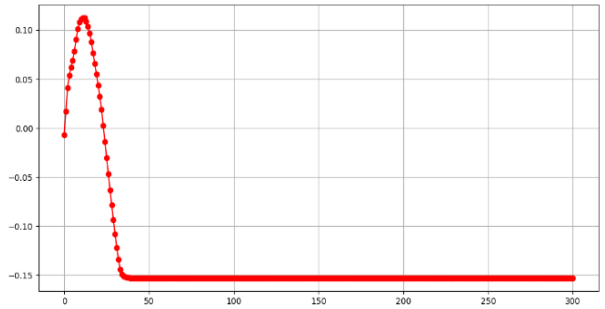


Рис. 12. Графік залежності середньої думки студентів потоку від часу.

Це пов’язано з тим, що у поділеному на групи потоці, студентів - однодумців об′єднують в одній групі, що призводить до зменшення активності дискусії між студентами, як це було в попередній моделі. Кінцевою середньою думкою все одно є майже нейтральне значення -0.16. Тому для формування правильного ставлення студентів до наукової діяльності викладачам, керівництву кафедри та факультету необхідно брати активну участь у формуванні думки студента.

Варіант 3. Поділ на групи та присутність впливу на 4 студентів

У варіанті 1 та варіанті 2 значення середньої думки потоку завжди прямує до нуля, бо звичайний стан для більшості студентів – нейтральний. Але на це можна впливати ззовні, проводячи агітацію студентів. Для впливу на думку студентів було вирішено обрати чотирьох студентів, з якими буде проводитися активна агітаційна робота (заохочення до написання робіт на наукові конкурси, написання тез та проведення досліджень). Для більшої ефективності було обрано студентів із найкращої групи (зі значенням думки від 0.5 до 1). В результаті роботи середнє ставлення потоку (Рис. 13) до наукової діяльності збільшилося на 0.45 відносно значення без втручання, що є досить значною різницею.

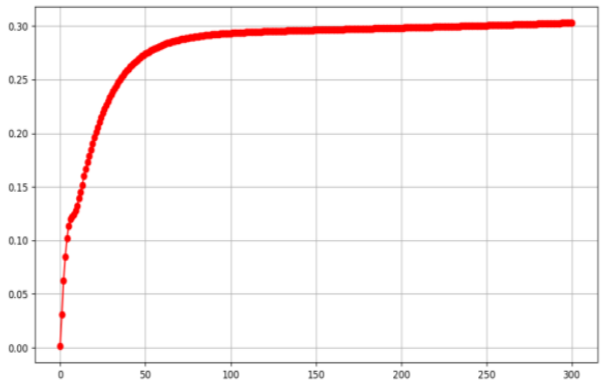


Рис. 13. Графік залежності середньої думки студентів потоку від часу.

Варіант 4. Поділ на групи та присутність впливу на 16 студентів

У цьому випадку кількість студентів, з якими проводиться агітація зросла до 16. В результаті збільшення агітації до 16 цього середнє ставлення потоку (Рис. 14) до наукової діяльності збільшилося на 0.63 (тобто становить 0.47) відносно значення без втручання. Водночас період насичення, коли швидкість зміни думки потоку зменшується, наступає раніше на 40 ітерацій автомату, що також є важливим показником, оскільки в такому випадку можуть економитися ресурси, що використовуються на агітацію.

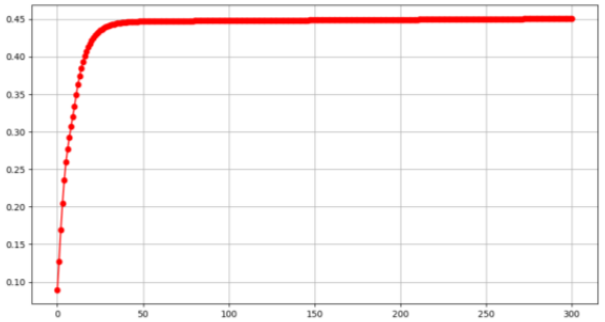


Рис. 14. Графік залежності середньої думки студентів потоку від часу.

Варіант 5. Поділ на групи та присутність впливу на 36 студентів

У цьому випадку кількість студентів, з якими проводиться агітація зросла до 36. Через що середнє ставлення потоку (Рис. 15) до наукової діяльності збільшилося на 0.79 (тобто становить 0.63) відносно значення без втручання. Досягнуте значення пербуває в межах, які встановлювалися під час формування найкращої групи (від 0.5 до 1). Це полегшує навчання студентів та співпрацю з викладачами. Воно стає ефективнішим.

Варіант 6. Поділ на групи та присутність впливу на 36 студентів з різних груп

Доцільним також є моделювання випадку, коли студенти, з якими має проводитися агітація, вибираються рівномірно в усіх групах потоку. Припустимо, студентів, з якими ведеться агітація, буде 36. У результаті цього середнє ставлення потоку (Рис. 16) до наукової діяльності збільшилося на 0.79 (тобто становить 0.63) відносно значення без втручання. Агітація відбулася досить ефективно, але й не на стільки, як у варіанті 5. У випадку ж варіанту 5 агітація відбувалася планомірно, без різких стрибків значень.

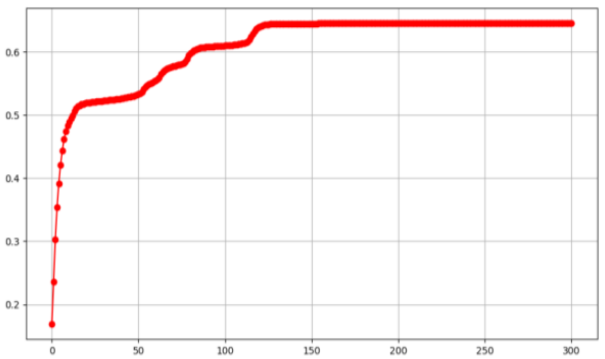


Рис. 15. Графік залежності середньої думки студентів потоку від часу.

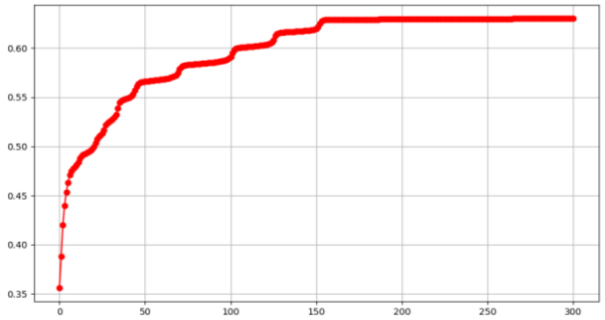


Рис. 16. Графік залежності середньої думки студентів потоку від часу.

На основі створених варіантів розвитку моделі навчального процесу за умови наукоцентричного підходу створимо порівняльну таблицю з початковою, кінцевою кількістю студентів, підготовлених до наукової діяльності та різницею між цими значеннями (Табл. 1).

Таблиця 1

Варіант	Початкова кількість	Кінцева кількість	Різниця кількостей
1	25	0	-25
2	25	0	-25
3	25	5	-20
4	25	66	41
5	36	77	41
6	36	83	47

Варто зазначити, що існуючі підходи до прогнозування поширення ідей у навчальному середовищі не враховують залучення студентів через інших студентів, а лише через викладачів. Це призводить до появи похибки в межах від 20% до 47%, що є значним відхиленням. Розроблена модель дає виграш в оцінці динаміки зміни уподобань студентів на рівні від 41% до 47%. Виходячи з результатів дослідження (Табл. 1) найефективнішим є 6 варіант розвитку подій, що робить активними у науковій діяльності 80% студентів потоку. Залежно від результатів опитування студентів та подальшого моделювання з різними варіантів розвитку може обиратися найбільш оптимальний. Невтручання керівництва факультету, кафедри та викладачів у процес формування ставлення до наукової діяльності призводить до практичної відсутності будь-якої активності студентів, що в свою чергу погіршує якість освіти студентів та формування висококваліфікованих спеціалістів.

Особливості програмної реалізації моделі

В умовах необхідності швидкого пошуку оптимального рішення шляхом розгляду великої кількості варіантів, питання ефективності програмної реалізації моделі постає особливо гостро. Тому для оптимізації моделі було вирішено реалізувати кінцеву версію моделі мовою програмування C#, оскільки дана мова має вищу швидкодію в порівнянні з аналогічними високорівневими мовами програмування [10] та безпосередньо пов'язана з реалізацією моделі, написаною мовою Python.

Основною особливістю моделі, що базується на використанні клітинних автоматів, є те, що розрахунки станів клітин автомату для наступної ітерації можна виконувати паралельно. Тож процес обрахунку можливо пришвидшити в декілька разів залежно від кількості доступних ядер процесору на комп'ютері, що використовується для обчислень.

Розглянемо послідовність обрахунку станів клітин для наступної ітерації автомату:

1. Отримання вхідних даних: розміри клітинного поля, значення клітин поля, значення радіусів взаємодії клітин поля.
2. Обрахунок різниці між значенням стану клітин та їх оточення (формула (3)).
3. Обрахунок стану клітин на наступній ітерації автомату (формула (4)).
4. Виведення отриманих значень для всього клітинного поля.

У нашому випадку розпаралелюванню піддається один процес, що виконується трьома етапами.

Для розпаралелювання було використано бібліотеку Task Parallel Library, а саме один з її класів – Parallel. В результаті порівняльного аналізу швидкодії алгоритмів було отримано наступні результати (Табл. 2) для обрахунків станів автомату.

На основі значень, наведених у таблиці 2 та додаткових даних, було побудовано графік залежності часу обрахунку від кількості клітин на полі (Рис. 17).

Таблиця 2

Кількість клітин\ Тип обчислень	Послідовне виконання	Паралельне виконання
100	30	394
2500	43	1000
10000	265	4438
409600	34704	108059
1000000	75976	150867
1638400	149098	241499
26214400	2686085	1407517

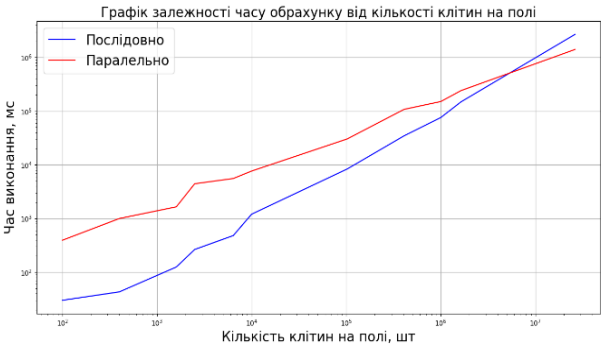


Рис. 17. Графік залежності часу обрахунку від кількості клітин на полі.

З графіку залежності часу обрахунку (Рис. 17) можна зробити висновок, що за невеликої кількості клітин в автоматі послідовне виконання є більш ефективним. Але, якщо кількість клітин на полі перевищує значення в приблизно 5400000 клітин, паралельне виконання розрахунків стає ефективнішим. Така кількість клітин може бути використана у разі збільшення охоплення моделі до розмірів невеликої країни, що загалом є можливим. Саме тому врахування особливостей моделі у її програмній реалізації є важливим аспектом для ефективного прогнозування та обрання керівничих рішень.

Висновки

1. У роботі збільшено активність наукової діяльності студентів за рахунок розробки моделі на базі клітинних автоматів із розширеним набором правил поведінки клітин. Отож, мета роботи досягнута.

2. Набули подальшого розвитку формалізовані та розширені правила клітинного автомату щодо замикання клітинного поля, визначення стану клітини та визначення околиці для окремих клітин.

3. У роботі вперше створено модель навчального процесу у наукоцентричному підході до освіти на базі моделі клітинних автоматів з урахуванням додаткових параметрів впливу параметрів визначення стану клітин, базового та модифікованого правил визначення околиці клітини, правила залежності коефіцієнту сприйнятливості від стану клітини та її оточення.

4. Розроблена модель дозволяє прогнозувати динаміку розповсюдження окремих думок в навчальному процесі та має достатній набір керуючих параметрів, щоб враховувати чинники реального світу в різних галузях для вирішення конкретних практичних задач. Вдосконалений клітинний автомат може бути використаний для моделювання як об'єктів соціуму, так і процесу розповсюдження ідей, що сприяють впровадженню наукоцентричного підходу.

5. У роботі розроблено програмні інструменти для аналізу динаміки розвитку клітинного автомату на прикладі моделі навчального процесу за умови наукоцентричного підходу. Розроблена модель дає виграш в оцінці динаміки зміни уподобань студентів на рівні від 41% до 47%, на відміну від існуючих підходів, що дають у прогнозуванні похибку від 20% до 47%.

6. Модель клітинних автоматів реалізована на трьох алгоритмічних мовах Python, MatLab та C#, що дозволяє обирати середовище моделювання за умови зміни особливостей постановки задачі.

7. Під час програмної реалізації моделі було виявлено умови, в яких паралельне виконання розрахунків моделі є доречним.

8. Напрямком подальших досліджень є розширення правил клітинного автомату, зокрема правила врахування непередбачуваних впливів ззовні студентського соціуму.

Література

1. G. Schneckenreither, N. Popper, F. Breitenecker. Methods for Cellular Automata and Evolution Systems in Modelling and Simulation. – 8th Vienna International Conferenceon Mathematical Modelling MATHMOD 2015, February 18 – February 20, 2015, Vienna, Austria, pp. 141 – 146. Available from: doi.org/10.1016/j.ifacol.2015.05.151
2. Шевченко Володимир, Берестов Деніс. Дослідження систем моделювання розповсюдження інформації в соціумі на базі клітинних автоматів. - XX Всеукраїнська науково-технічна конференція молодих вчених, аспі-

рантів та студентів. - Одеса, Квітень 21-22, 2020. – сс.225- 226.
3. Shevchenko Volodymyr; Georgi Dimitrov; Denys Berestov; Pepa Petrova; Igor Sinitcyn; Eugenia Kovatcheva; Ivan Garvanov; Iva Kostadinova One-way Function Based on Modified Cellular Automata in the Diffie-Hellman Algorithm for Big Data Exchange Tasks through Open Space. – DIGILIENCE 2020: Cyber Protection of Critical Infrastructures, Big Data and Artificial Intelligence. – Varna, September 30 – October 2, 2020. – pp.233-246. Available from: http://isij.eu/isij-47-digilience-2020-cyber-protection-critical-infrastructures-big-data-and-artificial
4. Shevchenko Volodymyr, Denis Berestov, Igor Sinitcyn Built-In Processor for Sharing Passwords Through the Open Information Space. - 2020 16-th International Conference Perspective Technologies and Methods in MEMS Design (MEMSTECH). Proceeding. - Lviv, April 22-26, 2020. - pp.40-44. Available from: ieeexplore.ieee.org/document/9109523
5. Shevchenko Volodymyr, Denis Berestov. Research of The Workability of One-Way Functions Based on Cellular Automata for The Modified Diffie-Helman Algorithm. - MSTIoE 2020-7. 7-th East European Conference on Mathematical Foundations and Software Technology of Internet of Everything.– Kyiv, December 22-23, 2020. pp.22-23
6. J. Freire, C. DaCamara. Using cellular automata to simulate wildfire propagation and to assist in fire management. - Natural Hazards and Earth System Sciences (NHESS), January 22, 2019, Lisbon, Portugal, pp. 169–179.
7. Alejandro Salcido. Equilibrium Properties of the Cellular Automata Models for Traffic Flow in a Single Lane // Cellular Automata - Simplicity Behind Complexity ed.by Salcido A. – 2011, pp. 159 – 192.
8. Lorenzo Piroli, Ignacio Cirac. Quantum Cellular Automata, Tensor Networks, and Area Laws. - Physical Review Letters, 6 November, 2020, Danvers, USA, p.5. Available from: doi.org/10.1103/PhysRevLett.125.190402
9. Kiray S.A., Kaptan F. The effectiveness of an integrated science and mathematics programme: Science-centred mathematics-assisted integration. - Energy Education Science and Technology Part B: Social and Educational StudiesVolume 4, Issue 2 - April 2012, pp. 943-956.
10. Z. Parveen, F. Nazish Performance comparison of most common high level programming languages. International Journal of Computing Academic Research, Volume 5, Number 5, October 2016, pp. 246-258.

Одержано 19.11.2021

Про авторів:

Шевченко Володимир Вікторович
студент бакалавр
Кількість публікацій в українських виданнях: 10.
Кількість публікацій в зарубіжних індексованих виданнях: 3.
http://orcid.org/0000-0002-2152-6816.

Берестов Денис Сергійович
асистент кафедри
Кількість публікацій в українських виданнях: 60.
http://orcid.org/0000-0002-3918-2978

Сініцин Ігор Петрович
завідувач відділом
Кількість публікацій в українських виданнях: 80.
http://orcid.org/0000-0002-4120-0784

Петрівський Володимир Ярославович
аспірант
Кількість публікацій в українських виданнях: 18.
http://orcid.org/0000-0001-9298-8244

Місце роботи авторів:

Факультет інформаційних технологій КНУ ім. Тараса Шевченка
вул. Богдана Гаврилишина, 24, Київ, Україна, 02000
E-mail: vladimir_337@ukr.net
Телефон: +380970166921

E-mail: berestov@ukr.net
Телефон: +380672247900

E-mail: vovapetrivskyi@gmail.com
Телефон: +380970487778

Інститут програмних систем НАН України
пр. Академіка Глушкова, 40, корпус 5, м. Київ, Україна, 03187
E-mail: ipsinitsyn@gmail.com
Телефон: +380672261313

УДК 004.43::004.94 https://doi.org/10.15407pp2021.04.082

А.Ю. Дифучин, І.В. Стеценко, Е.В. Жаріков

ГРАМАТИКА МОВИ ВІЗУАЛЬНОГО ПРОГРАМУВАННЯ ПЕТРІ-ОБ'ЄКТНИХ МОДЕЛЕЙ

Петрі-об'єктні моделі вирішують проблему тиражування фрагментів мереж Петрі з заданими параметрами та конструювання моделі системи з великої кількості елементів. Розроблена мова візуального програмування Петрі-об'єктних моделей дає можливість зменшити кількість помилок при конструюванні моделі за рахунок автоматизації кодування зв'язків між елементами моделі та графічного представлення моделі. Окрім тиражування Петрі-об'єктів, мова реалізує тиражування зв'язків між Петрі-об'єктами. Формалізація граматики мови візуального програмування представлена у вигляді правил виведення та зроблені висновки щодо її властивостей.

Ключові слова: формальна граматика, візуальне програмування, стохастична мережа Петрі, імітаційне моделювання.

Вступ

Програмне забезпечення імітаційного моделювання – це широкий клас програмних продуктів, призначених для побудови моделей, які відтворюють функціонування систем. Моделі використовують для дослідження властивостей систем, пошуку оптимальних умов, прогнозування. Моделі, побудовані з використанням мереж Петрі, рекомендовані для проєктування та аналізу програмного забезпечення стандартом ISO/IEC 15909-1:2004 [1]. Перевагами такого використання є візуалізація, математично формалізований опис, покрокове відтворення обчислювального процесу. Застосування мереж Петрі для розробки алгоритмів і програмного забезпечення з часом може втілитись у парадгму програмування на мережах Петрі [2]. Отже, мережі Петрі як засіб побудови моделей надзвичайно важливі для розвитку програмної інженерії. Недолік, з яким доводиться стикатись у розробці моделей мережами Петрі - це ускладнення сприйняття та побудови моделі при зростанні складності системи, що моделюється (складність може бути оцінена кількістю елементарних подій). Через нагромадження зв'язків та елементів перевага візуального сприйняття моделі зникає, модифікація параметрів моделі потребує значної кількості рутинних дій. Тому розвиток методів, що спрощують і прискорюють розробку моделей на основі мереж Петрі є актуальним науковим завданням.

Петрі-об'єктні моделі вирішують проблему тиражування фрагментів мереж Петрі з заданими параметрами та конструювання моделі системи з великої кількості елементів. Якщо система складається із сотень однотипних вузлів, опис кожного з яких потребує 10 переходів, 20 позицій та 40 дуг, то в термінах звичайної мережі Петрі довелось би створювати 1000 переходів, 2000 позицій, 4000 дуг та щонайменше 100 дуг для зв'язування фрагментів між собою. В термінах Петрі-об'єктної моделі один раз розроблена мережа Петрі, що описує функціонування одного вузла системи, використовується для створення 100 вузлів (із різними значеннями параметрів). Отже, розробка моделі значно спрощується і прискорюється. Скорочується також обсяг графічної інформації, яку потрібно зберігати для відтворення моделі. Широко відомий засіб для моделювання мережами Петрі CPNTools дає можливість використовувати фрагменти мережі Петрі повторно, проте налагоджувати і вбудовувати фрагмент у модель доведеться для кожного фрагменту окремо. Звісно, такий спосіб спрощує створення нового фрагменту мережі Петрі на основі вже існуючого, але не спрощує створення множинних наборів типових фрагментів. По суті, уможливлюється копіювання фрагментів і створення на їх основі нових, але не тиражування за заданим шаблоном з заданими параметрами. Зберігатись у CPNTools будуть усі скопійовані фрагменти

як наново створені, отже, на відміну від Петрі-об'єктного підходу, скорочення обсягів пам'яті на збереження фрагментів моделі не відбувається.

Петрі-об'єктні моделі дають можливість створювати моделі засобами мереж Петрі там, де раніше вони призводили до надмірного ускладнення: моделювання розповсюдження комп'ютерних вірусів [3], передача повідомлень у телекомунікаційних системах [4], моделювання алгоритмів паралельних обчислень [5]. Застосування Петрі-об'єктного підходу для моделювання систем різного призначення показало ефективність цієї технології конструювання моделей, що поєднує переваги об'єктно-орієнтованого підходу та стохастичних мереж Петрі. Проте кодування зв'язків між елементами мережі Петрі є процесом, схильним до помилок. Відшукання помилок у структурі моделі безпосередньо в коді потребує концентрації уваги. Якщо модель містить опис двох десятків подій, із цим ще можна впоратись. При більшій кількості подій процес розробки моделі настільки трудомісткий, що результати моделювання можуть не виправдати зусиль, витрачених на створення моделі.

Рішення цієї проблеми можна шукати у двох напрямках: 1) автоматизація, інтелектуалізація чи інші способи запобігання та виявлення помилок безпосередньо в коді, 2) візуалізація зв'язків та моделі в цілому для поліпшення сприйняття моделі, що будується. Підходи, які використовують виключно візуальне представлення моделі, примушують програміста відкривати та модифіковувати модель навіть зі зміною її параметрів. А якщо потрібна зміна водночас в кількох однотипних елементах, то доведеться повторювати рутинну дію. Такі дії швидше виконати безпосередньо в коді (одночасно потрібно забезпечити, щоб при відкритті моделі такі зміни потрапили у візуальне її представлення). Перевірити правильність зв'язків чи стану моделі в початковий момент часу з використанням інтелектуальних підходів не є можливим, оскільки можливі (допустимі) послідовності подій, які відповідають задуму моделі, є надзвичайно великою множиною навіть для відносно простих моделей.

Ідеальним вбачається рішення, коли програміст може застосовувати візуальне представлення і, водночас, використовувати кодоване представлення моделі. Проте перетворення візуального представлення моделі у кодоване відбувається, як правило, з втратою даних про розміщення графічних елементів. Тому перетворення з кодованого представлення моделі у візуальне її представлення потребує інтелектуальної обробки інформації про взаємозв'язки елементів. Ускладнює таке перетворення той факт, що не існує однозначної відповідності між кодованим представленням та візуальним представленням моделі, оскільки одній моделі може відповідати кілька варіантів візуального представлення з різним розміщенням графічних об'єктів.

Конфлікт між графічним представленням і кодованим представленням моделі зникає, якщо припустити, що візуальне представлення є водночас кодованим представленням моделі. Візуальні мови програмування використовують графічні об'єкти як алфавіт мови. Отже структури, складені з таких об'єктів є фактично кодом для представлення моделі.

Метою даного наукового дослідження є зменшення складності процесу конструювання моделі дискретно-подійної системи за рахунок використання мови візуального програмування Петрі-об'єктної моделі.

1. Формальні мови та засоби імітаційного моделювання дискретно-подійних систем

Із появою формальних граматик з'явились машино-незалежні мови програмування. Першою мовою програмування, побудованою на основі формальної граматики, стала мова ALGOL [7]. Мова імітаційного моделювання Simula, побудована на основі мови ALGOL, була задумана розробниками спочатку виключно як мова для цілей імітаційного моделювання дискретно-подійних систем, а стала першою об'єктно-орієнтованою мовою [6]. Її версія Simula 67 була написана вже як універсальна об'єктно-орієнтована мова програмування. Мовою Simula було написано провідне у свій час програмне забезпечення Arena. Ви-

користання блочних діаграм для побудови моделей і надзвичайно зручна розбудова ієрархічних моделей – основні переваги цього програмного забезпечення. Найбільш популярне в наші дні програмне забезпечення з імітаційного моделювання Simio по суті є продовженням концепції Arena з розвинутішими засобами анімації, зокрема 3D, та можливостями завдання розкладів [8].

Вказані програмні засоби та багато інших (AnyLogic, VisualSim, Simul8) ґрунтуються на описі дискретно-подійних систем у термінах блочних діаграм, що описують процес обробки об'єктів. Складні моделі зі специфічною взаємодією елементів важко вкласти у логіку спрацьовування достатньо обмеженого набору блоків, тому кількість блоків у бібліотеках постійно зростає, що негативно впливає на швидкість вивчення програмного забезпечення. Окрім того, логіка блочних діаграм завжди спирається на опис процесу обробки об'єктів, водночас управління ресурсами для обробки обмежено тільки їх кількістю, місткістю та розкладом. Деякі з існуючих програмних продуктів (Simio, Simul8) визнали обмеженість блочної побудови моделей і дають можливість розробникам створювати власні фрагменти коду і вбудовувати їх у моделі. Наприклад, Simio пропонує додавати написані на .NET фрагменти коду для передачі даних, опису специфічних алгоритмів вибору об'єктів з черг, опису серії експериментів [9]. Simula8 дає можливість додаткового опису функціональності блоків мовою Visual Logic[10].

Іншим підходом до побудови симуляції є розробка формалізації у вигляді мережі Петрі і запуск універсального алгоритму імітації для побудованої мережі. Програмне забезпечення CPNTools [11] використовує формалізацію опису моделі у вигляді ієрархічних розфарбованих мереж Петрі та функціональну мову CPN ML для опису даних, змінних та функцій. Мова CPN ML є розширенням мови функціонального програмування Standard ML. Використання функціональної мови програмування дає можливість CPNTools досягти необхідної гнучкості в побудові моделей, проте значно ускладнює сприйняття моделі з графічного представлення, оскільки, окрім графічних об'єктів та їхніх параметрів, опис моделі містить вирази-функції. Якщо CPN ґрунтується на математичному формалізмі розфарбованих мереж Петрі, то при додаванні до опису моделі фрагментів коду функціональною мовою математичний формалізм руйнується [12].

Отже, використання програмних засобів імітаційного моделювання свідчить про переваги використання графічного представлення моделі, такі як наочність представлення, швидкість розробки, зменшення кількості помилок у відтворенні структури. Водночас, забезпечити необхідну гнучкість розробки моделей не вдається без надання розробнику доступу до програмного коду моделі та/або надання дозволу вбудовувати фрагменти програмного коду у код моделі.

2. Поняття мови візуального програмування

Терміни «візуальне моделювання» та «візуальне програмування» суттєво відрізняються. Візуальне моделювання - це застосування графічних об'єктів для створення моделей. У загальному випадку візуальні моделі не призначені для обчислень. Наприклад, моделі UML використовують для опису програмного забезпечення, що проєктується. Візуальне програмування – це застосування графічних об'єктів для розробки програмного коду, який реалізує певний алгоритм обчислень. Відповідно, існує суттєва різниця між мовою візуального моделювання та мовою візуального програмування. Мова візуального моделювання – це набір графічних об'єктів для представлення систем, об'єктів або понять, що проєктуються або існують. Мова візуального програмування – це набір взаємопов'язаних графічних об'єктів (разом з чисельними параметрами), який однозначно може бути трансформований у програмний код.

На відміну від природних мов, мови програмування є формальними, тобто породжуються формальними граматиками. Опис будь-якої формальної граматики складається з алфавіту термінальних символів, алфавіту нетермінальних символів, початкового символу граматики та правил виведення, що використовують для розпіз- навання наборів символів. Алфавіти термінальних та

нетермінальних символів разом утворюють алфавіт мови. Граматика мови програмування породжує множини слів (виведені з термінальних символів за правилами виведення), що можуть бути інтерпретовані (перетворені) в модель обчислень. Приклади розробки формальних граматик, що орієнтовані на використання упорядкованих наборів символів, викладені у [13]. Математична теорія формальних граматик викладена у [14]. Формальне представлення граматики мови важливе для гарантування однозначності інтерпретації мовного виразу та однозначного перетворення його у алгоритм (перелік інструкцій для обчислень), що запускається на виконання.

Окрім алфавіту, неодмінними ознаками формальної мови є синтаксис мови та її семантика. Синтаксис визначає дозволені конструкції мови, семантика - зміст цих конструкцій. Синтаксис визначає набір правил, за якими складаються мовні конструкції. Розбір мовного виразу, складеного за синтаксичними правилами, виконується компілятором мови у відповідності до правил виведення (продукцій), що задані граматикою мови. Семантика – це зв'язок між синтаксисом мови та моделлю обчислення. Формальна семантика задається математичною моделлю, що описує обчислення у відповідності до виразу, заданого мовою. Синтаксичні помилки виявляються автоматизовано під час компіляції програмного коду, а семантичні – під час виконання програми.

Алфавіт мови візуального програмування – це набір графічних елементів, що виконують роль символів. У текстовій мові програмування слово (лексема) – це послідовність символів, утворене за правилами, що визначені синтаксисом мови. Розташування символів у послідовності зліва направо означає фактично їх з'єднання, і це з'єднання інтерпретується граматикою (при розборі виразу). У візуальній мові, що використовує графічне представлення для опису мовного виразу, на розташування елементів у графічному просторі не накладається ніяких обмежень. Тобто послідовності символів у тому розумінні, як вони присутні у традиційних мовах програмування, відсутні. Натомість у графічному представленні встановлюються зв'язки між елементами. Наприклад, якщо кінець дуги співпадає з місцем розташування вершини графу, то елемент дуга поєднаний з елементом вершини. Якщо такі зв'язки можуть утворювати фрагменти, які однозначно інтерпретуються, як лексеми мови, то графічне представлення може використовуватись для візуального представлення виразів програмного коду.

У разі текстової мови програмування текст перетворюється в алгоритм виконання обчислень за правилами, визначеними семантикою мови. У мові візуального програмування зображення, складене з взаємопов'язаних графічних елементів, трансформується у змістові конструкції, що запускаються на обчислення. Опис графічного елементу може супроводжуватись набором чисельних параметрів, що впливають на обчислення. Отже, виразом мови візуального програмування є набір взаємопов'язаних графічних об'єктів, що може бути запущений на обчислення.

Граматика текстової мови породжує послідовності символів термінального алфавіту, виведених з початкового символу. Граматика мови візуального програмування має породжувати графічні зображення, утворені з графічних елементів, що складають алфавіт термінальних символів мови. Далі на прикладі мови візуального програмування Петрі-об'єктних моделей, буде показано, що мова візуального програмування може бути визначена формальною граматикою так само, як і текстова мова програмування, за умови, що визначені зв'язки між графічними елементами.

2. Поняття Петрі-об'єктної моделі

Петрі-об'єктний підхід до розробки імітаційних моделей дискретно-подійних систем комбінує переваги використання стохастичних мереж Петрі та об'єктно-орієнтованої технології для опису моделей. Стохастична мережа Петрі з багатоканальними та конфліктними переходами визначена такими множинами елементів: позиції, переходи, дуги. Для зручності обробки графічного представлення будемо розрізняти вхідні та вихідні дуги переходу. Запуск переходу у мережах Петрі з часовими затримками здійснюється у два кроки: вхід марке-

Таблиця 1. Параметри елементів стохастичної мережі Петрі з багатоканальними та конфліктними переходами

Назва елементу	Параметр	Тип значення	Зміст параметру
Позиція	Кількість маркерів у позиції	Ціле невід'ємне число	Символізує виконання умови для запуску переходу. При достатній кількості маркерів у вхідних позиціях переходу, виконується умова входу маркерів в перехід.
Перехід	Часова затримка	Дійсне невід'ємне число	Затримка між входом та виходом маркерів з переходу, що реалізується в алгоритмі імітації очікуванням просування часу до відповідного значення модельного часу. Затримка відтворює виконання певних дій в реальній системі.
	Пріоритет	Ціле число	При вирішенні конфлікту переходи з найбільшим пріоритетом з рівною ймовірністю можуть здійснити свій запуск. Конфлікт переходів виникає, коли одночасно для більш, ніж одного переходу виконана умова входу маркерів в перехід.
Вхідна дуга	Кількість зв'язків	Натуральне число	Визначає кількість маркерів, що вилучаються при вході маркерів в перехід. Вхід маркерів в перехід здійснюється за умови, що в усіх вхідних позиціях переходу наявна кількість маркерів у кількості, зазначеній у кількості зв'язків відповідної вхідної дуги.
Вихідна дуга	Кількість зв'язків	Натуральне число	Визначає кількість маркерів, що додаються при виході маркерів з переходу. Вихід маркерів з переходу відбувається, коли сплине часова затримка, зазначена Запуск переходу здійснюється за умови, що в усі вихідні позиції переходу додаються маркери у кількості, залежно від кількості зв'язків відповідної вхідної дуги.

рів в перехід та вихід маркерів з переходу (після того, як сплинула часова затримка). Параметри, які визначені для елементів, представлені у таблиці 1.

Петрі-об'єкт за визначенням, введеним у [15], є об'єктом суперкласу, який містить методи для відтворення динаміки об'єкта у відповідності до мережі Петрі, що зберігається у спеціально призначеному для цього полі. Створюючи об'єкт конструктор суперкласу використовує мережу Петрі та, можливо, набір параметрів, тому об'єкт можна представити виразом:

$$o = (net, list),$$

де *o* - Петрі-об'єкт, *net* - мережа Петрі, *list* - список параметрів.

Петрі-об'єктна модель створюється з множини Петрі-об'єктів, динаміка яких взаємопов'язана через спільні позиції. Поєднання Петрі-об'єктів через спільні позиції гарантує, що динаміка моделі визначається мережею Петрі, яка є об'єднанням мереж Петрі-об'єктів [15]. Програмно спільні позиції реалізуються через присвоєння адрес пам'яті відповідних об'єктів-позицій, що можна представити наступним виразом:

$$o_u.net.p_b = o_v.net.p_a,$$

де o_u, o_v – Петрі-об'єкти, які з'єднуються, *o.net* – мережа Петрі-об'єкта *o*, *o.net.p* – позиція мережі Петрі-об'єкта *o*. Усі пари спільних позицій двох Петрі-об'єктів, що утворюють з'єднання між ними, будемо називати конектором:

$$connector(o_u, o_v) = \{(o_u.net.p_b, o_v.net.p_a)\}.$$

Модель утворюється з Петрі-об'єктів операцією агрегування (в термінах об'єктно-орієнтованої технології), тому модель можна представити наступним виразом:

$$m \longrightarrow\!\diamond \{o_{1,\ldots,} o_n\},$$
$$m.net = \bigcup_j o_j.net,$$

де m – модель, —◇ – символ операції агрегування, o_j – Петрі-об'єкти, які агрегуються, *m.net* – мережа Петрі моделі, *o.net* – мережа Петрі-об'єкта *o*.

Перетворення стану мережі Петрі всієї моделі, як доведено у [15], розпадається на перетворення стану мереж Петрі-об'єктів. Це збільшує швидкодію імітації, оскільки, по-перше, при відтворенні події відбуваються зміни тільки у відповідному Петрі-об'єкті, по-друге, пошук найближчої події відбувається переглядом стану усіх об'єктів моделі замість перегляду стану усіх переходів моделі.

Оскільки при з'єднанні кількох Петрі-об'єктів утворюється знову мережа Петрі, а не структура більш високого рівня, то можуть бути введені операції над Петрі-об'єктами. Операція тиражування *multiply* означає, що мережа Петрі використовується для створення групи Петрі-об'єктів. При цьому списки чисельних параметрів елементів мережі Петрі можуть відрізнятись (окрім кількості маркерів у позиціях, які використовуються для з'єднання з іншими Петрі-об'єктами):

$$g = multiply(net, lists, k) \Leftrightarrow$$
$$\Leftrightarrow \forall i: o_i = (net, list_i), i = 1..k,$$

де g – група Петрі-об'єктів, *net* - мережа Петрі, що використовується для створення групи об'єктів, $lists = \{list_i\}$ - множина списків параметрів, що застосовуються для створення мережі Петрі конкретного об'єкта, k – кількість створюваних об'єктів.

З'єднання Петрі-об'єкта *o* з групою g Петрі-об'єктів за правилом, сформульованим для пари Петрі-об'єктів, фактично тиражує зв'язок між Петрі-об'єктом *o* та кожним Петрі-об'єктом з групи g:

$$g.net.p_b = o.net.p_a \Leftrightarrow$$
$$\Leftrightarrow \forall o_i \in g:\ o_i.net.p_b = o.net.p_a.$$

Об'єкти, взаємопов'язані між собою, можуть утворювати колекції , для яких можна здійснювати тиражування за наступними правилами:

де f – група колекцій Петрі-об'єктів, *objects* - Петрі-об'єкти, що використовується для створення колекції, *connectors* – конектори між Петрі-об'єктами, k – кількість операцій тиражування колекції.

Інші об'єкти можуть створювати з'єднання з групою колекцій :

$$f.g_u.net.p_b = o.net.p_a \Leftrightarrow$$
$$\Leftrightarrow \forall u \in g_u: f.u.net.p_b = o.net.p_a,$$

де з'єднання встановлюється між Петрі-об'єктом *o* та групою Петрі-об'єктів g_u з колекції.

У [16] розглянуто застосування дворівневої структури графічного представлення Петрі-об'єктної моделі на прикладі моделі телекомунікаційної системи і визначені переваги використання мови візуального програмування. У наступному розділі статті для запропонованої мови даний опис у вигляді формальної граматики та зроблено висновки про її властивості.

3. Граматика мови візуального програмування Петрі-об'єктних моделей

Мова візуального програмування визначає набори графічних елементів (символів), що можуть бути перетворені на змістові конструкції Петрі-об'єктної моделі. Графічні елементи, передбачені для конструювання Петрі-об'єктної моделі, є символами мови. Набори елементів, які можуть бути перетворені на імітаційні моделі і запущені на виконання – це лексеми (слова) мови. Мережі Петрі-об'єктів – це лексеми в граматиці. Вирази мови – це фрагменти Петрі-об'єктної моделі.

Введемо алфавіт термінальних символів для опису мережі Петрі: *p* - позиція, *t* - перехід, *a* – дуга. Введемо алфавіт нетермінальних символів: *E* - комбінація термінальних символів, що задає одну вхідну дугу переходу, *Q* - комбінація термінальних символів, що задає одну вихідну дугу переходу, *N* - комбінація кількох нетермінальних символів *E* та *Q*, що може бути інтерпретована в мережу Петрі. Символом *I* позначимо початковий символ граматики. Вважатимемо,

$$f = multiply(objects, connectors, k) \Leftrightarrow$$
$$\forall o \in objects, \forall c \in connectors, \forall u, v \in objects | (u.net.p_b, v.net.p_a) \in c:$$
$$\begin{cases} g_o = multiply(o.net, lists, k), \\ \forall j: c_j = \left\{ (u_j.net.p_b, v_j.net.p_a) \,\middle|\, u_j \in g_{u_j}, v_j \in g_{v_j} \right\} \end{cases}, j = 1..k,$$

що дуга, яка поєднує елементи «позиція» та «перехід», визначає відношення «попередній – наступний» у послідовності символів, що задає лексему мови програмування. Правила виведення граматики для введених символів представлені у таблиці 2. Правила визначають, що усі набори символів інтерпретуються як трійки нетермінальних елементів *pat* та *tap*. У найпримітивнішому випадку мережа Петрі складається з однієї позиції (без переходів). Мережа Петрі може містити позиції, які не з'єднані з переходами (такі позиції зберігають інформацію про стан об'єкта, а зміни в такому стані можуть відбутись тільки в результаті подій, що відбуваються в інших Петрі-об'єктах).

Для опису Петрі-об'єктної моделі доповнимо алфавіт термінальних символів граматики такими символами: *o* - Петрі-об'єкт, *g* - група Петрі-об'єктів, *f* – група колекцій Петрі-об'єктів, *s* – позиція Петрі-об'єкта, що використовується для з'єднання з іншими Петрі-об'єктами, *l* – ототожнення позицій Петрі-об'єктів. Вважатимемо, що конектор, який поєднує Петрі-об'єкт з іншим Петрі-об'єктом, або Петрі-об'єкт з групою Петрі-об'єктів, визначає відношення «попередній-наступний» у послідовності символів, що задає лексему мови програмування. Доповнимо алфавіт нетермінальних символів граматики такими символами: *Z* – комбінація символів, що визначає пару з'єднаних Петрі-об'єктів, *Y* - комбінація символів, що визначає з'єднання Петрі-об'єкта з групою Петрі-об'єктів, *X* - комбінація символів, що визначає з'єднання Петрі-об'єкта з групою колекцій, *C* - конектор Петрі-об'єктів, що складається з кількох ототожнень позицій Петрі-об'єкта з позиціями іншого Петрі-об'єкта. Правила виведення 7-12 граматики (для введених символів) представлені у таблиці 3.

Таблиця 2. Правила виведення 1-6 граматики (синтаксис мови)

Правило	Зміст правила
1. $I \longrightarrow NI \mid N$	Для визначення Петрі-об'єктів має бути визначена одна або кілька мереж Петрі.
2. $N \longrightarrow EQ \mid QE$	Якщо вказана вхідна дуга, то обов'язково має бути вихідна. Якщо вказана вихідна дуга, то обов'язково має бути вхідна.
3. $N \longrightarrow EN \mid QN$	Може бути задано будь-скільки вхідних та вихідних дуг у довільному порядку.
4. $E \longrightarrow pat$	Трійка термінальних символів, що визначають вхідну дугу переходу
5. $Q \longrightarrow tap$	Трійка термінальних символів, що визначають вихідну дугу переходу
6. $N \longrightarrow p \mid pN$	Одна або декілька позицій в мережі Петрі

Таблиця 3. Правила виведення 7-12 граматики

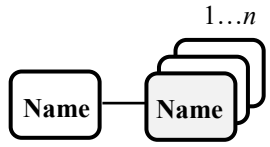
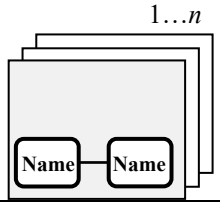
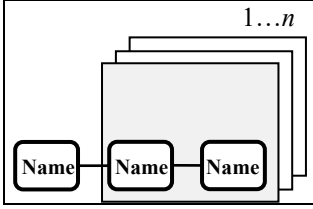
Правило	Зміст правила
7. $I \longrightarrow ZI \mid YI \mid XI$	Модель складається з трійок елементів, що визначають зв'язки між двома Петрі-об'єктами, між Петрі-об'єктом і групою Петрі- об'єктів, або групою колекцій Петрі-об'єктів та мереж Петрі для створення Петрі-об'єктів (див. правило 1)
8. $Y \longrightarrow oCg \mid g$	Трійка символів, що визначають зв'язок Петрі-об'єкта з групою Петрі-об'єктів. Ототожнення позицій, що вказані в конекторі, тиражуються для всіх Петрі-об'єктів групи. Допускається конструювання моделі з групи Петрі-об'єктів, що не мають зв'язків з іншими Петрі-об'єктами.
9. $X \longrightarrow oCf \mid f$	Трійка символів, що визначають зв'язок Петрі-об'єкта з групою Петрі-об'єктів, що входить до складу колекції. Ототожнення позицій, що вказані в конекторі, тиражуються для всіх Петрі-об'єктів групи. Допускається конструювання моделі з групи колекцій Петрі-об'єктів, що не мають зв'язків з іншими Петрі-об'єктами.
10. $Z \longrightarrow oCo \mid o$	Трійка символів, що визначають зв'язок Петрі-об'єкта з іншим Петрі-об'єктом. Допускається конструювання моделі з одного Петрі-об'єкта, що не має зв'язків з іншими Петрі-об'єктами.
11. $I \longrightarrow oI \mid gI$	Один Петрі-об'єкт, або група Петрі-об'єктів в моделі, або кілька Петрі-об'єктів без зв'язків, або кілька груп Петрі-об'єктів без зв'язків.
12. $C \longrightarrow slsC \mid sls$	Трійка символів, що визначають ототожнення позиції одного Петрі-об'єкта з позицією іншого Петрі-об'єкта. Конектор складається з одного або кількох ототожнювань позицій двох Петрі-об'єктів.

Графічні зображення символів мови представлені у таблиці 4. Параметри елементів, які представлені символами мови, мають у графічному представленні спеціально відведені поля. Параметр Name призначений для введення назви відповідного елементу.

Таблиця 4

Графічне зображення символів алфавіту мови візуального програмування Петрі-об'єктних моделей

Графічне представлення символу	Скорочене позначення символу	Інтерпретація
Name *k*	*p*	Позиція мережі Петрі з кількістю маркерів *k*.
Name *d*	*t*	Перехід мережі Петрі з часовою затримкою *d*, зі значенням пріоритету *r* (ширина прямокутника розраховується за формулою $w \cdot (1 + 0{,}5^r)$, де w – значення ширини, що відповідає пріоритету 1)
q	*a*	Дуга з кількістю зв'язків *q*. Значення *q*=1 є таким, що приймається за замовчуванням
1	*E*	Вхідна дуга переходу (з позиції)
1	*Q*	Вихідна дуга переходу (у позицію)
k	*N*	Фрагмент мережі Петрі, що складається з трьох вхідних дуг та трьох вихідних дуг: EEQEQQ.
Name placeName placeName placeName	*o*	Петрі-об'єкт із вказівниками на позиції, що можуть використовуватись для з'єднання з іншими Петрі-об'єктами.
placeName	*s*	Вказівник на позицію Петрі-об'єкта, що використовується для з'єднання
pName pName	*sls*	Ототожнення позиції Петрі-об'єкта з позицією іншого Петрі-об'єкта
	C	Конектор
Name **Name**	*Z*	Спрощене представлення Петрі-об'єкта та з'єднання двох Петрі-об'єктів
1…*n* **Name**	*g*	Група *n* Петрі-об'єктів, створених за однаковим шаблоном Петрі-об'єкта з назвою Name

	*Y*	З'єднання Петрі-об'єкта з групою *n* об'єктів
	*f*	Група *n* колекцій Петрі-об'єктів, створених за однаковим шаблоном. По суті, фрагмент моделі тиражується у кількості *n*.
	*X*	Група *n* колекцій Петрі-об'єктів, створених за однаковим шаблоном. По суті, це фрагмент моделі, що тиражується у кількості *n*.

Правила виведення 1-12 визначають граматику мови візуального програмування Петрі-об'єктних моделей. Наведена граматика є контекстно-вільною або граматикою типу 2 за ієрархією Хомського, оскільки інтерпретація будь-якого нетермінального символу не залежить від символів, які його оточують. Приклад виведення за визначеною граматикою (породження виразу з застосуванням послідовності правил 7, 10, 12, 7, 8, 12, 1, 3, 5, 2, 4, 5, 1, 3, 5, 3, 4, 2, 4, 5):

$$I \longrightarrow ZI \longrightarrow oCoI \longrightarrow oslsoI \longrightarrow oslsoYI \longrightarrow$$
$$\longrightarrow oslsooCgI \longrightarrow oslsooslsgI \longrightarrow$$
$$\longrightarrow oslsooslsgNI \longrightarrow oslsooslsgQNI \longrightarrow$$
$$\longrightarrow oslsooslsgtapNI \longrightarrow$$
$$\longrightarrow slsooslsgtapEQI \longrightarrow$$
$$\longrightarrow ooslsooslsgtappatQI \longrightarrow$$
$$\longrightarrow oslsooslsgtappattapI \longrightarrow$$
$$\longrightarrow oslsooslsgtappattapI \longrightarrow$$
$$\longrightarrow oslsooslsgtappattapN \longrightarrow$$
$$\longrightarrow oslsooslsgtappattapQN \longrightarrow$$
$$\longrightarrow oslsooslsgtappattaptapN \longrightarrow$$
$$\longrightarrow oslsooslsgtappattaptapEN \longrightarrow$$
$$\longrightarrow oslsooslsgtappattaptappatN \longrightarrow$$
$$\longrightarrow oslsooslsgtappattaptappatEQ \longrightarrow$$
$$\longrightarrow oslsooslsgtappattaptappatpatQ \longrightarrow$$
$$\longrightarrow oslsooslsgtappattaptappatpattap.$$

Виведення може бути також представлене у вигляді дерева виведення (рис. 1). Наведений ланцюжок відповідає Петрі-об'єктній моделі, що складається з двох Петрі-об'єктів одного типу та групи Петрі-об'єктів іншого типу. Мережа Петрі об'єктів першого типу складається з однієї вхідної дуги та двох вихідних дуг з відповідно визначеними для них позиціями та переходами, а мережа Петрі другого типу складається з двох вхідних дуг та двох вихідних дуг.

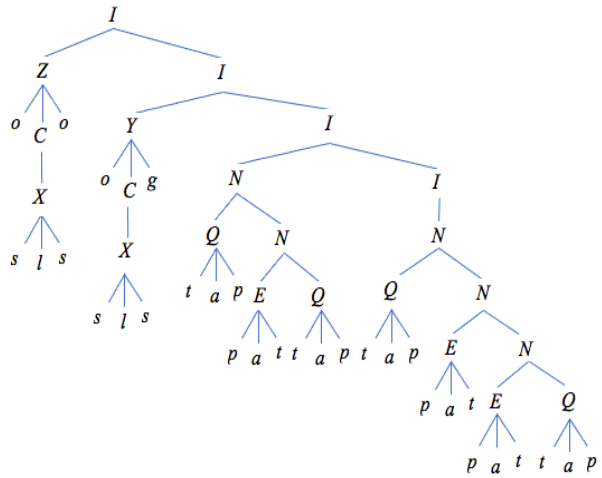


Рис. 1. Дерево виведення для моделі, представленої мовою візуального про-грамування.

При створенні елементів у графічному середовищі елементи одного типу зберігаються в масиві у послідовності створення розробником моделі. Враховуючи однозначність послідовності елементів у масивах, очевидно, що існує тільки одне дерево виведення для будь-якого візуального представлення моделі. Оскільки існує тільки одне дерево виведення для візуального представ-

лення моделі, то введена граматика є однозначною.

Алгоритми пошуку непродуктивних символів та недосяжних символів наведені у монографії [5]. Складаємо список нетермінальних символів, які можуть бути виведені з термінальних символів: *E, Q, N, C, Y, I*. Оскільки усі нетермінальні символи алфавіту потрапили до цього списку, то непродуктивних (зайвих) символів немає. Складаємо список нетермінальних символів, які можуть бути виведені з початкового символу: *I, Z, Y, N, C, E, Q*. Оскільки до списку потрапили усі нетермінальні символи граматики, то недосяжних символів у визначеній граматиці немає. Ланцюгових правил серед правил 1-11 немає. Отже, визначена граматика є приведеною.

Форма Бекуса-Наура – це інший спосіб введення граматики, який використовують, коли символів надто багато і правил виведення відповідно теж багато. Очевидно, що правила граматики 1-11 можна описати у вигляді розширеної нотації Бекуса-Наура.

Петрі-об'єктна-модель, описана візуальною мовою програмування, трансформується в алгоритмічну мову і запускається на виконання алгоритмом імітації. Правила перетворення візуальної моделі у модель обчислень визначають семантику мови.

4. Приклад представлення моделі мовою візуального програмування

Наведемо приклад розробки моделі з використанням розробленої мови візуального програмування. Побудуємо імітаційну модель, що відтворює обробку запитів клієнт-серверним застосунком. Запити надходять від користувачів двох типів: «авторів» та «гостей». Кожний тип користувача має свою логіку роботи з клієнт-серверним застосунком. Користувачі-автори, на відміну від користувачів-гостей, можуть додавати та змінювати інформаційний контент.

Загальною проблемою таких систем може бути довготривала обробка запиту через очікування доступу до бази даних та обробку даних серверним застосунком. Імітація системи на моделі може допомогти розробникам визначитися з параметрами обчислювальних ресурсів, які забезпечують прийнятне значення очікування відповіді на запит, та з вимогами до швидкодії окремих процесів обробки запитів.

Фрагмент моделі, що містить обробку запитів, які надходять від користувачів-гостей, представлений на рисунку 2. Мережа Server імітує обробку запитів на сервері. Оскільки запити надходять на пошук інформації або на перегляд, то використовуються два Петрі-об'єкти, створені з мережею Server, зі спільними позиціями-ресурсами Limit queue, Threads, DB access. Використання двох Петрі-об'єктів (замість одного) дозволяє відслідковувати обробку різних видів запитів окремо, а також використовувати в кожному фрагменті часові затримки, характерні для обробки відповідного запиту.

Позиції мереж Петрі, що можуть використовуватись для з'єднання з іншими Петрі-об'єктами, утворюють список позицій-сокетів Петрі-об'єкта. Такі позиції позначені сірим кольором на зображенні мережі Петрі (див. рис. 2) та представлені у списку досутпних для з'єднання позицій у зображенні Петрі-об'єкта (див. рис. 3). Три з'єднаних Петрі-об'єкти User, Server, Server утворюють фрагмент моделі, що імітує обробку запитів одного користувача. Щоб створити 50 користувачів, побудований фрагмент тиражуємо та з'єднуємо утворену групу колекцій з Петрі-об'єктами Server конектором Server-Server.

Модель, побудована з графічних елементів, перетворюється на обчислення алгоритмом імітаційного моделювання стохастичної мережі Петрі. Результати імітації Петрі-об'єктної моделі інформаційної системи наведені у публікації [17].

Висновки

Розроблена формальна граматика мови візуального програмування Петрі-об'єктних моделей. Синтаксичні правила складання мовних виразів визначені правилами виведення (продукціями), з яких випливає, що визначена граматика є контек-

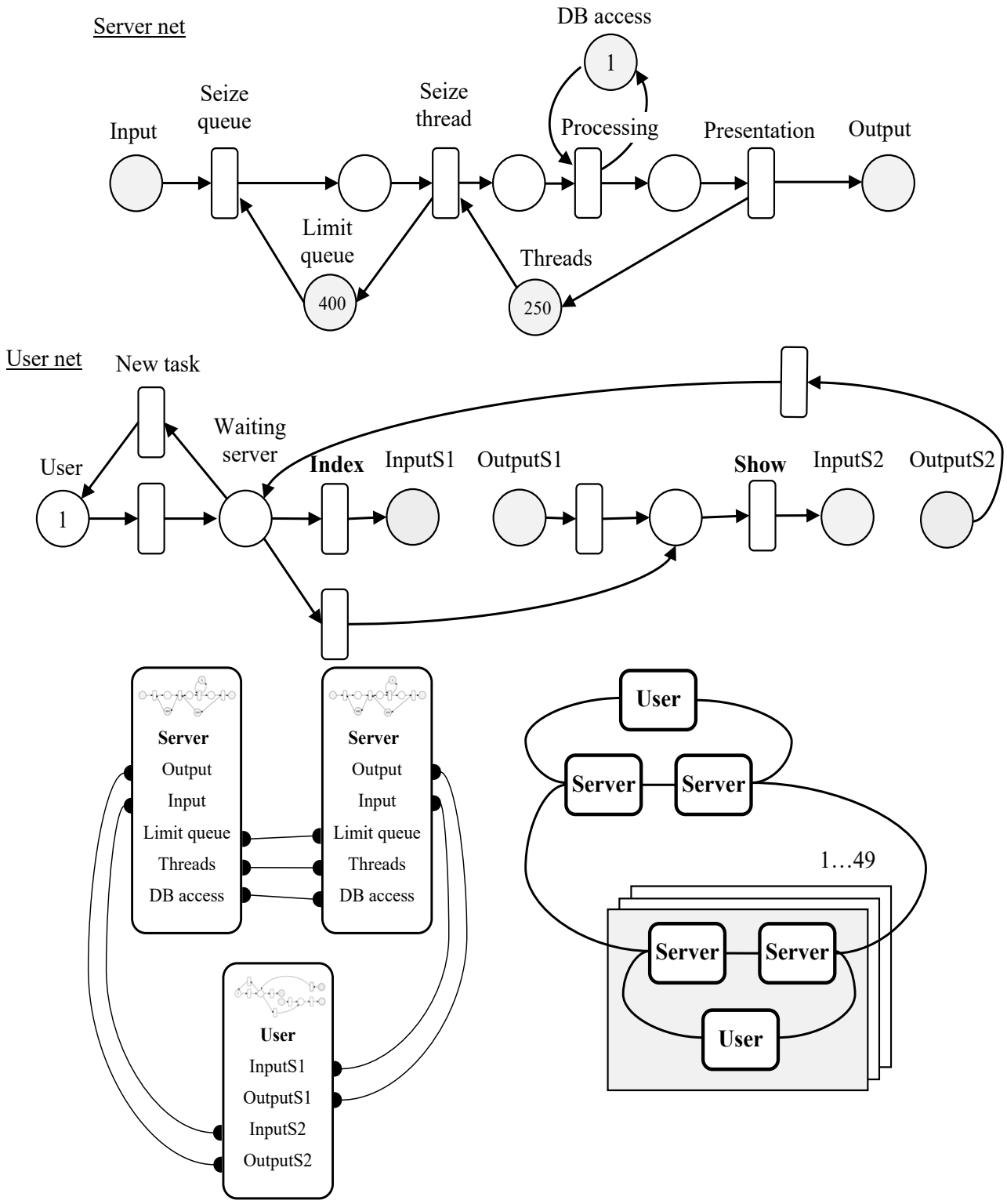


Рис. 2. Фрагмент Петрі-об'єктної моделі інформаційної системи, розроблений мовою ві-зуального програмування

стно-вільною і приведеною. Петрі-об'єктні моделі, породжені граматикою, можуть бути перетворені на обчислення, а саме запущені на виконання алгоритмом імітації. Семантика мови програмування визначається правилами перетворення мовних виразів на обчислення.

Поняття Петрі-об'єктної моделі розширено поняттями групи об'єктів, колекції об'єктів та групи колекції об'єктів, що дає змогу здійснювати тиражування об'єктів та зв'язків між об'єктами. Реалізація введених понять у мові візуального програмування уможливлює швидке створення та групу-

вання елементів моделі, динаміка яких визначається однаковим набором подій. За рахунок такого тиражування досягається також компактність візуального представлення для складних моделей.

Імітація побудованої моделі виконується алгоритмом імітаційного моделювання Петрі-об'єктної моделі.

References

1. ISO/IEC 15909-1:2004 Systems and software engineering — High-level Petri nets — Part 1: Concepts, definitions and graphical notation. [Online] – Available from: https://www.iso.org/standard/38225.html, last accessed 2020/08/28.
2. Zaitsev D.A.(2014) Paradigm of Computations on the Petri Nets, Automation and Remote Control, Vol. 75, No. 8, 1369–1383, https://doi.org/10.1134/S0005117914080025
3. Stetsenko I.V., Lytvynov V. (2020) Computer Virus Propagation Petri-Object Simulation. In: Palagin A., Anisimov A., Morozov A., Shkarlet S. (eds) Mathematical Modeling and Simulation of Systems. MODS 2019. Advances in Intelligent Systems and Computing, vol 1019, 103-112. Springer, Cham. https://doi.org/10.1007/978-3-030-25741-5_11
4. Shmeleva T.R., Stetsenko I.V. (2021) Modeling Unconditional Forwarding Decision Within Switching Lattice. In: Vorobiyenko P., Ilchenko M., Strelkovska I. (eds) Current Trends in Communication and Information Technologies. IPF 2020. Lecture Notes in Networks and Systems, vol 212, 171- 186. Springer, Cham. https://doi.org/10.1007/978-3-030-76343-5_10
5. Stetsenko I.V., Pavlov A.A., Dyfuchyna O. (2021) Parallel algorithm development and testing using Petri-object simulation. International Journal of Parallel, Emergent and Distributed Systems. Taylor & Francis. 1-16. https://doi.org/10.1080/17445760.2021.1955113
6. Dahl, O.-J., Myhrhaug, B., Nygaard, K. (1970). Simula information. Common base language. (Report).NorwegianComputingCenter.[Online] – Available from: https://bibsys-k.userservices.exlibrisgroup.com/view/delivery/47BIBSYS_UBO/12216823070002204
7. Johnson M., Zelenski J. Formal Grammars. (2012) [Online] – Available from: https://web.stanford.edu/class/archive/cs/cs143/cs143.1128/handouts/080%20Formal%20Grammars.pdf
8. Prochaska, K., Thiesing R. M. Introduction to Simio. (2016). Proceedings of the 2016 Winter Simulation Conference T. M. K. Roeder, P. I. Frazier, R. Szechtman, E. Zhou, T. Huschka, and S. E. Chick, eds.
9. Simio. The future of Simulation, Growing with you. [Online] – Available from: https://www.simio.com/about-simio/why-simio/simio-is-the-future-of-simulation-software-growing-with-you.php
10. Simul8. Visual Logic Tutorial. [Online] – Available from: https://www.simul8.com/support/help/doku.php?id=features:visual_logic:tutorial
11. CPNTools. [Online] – Available from: http://cpntools.org/, last accessed 2020/04/26.
12. Jensen, K., Kristensen L. M. (2015) Colored Petri nets: a graphical language for formal modeling and validation of concurrent systems. Communications of the ACM 58(6), 61-70. DOI: 10.1145/2663340
13. Формальні мови, граматики та автомати: Навчальний посібник/ Гавриленко С.Ю. – Харків: НТУ « ХПІ », 2021. – 133 с.
14. Формальні мови та автомати / підручник для студ спец. 124 Системний аналіз / І.Я.Спекторський, В.М.Статкевич; КПІ ім. Ігоря Сікорського. – Київ: КПІ ім. Ігоря Сікорського, 2019. – 167 с.
15. Стеценко И.В. (2011) Теоретические основы Петри-объектного моделирования систем. Математичні машини і системи, 136-148.
16. Stetsenko I.V., Dyfuchyn A. (2021) Petri-object Simulation Two Level Visual Programming Language. In: Palagin A., Anisimov A., Morozov A., Shkarlet S. (eds) Mathematical Modeling and Simulation of Systems. MODS 2020. Advances in Intelligent Systems and Computing, vol 1265. Springer, Cham. P. 266-276. (Scopus) https://doi.org/10.1007/978-3-030-58124-4_26
17. Stetsenko, I.V., Dyfuchyn, A.: Petri-object Simulation: Technique and Software. Information, Computing and Intelligent Systems 1, 51-59 (2020). https://doi.org/10.20535/2708-4930.1.2020.216057

Одержано 18.11.2021

Про авторів:

Дифучин Антон Юрійович,
аспірант 4 року навчання кафедри інформатики та програмної інженерії НТУУ "КПІ імені Ігоря Сікорського".
Кількість наукових публікацій в українських виданнях – 3.
Кількість наукових публікацій в іноземних виданнях – 3.
Індекс Хірша – 1.
https://orcid.org/0000-0002-1722-8840

Стеценко Інна Вячеславівна,
доктор технічних наук, професор, професор кафедри інформатики та програмної інженерії НТУУ "КПІ імені Ігоря Сікорського".
Кількість наукових публікацій в українських виданнях – понад 100.
Кількість наукових публікацій в іноземних виданнях – 12.
Індекс Хірша – 2.
http://orcid.org/ 0000-0002-4601-0058

Жаріков Едуард В'ячеславович,
в.о. зав. кафедри інформатики та програмної інженерії НТУУ "КПІ імені Ігоря Сікорського".
Кількість наукових публікацій в українських виданнях – понад 30.
Кількість наукових публікацій в іноземних виданнях – 29.
Індекс Хірша – 4.
http://orcid.org/ 0000-0003-1811-9336

Місце роботи авторів:

Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського»,
03056, м. Київ,
проспект Перемоги 37.
Тел.: (044) 236-9651
e-mail: difuchin@gmail.com
stiv.inna@gmail.com
zharikov.edward@gmail.com

UDC 004.93'1 http://doi.org/10.15407/pp2021.04.95

Y. Bezliudnyi, V. Shymkovych, A. Doroshenko

A MODEL OF CONVOLUTIONAL NEURAL NETWORK AND A SOFTWARE APPLICATION FOR TYPICAL INSECT PESTS RECOGNITION

A model of a convolutional neural network, a dataset for neural network training, and a software tool for the classification of typical insect pests have been developed, which allows recognizing the class of insect pests from an image. The structure of the neural network model was optimized to improve the classification results. In addition, the user interface, authentication, and authorization, data personalization, the presence of user roles and the appropriate distribution of functionality by role, the ability to view statistics on classified insects in a certain period of time were developed. Functional testing of the developed software application on a heterogeneous set of images of insects of 20 different classes was performed.
Keywords: image classification, convolutional neural networks, modular architecture, Python, Keras

Introduction

Applied entomology is a branch of biology that studies insects that damage plants, forests, crop products, and studies methods of insect pests control. Applied entomology is given considerable attention in the context of agronomy, which is quite rational, because the wrong definition of the type of insect pests on the field area, the choice of the wrong method of insect pests control or application of crop protection untimely cause a significant decline in yields and therefore loss of farmers. Classification of insects on plantations, in order to use appropriate methods of protection, as well as the collection of statistical data on the quantitative characteristics of crop damage by insect pests, is quite a difficult task, especially in a situation when different types of crops are grown on the single plantation. In addition, many species of insects migrate from one territory to another, some of the species are quite similar to each other, insects can damage plantations at several different stages of development. All of the above complicates the task of correct classification of insect pests by humans.

Artificial neural networks are an attempt to transfer a set of biological mechanisms occurring in the brain of animals to the category of computer systems [1]. Artificial neural networks can be used in facilities such as robotics [2], vehicle and unmanned aerial vehicle control, pattern recognition, analysis and decision making in IoT systems, spacecraft control, military equipment, and many other applications in modern technologies [3-5]. In these systems neural networks can be used for objects identification, prediction of the state of objects, recognition, clustering, classification, analysis of large amounts of data coming at high speed from a large number of devices and sensors, etc. [6-14]. Tasks that involve image classification are mostly based on the use of convolutional neural network technology. A convolutional neural network (CNN) is a type of multilayer perceptrons designed to minimize the amount of pre-processing of input data [15]. This type of neural network architecture has become popular as a tool for solving image recognition problems, as it has several advantages over other architectures, such as fewer learnable parameters; absence of image pixels memorization; the ability to perform part of the calculations in parallel; relative resistance to rotation and shift of images [16].

The purpose of this work is to automate the process of recognizing insect pests through the development of a new model of CNN, as well as learning database creation and the development of software application based on this model.

1. CNN model development

The CNN, in the general case, consists of 5 basic types of layers: input layers, convolutional layers, pooling layers, fully-con-

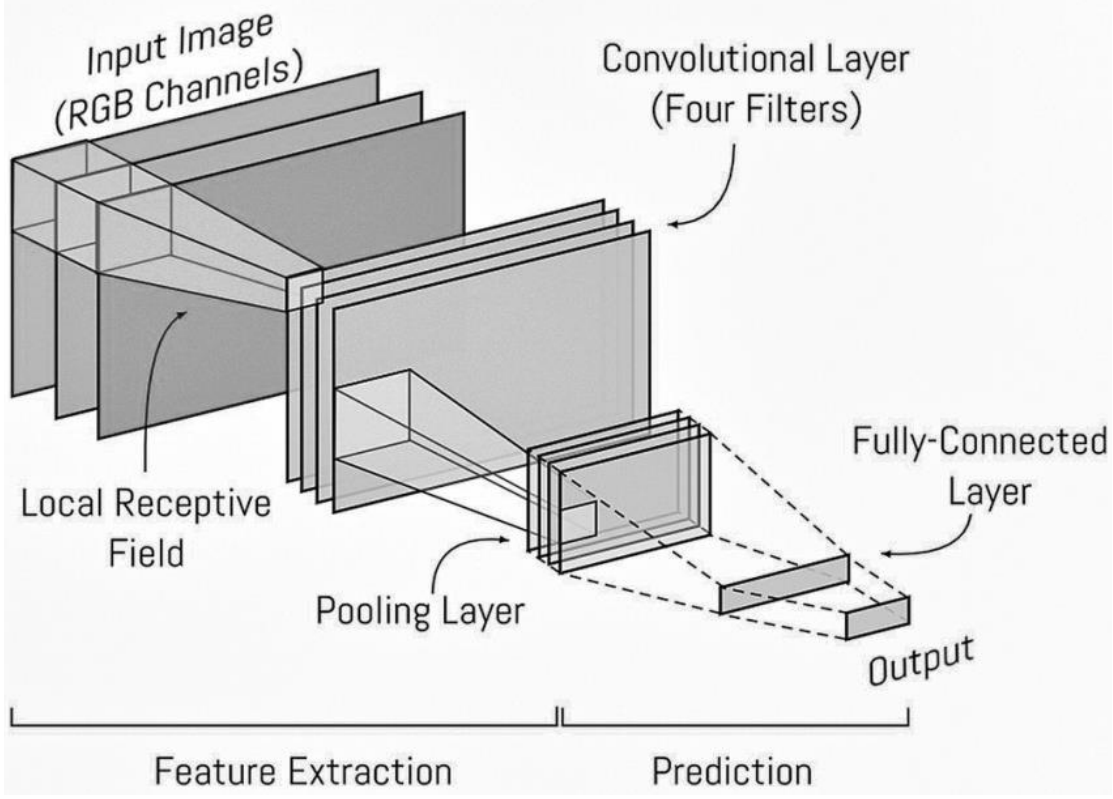


Fig. 1. The general structure of the CNN

nected layers and output layers. The general structure of the CNN is shown in Figure 1.

The input data is an image of $N \times N$ pixels and with K channels of the color model. The input layer of the CNN is a matrix of size $N \times N \times K$, each element of which is a numerical representation of the pixel regarding a particular color channel.

Convolutional layers consist of a set of feature maps, each of which has a scanning kernel (filter). The kernel is a system of scales (synapses). The kernel slides across the map of features and detects certain features of objects. During sliding on the feature map, the kernel performs a convolution operation on the map elements, the result of which is entered into the corresponding cell of the original feature map of a particular convolution layer. The convolution formula is as follows:

$$(A \circ G)[m,n] = \sum_{k,l} A[m-k,n-l] * G[k,l]$$

where A – the matrix of elements of the input feature map; G – the core of the convolution.

In this case, depending on the method of processing the edges of the output matrix, the output map of features may be less than, equal to, or greater than the input. According to the above, the convolutional layer, in a simplified form, is described by the following formula:

$$Y_{k,j} = s\left(\sum_i Y_{k-1,i} \circ G_{k,j} + B_{k,j}\right)$$

where $Y_{k,j}$ – the output of the k-th convolutional layer by the j-th feature; $B_{k,j}$ – the bias of the k-th convolutional layer by the j-th feature; $G_{k,j}$ – the kernel of the k-th convolutional layer by the j-th feature; s – activation function.

Unlike a standard neural network, each synapse in the convolutional layer of the CNN is not associated with all the synapses of the previous layer but is simply connected to nodes in a special region known as the local receptive field.

The pooling layer is usually used after the convolution layer. This type of layer was designed to simplify information and reduce the scale of feature maps made by convolutional layers. In other words, pooling layers create a compressed feature map from each feature map in the convolutional layers. Pooling layers are needed to reduce the time of computations and resolve problems with overtraining of the CNN. The pooling

operation can be performed in different ways, such as a geometric mean, harmonic mean, or maximum of passed arguments. Maximum pooling and average pooling are the two most common pooling operations.

Formally, the pooling layer can be described by the following formula:

$$Y_k = s(P(Y_{k-1}) + B_k)$$

where Y_k – the output of the k-th pooling layer; B_k – the bias of the k-th pooling layer; $P(...)$ – pooling operation; s – activation function.

Fully-connected layers are arranged after a sequence of convolutional layers and pooling layers. This part of the CNN contains a set of critical information obtained from all previous procedures performed inside of the CNN. The neurons of each map of the pooling layer (or convolutional layer, if it is the last before the fully-connected layers) are connected to one neuron of the fully connected layer.

The outputs of the last of the fully-connected layers form the output layer, which is known as the classifier. It determines the probability distribution of each label over N classes. The size of the last layer is equal to the number of classes that are classified by the neural network.

As part of this study, a number of decisions were made regarding the choice of CNN architecture, the algorithms used for input data processing as well as CNN training and interpretation of source data.

The first layer of the CNN is the input layer.

Input data must be normalized before processing by the CNN. For each specific pixel of the input image is its normalization in the range from 0 to 1 by the formula:

$$f(p, min, max) = \frac{p - min}{max - min}$$

where p – value of pixel; min – minimal value of pixel; max – maximal value of pixel. Images are passing to CNN in JPEG format with a RGB color model, which contains 3 channels with pixel values for a specific channel in the range from 0 to 255. Chosen image size is 100×100. This image size is sufficient to classify insect pests.

The input layer is followed by five convolutional layers, between which there are four pairs of normalization and pooling layers.

The size of the kernel in the convolutional layers is usually taken in the range from 3×3 to 7×7. If the size of the kernel is too small, CNN will not be able to distinguish any features; if it is too large then the number of connections between neurons increases. Also, the size of the kernel should be chosen in a way to make the size of the output maps even, which allows to not lose information when reducing the size of the data in the pooling layer described below.

The maximum pooling was chosen as the pooling operation, as it allows to better highlight sharp areas of the image, which is necessary to solve the problem posed.

In practice, the number of fully-connected layers is chosen not too big, usually no more than three. In this paper, it was chosen equal to three, which was determined by empirical method. Two dropout layers were also added between the fully-connected layers to avoid network overtraining.

The Adam algorithm [2] was chosen as an algorithm for gradient descent optimization, which was developed specifically for learning of deep neural networks. The Adam algorithm uses the power of adaptive learning methods that find individual learning metrics for each neural network parameter. Formally, this algorithm is described as follows:

1. Calculate the first and second-order moments of the gradient:

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t$$
$$v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t^2$$

where m_t – the first-order moment of the gradient with regard to model parameter θ_t; v_t – the second-order moment of the gradient with regard to model parameter θ_t; g_t – gradient value with regard to model parameter θ_t.

2. Calculate bias-corrected moments of the gradient:

$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t}$$

$$\hat{v}_t = \frac{v_t}{1 - \beta_2^t}$$

3. Update the values of model parameters:

$$\theta_{t+1} = \theta_t - \frac{\mu}{\sqrt{\hat{v}_t} + \varepsilon} \hat{m}_t$$

where μ – learning rate; ε – small constant.

The authors of the Adam algorithm suggest the use of the following coefficient values: $\beta_1 = 0.9;\ \beta_2 = 0.999;\ \varepsilon = 10^{-8}$.

The sparse categorical cross-entropy was chosen as a function of learning losses [17]. Cross-entropy losses determine the efficiency of the classification model, the output of which is the probability value in the range from 0 to 1. These losses increase when the predicted probability deviates from the actual label. The formula for cross-entropic losses is defined as:

$$Loss = -\sum_{c=1}^{M} y_{o,c} \ln(p_{o,c}),$$

where M – number of classes; y – binary indicator (0 or 1) that shows whether the label c is the correct classification for observation o; $p_{o,c}$ – the assumed probability that the observation o belongs to the class c.

The only difference between sparse categorical cross-entropy and categorical cross-entropy is the label format.

Each layer of the neural network has inputs with a corresponding probability distribution, which in the learning process is influenced by the randomness of the initialization of parameters and randomness in the input data. The influence of these sources of randomness on the probabilistic distribution of the values of the input data on the inner layers during training is described as the internal covariate shift. During the network learning phase, as the parameters of the previous levels change, the probability distribution of the input data to the current level changes accordingly, so that the current level requires constant readjustment to the new distributions. This problem is especially serious for deep networks, as small changes in the hidden layers closer to the input layer will be amplified when they propagate within the network, which will lead to a significant shift in the more distant hidden layers. Therefore, a method of batch normalization [18] has been proposed to reduce these undesirable changes, speed up learning and make the CNN more reliable. Formally, the batch normalization algorithm is defined as follows:

1. The mean μ_B and the variance σ_B^2 of a batch of m elements are computed:

$$\mu_B = \frac{1}{m}\sum_{i=1}^{m} x_i$$

$$\sigma_B^2 = \frac{1}{m}\sum_{i=1}^{m} (x_i - \mu_B)^2$$

2. Normalization of layer inputs is performed:

$$\overline{x_i} = \frac{x_i - \mu_B}{\sqrt{\sigma_B^2 + \varepsilon}}$$

where x_i – value of the i-th element of normalization input layer; $\overline{x_i}$ – normalized value of the i-th element of normalization input layer;

ε – small constant.

To restore the representation power of the network transformation of normalized values performed:

$$y_i = \gamma x_i + \beta$$

where $\mathbf{y}_i$ – value of the i-th element of normalization layer output; γ, β – parameters that are subsequently learned in the optimization process.

In addition to reducing the internal covariate shift, group normalization provides many other benefits. With using batch normalization technique, the network can use a higher learning rate without problem of gradients values vanishing or exploding.

2. Creating a learning dataset for CNN

The first step in creating a dataset for CNN learning was to determine the classes of insects that will be recognized and the minimum number of image examples in each class. As result, 20 most typical classes of insect pests were selected. The requirement to have at least 40 different images of insects of each class was set. In order to reduce the possibility of the CNN overtraining on the spatial location of insects in the images, for each original image in the database was added 3 more copies with different angles of rotation relative to the original (90°, 180°, 270°). After that images of each class were divided into training and test samples, in a ratio of 2:1.

Note that if the image size is too large, the computational complexity will increase, respectively, the restrictions on the speed of response will be violated. If too small images will be chosen, then the network will not be

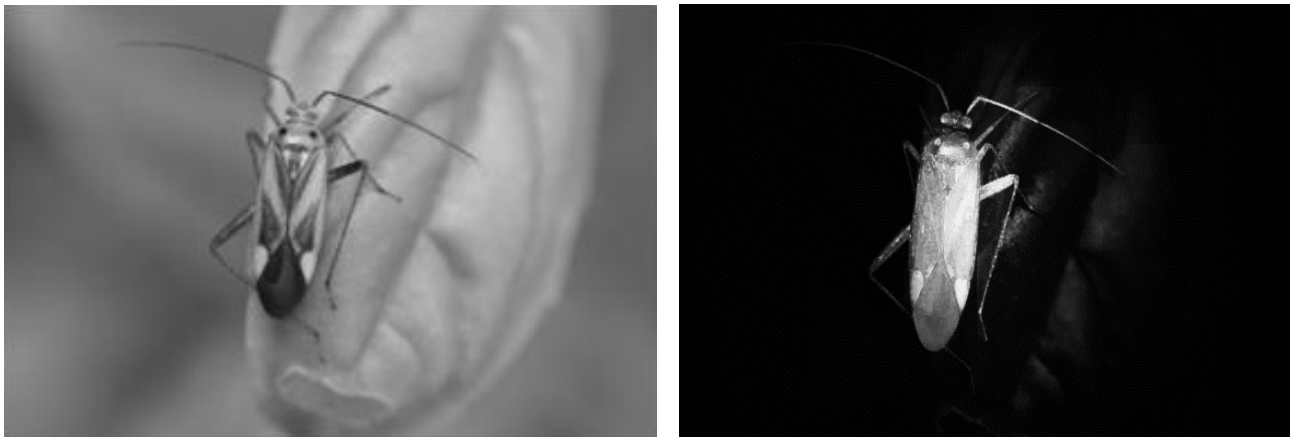


Fig. 2. Image of an insect (on the left) and its saliency map (on the right)

able to detect the key features of the objects depicted on them. In our case, the optimal image size was 100 × 100 pixels.

The dataset was partially populated with images from the IP102 dataset, which contains images of 102 classes of insects, with a total size of 75,000 images. However, most of the insect classes in this set not belong to the category of pests, so the rest of the classes were filled with images obtained through the use of Flickr API. Since the developed CNN is designed to recognize insects in images, the background information is irrelevant. Therefore, in order to improve the accuracy of classification, before scaling and expanding the number of images, an operation was performed to remove excess parts of the image using saliency maps [19]. The purpose of the saliency map is to display the degree of "informativeness" of each pixel, which is respectively greater for the pixels belonging to the part of the image where the insect is located, and less for the background pixels. An example of a saliency map is shown in Figure 2. During the learning dataset preparation, based on the saliency map obtained from the image, the excess parts of the images were cropped. The cropping process started from the image edges until the total "informativeness" of the pixels at the edge of an image passes the specified threshold. Testing on the selected set of images shows that the use of saliency maps allows to increase the insect to background ratio by an average of 30%.

In total, the created dataset contains 3000 images, which are arranged hierarchically by directories, according to their class and purpose (for training or testing).

3. Software application development

The developed application for insect pests recognition consists of three modules:

1. The graphical user interface module;
2. The module of customer accounting and service;
3. The module of image classification and processing.

The purpose of the first module is to provide a full graphical user interface consisting of a home page, pages for registration and login, a page for manual classification, a page for downloading new images of insects (for system administrators), and a page for viewing personal statistics. This module was implemented on the basis of Angular framework, using TypeScript language.

The module of customer accounting and service deals with such tasks as customer-related information processing and customer authorization and authentication. To implement this module, it was decided to choose Java programming language and frameworks such as Spring Security, Spring Boot, Spring Data.

The module of image classification and processing is the core of the developed application. This module consists of two main layers: the layer of classification and the layer of image processing and loading. The classification layer uses a trained CNN model to classify insects in the images that come with the request (images are pre-processed using saliency maps and scaled). The layer of image processing provides functionality that allows making such image transformations as rotation, scaling, highlighting the main part (using saliency maps), and so on. This module was implemented in Python

programming language. Tensorflow and Keras libraries were used to work with neural networks. OpenCV and Pillow libraries were chosen for imageprocessing. Numpy library was used to work with multidimensional arrays.

CNN implementation code in Python:

```
def get_model(image_size, classes_
num):
model = keras.Sequential([
keras.Input(image_size),  # Input layer
        keras.layers.experimental.preprocess-
ing.Rescaling(scale=1/255.0),  # Image scaling
        keras.layers.Conv2D(filters=16,
kernel_size=(7, 7), activation='relu',
padding='same'),  # Convolutional layer
        keras.layers.BatchNormalization(),  #
Normalization layer
        keras.layers.MaxPooling2D(pool_
size=(2, 2), strides=2),  # Pooling layer
        keras.layers.Conv2D(filters=32,
kernel_size=(5, 5), activation='relu',
padding='same'),
        keras.layers.BatchNormalization(),
        keras.layers.MaxPooling2D(pool_
size=(2, 2), strides=2),
        keras.layers.Conv2D(filters=32,
kernel_size=(3, 3), activation='relu',
padding='same'),
        keras.layers.BatchNormalization(),
        keras.layers.MaxPooling2D(pool_
size=(2, 2), strides=2),
        keras.layers.Conv2D(filters=64,
kernel_size=(3, 3), activation='relu',
padding='same'),
        keras.layers.Flatten(),  # Layer dimen-
sionality reduction
        keras.layers.Dense(units=2048,
activation='relu'),  # Fully-connected layer
        keras.layers.Dropout(rate=0.5),  #
Dropout layer
        keras.layers.Dense(units=2048,
activation='relu'),
        keras.layers.Dropout(rate=0.5),
        keras.layers.Dense(classes_num,
activation='softmax')  # Output layer
    ])
        return model
```

CNN training code in Python:

```
        def train_model(
        dataset_path=CURR_DATASET_
PATH,  # Path to the dataset root folder
        write_path=CURR_MODEL_PATH,  #
Path to the pretrained CNN model
    lr=0.0001,  # Learning rate
    epochs_num=15,  # Number of learning ep-
ochs
    image_size=(100, 100, 3),  # Image size
classes_num=20  # Number of classified classes
):
        training_data = keras.preprocessing.
image_dataset_from_directory(dataset_
path + '/train', batch_size=32, image_
size=image_size[:2])
    test_data = keras.preprocessing.image_da-
taset_from_directory(dataset_path + '/test',
batch_size=32, image_size=image_size[:2])
    if (os.path.exists(write_path)): shutil.
rmtree(write_path)
        model = get_model( image_
size=image_size, classes_num=classes_num)
        model.compile( optimizer=keras.opti-
mizers.Adam(lr=lr),
        loss=keras.losses.SparseCategorical-
Crossentropy(),
        metrics=['accuracy'])
    model.fit(training_data, epochs=epochs_
num, verbose=2)
    model.evaluate(test_data, verbose=2)
    model.save(write_path)
```

Optimal values of parameters such as kernel size and number of filters for convolutional layers, number of neurons for fully-connected layers, dropout value and learning rate are based on general recommendations for CNN construction and training described in section 2, as well as on values got using the Keras Tuner library, which provides ready-made solutions for neural network parameter optimization problems.

In addition to the modules described above, the structure of the software solution also includes the PostgreSQL database, which is accessed by the module of accounting and customer service.

4. Software application testing

PC characteristics on which the application was tested:

1. Processor clock speed – 2.6 GHz;
2. 6 physical cores, 12 logical processors;
3. 16 GB of RAM.

4. GPU integration absent.

The following technical requirements were set for the development system:

Accuracy of image classification –

not less than 95% on the training dataset, and not less than 60% on the test dataset;

2. The number of epochs of learning the neural network – 15;

3. The average image classification time – less than 0.5 s;

4. The average download time of the image from an external service – less than 0.5 s.

The learning results of the developed CNN model are follows:

1. Accuracy of image classification on the training set – 96.6%;

2. Accuracy of image classification on the test set – 65.4%;

3. The average training time for the era – 24 seconds.

The results of time testing for different request:

1. Classification of the image by the trained CNN – 115 ms;

2. Adding a new image to the training set (excluding the execution time of asynchronous processes) – 39 ms;

3. Downloading an image from an external service – 434 ms;

4. Obtaining classification statistics – 5 ms.

Testing results of the application are satisfying the technical requirements.

Conclusion

The model of the CNN and the training dataset have been developed to solve the problem of classification of typical insect pests. A software application for solving the problem of typical insect pests recognition by the transmitted image has been implemented, which allows to automate preventive protection against insect pests in case of integration with IoT-devices on fields. The implemented application satisfies the technical requirements for speed and quality of classification.

Further improvement of the quality of the CNN classification can be done with the help of such methods: GPU calculation acceleration; parallelization of data processing; construction of structurally more complex architectures of the CNN, etc.

References

1. *Dreyfus, G.* (2005) Neural networks: methodology and applications. Springer-Verlag, Berlin. 498 p. https://doi.org/10.1007/3-540-28847-3
2. *Doan, Q. V., Le, T. D., Le, Q. D., & Kang, H.-J.* (2018) A neural network–based synchronized computed torque controller for three degree-of-freedom planar parallel manipulators with uncertainties compensation. International Journal of Advanced Robotic Systems. 15(2). pp. 1-13. https://doi.org/10.1177/1729881418767307
3. *Shymkovych, V., Telenyk, S., Kravets, P.* (2021) Hardware implementation of radial-basis neural networks with Gaussian activation functions on FPGA. Neural Computing and Applications. 33. pp. 9467-9479. https://doi.org/10.1007/s00521-021-05706-3
4. *Melchert, F., Bani, G., Seiffert, U., Biehl, M.* (2020) Adaptive basis functions for prototype-based classification of functional data. Neural Computing and Applications. 32. pp. 18213-18223. https://doi.org/10.1007/s00521-019-04299-2
5. *Goncalves, S., Cortez, P., Moro, S.* (2020) A deep learning classifier for sentence classification in biomedical and computer science abstracts. Neural Computing and Applications. 32. pp. 6793-6807. https://doi.org/10.1007/s00521-019-04334-2
6. *Kravets, P., Shymkovych, V.* (2020) Hardware Implementation Neural Network Controller on FPGA for Stability Ball on the Platform. In: Hu Z., Petoukhov S., Dychka I., He M. (eds) Advances in Computer Science for Engineering and Education II. ICCSEEA 2019. Advances in Intelligent Systems and Computing. 938. pp. 247-256. https://doi.org/10.1007/978-3-030-16621-2_23
7. *P., Kravets, V., Nevolko, V., Shymkovych, L., Shymkovych* (2020) Synthesis of High-Speed Neuro-Fuzzy-ControllersBased on FPGA. 2020 IEEE 2nd International Conference on Advanced Trends in Information Theory (ATIT). pp. 291-295. https://doi.org/10.1109/ATIT50783.2020.9349299.
8. *Passalis, N., Tefas, A.* (2020) Continuous drone control using deepreinforcement learning for frontal view person shooting. Neural Computing and Applications. 32. pp. 4227-4238. https://doi.org/10.1007/s00521-019-04330-6
9. *Zhao, Z., Zheng, P., Xu, S., Wu, X.* (2019) Object detection with deep learning: a review. IEEE Transactions on Neural Networks and Learning Systems. 30. pp. 3212–3232. https://doi.org/10.1109/TNNLS.2018.2876865
10. *Kravets, P.I., Shimkovich, V.N., Ferens, D.A.* (2015) Method and algorithms of implementa-

tion on PLIS the activation function for artificial neuron chains. Elektronnoe Modelirovanie. 37(4). pp. 63-74. (in Russian)
11. *Hong, Q., Li, Y., Wang, X.* (2020) Memristive continuous Hopfield neural network circuit for image restoration. Neural Computing and Applications. 32. pp. 8175-8185. https://doi.org/10.1007/s00521-019-04305-7
12. *V., Shymkovych, V., Samotyy, S., Telenyk, P., Kravets, T., Posvistak* (2018) A real time control system for balancing a ball on a platform with FPGA parallel implementation. Technical Transactions. Poland. 5. pp. 109-117. https://doi.org/10.4467/2353737XCT.18.077.8559
13. *Shao, L., Zhu, F., Li, X.* (2015) Transfer learning for visual categorization: a survey. IEEE Transactions on Neural Networks and Learning Systems. 26. pp. 1019-1034. https://doi.org/10.1109/TNNLS.2014.2330900
14. *P.I., Kravets, T.I., Lukina, V.M., Shymkovych, I.I., Tkach* (2012) Development and research the technology of evaluation neural network models MIMO-objects of control. Visnyk NTUU "KPI" Informatics operation and computer systems. 57. pp. 144-149. (in Ukrainian)
15. *Y., LeCun, B., Boser, J.S., Denker, D., Henderson, R.E., Howard, W., Hubbard, L.D., Jackel* (1989) Backpropagation applied to handwritten zip code recognition. Neural computation. 1. pp. 541-551.
16. *Asifullah Khan, Anabia Sohail, Umme Zahoora, Aqsa Saeed Qureshi* (2020) A survey of the recent architectures of deep convolutional neural networks. Artificial Intelligence Review. 53. pp. 5455-5516. https://doi.org/10.1007/s10462-020-09825-6
17. *Kingma, D.P., Ba, J.* (2014) Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980.
18. *H., Yessou, G., Sumbul, B., Demir* (2020) A Comparative Study of Deep Learning Loss Functions for Multi-Label Remote Sensing Image Classification. IGARSS 2020 - 2020 IEEE International Geoscience and Remote Sensing Symposium. pp. 1349-1352. https://doi.org/10.1109/IGARSS39084.2020.9323583
19. *S., Ioffe, C., Szegedy* (2015) Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. Proceedings of the 32nd International Conference on Machine Learning, PMLR. 37. pp. 448-456
20. *T.N., Mundhenk, B.Y., Chen, G., Friedland* (2020) Efficient Saliency Maps for Explainable AI. European Conference on Computer Vision (ECCV). https://arxiv.org/abs/1911.11293

Received: 17.11.2021

About authors:

Bezliudnyi Yurii Serhiyovych,
5th year student of the National Technical University of Ukraine "Igor Sikorsky Kyiv Polytechnic Institute".
https://orcid.org/0000-0003-1950-9790

Shymkovych Volodymyr Mykolayovych,
Candidate of Technical Sciences,
Associate Professor of the Department of Information Systems and Technologies of the National Technical University of Ukraine "Igor Sikorsky Kyiv Polytechnic Institute". Number of scientific publications in Ukrainian publications – more than 30.
Number of scientific publications in foreign publications – more than 10.
Hirsch index – 3.
https://orcid.org/0000-0003-4014-2786

Doroshenko Anatoliy Yukhymovych,
Doctor of Physical and Mathematical Sciences, Professor.
Head of the Department of Computation Theory, Professor of the Department of Information Systems and Technologies of the National Technical University of Ukraine "Igor Sikorsky Kyiv Polytechnic Institute". Number of scientific publications in Ukrainian publications - more than 190.
Number of scientific publications in foreign publications - more than 80.
Hirsch index - 6.
http://orcid.org/0000-0002-8435-1451

Affiliation:

National Technical University of Ukraine "Igor Sikorsky Kyiv Polytechnic Institute", 37 Peremohy Avenue

Institute of Software Systems of the National Academy of Sciences of Ukraine, 03187, Kyiv-187, 40 Akademika Glushkova Avenue.
Phone: (044) 526 3559
E-mail:
uraura05052000@gmail.com,
v.shymkovych@kpi.ua,
doroshenkoanatoliy2@gmail.com

УДК 94::004:004.[235+43+655.3] https://doi.org/10.15407/pp2021.04.103

Ю.О. Ющенко

РОЗРОБКА АРХІТЕКТУРИ КОМП'ЮТЕРА «КИЇВ» ЗА КОНЦЕПЦІЄЮ АДРЕСНОГО МЕТОДУ ПРОГАМУВАННЯ

Описано поступовий перехід від розрахунків логарифмічними лінійками та арифмометрами до використання мови високого рівня з вказівниками та складними ієрархічними структурами. В статті розглянуто фактори, які сприяли цьому важливому технологічному переходу. Робота відновлює загублену ланку історії виникнення в Україні опосередкованої адресації (вказівників) вищих рангів та складних ієрархічних структур. Надаються підтвердження винайдення вказівників та складних структур даних київськими вченими, описано апаратну реалізацію в комп'ютері «Київ» «штрих-операції» (розіменування вказівників) та операцій задання циклів. У роботі обґрунтовується високорівневість програмування у командах комп'ютера «Київ» шляхом порівняння з мовою програмування високого рівня – Plankalkül.

Ключові слова: історія, штрих-операція, опосередкована адресація, вказівники, програмування, деревоподібні формати, масиви, списки, структури, абстрактні типи даних.

Вступ

Наразі спостерігається підвищення цікавості людей до історії інформаційних технологій (ІТ), що обумовлено лавиноподібним зростанням значущості ІТ у всіх сферах науки, виробництва, економіки, політики та взагалі у всіх видах діяльності та дозвілля людей. У світі особлива увага приділяється зародженню інформаційних технологій.

Зародженню перших комп'ютерів та виникнення програмування присвячено багато публікацій. Однак, на думку автора, втрачена важлива ланка в історії ІТ, що б мала пояснити перехід від низькорівневого програмування до високорівневого. Стаття, на основі наукових публікацій [2-5, 7, 11, 10, 14, 20, 21, 26, 29, 39-42], спогадів очевидців [12, 13, 15, 18, 19, 22, 23, 30, 31, 38, 46], історичних публікацій [9, 24, 32-34] та спогадів К.Л. Ющенко, відновлює цю загублену ланку.

Мета статті відновити загублену в історії ланку зародження в Україні програмування високого рівня з потужними засобами, які використовуються в усіх сучасних технологіях програмування. Метою статті є визначення факторів, які сприяли винайденню високорівневого Адресного програмування з потужними можливостями адресації.

На зародження в Україні високорівневого програмування вплинуло використання інструкцій для розрахунків арифмометрами та досвід програмування на комп'ютері МЕСМ[1] [11 (стор. 36: Т. 2), 18, 43] з обмеженими ресурсами. Під час розробки асинхронного комп'ютера «Київ» важливу роль відіграли висококваліфіковані інженери, які розроблювали МЕСМ [15].

Виникненню програмування високого рівня сприяв операторний метод програмування [11], до якого О.А. Ляпунов прийшов під час досліджень програмування на МЕСМ у Феофанії влітку 1952р. [44]. 1953 року в наукових статтях, семінарах та лекціях О.А. Ляпунов оприлюднює операторний метод програмування [9, 11, 18, 21, 22, 29, 43].

Можливість динамічної модернізації програм МЕСМ наштовхнула на ідею створення однією програмою інших програм, тобто програмуючих програм, які згодом було названо компіляторами та трансляторами.

1. Діяльність Б.В. Гнєденка

Б.В. Гнєденко 1949 року, на семінарі із розробки МЕСМ, порадив С.О. Лебедєву збільшити розрядність комірок пам'яті з 12 до 17 [13, 18, 23, 24, 30]. Без цього збіль-

[1] Російською "МЭСМ" (малая электронная счетная машина).

шення комп'ютер не був би придатний для розрахунків.

Б.В. Гнєденко запропонував долучитися до програмування В.С. Королюку, який під час навчання в аспірантурі у А.М. Колмогорова, відвідував московські наукові семінари із програмування та слухав лекції О.А. Ляпунова [13, 18].

Після повернення з НДР Б.В. Гнєденко очолює лабораторію обчислювальної техніки, яка утворюється шляхом об'єднання обчислювальної лабораторії ІМ УН УРСР (керівник К.Л. Ющенко) [31] з лабораторією моделювання та обчислювальної техніки Інституту електродинаміки (керівник С.О Лебедєв).

В.С. Королюк разом із К.Л. Ющенко з 1955/56р. навчального року починають читати лекції у КДУ та КПІ по операторному та адресному методах програмування, організованих Б.В. Гнєденком [13, 23, 46]. 1957 року К.Л. Ющенко та В.С. Королюк опублікували у КДУ обмеженим накладом перший у СРСР підручник із програмування з описом операторного та адресного методів програмування [46 (стор. 36)]. Цей підручник побачив світ 1961р. та перекладався й видавався у європейських країнах: Угорщині [5], Чехословаччині, східній Німеччині [4], Югославії та Данії. 1963р. додатки до цього підручника з описом архітектури п'яти радянських комп'ютерів, включаючи комп'ютер «Київ», перекладаються англійською мовою та публікуються у США 1963р. [7]. А 1969р., після 12 років від часу написання, видають французькою у Франції [3]. У підручнику описується концепція опосередкованої адресації (Pointers) вищих рангів, відношення слідування, а у додатку з описом архітектури комп'ютера «Київ» детально описано операції модифікації адрес із можливістю виконання «штрих-операції» (розіменування Pointers) та операцій задання циклів.

Очолюючи лабораторію обчислювальної техніки, Б.В. Гнєденко організовує наукові семінари із проєктування архітектури великого універсального асинхронного керуючого комп'ютера «Київ» та автоматизації програмування. Як науковий керівник розробки комп'ютера заглиблювався в деталі архітектури та тонкощі програмування [13, 23, 24, 46].

2. Актуальність

У світі приділяється велика увага історії зародження програмування, створено багато музеїв історії інформаційних технологій, існує багато публікацій, знято низку документальних та художніх фільмів.

На підставі детального дослідження та аналізу доступних джерел автор дійшов висновку, що важлива ланка переходу від програмування низького рівня до програмування високого рівня [9, 11, 12, 18, 21, 22, 30, 38] не вповні описана в історії ІТ. Лише частково цей перехід заповнює операторний метод програмування, який по праву вважається передумовою зародження програмування високого рівня у СРСР.

В Україні перехід від програмування низького рівня до високорівневого програмування відбувався одночасно з включенням в нього опосередкованої адресації (Pointers) вищих рангів та універсальних засобів групування даних і підпрограм у складні ієрархічні структури (деревоподібні формати[2]). Основний науковий результат цієї статті полягає у визначенні передумов та причинно-наслідкових зв'язків, які сприяли та привели до винаходу киянами потужних засобів програмування, без яких не обходиться жодна сучасна технологія програмування. Однак історики інформаційних технологій ніколи не зазначають про цей значний внесок українців у зародження та розвиток програмування.

Історія та передумови виникнення найпотужніших засобів у програмуванні дуже цікава програмістам, але досі залишалася не розкритою.

Важливість статті полягає у детальному аналізі передумов поступового зародження в Україні програмування високого рівня з вказівниками та універсальними можливостями групування та з'єднання даних і підпрограм у деревоподібні формати (складні ієрархічні структури), яким подібні абстрактні типи даних.

[2] В оригінальних працях [40] використовується термін «деревообразные форматы» (рос.).

Чомусь, ані державні структури, ані відомі ІТ-компанії не виявляють зацікавлення у визнанні світовою спільнотою важливих досягнень українців в ІТ. Українські ЗМІ, музеї та інші державні та недержавні установи мають приділяти гідну увагу внеску українських вчених у розвиток світових ІТ. Україна має донести, як до своїх співвітчизників, так і до міжнародної спільноти, інформацію про всесвітньо значущі досягнення українців.

Світова спільнота докладає багато зусиль для збереження історії зародження та розвитку ІТ. Визнання всім світом суттєвого внеску українців у зародження потужних засобів програмування має велике значення для збереження всесвітньої історії ІТ та дуже важливе для національно патріотичного виховання молоді.

3. Публікації по Адресному програмуванню

Існує багато підтверджень того, що саме в Феофанії у 50-х роках минулого сторіччя були винайдені фундаментальні засоби універсального групування, з'єднання даних та підпрограм у складні ієрархічні структури. Зазначені підтвердження наявні у публікаціях із Адресної мови програмування, із архітектури комп'ютера «Київ» та в інших джерелах, зокрема, в [11, 20]. Підручник із програмування, який публікувався багатьма мовами, містить опис опосередкованої адресації (вказівників) вищих рангів. У додатках до підручника містяться підтвердження апаратної реалізації в комп'ютері «Київ» операцій маніпуляцій з адресами 2-ого рангу (вказівниками) та задавання циклів. Окрім підручника з додатками вичерпні та беззаперечні підтвердження винаходу українцями вказівників опубліковані 1966р. у США [2] в перекладеній англійською мовою монографії, присвяченій комп'ютеру «Київ» з описом Адресної мови програмування [10].

Однак, у закордонних джерелах з історії програмування не згадуються випередження українцями світових досягнень на багато років. Українським програмістам також мало відомо про Адресну мову, а про те, що в ній були вказівники, складні ієрархічні структури та декларативні можливості [1-5, 11, 10, 20, 39-42, 45] вони не вірять.

Головні підтвердження винаходу українськими вченими вказівників та складних ієрархічних структур наявні в архітектурі асинхронного комп'ютера «Київ». Так, його Ф-операція може виконувати «штрих-операції» (розіменування Pointers), а групові операції дозволяють задавати цикли [9 (Т. 1 стор. 489)]. При цьому мови програмування ФОРТРАН, КОБОЛ та АЛГОЛ-60, які помилково вважаються першими мовами програмування високого рівня, з'являються декількома роками пізніше за Адресну мову.

Опублікований у США 1963р. опис архітектури комп'ютера «Київ» [7] свідчить, що американським ученим у 1964р., на час винаходу ними вказівників (Pointers), вже було відомо про винахід киянами опосередкованої адресації (Pointers). За таких обставин можна припустити, що американські вчені досі не звернули увагу на доступні їм та наявні у США матеріали [2, 7] з Адресної мови програмування та комп'ютеру «Київ».

4. Причини невідомості

Розглянемо причини, які пояснюють, чому випередження українців невідомі за кордоном та маловідомі в самій Україні.

Перша причина полягає у тому, що до 1956р. розв'язувалися задачі для оборонної космічної галузі, включаючи розрахунки для проєктування водневої бомби, наслідків її вибуху, розрахунки траєкторій польотів дальніх балістичних та космічних ракет [15, 18, 30, 31]. Ці роботи були під грифом суворої державної таємниці, що було перепоною в оприлюдненні будь-яких матеріалів.

Друга причина криється у завантаженості МЕСМ. Загальновідомо, що у 1952/53 роках МЕСМ був фактично єдиним у СРСР комп'ютером, який регулярно використовувався для розв'язку задач [15 (стор. 53)]. Перелік цих задач опубліковано у джерелах [18, 30, 31, 43].

Третя причина полягає у недооціненні значущості програмування у часи зародження інформаційних технологій. Законів щодо захисту прав на інтелектуальну

власність у Радянському союзі не існувало. Вироби інженерів є матеріалізованими, їх можна побачити та доторкнутися до них, а програму ні побачити, ні відчути на дотик нема як. Оплата праці програмістів була значно меншою від зарплатні інженерів.

Четверта причина криється у тому, що за тоталітарного режиму не надто вірили, що у «провінційному» Києві, без фінансування з центру, Москви, можливі якісь суттєві досягнення. В СРСР увагу та розголос набували лише ті досягнення, які керівникам від науки вдавалося заявляти за свої [13, 24, 32].

П'ята причина полягає у тому, що тоталітарним комуністичним режимом було прийнято рішення переходу на мови ФОРТРАН, КОБОЛ та АЛГОЛ-60 та забороні подальшої роботи над Адресною мовою, яка була значно потужнішою за ці мови та мала очевидні переваги над ними. Було також заборонено публікувати та виступати на семінарах чи конференціях. Розпочаті роботи із реалізації компілятора з Адресної мови програмування для БЕСМ-6 було припинено [32, 44]. В останній публікації з Адресної мови [26] автори намагаються відстояти Адресну мову та демонструють її переваги над мовами ФОРТРАН, КОБОЛ та АЛГОЛ-60, але «вирок» було винесено. Однак, саме тоді перекладається та видається підручник з Адресної мови [14] у багатьох країнах східної Європи [4, 5] та у Франції [3], а 1966р. монографія [10] з описом архітектури комп'ютера «Київ» та Адресної мови публікується англійською у США [2].

Рано чи пізно людству стануть відомі видатні винаходи українців. Визнання значного внеску українців у технології програмування стримується певними обставинами. Багато українських програмістів не вірять або не розуміють, що вказівники є повним аналогом опосередкованої адресації 2-ого рангу. Обмаль публікацій з деревоподібних форматів та їх застосування. Закордонні програмісти повідомлення про винаходи в Україні вказівників 1955р. відносять до «фейкових» та вважають, що Україна мала достатньо часу для визнання своєї першості у таких важливих для людства винаходах.

5. Передумови зародження Адресного програмування

Вельми часто у джерелах зустрічається твердження, що перші комп'ютери розроблювались інженерами без участі математиків. Однак це не відповідає дійсності, бо 1948 року С.О. Лебедєв до семінарів по проєктуванню комп'ютера МЕСМ, зокрема, для визначення операцій, залучав видатних математиків: А.О. Дородніцина, К.А. Семендяєва, М.О. Лаврентьєва, Б.В. Гнєденка, О.Ю. Ішлінського, О.О. Харкевича та ін. [13, 18, 24]. Також у розробці комп'ютера «Київ», як і інших київських комп'ютерів, безпосередньо брали участь математики. Керівником розробки комп'ютера «Київ» був математик Б.В. Гнєденко. Інженер Л.Н. Дашевський керував розробкою інженерної складової. Розробкою архітектури та математичного (програмного) забезпечення опікувалася К.Л. Ющенко.

Розробка математичних вимог до комп'ютера «Київ» відбувалася паралельно з роботою над створенням та розвитком Адресної мови [10 (стор. 53)].

Отже, розвиток технічної бази та програмного забезпечення в Україні відбувався нерозривно один від одного. З одного боку архітектура та можливості реалізації операцій впливали на можливості та методи програмування, а з іншого – на вибір операцій вплинули особливості Адресної мови програмування. Система операцій розроблювалось з урахуванням потреб зручності програмуванні та спрямовувалась на легкість складання та сприйняття програм. До системи команд включались «скорочення» (мнемонічні коди), які застосовувались програмістами на МЕСМ [43].

Автоматизація програмування у Києві відбувалась незалежно та вельми відірвано від світового розвитку ІТ [20]. Є можливість чітко визначити особливості поступового переходу київськими математиками від розрахунків за інструкціями логарифмічними лінійками та арифмометрами до автоматичних розрахунків програм мовою програмування високого рівня з опосередкованою адресацією (вказівниками) та деревоподібними форматами. Вирішальну роль у зазначеному переході віді-

грає унікальна архітектура МЕСМ, яка наявністю динамічної зміни програм підказала можливість створення програмою іншої програми (компіляторів та трансляторів) [11]. Без наявності у Феофанії МЕСМ нічого б не могло бути винайдено.

Чи могло програмування високого рівня зародитися ще до появи комп'ютера? Саме так відбувалося в Києві.

Неможливо визначити дату виникнення програмування високого рівня, оскільки зародження його відбувалось поступово та почалося ще до використання комп'ютера в інструкціях для розрахунків логарифмічними лінійками та арифмометрами.

6. Інструкції розрахунків арифмометрами

Розповсюджена думка, що на зміну програмуванню у двійкових машинних командах одразу прийшло програмування високого рівня, є хибною. Досвід складання інструкцій для проведення розрахунків з використанням арифмометрів, логарифмічних лінійок та рахівниць можна вважати першим кроком до зародження програмування високого рівня.

У 1950 році для розв'язку навігаційних задач за допомогою математичної теорії гіроскопічних систем О.Ю. Ішлінського, ним, директором математичного інституту АН УРСР, була створена лабораторія[3] методів обчислень та розрахунків. Керівником лабораторії було призначено К.Л. Рвачову (після одруження - Ющенко) [31].

Математичний інститут задля полегшення проведення розрахунків придбав електронні лічильні арифмометри RheinMetal, які розмістили у підвальному приміщені президії АН УРСР [30, 31]. Окрім електронних арифмометрів, використовувались і механічні арифмометри, логарифмічні лінійки, а для фіксації проміжних результатів і звичайні рахівниці. Також, під рукою, були таблиці значень елементарних функцій.

Математики обчислювальної лабораторії умовно розділялись на дві групи. Представники однієї групи «методів обчислень» за завданням видатних математиків, згідно описаних ними загальних принципів розв'язку задач, формулювали завдання математикам-обчислювачам з іншої групи у вигляді **інструкцій** з детальним описом дій. Ці інструкції писались природною мовою з використанням формальних математичних позначок. Зокрема, інструкції містили формули, розгалуження з перевіркою умов, елементарні функції та математичні позначки: $\forall$, $\exists$, $\sum$, $\prod$ та інші. Вочевидь, що при виборі операцій МЕСМ, було враховано існуючі на той час методи проведення обчислень із цими позначками. У наступному розділі буде детально розглянуто, як такі позначки перетворювались у двійкові команди МЕСМ.

Окремі пункти інструкцій являли собою перевірку умов із визначенням переходу на попередній пункт (циклування). Обчислювальна лабораторія розросталася відповідно до потреб проведення розрахунків для різних задач народного господарства. До лабораторії відряджалися математики з різних підрозділів математичного інституту.

На той час у Феофанії, передмісті Києва, тривала розробка комп'ютера МЕСМ, який планувався та призначався для використання математиками ІМ АН УРСР. Співробітники обчислювальної лабораторії отримували доступ до державної таємниці.

7. Програми на МЕСМ

Із 12.01.1952р. частину співробітників обчислювальної лабораторії, включаючи К.Л. Ющенко (Рвачову) та її чоловіка, Ющенка О.А., було відряджено до Феофанії для експлуатації МЕСМ, а частина продовжувала рахувати арифмометрами.

Усе, що стосувалося комп'ютера МЕСМ, знаходилося під грифом суворої державної таємниці.

Експлуатація МЕСМ внесла корективи у розподіл праці між існуючими групами математиків лабораторії. Відтепер з'явилась можливість перекласти частину розрахунків на комп'ютер. Для використання комп'ютера частина математиків

[3] Іноді у джерелах лабораторію називають обчислювальною лабораторією. Згодом цю лабораторію перейменували у лабораторію методів обчислень та програмування.

опанувала складання програм. Обчислювачам, які «обертали» ручку арифмометра доручили вводити програми та дані до пам'яті МЕСМ.

Процес введення команд у пам'ять комп'ютера був не з простих: команди вводились по «0» та «1» шляхом вставки в отвори штекерів, опускання і піднімання тумблерів [11, 18].

Обчислювачі записували олівцем у зошити результати роботи комп'ютера з оперативної пам'яті: кожен біт оперативної пам'яті МЕСМ відображався на фанерних шафах у вигляді сяючої або згаслої лампочки. Для виконання цих робіт штат обчислювачів поповнюється студентками та дівчатами з середньою освітою [31].

Для введення програм застосовувались перфострічки. Був магнітний барабан для збереження програм та даних, а 1953р. підключають і друкарський пристрій.

Для зручності складання програм для МЕСМ математики замінюють коди операції позначками, а операнди команд мнемонічними літерами. Згодом ці мнемонічні коди були вдосконалені внесенням в них простих арифметичних виразів та шаблонів циклів [43]. Обчислювачі, виконуючи введення та виведення даних, допомагали програмістам та звільняли їх від рутинних дій [43].

Так поступово відбувалось наближення до високорівневого програмування. В документації по МЕСМ містяться приклади програм із циклами. Можливості команди складання команд МЕСМ дозволяють модифікувати команди в тілі циклів для отримання доступу до значень у сусідніх, послідовно розташованих комірках оперативної пам'яті комп'ютера, які у операторному методі програмування отримали назву «елементи масивів».

Обмеженість ресурсів МЕСМ відіграла позитивну роль, оскільки змусила програмістів знаходити вишукані прийоми програмування [11, 18].

Математичні позначки суми $\sum$ та добутку $\prod$ у формулах за своїм математичним значенням представляють визначення дій, що повторюються циклічно. Під позначками $\sum$ та могли міститись елементи векторів або матриць: $A = \sum_{i=1}^{n} a_i$, або не міститись:

$$e^x = \sum_{n=1}^{\infty} \frac{x^n}{n!}, \quad -\infty < x < \infty$$

.

Позначки $\sum$ та $\prod$ без індексів просто записувались у двійкових командах МЕСМ. Із документацією по МЕСМ були надані приклади, як із використанням машинних команд записувати послідовне звернення до значень елементів векторів та матриць в циклічно повторюваних діях.

Необхідність провести однакові розрахунки для кожного значення з певної множини даних позначались квантором загальності $\forall$. Це визначало потребу у повторенні дій для кожного елементу з множини значень, тобто повторювати дії на кшталт циклування. Саме ці потреби привели до використання опосередкованої адресації 2-ого рангу (вказівників), що дозволило економити час та зменшити ймовірність помилок під час введення даних.

А адресація 2-ого рангу до адрес підпрограм дозволила підвищити універсальність програм та підпрограм [43]. Вишукані прийоми використання опосередкованої адресації 2-ого рангу, які винайшли кияни, не описано в документації по МЕСМ та відсутні в операторному програмуванні. Подальше узагальнення опосередкованої адресації на адресацію вищих рангів дозволило з використанням «штрих-операції» (розіменування вказівників) формалізувати відношення слідування комірок пам'яті, які розташовані в пам'яті комп'ютера за довільними адресами.

Автоматизації програмування спрямовувались на полегшення роботи програмістів. Мнемонічні коди намагалися наблизити до інструкцій, які складали для розрахунків арифмометрами, що викликало набуття мнемонічними кодами ознак високорівневого програмування.

У програмуванні на МЕСМ також використовувався відомий бібліотечний метод програмування [11, 21, 22]. Для МЕСМ була розроблена бібліотека стандартних підпрограм з підпрограми елементарних функцій.

Інженери, які розробили МЕСМ, не збиралися на цьому зупинятися, до їхніх планів входила розробка великого київського комп'ютера, архітектуру якого розробили у 1952-1954р. Офіційно почали збирати комп'ютер у 1954р, який згодом у 1956р. отримав назву «Київ». Досвід використання МЕСМ та роботи з автоматизації програмування підказали математикам та інженерам важливі та унікальні архітектурні рішення для асинхронного комп'ютера «Київ» широкого призначення.

8. Високорівневі машинні програми комп'ютера «Київ»

«Київ» та Адресна мова. Ученій Катерині Ющенко доручили розробку математичного забезпечення комп'ютера «Київ», включаючи розробку системи команд, системи кодування даних у пам'яті, бібліотечних підпрограм та інші питання, пов'язані з програмуванням.

Процес написання програм у машинних кодах для МЕСМ був трудомістким та незручним, оскільки команди для виконання процесором записувались у двійковому коді, незручному для сприйняття людиною.

Під час розробки комп'ютера «Київ» враховувалися потреби полегшення роботи не лише програмістам, а й обчислювачам. К.О. Шкабара та К.Л. Ющенко ще задовго до початку робіт з проєктування комп'ютера «Київ», обговорювали потребу та можливість апаратної реалізації вдосконалених мнемонічних кодів програм, можливості модифікації адрес для циклів. Проводилися дослідження заміни введення даних по окремим бітам на введення символів[4].

Досвід розробки дослідницького компілятора виразів для МЕСМ та мова програмування, яка розроблювалась, вплинули на систему команд комп'ютера «Київ» [9 (Т. 1 стор. 30, 43)]. Система команд комп'ютера широкого призначення мала враховувати потреби розробки програмуючої програми (компілятора), можливості обробки текстів та бути зручною для розв'язку логічних та інших задач.

На відміну від асинхронного комп'ютера «Київ» широкого призначення, комп'ютери, розроблені до 1955р., призначалися виключно для математичних розрахунків [11, 26] шляхом виконання програм у двійкових машинних кодах або дуже до них наближених. Наприклад, до пам'яті комп'ютера ЕДСАК програми на асемблері вводились спеціальним пристроєм без задіяння процесора.

У комп'ютері «Київ» уперше в світі була апаратно реалізована адресація 2-ого рангу та можливість задання циклів [11 (стор. 489)].

Відтак комп'ютер широкого призначення «Київ» був першим у світі комп'ютером, у системі операцій якого було закладено потужні засоби програмування, які дозволяли визначати та обробляти спискові структури.

Оскільки операції комп'ютера «Київ» були ближче до мнемонічних кодів, ніж команди МЕСМ, то ті ж програми у мнемонічних кодах легше вводились до пам'яті комп'ютера «Київ» ніж до МЕСМ. Згодом, як зазначається в анотації до [40], Адресна мова набуває призначення для перенесення програм між комп'ютерами з різною архітектурою.

Автоматизацію програмування згодом називають адресним методом програмування, а сам термін «мова програмування» кияни починають застосовувати лише з 1958р. Деякий час вчені вважали, що Адресна мова створена у момент початку використання терміну «мова програмування» для адресного методу програмування. У ті роки ОЦ АН УРСР набував обертів та для більшої вагомості його досягнень було оголошено, що Адресну мову запропоновано в його стінах. Згодом було визначено, що Адресна мова створена 1955р., саме тоді, коли було запропоновано адресний метод програмування та запущено процесор комп'ютера «Київ» з апаратно реалізованою головною, базовою складовою Адресної мови програмування [11 (Т. 1 стор. 61)].

Згідно зі спогадами інженерів та очевидців С.Б. Погребинського і Б.М. Ма-

[4] В комп'ютері «Київ» символи кодуються трьома вісімковими знаками.

линовського комп'ютер «Київ» був розроблений 1956р [18]. Під час завершальної стадії розробки до комп'ютера «Київ» приєднувались унікальні зовнішні пристрої введення та виведення даних. Після перевезення Комп'ютера «Київ» з Феофанії на вул. Лисогірську 14.03.1958р. відбувся його повторний запуск. Порівняно з МЕСМ «Київ» мав не лише значно простіше програмування, а й мав продуктивність у сотні разів більшу ніж МЕСМ. Програми, які виконувались на МЕСМ декілька годин «Київ» виконував за лічені хвилини. 1956р. було прийнято рішення передати МЕСМ до КПІ [15, 18] для навчання студентів. Восени 1959р. було завершено перевезення МЕСМ з Феофанії до КПІ.

Команда складання команд. Широкі можливості команди складання команд МЕСМ та, водночас, незручність її використання, визначили напрями її вдосконалення та узагальнення. Також стало зрозуміло, що для прискорення динамічної модифікації адрес, необхідно мати спеціалізовані комірки у складі самого процесора, тобто регістри. Регістри модифікації адрес виконують більше функцій, аніж загальновідомі адресні регістри. Регістри модифікації адрес призначались, окрім прискорення доступу за індексами до елементів масивів, ще й для опосередкованої адресації 2-ого рангу – «штрих-операції» (розіменування вказівників) [2, 11 (Т. 1 стор. 489), 10].

Розробка системи команд комп'ютера «Київ» вдосконалила можливості динамічної зміни програм МЕСМ для забезпечення можливості та зручності розробки програмуючих програм (компіляторів). Це було революційним рішенням та видатним винаходом у галузі ІТ. Динамічну зміну програм комп'ютера «Київ» забезпечують групові операцій з використанням регістру модифікації адрес. До групових операцій належать три операції, включаючи Ф-операцію, яка заповнювала значенням регістр модифікації адрес [2-5, 7, 11, 10] та ще дві групові операції.

В окремому розділі глави ІІІ в монографії [2, 10] детально описано групові операції комп'ютера «Київ» та зокрема Ф-операція. З цього розділу видно, що, окрім розіменування вказівників, система операцій комп'ютера «Київ» передбачає задання даних, організованих у списковi ланцюжки, ідентичні найпростішому типу абстрактних типів даних. Також легко побачити, що Ф-операція дозволяє визначати та обробляти дані, які групуються у спискові ланцюжки, ідентичні абстрактним типам даних – однозв'язним лінійним спискам та іншим складним ієрархічним структурам. На відміну від абстрактних типів даних, елементами цих структур можуть бути окрім даних і підпрограми, аналогічно ООП.

Групові операції визначають початок та кінець циклування та фактично являють собою заголовки циклів імперативних мов програмування або, як визначено у Т. 1 на стор. 489 в [11], – *«операції для задавання циклів»*. Задання циклів операціями в комп'ютері «Київ» має унікальну можливість – **задавати цикли за елементами «списків»**.

Слід зазначити, що Ф-операція уможливлює визначення одного з двох типів відношень слідування:

1) послідовне слідування комірок оперативної пам'яті (для послідовного звернення до елементів списків та простих структур);

2) слідування, яке визначається послідовними значеннями вказівників («штрих-операцією»), які визначають «список» (для послідовного звернення до елементів «списків»).

Композиція цих двох відношень слідування в Адресній мові використовується для визначення довільних складних ієрархічних структур. Елементами, які належать послідовним слідуванням комірок, можуть бути адреси, які «породжують» нові «адресні» ланцюжки, що утворюються застосуванням «штрих-операції». Ідентично в Асоціативному програмуванні будуються складні спискові структури: *«у структуру спискового члену можуть входити описи списків»* [20 (стор. 110)].

У комп'ютері «Київ» є команда з назвою «команда складання команд», яка ідентична однойменній команді МЕСМ. Вона може бути використана для вико-

нання «штрих-операції» (розіменування Pointers).

Окреме, принципове значення має можливість використання Ф-операції з операцією завершення групових операцій для виконання багатократного виконання «штрих-операції», тобто для багатократного розіменування вказівників (Multiple indirection of Pointers).

Таким чином операції групові операції дозволяють задавати цикли зі змінною циклу та/або по елементам масивів (у загальному випадку по елементам багатовимірних масивів), а Ф-операція з ЗГО – цикли по елементам «списків». Групові операції являють собою апаратну реалізацію обробки складних ієрархічних структур (деревоподібних форматів) та призначені для прискорення їх обробки [15, 18].

В тілі циклу можуть використовуватися команди умовного чи безумовного переходу для «дострокового» виходу з циклу.

Високорівневість операцій. Відтак, у системі команд комп'ютера «Київ» наявні можливості організації циклічних процесів з послідовної обробки змінних числових значень, елементів масивів і елементів «списків».

До унікальних можливостей комп'ютера «Київ» належить умовна передача управління за опосередкованою адресою вищих рангів та відносна передача управління, яка дозволила завантажувати програми без будь-яких її змін у довільну ділянку оперативної пам'яті так, щоб робота програми не залежала від її місця розташування в оперативній пам'яті комп'ютера [2, 11 (Т. 1 стор. 61) 12, 10].

Система команд комп'ютера «Київ» містить, окрім групових операції модифікації адрес (засоби організації циклів), умовні та безумовні переходи, розгалуження можливості організувати цикли з перевіркою умов завершення виконання циклів. Команди вхідної мови програмування комп'ютера «Київ» мають оператори засилання (присвоєння) та можливість отримувати значення безпосередньо за адресою та за опосередкованою адресою 2-ого (адреса адреси або вказівник) та більш високого рангу. Вхідна мова комп'ютера «Київ» має засоби звернення до елементу масиву за його порядковим номером (індексом).

Відтак, вхідна мнемонічна мова комп'ютера «Київ» з можливістю задавати цикли та з використанням опосередкованої адресації визначати та оброблювати складні спискові структури (деревоподібні формати) [11 (Т. 1 стор. 489)] **являє собою мову програмування високого рівня, оскільки має більше можливостей ніж мова «Планкалкюль» Конрада Цузе,** яка визнана першою мовою програмування високого рівня.

Українські вчені мають усі підстави для визнання міжнародною спільнотою першості України у винаході як самих вказівників, так і у винаході списків та складних структур – деревоподібних форматів [20, 26, 40]. Абстрактні типи даних являють собою частковий випадок деревоподібних форматів, оскільки в деревоподібних форматах допускаються циклічні посилання на довільні комірки оперативної пам'яті, а як їхні елементи можуть використовуватися, подібно до методів ООП, підпрограми.

У комп'ютері «Київ» було вперше застосовано технологію використання змінно спаяних модулів з бібліотеками підпрограм, що дозволило суттєво економити оперативну пам'ять комп'ютера при використанні підпрограм.

Обробка чисел із плаваючою точкою була реалізована програмно. Прийняття такого рішення дозволило спростити інженерну компоненту комп'ютера «Київ», зменшити вартість та прискорити його розробку. Ці фактори переважили швидкість розрахунків з плаваючою точністю, оскільки було зрозуміло, що «Київ» проводитиме обчислення значно швидше за МЕСМ.

9. Унікальність концепції Адресного програмування

У цьому розділі на прикладах показано, що циклування в Адресній мові є більш загальним та, на відміну від імперативних мов програмування, припускає визначення в заголовках циклу перегляд елементів «списків». Також у цьому розділі визначено окремі унікальні можливості Адресного програмування щодо обробки

даних, які містяться в складних ланцюжкових ієрархічних структурах – деревоподібних форматах.

Необхідно нагадати, що концепція деревоподібних форматів базується на двох типах відношення слідування комірок пам'яті. Перший тип відношення слідування визначається лінійним порядком слідування адрес в оперативній пам'яті. Другий тип адресного відношення слідування задається «штрих-операцією» та визначає спискові ланцюжки. Як композиція двох типів слідування дозволяє визначати довільні ієрархічні структури визначалось у попередньому розділі. Зауважимо, що елементами деревоподібних форматів можуть бути, окрім самих даних, і адреси підпрограм (подібно ООП).

Нагадаємо, що операції комп'ютера «Київ» дозволяють задавати цикли поза обома відношеннями слідування.

Адресна мова програмування передбачає задання циклів не лише зі змінними циклу арифметичного типу, а й за адресами (тип Pointer). Можна визначати перший елемент, із яким має виконуватись перший раз тіло циклу, наприклад «головою списку». Як крок (**STEP**) у циклах типу «FOR і FROM 1 STEP 1 TO N» можна вказати адресне слідування – «штрих-операцію» (розіменування вказівників). В імперативних мовах з вказівниками необхідно використати Until або While. В табл. 1 наведено приклад фрагментів мовою C++ та Адресною мовою, в яких циклічно виконуються дії F із кожним елементом масиву та списку. Адресний тип відношення слідування (з використанням «штрих-операції») надає Адресній мові програмування можливість оперувати поняттям «множина».

Літери ***Ц*** та ***П*** – це позначки формули циклювання та формули входження (виклику) підпрограм. Літера π є аналогом змінної циклу **i** та змінної **Current**, значення яких зберігається у регістрах. Запису ***Ц***{ List, (''∅) ⇒ π } відповідають групові операції комп'ютера «Київ». Групові операції та використання регістру, а не комірок оперативної пам'яті, забезпечують прискорену обробки складних структур [2, 11, 10, 40, 41].

В Адресній мові програмістам надається можливість оперувати поняттям «множина» у визначенні повторювальних дій з кожним елементом множини або для тих елементів множини, які задовольняють визначену програмістом умову. Циклування в Адресній мові дозволяє задавати цикли по множинам, які є об'єднанням інших множин. Крім сказаного, велике значення для зручності програмістів має можливість визначати однією формулою циклування («заголовком циклу») одразу і вкладені цикли. Для задання циклу програмісту достатньо визначити множину та один із типів відношення слідування на ній.

У формулі циклування можна використовувати конструкції фільтрування, аналогічні умовам Where мови SQL.

Функція «map», яка наразі широко використовується у мовах програмування, природно визначається формулою циклування в Адресній мові.

Широкі можливості визначення відношення адресного слідування у формулах циклування дозволяють реалізовувати різноманітні схеми огляду областей доступності та деревоподібних форматів, що відповідає сучасному терміну «спосіб об-

Таблиця 1

Приклади фрагментів програм із циклом за елементами масиву та елементами списку

	C++	**Адресна мова**	**Пояснення на мові, подібній до C++**
Цикл за елементами масиву	**for** int **i** (**0**; **n**, **++i**) { F(M[**i**]); }	***Ц***{ **0**, (∅⊕**1**, **n**) ⇒ π } ***П*** F{π }	–
Цикл за елементами списку	List *Current =Head; while (Current -> Next != Null) { F(Current->info); Current = Current -> Next; }	***Ц***{ List, (''∅) ⇒ π } ***П*** F{π ⊕1}	// подібного в сучасних мовах немає: for *Pi (List; Null, *List) { F(Pi); }

ходу дерев» в ширину, глибину, праворуч чи ліворуч.

При визначенні циклування програмісту надається можливість користуватися поняттями «множина» та «підмножина».

В сучасних імперативних мовах аналогу «мінус штрих-операції», яка є оберненою до «штрих-операції» (розіменуванню Pointers) не існує. Ця операція має суто декларативний характер та дозволяє «пробігати» елементи однозв'язних лінійних списків у зворотному порядку [39, 40, 45].

Окрім того, «мінус штрих-операція» дозволяє визначати дерева шляхом визначення лише батька для вузлів дерева. В імперативних мовах із використанням концепції абстрактних типів даних для вузлів необхідно визначити список синів, наприклад: «лівий син», «правий син», «середній син» [45].

В Адресному програмуванні визначено поняття багатовимірного адресного сортування, яке дозволяє мати одночасно декілька різних варіантів сортування елементів множини. При адресному сортуванні відсутня необхідність змінювати місцезнаходження елементів множини під час їх сортування [25, 45].

10. Використання та значення Адресної мови програмування

Адресна мова програмування вплинула на вибір операцій не лише для комп'ютера «Київ», а й для інших київських комп'ютерів «Дніпро», «Промінь», «Дніпро-2», «Мір» [9 (Т. 1 стор. 30), 35 (стор. 10), 31]. Ідеї киян щодо модифікації адрес, були враховані в архітектурі інших радянських комп'ютерів, зокрема в «М-20» і архітектурно сумісними з нею: БЕСМ-3, БЕСМ-3М та БЕСМ-4 [8], в комп'ютерах серій «Мінськ», «Урал» та інших.

У відео [6 з 1:31:52] доцент факультету прикладних наук Українського Католицького Університету Олег Фаренюк, фахівець у галузі архітектури комп'ютерів, висунув та обґрунтував тезу: «*можна вважати, що «Київ» реалізував ідею RISC за кілька десятиліть до появи цього терміну*.»

Адресна мова використовувалась у розробці унікальних систем, зокрема, для реалізації інформаційної системи «Автодиректор» – табличної бази даних реляційного типу [18].

В монографії по комп'ютеру «Київ» [2, 10 (стор. 97-110)] наведено перелік реалізованих програм та систем. Це система автоматизованого проєктування електричних плат, декілька компіляторів, емулятор архітектури комп'ютера «Дніпро», інші розв'язки методами навчання задач розпізнавання образів, серед них, простих геометричних фігур, друкованих рукописних букв і чисел. І це не повний перелік реалізованих комп'ютером «Київ» задач.

Слід зазначити, що Адресна мова передбачає паралельне виконання операторів [2-5, 10, 40-42]. Можливість паралельного виконання програм Адресною мовою була використана для реалізації паралельних асинхронних процесів комп'ютера «Київ».

У період з 1956р. до кінця 1959р. в Адресній мові були конкретизовані та уточнені поняття слідування адрес, області доступності, введено поняття деревоподібних форматів [26, 40, 45]. Досліджувались методи та розроблювались схеми огляду даних, областей доступності або деревоподібних форматів. Надзвичайно важливого подальшого розвитку набули теоретичні дослідження деревоподібних форматів, які мали практичне значення для застосування універсальних засобів групування даних та процедур їх обробки [40].

До одного з найцікавіших та найпотужніших засобів розвитку Адресної мови програмування належить винахід «мінус штрих-операції», яка є оберненою до «штрих-операції» (розіменування вказівників) [11, 39, 40, 45]. Ця операція була реалізована В.П. Сьомиком у компіляторах Адресної мови [39]. «Мінус штрих-операція» носить виключно декларативний характер та тому не має аналогів у сучасних імперативних мовах програмування. В [45] описано порівняння можливостей «мінус штрих-операції» з можливостями сучасних декларативних засобів програмування та зроблено висновок, що Адресна мова програмування має вичерпні декларативні можливості.

Для комп'ютера «Київ» були розроблені унікальні пристрої введення та виведення даних. До цих пристроїв належать

пристрої введення графічних зображень з паперових носіїв інформації, фотоплівок і фотопластин [18] та пристрої друку зображень на папері [18].

Інформація в пам'яті комп'ютера про зображення зберігалась одним з двох способів: шляхом представлення по точкам або аналітичними функціями. На Адресною мовою були розроблені бібліотеки програм для обробки зображень, зокрема, розв'язувались задачі перетворення точкового представлення зображення у аналітичну форму. Проблеми перетворення точкового представлення зображень у аналітичне спонукали видатного математика В.Л. Рвачова до винайдення R-функцій, які згодом «*були успішно використані в задачах розпізнавання образів*» [34 (стор. 50)]. Це були значні національні досягнення, що перевершували тогочасні закордонні винаходи. Наявність пристроїв введення зображень в пам'ять комп'ютера та потужні можливості опосередкованої адресації вищих рангів дозволили українським вченим розробити систему розпізнавання простих геометричних фігур, друкованих літер та цифр [2, 7, 10, 18, 34].

Для комп'ютера «Київ» був розроблений пристрій отримання та передачі даних через телеграфні лінії зв'язку. Цю можливість було використано для віддаленого керування технологічними процесами на сталеплавильному заводі в місті Кам'янське (колишній Дніпродзержинськ), який знаходився від комп'ютера «Київ» на відстані 500 км. [18].

Другий примірник комп'ютера «Київ» було зібрано у 1959р. на замовлення Об'єднаного інституту ядерних досліджень (м. Дубна), де він понад десять років успішно використовувався для розв'язку задач [18, 30].

На комп'ютері «Київ» було створено емулятор комп'ютера «Дніпро». На цьому емуляторі розроблено компілятор Адресної мови для комп'ютера «Дніпро» [2, 10]. Це мало велике значення, оскільки дозволило ще до моменту запуску цієї машини, мати готовий компілятор Адресної мови та низку інших програм [2, 11, 10]. Це унікальний випадок в історії зародження інформаційних технологій, коли написання та налагодження програм здійснювалось для ще неіснуючого комп'ютера «Дніпро».

На комп'ютері «Київ» проводилися роботи із автоматизованого проєктування електричних схем вузлів для комп'ютерів, що знаходилися на стадії розробки. Це давало можливість віртуально тестувати електричні схеми без їх фактичного виготовлення. З використанням комп'ютера «Київ» та Адресної мови програмування розроблено комп'ютери «Дніпро», Промінь, комп'ютери серії «МІР» та комп'ютер «Дніпро-2» [11, 31].

Адресною мовою програмування було написано низку синтаксичних аналізаторів та розроблено цілий ряд компіляторів з інших мов програмування [11, 12, 33].

Наведемо цитату [11 (Т. 1 стор. 63)]: *«Завдяки можливості описувати адреси як функції якихось параметрів Адресної мови можна описувати й довільні схеми огляду інформації та складні інформаційно-логічні й економічні алгоритми і складні процеси перегляду й пошуку інформації, організованої в ланцюгові списки і спискові структури; алгоритм, процеси такого роду не можна описувати за допомогою алгоритм, мов типу АЛГОЛ, не залучаючи допоміжних засобів. У цьому відношенні Адресна мова випередила алгоритмічні мови, створені за кордоном для спискової обробки символічних виразів (напр., ЛІСП тощо).»*.

Адресна мова належить до мов функційного програмування («мов спискових»)[5] [11 (Т. 2 стор. 86, 87), 20 (стор. 107)]. Вона запропонована на декілька років раніше за мову LISP, яку помилково вважають першою мовою функційного програмування.

Адресна мова програмування реалізована на багатьох радянських комп'ютерах 1-ого та 2-ого поколінь, зокрема, на комп'ютерах «Київ», М-20 та сумісних з ним БЕСМ-3, БЕСМ-3М та БЕСМ-4, комп'ютерах серій «Урал» та «Мінськ» та

[5] У 50-х та 60-х роках замість «функціональні мови» використовувався термін «мови спискові».

використовувалася програмістами постсоціалістичного простору понад 20 років [18, 33]. Концепції рангу адрес та деревоподібних форматів увійшли до багатьох радянських мов програмування, зокрема, до: АЛГЕМ, А-КОБОЛ [11 (Т. 1 стор. 63)] та А-АЛГАМС [28].

Машинно-орієнтовані мови програмування АВТОКОД, зокрема для комп'ютера «Дніпро-2» [27, 28], включають засоби опосередкованої адресації вищих рангів Адресної мови програмування, що дозволяє визначати та оброблювати складні ієрархічні структури даних.

Видатний учений А.І. Кітов, відомий теорією Асоціативного програмування (1963р.) та мовою АЛГЕМ, в [20 (стор. 113)] зазначив, що принцип рангу адреси та адресних функцій Адресної мови програмування використано в Асоціативному програмуванні та мові АЛГЕМ, де складні ієрархічні структури утворюються використанням двох типів членів списків: вузлових та гніздових. Ці типи ідентичні двом типам відношень слідування Адресного програмування. У термінології Асоціативного програмування [20] збережено поняття «адреса» Адресної концепції програмування, а поняття «адресне співвідношення» ідентичне формулі засилання та відповідає адресному відображенню.

Система програмування Альфа-мовою (1963р.) для комп'ютера М-20 (та сумісних з ним: БЕСМ-3, БЕСМ-3М та БЕСМ-4), яка розроблена у середині 60-х років, містить засоби, які термінологічно, семантично та частково синтаксично ідентичні засобам Адресної мови програмування. Це засоби визначення циклів та спискових структур розміщення інформації. Система АЛГІБР є модифікацією Альфа-мови для БЕСМ-6 [11 (Т. 1 стор. 102-104), 16, 17].

Висновки

У статті розглянуто поступове зародження Адресної мови програмування з опосередкованою адресацією вищих рангів та складними ієрархічними структурами: деревоподібними форматами. Визначено вплив унікальної архітектури МЕСМ на винахід Адресної мови.

Стаття відновлює забуту та втрачену сторінку виникнення програмування високого рівня в Україні.

Світова спільнота ще не визнала першість винаходу вказівників за Україною та продовжує помилково вважати, що перші вказівники винайдені Гарольдом Лоусоном 1964р.

В статті наведено та детально розглянуто наявні публікації щодо підтверджень винаходу українськими вченими потужних засобів програмування: опосередкованої адресації (вказівників) вищих рангів та деревоподібних форматів, до яких подібні абстрактні типи даних. До переліку підтверджень належить низка публікацій з Адресної мови програмування та архітектури комп'ютера «Київ», перекладених багатьма мовами та виданих за кордоном.

Результати статті стануть в пригоді для визнання світовою спільнотою першості за Україною винаходу вказівників та складних ієрархічних структур у програмуванні високого рівня.

Стаття має за мету популяризацію досягнень українських науковців у часи зародження інформаційних технологій та має важливе значення для національно-патріотичного виховання молоді, оскільки спонукає молодь до наукових досліджень.

Стаття має значення для збереження всесвітньої історії зародження та розвитку інформаційних технологій, зокрема історії зародження програмування високого рівня з вказівниками та універсальними потужними засобами групування і зв'язування даних та процедур у складні спискові ієрархічні структури.

Література

1. Alvaro Videla, M 2018, 'Kateryna L. Yushchenko — Inventor of Pointers' A Computer of One's Own – Pioneers of the Computing Age, blog post, 8 December, viewed 24 October 2021, <https://medium.com/a-computer-of-ones-own/kateryna-l-yushchenko-inventor-of-pointers-6f2796fa1798?fbclid=IwAR3fcqmC0COfy5EqyIHBrlQhCpno5MUFZjCUQ-SM-v-xhD0g3xbj_P2SRCM>.
2. Glushkov V.M., & Yushchenko E.L., D 1966, The Kiev Computer; a Mathematical Description, USA, Ohio, Translation Division,

Foreign Technology Div., Wright-Pattenon AFB, 234p., ASIN: B0007G3QGC.
3. Gnedenko B.V., Koroliouk V. S. & Iouchtchenko E.L., D 1969, Eléments de programmation sur ordinateurs, Paris, Dunod, 362p., ASIN: B0014UQTU0, viewed 24 October 2021, <https://files.infoua.net/yushchenko/Elements-de-programmation-sur-ordinateurs_BGnedenko-VKoroliouk-EIouchtchenko_1969_France_OCR.pdf>.
4. Gnedenko B.V., Koroljuk V.S. & Justschenko E.L., D 1964, Elemente der Programmierung, DDR, Leipzig, Verlag: B. G. Teubner, 327 oldal.
5. Gnedenko B.V., Korolyuk V.S. & Juscsenko E.L. D 1964, Bevezetès a progamozásba, – I, II. – Magyarország, Budapest, Uj technica.
6. Sensei Yuri. Унікальність комп'ютера «Київ», 2021. YouTube. URL: https://youtu.be/3ohE3njX8P0?t=5512 (дата звернення: 17.11.2021).
7. Ware Willis H. & Holland Wade B., D 1963, Willis H. Ware. Soviet Cybernetics Technology: III, Programming Elements of the BESM, STRELA, Ural, M-3, and Kiev Computers, Translated by A.S. Kozak, RAND Corporation, 91p., Series: Research Memoranda, USA, viewed 24 October 2021, 70p., viewed 24 October 2021, <https://files.infoua.net/yushchenko/Soviet-Cybernetics-Technology-III_1963.pdf>.
8. Вакуленко С. Вычислительная машина «Киев», математическое описание. Dreamwidth.org. URL: https://vak.dreamwidth.org/771441.html (дата звернення: 31.10.2021).
9. Васильев Ю. Его оружие — математика К 60-летию Алексея Андреевича Ляпунова. / в кн.: Алексей Андреевич Ляпунов / Редакторы-составители Н.А. Ляпунова, Я.И. Фет. – Новосибирск : Филиал «Гео» Изд. СО РАН, Издательство ИВМиМГ СО РАН, 2001. – 502 с., – URL: www.ict.nsc.ru/jspui/bitstream/ICT/1260/1/Lyap_2001.pdf (дата звернення: 11.10.2021).
10. Глушков В.М., Вычислительная машина «Киев». Математическое описание / В.М. Глушков, Е.Л. Ющенко. // К. : – Гостехиздат УССР, 1962. – 183 с. : ил., – URL: Vychislitelnaya-mashyna-Kiev_VHlushkov_EYushchenko_1962.pdf (infoua.net) (дата звернення: 21.10.2021).
11. Глушков В.М. (відпов. ред.), Енциклопедія кібернетики. К. : Головна ред. укр. рад. енциклопедії, 1973. Т. 1 : А-Л. 584 с., Т. 2 : М - Я. 574 с. URL: https://files.infoua.net/yushchenko/Encyklopediya-kibernetyky_T1-2_VHlushkov-ta-in_1973_UKR_DJVU.zip (дата звернення: 24.10.2021).
12. Глушков В.М. и др., Основные направления развития цифровой вычислительной техники : обзор / В.М. Глушков, Б.Н. Малиновский, З.Л. Рабинович, Е.Л. Ющенко. – М. : – ЦНИИТЭИ приборостроения, 1967. – 96 с.
13. Гнеденко Б.В. Воспоминания. Моя жизнь в математике и математика в моей жизни. / Б.В. Гнеденко. // под ред. Д.Б. Гнеденко / (Наука в СССР. Через тернии к звездам), – М. : Ленанд, 2015. – 624 с., ISBN 978-5-9710-1416-4.
14. Гнєденко Б.В., Элементы программирования / Гнєденко Б.В., Королюк В.С., Ющенко Е.Л.; // М. : – ГИФМЛ, 1961. – 348 с., – URL: Elementy-programmirovaniya_BGnedenko-VKoroljuk-EYushchenko_1961.pdf (infoua.net) (дата звернення: 23.10.2021).
15. Дашевский Л.Н., Шкабара Е.А., Как это начиналось. // Новое в жизни, науке, технике. Сер. Математика, кибернетика; № 1 / М. : Знание, 1981. – 64 с., – URL: kak_eto_nachinalos.pdf (computer-museum.ru) (дата звернення: 23.10.2021).
16. Ершов А. П., Руководство к пользованию системой Альфа. / Ершов А. П., Кожухин Г.И., Поттосин И.В. // Новосибирск : Наука, 1968. – 180 с.
17. Ершов А.П., Входной язык системы автоматического программирования. // Ершов А.П., Кожухин Г.И., Волошин Ю.М. / М. : 1961. – С.173-174.
18. Європейський віртуальний музей історії інформаційних технологій в Україні., – URL: http://www.icfcst.kiev.ua/MUSEUM/ (дата звернення: 07.10.2021).
19. Іваненко Л.М. МЭСМ та її люди на відстані літ С.29-35 // В кн.: Видатні конструктори України. За матеріалами наукових читань з циклу "Видатні конструктори України", проведених у 2001–2018 роках. Т. 8. / кер. гр. уклад. М.Ю. Ільченко; за редакцією Б.Є. Патона, М.З. Згуровського. – К. : ВПК "Політехніка", 2018. – 256 с., – URL: http://biography.nbuv.gov.ua/rating/r2018/txt/g3/1007.pdf (дата звернення: 23.10.2021).

20. Китов А.И. Программирование информационно-логических задач. М. : Советское радио, 1967. 124 с., – URL: http://library.dnu.dp.ua/0318.rtf (дата звернення: 26.10.2021).
21. Китов А.И., Криницкий Н. А., Комолов П.Н., Элементы программирования (для электронных цифровых машин). / Отв. ред. Китов А.И. М. : изд-во Артиллерийской академии им. Дзержинского. // М. : – 1956. – 288 с.
22. Китов В.А., Прохоров С.А. Становление программирования в СССР с 1950-го по 1960 год. Виртуальный компьютерный музей, Материалы международной конференции SORUCOM 2011. – URL: https://www.computer-museum.ru/histsoft/1950_1960_sorucom_2011.htm (дата звернення: 29.09.2021).
23. Коваленко І. М. Граничні задачі життя. До 90-річчя академіка НАНУ В.С. Королюка / І.М. Коваленко, О.А. Летичевський, І.В. Сергієнко // ISSN 1027-3239. Вісн. НАНУ, № 8, 2015. С.95-99, – URL: http://www.visnyk-nanu.org.ua/sites/default/files/files/Visn.2015/8/15.Kovalenko.pdf (дата звернення: 26.10.2021).
24. Кратко М. «З історії розвитку інформатики в Україні: [історія створення електронних обчислювальних машин]», Світогляд. № 6, 2009. – С.56-62, – URL: https://www.mao.kiev.ua/biblio/jscans/svitogliad/svit-2009-20-6/svit-2009-20-6-56-kratko.pdf?fbclid=IwAR2mHMXkg_cvLKzi4xPywb2e-dGLXxE-k9IdW0pX6q1Yy7uyDkGB35eMAGY (дата звернення 21.10.2021).
25. Крещенко Т.О. Метод кластеризації з використанням багатовимірного адресного сортування. / Крещенко Т.О., Ющенко Ю.О. // Наукові записки НаУКМА. – Т. 3 : Комп'ютерні науки. – К. : – 2020. – С.83–87, – URL: https://doi.org/10.18523/2617-3808.2020.3.83-87 (дата звернення 12.10.2021).
26. Кулинкович А.Е., Ющенко Е.Л., О базовом алгоритмическом языке. / Кулинкович А.Е., Ющенко Е.Л., в журн.: «Кибернетика», К. : № 2, 1965. С.3-9, – URL: https://files.infoua.net/yushchenko/O-bazovom-algoritmicheskov-yazyke_AKulinkovich_EYushchenko_1965.pdf (дата звернення 12.10.2021).
27. Лаврищева Е.М. АКД – Автокод машины «ДНЕПР-2». / Лаврищева Е.М., Никитин А.И., Усенко Л.Г., Ющенко Е.Л. // К. : Ин-т кибернетики АН УССР, 1969. – 97 с.
28. Лаврищева Е.М., Л.Г.Борисенко, Усенко Л.Г., Ющенко Е.Л. Транслятор Д-АЛГАМС для УВК Днепр-2. / Лаврищева Е.М., Борисенко Л.Г., Усенко Л.Г., Ющенко Е.Л. // К. : ИК АН УССР, 1971. – 246 с.
29. Лебедев С.А., Дашевский Л.Н., Шкабара Е.Л., Малая электронная счетная машина. // М. : – Из-во АН СССР, 1952. – 162 с.
30. Маліновський Б.М., Історія обчислювальної техніки в обличчях. // К. : Фірма «КІТ», ПТОО «А.С.К.» 1995. 245 с., – URL: http://icfcst.kiev.ua/MUSEUM/TXT/Malinovsky_history_ukr.pdf (дата звернення 21.10.2021).
31. Міщенко Н., Інститут кібернетики НАНУ: все починалося у Феофанії… (1956–1958) : спогади. Ч. 1 / Надія Міщенко // Кібернетика, інформатика і довкола… : наукові праці, повідомлення, спогади., – URL: https://cyberua.info/novyny/ik-nanu-vse-pochynalosja-u-feofaniji-spohady-ch1-nadija-mishchenko/ – (дата звернення: 26.10.2021).
32. Олешко А.Л., Коваленко (Юфит) Нонна Наумовна (воспоминания) / Газета «Південна зоря». – Бердянська суспільно-політична газета. – № 1306, архів: С.295-344 – URL: http://pivdenka.berdyansk.net/assets/files/book/Odnokursniki/odnokursniki-chast08.pdf?fbclid=IwAR3XkFHgPhLhmm-ScbTbS9cjs3LHthBENTp-JsFQmfZ3eSM-j-4UsmeS9Ik (дата звернення: 26.10.2021).
33. Перевозчикова О.Л., Школа теории программирования Е.Л. Ющенко. К : – Наука та наукознавство. – 2007. – № 4. – С.114-146, – URL: http://dspace.nbuv.gov.ua/handle/123456789/49192?show=full (дата звернення: 05.10.2021).
34. Подгаєцький О. Еволюція розробок у галузі штучного інтелекту в Україні та світі (Evolution of research in the field of artificial intelligence in Ukraine and the World) / Oleksandr Podgayetsky // Дослідження з історії техніки. – № 16, Харків : – 2012. – URL: https://ela.kpi.ua/bitstream/123456789/7703/1/RHT-issue-16-title-05-Podgayetsky.pdf (дата звернення: 07.10.2021).
35. Семик В.П. Транслятор с адресного языка для ЭВМ «Минск-1» / Семик В.П., Згурский А.Д., Шкляров Л.И., Глушко А.Г. // Сб. «Приборы и устройства средств авто-

матики и телемеханики». Вып.1. Изд. ХГУ. – г.Харьков : 1965. С.96-98.
36. Семик В.П., Ющенко Е.Л. Адресный язык и принцип адресности в алгоритмических языках. // Сб. «Информационные системы». Изд. ВИНИТИ АН СССР. – М. : 1964. С.9-14.
37. Семик В.П., Ющенко Е.Л. Возможности адресного языка для описания информационно-логических алгоритмов. // Сб. «Информационные системы». Изд. ВИНИТИ АН СССР. – М. : 1964. С.5-8.
38. Сергієнко І.В., Інформатика в Україні: становлення, розвиток, проблеми / відп. ред.: Ю.В. Капітонова, Т.Т. Лебедєва; рец.: Н.З. Шор, О.В. Палагін, К.Л. Ющенко; НАНУ, Ін-т кібернетики // – К. : Наукова думка, 1999. 354 с., – ISBN 966-00-0540-7.
39. Сьомик В.П., Про розширення поняття рангу адреси. // Сб. «Обчислювальна математика і техніка. Вид. АН УРСР. – К. : 1963. С.61-65.
40. Ющенко Е.Л., Адресное программирование // К. : – Гос. издательство технической литературы, УРСР, 1963. – 287 с., – URL: https://files.infoua.net/yushchenko/Adresnoe-programmirovanie_EYushchenko_1963.pdf (дата звернення: 03.10.2021).
41. Ющенко Е.Л., Адресное программирование и особенности решения задач на машине «УРАЛ» / под ред. Бушко-Жук. // К. : Киев. высш. инженерное радиотехническое училище войск противовоздуш. обороны страны.: 1960. – 192 с.
42. Ющенко Е.Л., Адресный язык (Тема 5) // Кибернетика на транспорте: Заочный семинар. / Киевский дом Научно-технической пропаганды / – К. : – 1962. – 32 с., – URL: Kibernetika-na-transporte_Adresnyy-yazyk_KYushchenko_1962.pdf (infoua.net) (дата звернення: 12.10.2021).
43. Ющенко Ю.О., Pointers в програмах на «МЕСМ» / Для Європейського віртуального комп'ютерного музею історії інформаційних технологій в Україні: //К. : 2021, 9 с., – URL: http://www.icfcst.kiev.ua/MUSEUM/TXT/YuriYushchenko_u.pdf (дата звернення: 29.09.2021).
44. Ющенко Ю.О., Катерина Логвинівна Ющенко – винахідниця Pointers та авторка однієї з перших в світі мов програмування високого рівня // Газета «Світ», № 5-6, 10.02.2021 р., Видав-во НАНУ та МОНУ. К. : – 2021. – С.2-3, – URL: https://www.nas.gov.ua/UA/Messages/Pages/View.aspx?MessageID=7487&fbclid=IwAR1BqiFUu7OGwN1knFbAUQt4sPeDKYZLkLeRIABFxoX-RlgaxL_gtISJz-g (дата звернення: 23.10.2021).
45. Ющенко Ю.О., Окремі аспекти декларативності «мінус штрих-операції» // Наукові записки НаУКМА. – Т. 3 : Комп'ютерні науки. – К. : – 2020. – С.19–26, – URL: http://nrpcomp.ukma.edu.ua/article/view/220657?fbclid=IwAR0N65fNM1liyasK84z0OsEmzenb10FAy8IksOv0pzV6JSX54TJpBf0-V7o (дата звернення: 03.10.2021).
46. Ядренко М.Й. Борис Володимирович Гнєденко – фундатор кафедри теорії імовірностей в Київському університеті. // Теорія імовірностей та математична статистика. Вип. 56, К : – 1997. – С.32-39, – URL: https://probability.knu.ua/tims/issues-new/56/PDF/6.pdf (дата звернення: 07.09.2021)

Одержано: 23.10.2021

Про автора:

Ющенко Юрій Олексійович,
к.ф.-м.н., доцент факультету інформатики
Національного університету
Києво-Могилянська Академія.
Кількість публікацій в українських журналах – 16,
Індекс Хірша - 3.
ORCID: 0000-0003-4602-1774.

Місце роботи автора:

Національний університету
Києво-Могилянська Академія
Вул. Григорія Сковороди, 2, Київ, Україна, 04655.
Електронна пошта: yury.yuschenko@ukma.edu.ua.
Тел.: +38(093)3784051.

УДК 519.164:004.14

Засоби забезпечення та засвідчення якості контекстно-залежної композиції семантичних Веб-сервісів / П.І. Андон, О.О. Слабоспицька.

Розроблений авторами метод динамічної композиції адаптивного семантичного Веб-сервісу розвинуто алгоритмами узгодженого застосування спеціальної OWL-S-формалізації його запитаного контексту на етапах життєвого циклу та опрацювання циклічних залежностей на функціональному рівні композиції за допомогою формалізмів Сервісу-вузла, Сервісу-посередника й Спрощення – для сталого забезпечення якості формованого Веб-сервісу. Запропоновано засвідчення рівня якості для причетних сторін шляхом динамічної верифікації, зокрема, перевірки відповідності запитаному контексту та живості, з використанням процесного числення контекстно-залежних амбієнтів. Надані алгоритми підвищують відповідність формованого Веб-сервісу очікуванням запитувачів та уможливлюють його стале контекстно-залежне уточнення для уніфікованої підтримки як змінних розподілених ділових процесів сучасних організацій, так і потреб споживачів певної галузі.

Ключові слова: OWL-S, семантичний Веб-сервіс, динамічна контекстно-залежна композиція Веб-сервісів, циклічна залежність, засвідчення якості, граф композиції, числення амбієнтів.

UDK 519.164:004.14

Means for Quality Implementation and Assurance of Context-Aware Semantic Web Service Composition / P.I. Andon, O.O. Slabospitska.

Author's Method for Adaptive Semantic Web Service dynamic composition is enriched with the algorithms for both its Context dedicated OWL-S-specification consistent using over its Life Cycle and Cyclic Dependencies at the function level solving with the Formalisms proposed of Knot-service, Proxy-service and Simplification – for sustain Quality implementing of Web Service being formed. Quality assurance for stakeholders is proposed through dynamic verification, in particular meeting the specified context and liveness, with the process Calculus of Context-aware Ambients. The algorithms proposed increase correspondence of Web Service being formed with Customers' expectations and enable its sustain context-aware tailoring for unified support of both modern organizations' variable and distributed Business Processes and some problem field Customers' needs.

Keywords: OWL-S, Semantic Web service dynamic context-aware Web service composition, cyclic dependency, quality assurance, composition graph, ambient calculus.

УДК 004.6

Методи та технології управляння моделями представлення знань, базованих на онтологіях в контексті велики х даних / О. Новицький

Онтологічні моделі представлення знань у контексті великих даних є одним із способів зменшити складність обробки даних за допо-

UDC 004.6

Methods and techniques for management of ontolodgy-based knowledge representation models in the context of BIG data / O. Novytskyi

Ontology-based knowledge representation models in the context of big data are one way to reduce complexity for data processing across meth-

могою семантичних методів. У статті розглянуто методи і засоби ефективного управління моделями на основі онтологій, які покращують системи великих даних. Для цього випадку мова вираження обмежень форм (SHACL) для перевірки інформації була розглянута як ключовий метод. У статті також досліджуються та розглядаються представлення знань та методи виводу. Належне управління моделями представлення знань на основі онтології за допомогою запропонованих методів і засобів забезпечує покращену інтеграцію даних, якість великих даних та інтеграцію бізнес-процесів.

Ключові слова: модель на основі онтології, онтології, великі дані, вивід, система представлення, онтології, мова обмежень форм, перевірка інформації, моделі представлення знань на основі онтологій

ods of semantic description. This research paper aims at providing an overview of the methods and techniques for efficient management of the ontology-based models that improve big data systems. For this case, the shapes constraint language (SHACL) for information validation was reviewed as the key method. The knowledge representation systems and reasoners are studied and reviewed in the paper as well. It describes approaches based on ontologies in the context of big data. The proper management of ontology-based knowledge representation models through offered methods and techniques brings improved data integration, big data quality, and business process integration.

Key words: ontology-based model, ontologies, big data, reasoners, representation system, ontologies, shapes constraint language, information validation, ontology-based knowledge representation models

УДК 681.3

UDC 681.3

Дослідження принципів, моделей та методів парадигми менеджменту наукових даних FAIR для аналізу метаданих BIG DATA / Ю.В. Рогушина, І.Ю. Гришанова.

Розглянуто базові принципи, моделі та методи парадигми менеджменту наукових даних FAIR (Findable, Accessible, Interoperable, Reusable як окремого випадку великих даних (Big Data), яка орієнтована на повторне використання результатів наукових досліджень. Проаналізовано, як властивості даних FAIR сприяють уніфікації й об'єднанню наукової інфраструктури у парадигмі відкритої науки. Запропоновано методи та програмні засоби, за допомогою яких властивості даних FAIR можуть відтворюватися у семантично розмічених Wiki-ресурсах, що побудовані на основі Semantic MediaWiki.

Ключові слова: метадані, Big Data, семантичні Wiki-ресурси

Study of principles, models and methods of FAIR paradigm of scientific data management for analysis for BIG data metadata / I. Grishanova, J. Rogushina.

In this research work we considere the basic principles, models and methods of the FAIR (Findable, Accessible, Interoperable, Reusable) scientific data management paradigm as a separate case of Big Data. This paradigm is focused on the reuse of scientific research results.Basic principles of FAIR apply to three types of entities: data (or any digital object), metadata (information about this digital object) and infrastructure. Information on various projects, initiatives and communities working on solving the problems of scientific data and their metadata integration is examined.
We analyze how the properties of FAIR data contribute to the unification and integration of the scientific infrastructure in the paradigm of open science, which is based on free access to research results and open data.

Keywords: metadata, Big Data, semantic Wiki resources

УДК 004.94

60 років базам даних/ В.А. Резніченко.

Наводиться огляд досліджень і розробок баз даних з моменту їх виникнення в 60-х роках минулого століття і по сьогодні. Виділяються наступні етапи: виникнення і становлення, бурхливий розвиток, епоха реляційних баз даних, розширені реляційні бази даних, пост-реляційні бази даних і великі дані. На етапі становлення описуються системи IDS, IMS, Total і Adabas. На етапі бурхливого розвитку висвітлені питання архітектури баз даних ANSI/X3/SPARC, пропозицій КОДАСИЛ, концепції і мов концептуального моделювання. На етапі епохи реляційних баз даних наводяться результати наукової діяльності Е. Кодда, теорія залежностей і нормальних форм, мови запитів, експериментальні дослідження і розробки, оптимізація та стандартизація, управління транзакціями. Етап розширених реляційних баз даних присвячений опису темпоральних, просторових, дедуктивних, активних, об'єктних, розподілених та статистичних баз даних, баз даних масивів, а також машин баз даних і сховищ даних. На наступному етапі розкрита проблематика постреляційних баз даних, а саме, NOSQL-, NewSQL- і онтологічних баз даних. Шостий етап присвячено розкриттю причин виникнення, характерних властивостей, класифікації, принципів роботи, методів і технологій великих даних. Нарешті, в останньому розділі подається короткий огляд досліджень і розробок по базах даних в Радянському Союзі.

Ключові слова: Типи баз даних: ієрархічна, мережева, реляційна, навігаційна, темпоральна, просторова, просторово-темпоральна, просторово-мережева, об'єктів, що переміщуються, дедуктивна, активна, об'єктно-орієнтована, об'єктно-реляційна, розподілена, паралельна, масивів, статистична, багатовимірна, машина баз даних, сховища даних, NoSQL, ключ-значення, стовпчикова, документно-орієнтована, графова, мультимодельна, хмарна, наукова, багатозначна, XML, NewSQL, онтологічна, великі дані.

UDC 004.94

60 Years of Databases/ V.A. Reznichenko.

The article provides an overview of research and development of databases since their appearance in the 60s of the last century to the present time. The following stages are distinguished: the emergence formation and rapid development, the era of relational databases, extended relational databases, post-relational databases and big data. At the stage of formation, the systems IDS, IMS, Total and Adabas are described. At the stage of rapid development, issues of ANSI/X3/SPARC database architecture, CODASYL proposals, concepts and languages of conceptual modeling are highlighted. At the stage of the era of relational databases, the results of E. Codd's scientific activities, the theory of dependencies and normal forms, query languages, experimental research and development, optimization and standardization, and transaction management are revealed. The extended relational databases phase is devoted to describing temporal, spatial, deductive, active, object, distributed and statistical databases, array databases, and database machines and data warehouses. At the next stage, the problems of post-relational databases are disclosed, namely, NOSQL-, NewSQL- and ontological databases. The sixth stage is devoted to the disclosure of the causes of occurrence, characteristic properties, classification, principles of work, methods and technologies of big data. Finally, the last section provides a brief overview of database research and development in the Soviet Union.

Keywords: Database types: hierarchical, network, relational, navigational, temporal, spatial. spatio-temporal, spatio-network, moving objects, deductive, active, object-oriented, object-relational, distributed, parallel, arrays, statistical, multidimensional, database machines, data warehouse, NoSQL, key-value , triple store, column-oriented, document- oriented, graph-oriented, multimodal, cloud, scientific, multi-valued, XML, NewSQL, ontological, Big Data.

УДК004.942:004.021:004.942

Інформаційна технологія забезпечення живучості сенсорних мереж / В.Я. Петрівський, В.Л. Шевченко, О.С. Бичков, І.П. Сініцин.

У сучасному технологічному світі сенсори та сенсорні мережі широко використовуються у більшості сфер людської діяльності. Одним із ключових інженерних завдань під час проєктування сенсорних мереж є питання забезпечення живучості мережі. У статті представлено алгоритми забезпечення живучості сенсорних мереж на основі попередньої оцінки даної властивості. Оцінка живучості залежить від топології мережі. Підвищення показника живучості сенсорної мережі досягається шляхом включення до мережі додаткових датчиків. Запропоновано алгоритм знаходження позиції додаткових датчиків, що враховує радіус покриття сенсорів та необхідність наявності перетину зон покриття датчиків для забезпечення обміну інформацією. Представлено ітераційний алгоритм забезпечення живучості мережі за наявності та врахування динамічних датчиків. Результати обчислювальних експериментів, що представлені у роботі, підтверджують ефективність запропонованих підходів.

Ключові слова: сенсори, сенсорні мережі, живучість сенсорних мереж, інформаційна технологія

UDC 004.942:004.021:004.942

Information technology to ensure the survivability of sensor networks / V.Y. Petrivskyi, V.L. Shevchenko, O.S. Bychkov, I.P. Sinitsyn.

In the modern technological world, sensors and sensor networks are widely used in the all spheres of human activity. One of the key engineering tasks in the design of sensor networks is to ensure the survivability of the network. The article presents algorithms for ensuring the survivability of sensor networks based on a preliminary assessment of this property. Estimation of survivability depends on the network topology. Increasing the survivability of the sensor network is achieved by including additional sensors in the network. An algorithm for finding the position of additional sensors in the stationary sensor network is proposed. Proposed approach takes into account the radius of coverage of the sensors and the need for the intersection of the coverage areas of the sensors to ensure the exchange of information. An iterative algorithm for ensuring the survivability of the network in the presence and consideration of dynamic sensors is presented. The results of computer simulation experiments presented in the paper confirm the effectiveness of the proposed approaches.

Keywords: sensors, sensor networks, survivability of sensor network, information technology

УДК 004.942

Особливості комп'ютерного моделювання розповсюдження думок у соціумі на прикладі моделі студентської спільноти / В.В. Шевченко, Д.С. Берестов, І.П. Сініцин, В.Я. Петрівський.

У статті розглядаються особливості комп'ютерного моделювання, засновані на використанні теорії клітинних автоматів як основного інструменту для моделювання процесів формування суспільної думки. Об'єктом моделювання було обрано соціальну поведінку в контексті прийняття навчального матеріалу в рамках освітнього процесу. Як інструмент моделювання була обрана теорія клітинних автоматів, оскільки вона є відносно простим і водночас ефективним мето-

UDC 004/942

Peculiarities of Computer Modeling of Thought Dissemination in Society on the Example of Student Society / V.V. Shevchenko, D.S. Berestov, I.P. Sinitsyn, V.Y. Petrivskyi.

The paper considers methods of mathematical modeling based on the use of the theory of cellular automata as the main tool for modeling the processes of public opinion formation. As the object of modeling was chosen social behavior in the context of the acceptance of educational material in the science-centered approach. The theory of cellular automata was chosen as a modeling tool because it is a relatively simple and at the same time effective method for modeling the interaction of single-type objects. The developed

дом моделювання взаємодії однотипних об'єктів. Розроблена модель має розширені правила визначення стану клітини і визначення близькості клітини. Також була виведена залежність коефіцієнта сприйняття інформації від різниці між станом клітини і її оточення. Ці удосконалення дозволяють використовувати модель для прогнозування змін в уподобаннях соціальних груп.

Ключові слова: клітинні автомати, навчальний процес, моделювання

basic cellular automata of the model has extended rules of determination of cell state and determination of cell vicinity. It has also been derived the dependence of the information perception coefficient on the difference between the state of the cell and its surroundings. These improvements allow the model to be used to predict changes in social group preferences.

Keywords: cellular automata, learning process, modeling

УДК 004.43::004.94

Граматика мови візуального програмування Петрі-об'єктних моделей / А.Ю. Дифучин, І.В. Стеценко, Е.В. Жаріков.

Імітаційне моделювання широко використовується у дослідженні властивостей систем, пошуку оптимальних умов, прогнозування. Мережі Петрі є універсальним інструментом для формалізації дискретно-подійних систем. Недоліком мереж Петрі є зростання кількості візуальних елементів під час проєктування складних систем. Через нагромадження зв'язків та елементів перевага візуального сприйняття моделі зникає, модифікація параметрів моделі потребує значної кількості рутинних дій. Петрі-об'єктні моделі вирішують проблему тиражування фрагментів мереж Петрі з заданими параметрами та конструювання моделі системи з великої кількості елементів. Однак Петрі-об'єктна технологія не позбавляє необхідності зв'язування елементів моделі. Розроблена мова візуального програмування Петрі-об'єктних моделей дає можливість зменшити кількість помилок при конструюванні моделі за рахунок автоматизації кодування зв'язків між елементами моделі та графічного представлення моделі. Окрім тиражування Петрі-об'єктів, мова реалізує тиражування зв'язків між Петрі-об'єктами. За рахунок такого тиражування досягається також компактність візуального представлення для складних моделей. Реалізація введених понять у мові візуального програмування дає можливість швидко створювати та групувати елементи моделі. Алфавіт мови складається із графічних елементів, передбачених для конструювання Петрі-об'єктної моделі. Синтаксис мови визначений правилами конструювання виразів із послідовності символів алфавіту мови. Семантика мови програмування визначається правилами перетворення мовних виразів в обчислення. Формалізація гра-

UDC 004.43::004.94

The grammar of Petri-object model visual programming language / Anton Dyfuchyn, Inna V. Stetsenko, Eduard Zharikov.

Petri-object models solve the problem of replicating fragments of Petri nets with given parameters and constructing a model from a large number of elements. The developed visual programming language of Petri-object models gives a possibility to reduce the number of errors during a model construction by automating the coding links between elements and graphical representation of a model. In addition to replicating Petri objects, the visual language implements replication of links between Petri objects. Formalization of the visual programming language grammar is presented in the form of production rules. A conclusion about grammar properties has been drawn.

Keywords: Key words: formal grammar, visual programming, stochastic Petri nets, simulation.

матики мови візуального програмування представлена у вигляді правил виведення та зроблені висновки про її властивості.

Ключові слова: формальна граматика, візуальне програмування, стохастична мережа Петрі, імітаційне моделювання.

УДК 004.93’1

Модель згорткової нейронної мережі та програмний засіб для класифікації типових комах-шкідників/ Ю.С. Безлюдний, В.М. Шимкович, А.Ю. Дорошенко.

Розроблено модель загорткової нейронної мережі, базу даних для навчання нейронної мережі та програмний засіб для класифікації типових комах-шкідників, що дозволяє здійснювати розпізнавання класу комахи-фітофага за переданим зображенням. Виконано оптимізацію структури моделі нейронної мережі задля покращення результатів класифікації. Проаналізувавши типові архітектури згорткових нейронних мереж та наклавши на них часові та ресурсні обмеження, було обрано наступну послідовність шарів вихідної загорткової нейронної мережі: один вхідний шар; п’ять згорткових шарів, між якими знаходяться чотири пари шарів нормалізації та об’єднання; три повністю зв’язані шари з двома шарами розрідження між ними. Розроблено власну базу даних для навчання нейронної мережі. Сумарно створений набір даних містить 3000 зображень, які впорядковані ієрархічно по директоріям, відповідно до класу та призначення (для навчання чи тестування). Додатково здійснено розробку інтерфейсу користувача, передбачено автентифікацію та авторизацію, персоналізацію даних, наявність ролей користувачів та відповідний розподіл функціоналу за ролями, створено можливість перегляду статистики по класифікованим комахам, у певному проміжку часу. Проведено функціональне тестування розробленого застосунку на гетерогенному наборі зображень комах 20 різних класів. Результати тестування на CPU середнього часу обробки запитів до застосунку: класифікація зображення натренованою згортковою нейронною мережею – 115 мс; додавання нового зображення до тренувального набору (без урахування часу виконання асинхронних процесів) – 39 мс; завантаження зображення зі зовнішнього сервісу – 434 мс; отримання статистики з класифікації – 5 мс. Подальше покращення якості

UDC 004.93’1

Convolutional neural network model and software for classification of typical pests / Y.S. Bezliudnyi, V.M. Shymkovysh, A.Yu. Doroshenko.

A model of a convolution neural network, a database for neural network training, and a software tool for classifying typical insect pests have been developed. It allows recognizing the class of phytophagous insects from the transmitted image. The structure of the neural network model has been optimized to improve the classification results. After analyzing the typical architectures of convolution neural networks and imposing time and resource constraints on them, the following sequence of layers of the original convolution neural network was selected: one input layer; five convolution layers, between which there are four pairs of normalization and aggregation layers; three fully connected layers with two layers of rarefaction between them. Developed its own database for neural network training. In total, the created data set contains 3000 images, which are arranged hierarchically by directories, according to their class and purpose (for training or testing). Additionally, the user interface was developed, authentication and authorization, data personalization, the presence of user roles and the appropriate distribution of functionality by role, the ability to view statistics on classified insects in a certain period of time. Functional testing of the developed application on a heterogeneous set of images of insects of 20 different classes was carried out. The results of testing on the CPU of the average processing time of requests to the application: the classification of the image by a trained convolution neural network – 115 ms; adding a new image to the training set (excluding the execution time of asynchronous processes) – 39 ms; image download from external service – 434 ms; receiving classification statistics – 5 ms. Further improvement of the quality of classification can be done with the help of: calculations using video cards; means of parallel data processing;

класифікації може бути виконане за допомогою: обчислень з використанням відеокарт; засобів паралельної обробки даних; побудови конструктивно складніших архітектур згорткових нейронних мереж тощо.

Ключові слова: класифікація зображень, згорткові нейронні мережі, модульна архітектура, Python, keras

construction of structurally more complex architectures of convolution neural networks, etc.

Keywords: image classification, convolution neural networks, modular architecture, Python, keras.

УДК 94::004:004.[235+43+655.3]

UDC 94::004:004.[235+43+655.3]

Розробка архітектури комп′ютера «Київ» за концепцією адресного методу програмування / Ющенко Ю.О.

Описано поступовий перехід від розрахунків логарифмічними лінійками та арифмометрами до використання мови високого рівня з вказівниками та складними ієрархічними структурами. В статті розглянуто фактори, які сприяли цьому важливому технологічному переходу. Робота відновлює загублену ланку історії виникнення в Україні опосередкованої адресації (вказівників) вищих рангів та складних ієрархічних структур. Надаються підтвердження винайдення вказівників та складних структур даних київськими вченими, описано апаратну реалізацію в комп'ютері «Київ» «штрих-операції» (розіменування вказівників) та операцій задання циклів. У роботі обґрунтовується високорівневість програмування у командах комп'ютера «Київ» шляхом порівняння з мовою програмування високого рівня – Plankalkül.

Ключові слова: історія, штрих-операція, опосередкована адресація, вказівники, програмування, деревоподібні формати, масиви, списки, структури, абстрактні типи даних.

Invention of a computer "Kyiv" architecture using a concept of Addressed Programming Language / Yuschenko Yu.O.

The article is devoted to the history of the origin of high-level programming in Ukraine. The transition from calculations by arithmometers and logarithmic rulers to solving problems on the computer "Kyiv" using pointers and tree-like formats (abstract data types are analogous) is described. The factors that contributed to this transition include: the experience of providing instructions for calculations by arithmometers, and the experience of programming on MESM. As a result a computer "Kyiv" has been developed with a hardware-implemented possibility of high-level programming, invention of the Addressed Programming Language with indirect addressing (pointers), tree formats and declarative capabilities. Hardware-realized demining of pointers in the computer "Kyiv" is one of the outstanding inventions of Ukrainian engineers and mathematicians at the initial stage of the information technologies development. At that time it was significantly ahead of the world technologies. Programming in computer "Kyiv", unlike Plankalkül, could identify and process complex structures. The paper describes the individual applications of the Address Programming Language, which was implemented on many Soviet-made computers and has been used by programmers for more than 20 years. Due to the so-called "Iron Curtain", scientists in the field of programming outside the post-socialist world still do not know about the invention of pointers by Kiev scientists. A textbook describing the Addressed Programming Language was translated into many languages. A monograph with a description of the computer architecture "Kyiv" and of the Addressed Programming Language was translated into English and published in the United States in 1966.

Keywords: history, stroke-operation, indirect addressing, pointers, programming, tree-like formats, arrays, lists, structures, abstract data types.

ДО УВАГИ АВТОРІВ!

У журналі «Проблеми програмування» публікуються наукові матеріали, які раніше не публікувалися в інших виданнях.

Мова статті: українська, англійська.* Обсяг статті - від 6 до16 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: alengoro@isofts.kiev.ua. FAX: +380 (44) 526 6263, Телефон: +380 (96) 418 3082.

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 – 11пт. 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. Джерела з заголовками на латиниці наводяться без перекладу. Інші джерела подаються мовою оригіналу. Приклади оформлення бібліографічних посилань згідно з вимогами ***Harvard Style наведені в багатьох публікаціях, наприклад: 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 – англійською і українською 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 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".

*16.07.2020 р. набули чинності положення Закону України «Про забезпечення функціонування української мови як державної». Відповідно до статті 22 «Державна мова у сфері науки» у наукових виданнях не повинно бути вміщено матеріалів іншими мовами, окрім державної, англійської та мов ЄС.

Примітка: Підписний індекс журналу «Проблеми програмування» – **90853**.

Індекс 90853