НАЦІОНАЛЬНА АКАДЕМІЯ НАУК УКРАЇНИ
ІНСТИТУТ ПРОГРАМНИХ СИСТЕМ

ISSN 1727-4907

П Р О Б Л Е М И ПРОГРАМУВАННЯ

НАУКОВИЙ ЖУРНАЛ

P R O B L E M S IN PROGRAMMING

SCIENTIFIC JOURNAL

2019

№ 3

Теми випуску:

- ***Теоретичні та методологічні основи програмування***
- ***Методи та засоби програмної інженерії***
- ***Моделі та засоби систем баз даних і знань***
- ***Експертні та інтелектуальні інформаційні системи***
- ***Прикладні засоби програмування та програмне забезпечення***

НАЦІОНАЛЬНА АКАДЕМІЯ НАУК УКРАЇНИ
ІНСТИТУТ ПРОГРАМНИХ СИСТЕМ

П Р О Б Л Е М И ПРОГРАМУВАННЯ

науковий журнал

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

Адреса для кореспонденції

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

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

Затверджено до друку вченою радою Інституту програмних систем НАН України.
Протокол № 5 від 13.06.2019 р.

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

Підписано до друку 19.08.2019. Формат 60x84/8. Папір офс. Ум. друк. арк. 15,11.
Обл.-вид. арк. 12,41. Тираж 120 прим. Ціна договірна. Замовл.

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

Свідоцтво суб'єкта видавничої справи ДК № 544 від 27.07.2001

НАЦІОНАЛЬНА АКАДЕМІЯ НАУК УКРАЇНИ
ІНСТИТУТ ПРОГРАМНИХ СИСТЕМ

ПРОБЛЕМИ ПРОГРАМУВАННЯ

науковий журнал

№ 3 липень – вересень 2019

Заснований у березні 1999 р.

ЗМІСТ

Свідоцтво про державну реєстрацію КВ № 7490 від 01.07.2003

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

ISSN 1727-4907

NATIONAL ACADEMY OF SCIENCES OF UKRAINE
INSTITUTE OF SOFTWARE SYSTEMS

PROBLEMS IN PROGRAMMING

scientific journal

№ 3 July – September 2019

Founded in March, 1999

CONTENTS

УДК 004.8 https://doi.org/10.15407/pp2019.03.003

А.Ф. Кургаев

МОДИФИКАЦИЯ МЕТАЯЗЫКА НОРМАЛЬНЫХ ФОРМ ЗНАНИЙ

Набор базовых отношений метаязыка нормальных форм знаний (альтернатива, конкатенация, отрицание и итерация) расширен двумя новыми отношениями: обязательности (не нулевым числом повторений), необязательности некоторой структуры и структурными скобками. Введение новых отношений выполнено описанием их структур в базовых отношениях метаязыка нормальных форм знаний. Даны текстовая и графическая формы самоописания модифицированного метаязыка нормальных форм знаний, расширенного стилистическими отношениями описания информационных структур. Приведены примеры графических вариантов представления структур новых отношений терминов, использованных в самоописании модифицированного метаязыка нормальных форм знаний.

Ключевые слова: модифицированный метаязык, самоописание метаязыка, граф самоописания метаязыка, отношение обязательности, отношение необязательности, структурные скобки.

Введение

Цель создания систем обработки знаний – дать возможность любому пользователю ставить и решать на компьютере проблемы, трудные для обычного программирования, – "создать одну систему для всех классов пользователей непросто, но это должно быть сделано" [1].

Нужна концепция построения систем, все функции (приобретение, передача, представление и использование) обработки знаний которых связаны едиными механизмами [2]. Приоритет концептуального единства утверждается и в [3, с. 36–38]: "концептуальное единство является самым важным соображением при проектировании системы. … Каждая часть должна использовать одну и ту же технику синтаксиса и одинаковые понятия в семантике ... простота в использовании диктует требования единообразия, то есть концептуального единства при проектировании". По-сути, тем самым на компьютерные системы распространено требование простоты объяснительных теорий [4].

Эта цель достижима лишь на пути создания информационного компьютера с внутренним языком, достаточным для эффективного представления и использования разнообразной информации и способным овладеть всей нашей системой знаний в процессе общения с людьми и реальным миром.

Для строгого и точного описания синтаксических структур языков программирования используют специальные метаязыки (языки для описания других языков) [5]. Наиболее распространенными метаязыками являются Бекусо – Науровы формы (BNF) и EBNF (extended BNF) [6]. Однако, в качестве языка представления знаний метаязык EBNF (и все другие известные метаязыки) не является функционально полным и потому не пригоден для представления произвольных знаний.

Для достижения функциональной полноты метаязык нормальных форм знаний (НФЗ) для описания произвольных знаний наделен операциями анализа и порождения над произвольными структурами понятий, связанных отношениями конкатенации (последовательности), альтернативы, отрицания и итерации [7–9]. Эти, базовые отношения, дополняют стилистическими для упрощения описания и восприятия структур понятий. Например, метаязык EBNF, кроме базовых отношений, содержит отношения [6]: необязательности (необязательный элемент выделяют квадратными скобками); обязательности – одного или большего числа вхождений элемента (обязательный элемент выделяют надстрочным знаком +, примыкающим к этому элементу) и структурные круглые скобки.

Для расширения выразительных возможностей метаязыка НФЗ представляется важным распространить операционные возможности метаязыка НФЗ на отношения в описании информационных структур, принятые в известных метаязыках. В связи с этим, цель статьи –это расширение множества отношений метаязыка НФЗ сервисными отношениями обязательности, необязательности некоторой структуры и структурными скобками.

1. Текстовое самоописание модифицированиого метаязыка

Текстовое самоописание метаязыка НФЗ, расширенного отношениями обязательности, необязательности некоторой структуры и структурными скобками, представляется такой структурой:

1. description = (: determination :) ;
2. determination = [negativ] nameConcept definition bodyDeterm endDeterm;
3. nameConcept = identifier / integer / chainSigns;
4. identifier = letter (letter / decimalDigit);
5. integer = (: decimalDigit :);
6. chainSigns = (: ^metaSign sign :);
7. bodyDeterm = structure / terminal;
8. terminal = (: space :);
9. structure = singleDefinit (separator singleDefinit);
10. singleDefinit = [negativ] primary (concatenate [negativ] primary);
11. primary = groupedSeq / optionalSeq / iterationSeq / mandatorySeq / {nameConcept / line} [analysis / traceAnalysis / generation];
12. optionalSeq = startOptionSymb bodyDeterm endOptionSymb;
13. groupedSeq = startGroupSymb bodyDeterm endGroupSymb;
14. iterationSeq = startIterationSymb bodyDeterm endIterationSymb;
15. mandatorySeq = startMandatorySymb bodyDeterm endMandatorySymb;
16. line = quotationMark nameConcept quotationMark;

где definition – разделитель двух частей определения, изображается символом '=';

separator – отношение альтернативного выбора изображается символом '/';

concatenate – отношение конкатенации изображается символом space ' ';

startIterationSymb, endIterationSymb – пара скобок '(' и ')', обрамляющих итерируемый элемент;

startMandatorySymb, endMandatorySymb – пара скобок '(:' и ':)', обрамляющих обязательный элемент (не нулевое число его повторений);

startOptionSymb, endOptionSymb – пара скобок '[' и ']', обрамляющих необязательный элемент;

startGroupSymb, endGroupSymb – пара структурных скобок '{' и '}';

negativ – отношение отрицания изображается символом '^';

endDeterm – конец определения изображается символом ';';

quotationMark – текстовая кавычка, изображается символом '";

analysis – режим анализа изображается символом ‘?’;

traceAnalysis – режим анализа со следом изображается символом '#';

generation – режим порождения изображается символом '!';

letter = 'A' / 'B' / 'C' /…/ 'Z' / 'a' / 'b' / 'c' /.../ 'z';

decimalDigit = '0' / '1' / '2' / '3' / '4' / '5' / '6' / '7' / '8' / '9';

sign = '-' / '&' / '%' / '$' / '@' / '~' / ':' / '<' / '>' / … / ',' / '.' / '_';

metaSign = '(' / ')' / space / '/' / '=' / '?' / '#' / '!' / ';' / '"/ '"' / '{' / '}'.

Приведенное самоописание метаязыка НФЗ состоит из непустой последовательности определений, в каждом из которых нетерминал, заданный слева от разделителя definition именем понятия

(возможно, с предшествующим отрицанием), определяется некоторой структурой (именованной bodyDeterm) отношений нетерминалов и терминалов, указанной справа от знака definition. Набор отношений метаязыка НФЗ (альтернатива, конкатенация, отрицание и итерация) расширен в модифицированном метаязыке НФЗ двумя новыми отношениями: не нулевым числом повторений некоторой структуры, необязательности некоторой структуры и структурными скобками (groupedSeq).

При этом, любая structure – это альтернатива конкатенаций, возможно, отрицаемых первичных (primary), в качестве которых может быть обязательно, необязательно, сгруппирован или повторено произвольное число раз bodyDeterm, или, даже, неименованная последовательность. Первый элемент этой последовательности – это неименованная альтернатива терминов (nameConcept и line), а второй – неименованная альтернатива необязательных терминов analysis, traceAnalysis и generation.

2. Граф самоописания модифицированиого метаязыка НФЗ

Используя графические средства метаязыка НФЗ (см. рис. 1), в форме рис. 2 и 3 разработан граф самоописания модифицированиого метаязыка НФЗ, эквивалентный приведенному текстовому.

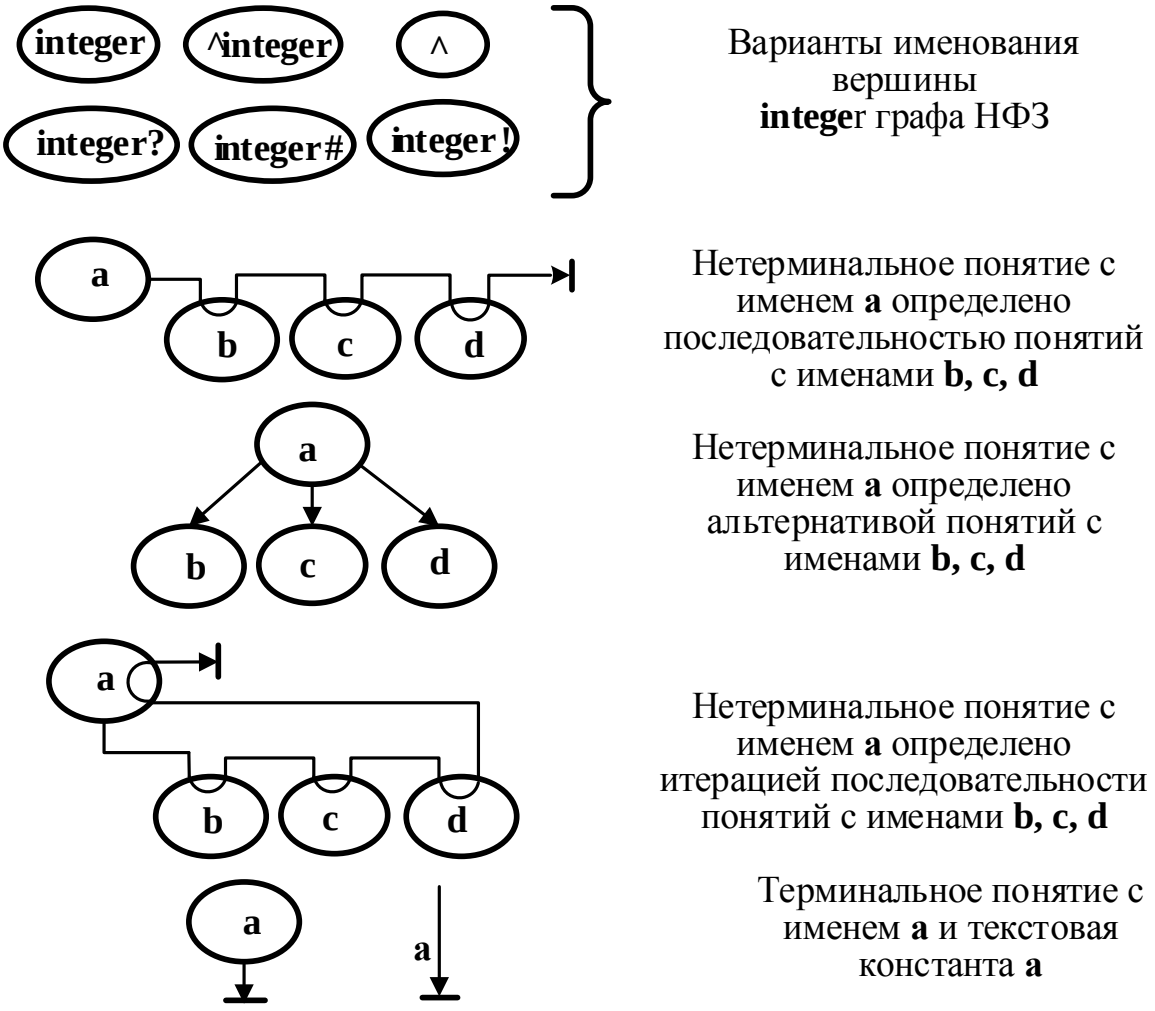


Рис. 1. Графические средства метаязыка НФЗ [6–8]

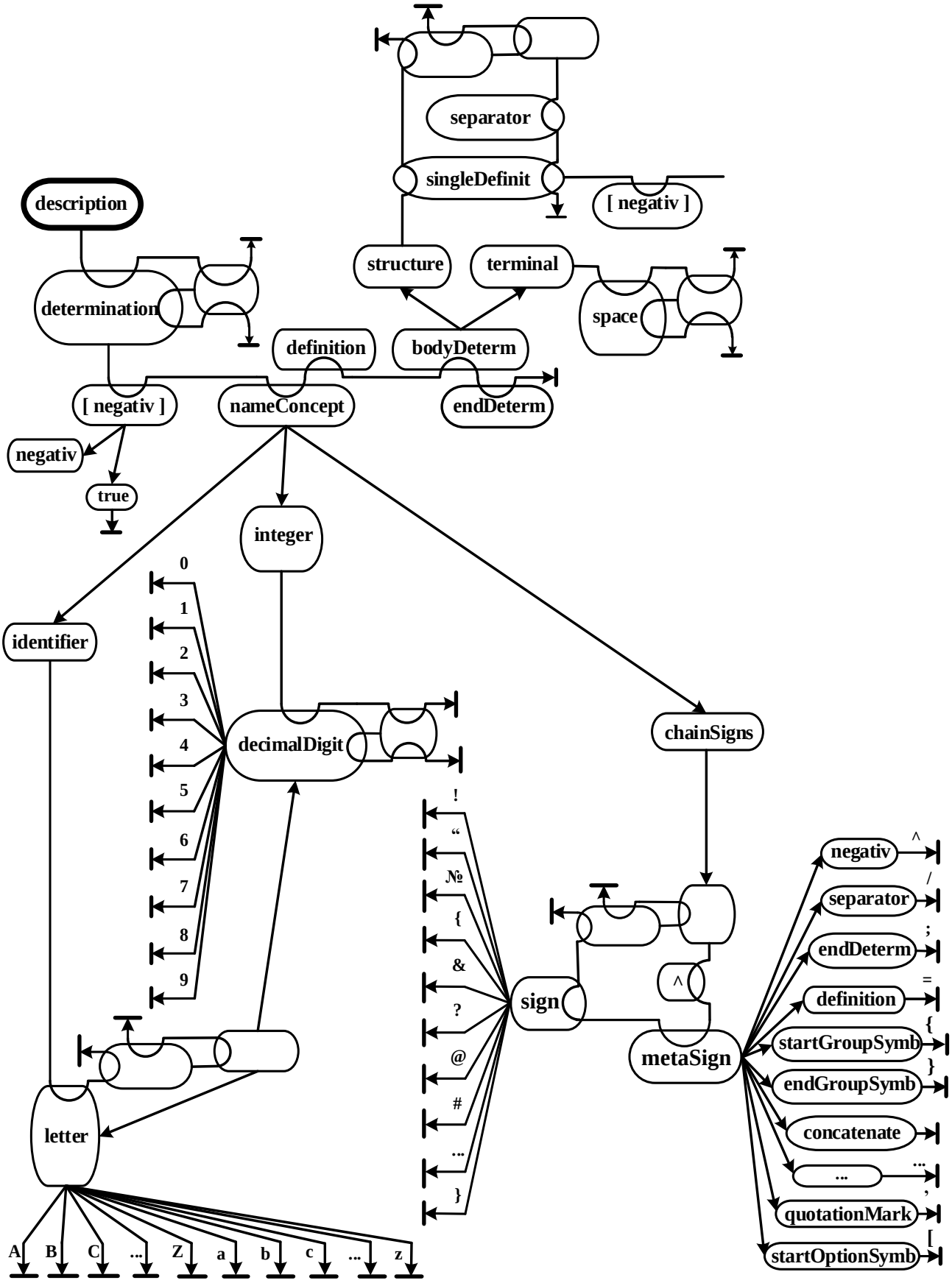


Рис. 2. Первая страница графа самоописания расширенного метаязыка НФЗ

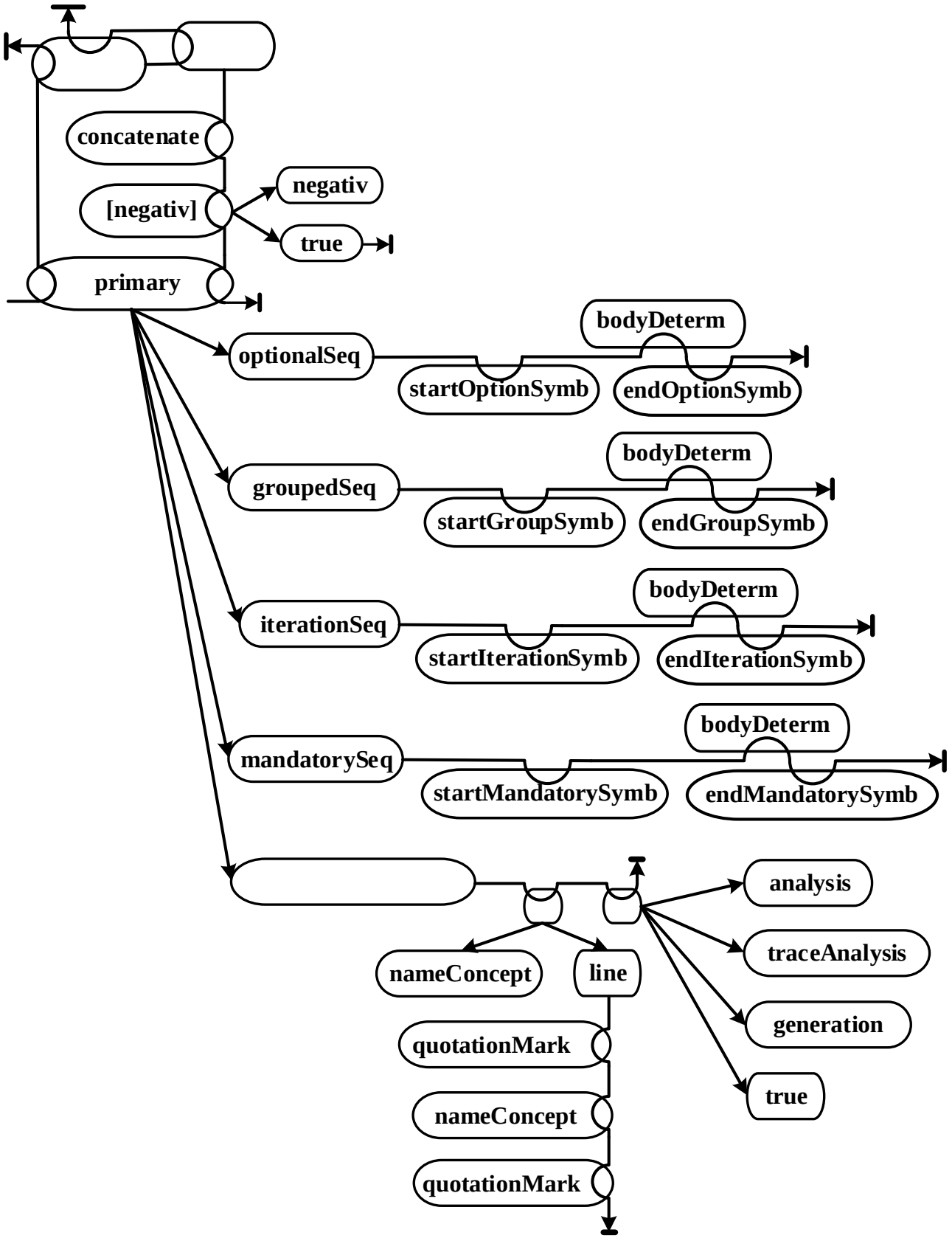


Рис. 3. Вторая страница графа самоописания расширенного метаязыка НФЗ

3. Графическое представление новых отношений расширенного метаязыка НФЗ

Графические средства модифицированного метаязыка НФЗ развивают графические средства метаязыка НФЗ. Примеры характерных вариантов симметричных изображений графов новых отношений, использованных в самоописании модифицированного метаязыка НФЗ показаны на рис. 4 – 8.

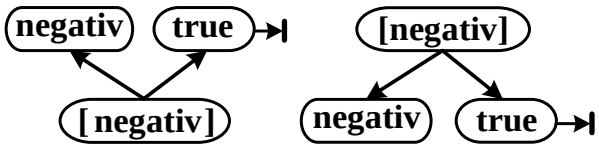


Рис. 4. Симметричные варианты графа необязательности (optionalSeq) термина negativ

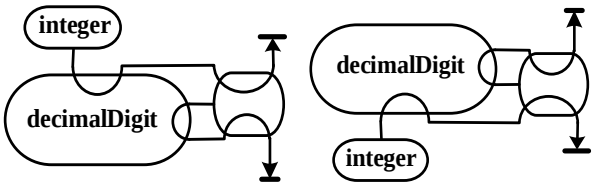


Рис. 5. Симметричные варианты графа обязательности (mandatorySeq) термина decimalDigit, использованном в текстовом определении:
integer = (: decimalDigit :);

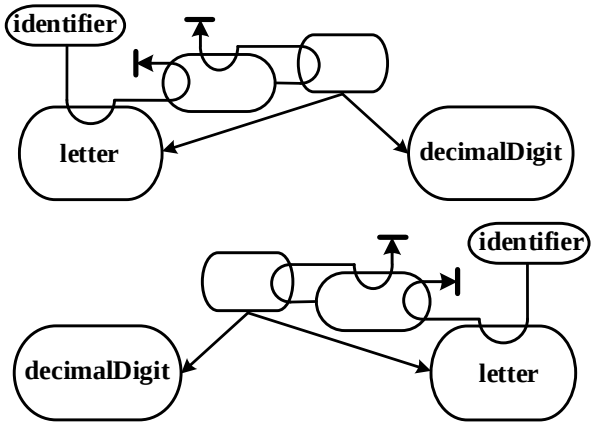
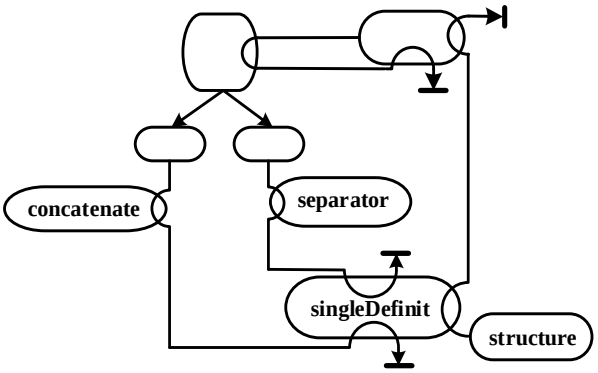


Рис. 6. Симметричные варианты графа определения идентификатора identifier, начинающегося буквой, с последующим повторением (произвольное число раз) неименованной альтернативы терминов letter и decimalDigit



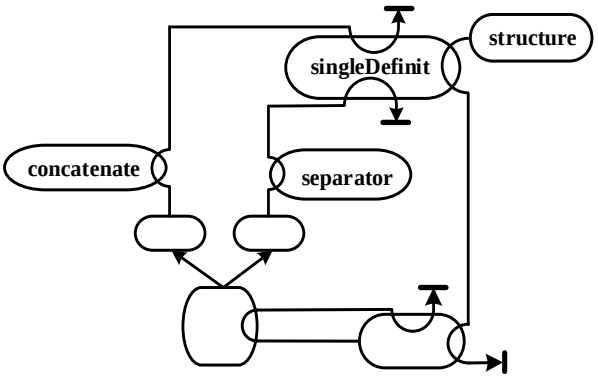


Рис. 7. Симметричные варианты графа описания термина structure:
structure = singleDefinit (separator singleDefinit);

Определение термина structure начинается термином singleDefinit и продолжается повторением (произвольное число раз) неименованной последовательности separator и singleDefinit, определенного ненулевой последовательностью возможно негативных терминов primary.

В свою очередь, primary есть альтернатива именованных терминов (groupedSeq, optionalSeq, iterationSeq, mandatorySeq) и неименованной последовательности, первым элементом которой является неименованная альтернатива терминов (nameConcept и line), а вторым – неименованная альтернатива необязательных терминов analysis, traceAnalysis и generation.

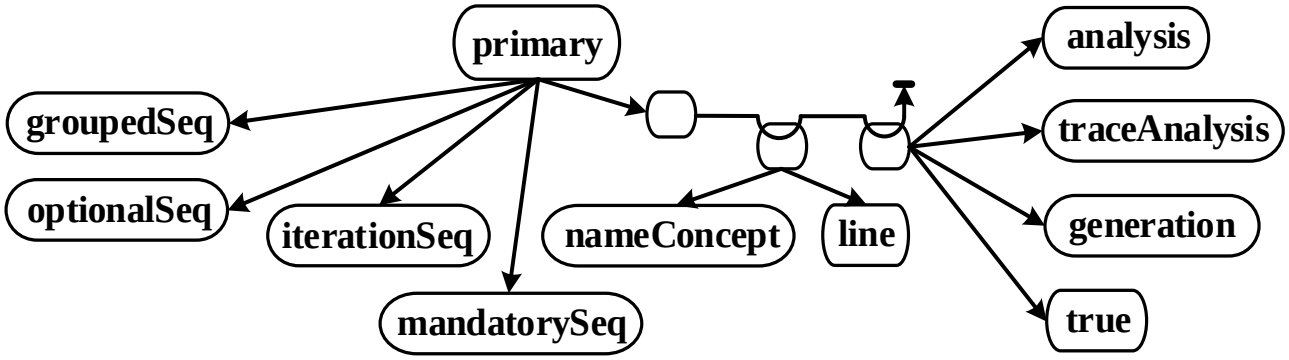


Рис. 8. Граф описания термина primary

Заключение

Предложено формальное описание метаязыка нормальных форм знаний, расширенного стилистическими отношениями описания информационных структур. Показано, что введение новых отношений осуществляется достаточно просто – описанием их структур в базовых отношениях метаязыка НФЗ. Представленный вариант расширения метаязыка НФЗ не исчерпывает всех возможностей расширения этого метаязыка.

Литература

1. Норман Д. Память и научение. Пер. с англ. М.: Мир, 1985. 160 с.
2. Осуга С. Обработка знаний. Пер. с яп. М.: Мир, 1989. 292 с.
3. Брукс Фредерик. Мифический человеко-месяц или как создаются программные системы. https://nsu.ru/xmlui/bitstream/handle/nsu/8870/Frederick_Brooks.pdf
4. Поппер К. Логика и рост научного знания. Избранные работы. Пер. с англ. М.: Прогресс, 1983. 496 с.
5. SoftCraft разноликое программирование: Основы разработки трансляторов. http://sl-ur.narod2.ru/studentu/vtoroi_semestr_2010_g/pyavu/Osnovy_razrabotki_translyatorov_.rar
6. International Standard ISO/IEC 14977: 1996(E). Электронный ресурс. Режим доступа: http://www.cl.cam.ac.uk/~mgk25/iso-14977.pdf
7. Спосіб представлення і використання знань. О.П. Кургаєв, С.М. Григор'єв. Патент на корисну модель UA 92484 U, 2014, Бюл. № 16.
8. Кургаев А.Ф., Григорьев С.Н. Нормальные формы знаний. Допов. нац. акад. наук Укр. 2015. № 11. С. 36–43. doi: https://doi.org/10.15407/dopovidi2015.11.036
9. Кургаев А.Ф., Григорьев С.Н. Метаязык нормальных форм знаний. *Кибернетика и системный анализ.* 2016. Том. 52. № 6. С. 11–20. DOI 10.1007/s10559-016-9885-3

References

1. Norman, Donald A. (1985). LEARNING AND MEMORY. W. H. Freeman and Company San Francisco.
2. Osuga, S. (1989) Treatment of knowledge. Мир, Moskow (in Russian).
3. Brooks Frederick P .Jr. The Mythical Man-Month. https://is.muni.cz/www/jirqa/The.Mythical.Man.Month.F.Brooks.pdf
4. Popper, K.: Evolutionary Epistemology. Evolutionary Theory: Paths into the Future. Ed. by J. W. Pollard. Ch. 10, pp. 239-255. Wiley, Chichester and New York (2002)
5. SoftCraft various programming: Bases of development of translators. http://sl-ur.narod2.ru /studentu/vtoroi_semestr_2010_g/pyavu/Osnovy_razrabotki_translyatorov_.rar
6. International Standard ISO/IEC 14977: 1996(E). Retrieved from http://www.cl.cam.ac.uk/~mgk25/iso-14977.pdf
7. Kurgaev A., Grygoryev S. Utility model patent UA 92484 U, 2014, Bulletin No 16 (in Ukrainian).

8. Kurgaev, A. & Grygoryev, S. (2015). The normal forms of knowledge. Dopov. nac. akad. nauk Ukr. N 11. P. 36-43 (in Russian). doi: https://doi.org/10.15407/dopovidi2015.11.036
9. Kurgaev, A., Grygoryev, S. (2016). Metalanguage of Normal Forms of Knowledge. Cybernetics and Systems Analysis. 52(6), 839-848. doi: http://link.springer.com/article/10.1007/s10559-016-9885-3.

Получено 13.05.2019

Об авторе:

Кургаев Александр Филиппович,
доктор технических наук,
профессор, ведущий научный сотрудник.
Количество научных публикаций в украинских изданиях – более 240.
Количество научных публикаций в зарубежных индексированных изданиях – 22,
h-індекс (Google Scholar): 6
http://orcid.org/0000-0001-5348-2734.

Место работы автора:

Институт кибернетики имени
В.М. Глушкова НАН Украины,
03187, Киев-187,
проспект Академика Глушкова, 40.
E-mail: afkurgaev@ukr.net

УДК 004.42:510.69 https://doi.org/10.15407/pp2019.03.011

О.С. Шкільняк

ВІДНОШЕННЯ ЛОГІЧНОГО НАСЛІДКУ В ЛОГІКАХ ЧАСТКОВИХ ПРЕДИКАТІВ З КОМПОЗИЦІЄЮ ПРЕДИКАТНОГО ДОПОВНЕННЯ

Досліджено нові програмно-орієнтовані логіки часткових предикатів з операцією (композицією) предикатного доповнення, такі логіки названо LC. Для першопорядкових LC запропоновано низку відношень логічного наслідку та відношень логічного наслідку за умови невизначеності. Досліджено властивості цих відношень, встановлено співвідношення між ними. Для відношень типів $|=_T$ та $|=_F$ доведено теорему про елімінацію умов невизначеності. Для запропонованих відношень описано умови їх гарантованої наявності, наведено властивості декомпозиції формул та елімінації кванторів.

Ключові слова: логіка, частковий предикат, композиційна алгебра, логічний наслідок.

Вступ

Розвиток інформаційних технологій та розширення сфери їх застосування роблять першочерговою задачу створення надійних і ефективних програмних систем. Недавні авіакатастрофи з літаками Boeing 737 MAX 8 яскраво це засвідчують. В основі таких систем лежить апарат математичної логіки (див., напр., [1]). Зазвичай це класична логіка [2] предикатів та базовані на ній спеціальні логіки. Проте класична логіка має [3] принципові обмеження, які ускладнюють її використання. Тому проблема побудови нових, програмно-орієнтованих логічних формалізмів набуває особливої актуальності. Такими формалізмами є композиційно-номінативні логіки (КНЛ) часткових квазіарних предикатів. Описано та досліджено [3–6] низку класів КНЛ на різних рівнях абстрактності й загальності.

Важливим різновидом програмно-орієнтованих логік, які успішно застосовуються в системах верифікації програм, є логіки Флойда-Хоара [7]. Ці логіки використовують тотальні перед- та після-умови (предикати), тому запропоновано [8, 9] їх розширення на випадок часткових предикатів. Введення спеціальної операції (композиції) предикатного доповнення [9] є продуктивним напрямком такого розширення. КНЛ часткових предикатів з композицією предикатного доповнення запропоновано в [10], їх названо LC. Дослідженню пропозиційних LC присвячена робота [10].

Метою даної роботи є вивчення семантичних властивостей першопорядкових LC. Запропоновано низку відношень логічного наслідку в LC та відношень логічного наслідку за умови невизначеності. Досліджено властивості цих відношень, встановлено співвідношення між цими. Для відношень типів $|=_T$ та $|=_F$ доведено теорему про елімінацію умов невизначеності. Для цих відношень описано умови їх гарантованої наявності, наведено властивості декомпозиції формул та елімінації кванторів.

Поняття, які тут не визначаються, будемо тлумачити в сенсі [3, 5].

1. Композиційні системи логік з предикатним доповненням

Під V-A-квазіарним предикатом розуміємо часткову неоднозначну функцію вигляду $P : {}^{V}A \to \{T, F\}$. Тут $\{T, F\}$ – множина істиннісних значень, ${}^{V}A$ – множина всіх V-A-іменних множин.

Множину всіх значень, які неоднозначний предикат P може приймати на аргументі (даному) $d \in {}^{V}A$, позначаємо $P[d]$.

В цій роботі розглядаємо неоднозначні (недетерміновані) предикати реляційного типу, або R-предикати.

Кожний R-предикат Q можна однозначно задати за допомогою 2-х множин:

– область істинності $T(Q) = \{d \mid T \in Q[d]\}$;

– область хибності $F(Q) = \{d \mid F \in Q[d]\}$.

Область невизначеності R-предиката Q визначається через $T(Q)$ та $F(Q)$ так:

$$\perp(Q) = \overline{T(Q) \cup F(Q)} = \overline{T(Q)} \cap \overline{F(Q)}.$$

V-A-квазіарний *R*-предикат *Q*:

– частковий однозначний або *P*-предикат, якщо $T(Q)\cap F(Q)=\varnothing$;

– тотальний, або *T*-предикат, якщо $T(Q)\cup F(Q)={}^{V}A$; тоді $\perp(Q)=\varnothing$.

Для *P*-предикатів пишемо $Q(d)\downarrow$, якщо значення $Q(d)$ визначене, та пишемо $Q(d)\uparrow$, якщо значення $Q(d)$ невизначене.

Тому для *P*-предикатів:

$$T(Q)=\{d \mid Q(d)\downarrow = T\},$$

$$F(Q)=\{d \mid Q(d)\downarrow = F\}.$$

Тоді $\perp(Q)=\{d \mid Q(d)\uparrow\}$.

Класи *V-A*-квазіарних *R*-предикатів та *P*-предикатів позначимо Pr^{V-A} та PrP^{V-A}. Маємо [5] 4 константних *R*-предикати T, F, $\perp$, Υ: $T(\mathrm{T})=F(\mathrm{F})={}^{V}A$, $T(\mathrm{F})=F(\mathrm{T})=\varnothing$, $T(\perp)=F(\perp)=\varnothing$, $T(\Upsilon)=F(\Upsilon)={}^{V}A$.

Опишемо композиції *R*-предикатів.

Композиції пропозиційного рівня називають логічними зв'язками. За базові логічні зв'язки *R*-предикатів візьмемо $\neg$ та $\vee$, вони задаються так:

$$T(\neg P)=F(P);\ F(\neg P)=T(P);$$

$$T(P\vee Q)=T(P)\cup T(Q);\ F(P\vee Q)=F(P)\cap F(Q).$$

Композиції $\rightarrow$, & є *похідними*, вони виражаються через базові композиції $\neg$, $\vee$:

$$P\&Q=\neg(\neg P\vee\neg Q);\ P\rightarrow Q=\neg P\vee Q.$$

Для *R*-предикатів $\perp(\neg Q)=\perp(Q)$;

$$\perp(P\vee Q)=(\perp(P)\cap\overline{T(Q)})\cup(\perp(Q)\cap\overline{T(P)}).$$

1-арна композиція реномінації $\mathrm{R}_{\bar{x}}^{\bar{v}}: Pr^{V-A}\rightarrow Pr^{V-A}$ задається умовою:

$$\mathrm{R}_{\bar{x}}^{\bar{v}}(P)[d]=P\,[\mathrm{r}_{\bar{x}}^{\bar{v}}(d)].$$

Тут $\mathrm{r}_{\bar{x}}^{\bar{v}}$ – операція реномінації [3, 5].

$\mathrm{R}_{\bar{x}}^{\bar{v}}$ можна визначити через області істинності та хибності предиката $\mathrm{R}_{\bar{x}}^{\bar{v}}(P)$:

$$T(\mathrm{R}_{\bar{x}}^{\bar{v}}(P))=\{d \mid \mathrm{r}_{\bar{x}}^{\bar{v}}(d)\in T(P)\};$$

$$F(\mathrm{R}_{\bar{x}}^{\bar{v}}(P))=\{d \mid \mathrm{r}_{\bar{x}}^{\bar{v}}(d)\in F(P)\}.$$

Визначальними для першопорядкових логік є 1-арні композиції *квантифікації*. За базову візьмемо $\exists x$, задамо її так:

$$T(\exists xP)=\bigcup_{a\in A}\{d \mid d\nabla x\mapsto a\in T(P)\};$$

$$F(\exists xP)=\bigcap_{a\in A}\{d \mid d\nabla x\mapsto a\in F(P)\}.$$

Композиція $\forall x$ є *похідною*:

$$\forall xP=\neg\exists x\neg P.$$

Маємо $\perp(\exists xP)=$

$$=\overline{T(\exists xP)\cup F(\exists xP)}=\overline{T(\exists xP)}\cap\overline{F(\exists xP)}.$$

Звідси $\perp(\exists xP)=$

$$=\bigcap_{a\in A}\{d \mid d\nabla x\mapsto a\in F(P)\cup\perp(P)\}\cap$$
$$\cap\bigcup_{a\in A}\{d \mid d\nabla x\mapsto a\in\perp(P)\}.$$

Зокрема, для *P*-предикатів маємо

$$\perp(\exists xQ)=\bigcap_{a\in A}\{d \mid Q(d\nabla x\mapsto a)\neq T\}\cap$$
$$\cap\bigcup_{a\in A}\{d \mid d\nabla x\mapsto a\in\perp(P)\}.$$

Таким чином, для *P*-предикатів:

$\exists xQ(d)\uparrow \Leftrightarrow Q(d\nabla x\mapsto b)\uparrow$ для деякого $b\in A$ та неможливо $\exists xQ(d)=T$.

Характерна особливість LC – наявність спеціальної немонотонної 1-арної композиції предикатного доповнення $\sim$.

Композицію $\sim$ можна задати так:

$$T(\sim P)=\perp(P)=\overline{T(P)\cup F(P)}=\overline{T(P)}\cap\overline{F(P)},$$

$$F(\sim P)=\varnothing.$$

Тоді маємо $\perp(\sim P)=T(P)\cup F(P)$.

Для *P*-предикатів $\sim$ задається так:

$$(\sim P)(d)=\begin{cases} T, & \text{якщо } P(d)\uparrow,\\ \text{невизначене}, & \text{якщо } P(d)\downarrow.\end{cases}$$

Отже, композиція $\sim$ немонотонна.

Класи *P*-предикатів та *R*-предикатів замкнені щодо композицій $\neg$, $\vee$, $\rightarrow$, &, $\mathrm{R}_{\bar{x}}^{\bar{v}}$, $\exists xQ$, $\forall xP$, $\sim$.

$\sim Q$ є *P*-предикатом для довільного $Q\in Pr^{V-A}$, тому клас *T*-предикатів незамкнений щодо $\sim$. Отже, для *T*-предикатів LC не мають смислу.

Тому розглядаємо загальний клас LC – логіки *R*-предикатів з композицією $\tilde{}$, та їх підклас LCP – LC *P*-предикатів.

LC пропозиційного, реномінативного, чистого першопорядкового рівнів називаємо PLC, RLC, QLC. LC *P*-предикатів відповідних рівнів називаємо PLCP, RLCP, QLCP. Множини базових композицій PLC, RLC, QLC – це множини $C_{PLC}=\{\neg,\vee,\tilde{}\}$, $C_{RLC}=\{\neg,\vee,\mathrm{R}^{\bar{v}}_{\bar{x}},\tilde{}\}$, $C_{QLC}=\{\neg,\vee,\mathrm{R}^{\bar{v}}_{\bar{x}},\tilde{},\exists x\}$.

Семантичною основою LC є композиційні предикатні системи вигляду CS = (^{V}A, Pr^{V-A}, C_{LC}), де C_{LC} – множина базових композицій відповідного рівня. Отже, маємо композиційні LC-системи: пропозиційну (^{V}A, Pr^{V-A}, C_{PLC}), реномінативну (^{V}A, Pr^{V-A}, C_{RLC}), чисту першопорядкову (^{V}A, Pr^{V-A}, C_{QLC}). Відповідно отримуємо композиційні LC-алгебри *R*-предикатів: реномінативну $APLC=(Pr^{V-A}, C_{PLC})$, реномінативну $ARLC=(Pr^{V-A}, C_{RLC})$, чисту першопорядкову $AQLC=(Pr^{V-A}, C_{QLC})$.

Клас *P*-предикатів замкнений щодо $\tilde{}$, тому в *APLC*, *ARLC*, *AQLC* виділені підалгебри *APLCP*, *ARLCP*, *AQLCP* з носієм PrP^{V-A} – композиційні LC-алгебри *P*-предикатів пропозиційного, реномінативного, чистого першопорядкового рівнів.

Опишемо основні властивості композицій LC.

Не пов'язані з $\tilde{}$ властивості пропозиційних композицій, композицій реномінації та квантифікації в LC аналогічні властивостям цих композицій у традиційних логіках квазіарних предикатів (див. [3–5]).

На пропозиційному рівні властивості композиції $\tilde{}$ описано в [10]. Зокрема:

$$\tilde{}\neg P=\tilde{}P;\quad \tilde{}\tilde{}\tilde{}P=\tilde{}P;\quad \tilde{}\tilde{}\tilde{}\tilde{}P=\tilde{}\tilde{}P.$$

Для *P*-предикатів додатково маємо

$$\tilde{}\tilde{}P=P\vee\neg P.$$

Твердження 1. Маємо

$$F(\neg\tilde{}P)=T(\tilde{}P)=\perp(P)=\overline{T(P)}\cap\overline{F(P)},$$

$$T(\neg\tilde{}P)=F(\tilde{}P)=\varnothing,$$

$$\perp(\neg\tilde{}P)=\perp(\tilde{}P)=T(P)\cup F(P);$$

$$T(\mathrm{R}^{\bar{v}}_{\bar{x}}(\tilde{}P))=T(\tilde{}(\mathrm{R}^{\bar{v}}_{\bar{x}}(P))),$$

$$\perp(\mathrm{R}^{\bar{v}}_{\bar{x}}(\tilde{}P))=\perp(\tilde{}(\mathrm{R}^{\bar{v}}_{\bar{x}}(P))),$$

$$F(\mathrm{R}^{\bar{v}}_{\bar{x}}(\tilde{}P))=F(\tilde{}(\mathrm{R}^{\bar{v}}_{\bar{x}}(P)))=\varnothing.$$

Твердження 2. Властивість R$\tilde{}$-дистрибутивності:

$$\mathrm{R}\tilde{}\ )\ \mathrm{R}^{\bar{v}}_{\bar{x}}(\tilde{}P)=\tilde{}\mathrm{R}^{\bar{v}}_{\bar{x}}(P).$$

Твердження 3. Маємо

$$\perp(\tilde{}\exists xP)=T(\exists xP)\cup F(\exists xP);$$

$$T(\tilde{}\exists xP)=\perp(\exists xP);\quad F(\tilde{}\exists xP)=\varnothing.$$

Твердження 4. Дія композиції $\tilde{}$ на константні предикати T, F, $\perp$, Υ:

$$\tilde{}(\perp)=\mathrm{T};\ \tilde{}(\mathrm{T})=\tilde{}(\mathrm{F})=\tilde{}(\Upsilon)=\perp.$$

Отже, множини $S_R=\{\mathrm{T},\mathrm{F},\perp,\Upsilon\}$ та $S_P=\{\mathrm{T},\mathrm{F},\perp\}$ замкнені щодо $\neg,\vee,\mathrm{R}^{\bar{v}}_{\bar{x}},\exists x,\ \tilde{}$.

Тому можна виділити підалгебри $AQsC=(S_R, C_{QLC})$ та $AQsCP=(S_P, C_{QLC})$ алгебр *AQLC* та *AQLCP*. Аналогічні алгебри *APsC*, *ARsC* з носієм S_R та *APsCP*, *ARsCP* з носієм S_P виділяємо на пропозиційному та реномінативному рівнях; для них маємо $APsC\prec APLC$, $ARsC\prec ARLC$, $APsCP\prec APLCP$, $ARsCP\prec ARLCP$.

Опишемо мови логік з композицією предикатного доповнення. Розглядаємо загальний випадок першопорядкових LC.

Алфавіт мови:

– множина *V* предметних імен (змінних),

– множина *Ps* предикатних символів,

– множина $C_{QLC}=\{\neg,\vee,\tilde{},R^{\bar{v}}_{\bar{x}},\exists x\}$ символів базових композицій.

Задамо множину *Fr* формул мови. Маємо $Ps\subseteq Fr$, далі задаємо індуктивно:

$$\Phi,\Psi\in Fr\Rightarrow\neg\Phi,\vee\Phi\Psi,\tilde{}\Phi,R^{\bar{v}}_{\bar{x}}\Phi,\exists x\Phi\in Fr.$$

Розширена сигнатура мови QLC – це $\Sigma=(V, V_T, C_{QLC}, Ps)$, де $V_T\subseteq V$ – множина тотально неістотних [5] імен.

Інтерпретуємо мову на композиційних системах CS = $({}^{V}A, Pr^{V-A}, C_{QLC})$.

Задаємо тотальне однозначне $I : Ps \to Pr^{V-A}$, яке продовжимо до відображення інтерпретації формул $I : Fr \to Pr^{V-A}$:

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

$$I(\tilde{}\Phi) = \tilde{}(I(\Phi)),$$

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

Інтерпретація мови сигнатури Σ – це $\boldsymbol{J} = (\mathrm{CS}, \Sigma, I)$; скорочено її позначаємо (A, I). Предикат $\boldsymbol{J}(\Phi)$ позначимо Φ_J.

Для довільної $\Gamma \subseteq Fr$ позначимо $\sigma(\Gamma)$ множину всіх $p \in Ps$, які входять до складу $\Phi \in \Gamma$; позначимо $nm(\Gamma)$ множину всіх $x \in V$, які фігурують у формулах $\Phi \in \Gamma$.

Для формул мови QLC визначимо множини гарантовано неістотних імен за допомогою відображення $\nu : Fr \to 2^{V}$ так, як для традиційних логік квазіарних предикатів [3, 5], додаючи пункт $\nu(\tilde{}\Phi) = \nu(\Phi)$.

Для довільної $\Gamma \subseteq Fr$ задаємо

$$\nu(\Gamma) = \bigcap_{\Phi\in\Gamma} \nu(\Phi),\ fu(\Gamma) = V_T \setminus nm(\Gamma).$$

Подібним чином, опускаючи пункти для $\exists x$, описуємо мову RLC.

При описі мови PLC опускаємо пункти для $R^{\bar{v}}_{\bar{x}}$ та $\exists x$, а також поняття, пов'язані з неістотністю імен.

Виділення класів предикатів виділяє відповідні класи інтерпретацій, які називають [3, 5] семантиками. Зокрема, це *R*-семантика, *P*-семантика, *T*-семантика.

Клас *T*-предикатів незамкнений щодо $\tilde{}$, тому для LC *T*-семантика *малозмістовна*. Змістовними для LC є *R*-семантика та *P*-семантика, їх позначаємо $\boldsymbol{R_C}$ та $\boldsymbol{P_C}$.

Фіксуючи пропозиційний, реномінативний, чистий першопорядковий рівні розгляду, отримуємо семантики $\boldsymbol{R_{PC}}$ та $\boldsymbol{P_{PC}}$, $\boldsymbol{R_{RC}}$ та $\boldsymbol{P_{RC}}$, $\boldsymbol{R_{QC}}$ та $\boldsymbol{P_{QC}}$.

Властивості композицій LC можна подати із використанням формул мови.

Наведемо для прикладу основні властивості, пов'язані з реномінацією.

R) $\mathrm{R}(\Phi)\mathrm{J} = \Phi\mathrm{J}$;

RI) $R^{z,\bar{v}}_{z,\bar{x}}(\Phi)_J = R^{\bar{v}}_{\bar{x}}(\Phi)_J$;

RU) $R^{z,\bar{v}}_{y,\bar{x}}(\Phi)_J = R^{\bar{v}}_{\bar{x}}(\Phi)_J$, де $z \in \nu(\Phi)$;

R¬) $R^{\bar{v}}_{\bar{x}}(\neg\Phi)_J = \neg R^{\bar{v}}_{\bar{x}}(\Phi)_J$;

R∨) $R^{\bar{v}}_{\bar{x}}(\Phi \vee \Psi)_J = R^{\bar{v}}_{\bar{x}}(\Phi)_J \vee R^{\bar{v}}_{\bar{x}}(\Psi)_J$;

RR) $R^{\bar{v}}_{\bar{x}}(R^{\bar{w}}_{\bar{y}}(\Phi))_J = R^{\bar{v}}_{\bar{x}} \circ^{\bar{w}}_{\bar{y}} (\Phi)_J$;

$R\tilde{}$) $R^{\bar{v}}_{\bar{x}}(\tilde{}\Phi)_J = \tilde{} R^{\bar{v}}_{\bar{x}}(\Phi)_J$

$R\tilde{}$) $R^{\bar{v}}_{\bar{x}}(\tilde{}\Phi)_J = \tilde{} R^{\bar{v}}_{\bar{x}}(\Phi)_J$

Ren) $\exists y(\Phi)_J = \exists z\ R^{y}_{z}(\Phi)_J$, де $z \in \nu(\Phi)$;

$R\exists s$) $R^{\bar{v}}_{\bar{x}}(\exists y\Phi)_J = \exists y R^{\bar{v}}_{\bar{x}}(\Phi)_J$,

де $y \notin \{\bar{v}, \bar{x}\}$;

$R\exists$) $R^{\bar{v}}_{\bar{x}}(\exists y\Phi)_J = \exists z\ R^{\bar{v}}_{\bar{x}} \circ^{y}_{z} (\Phi)_J$,

де $z \in fu(R^{\bar{v}}_{\bar{x}}(\exists y\Phi))$.

2. Відношення логічного наслідку в LC

На множинах формул мови LC можна визначити низку відношень логічного наслідку. Зокрема, на мову LC природним чином поширимо відомі [5] відношення ${}^{P}|\!=_{IR}$, ${}^{P}|\!=_{T}$, ${}^{P}|\!=_{F}$, ${}^{P}|\!=_{TF}$, ${}^{R}|\!=_{TF}$. Такі відношення в LC будемо позначати ${}^{Pc}|\!=_{IR}$, ${}^{Pc}|\!=_{T}$, ${}^{Pc}|\!=_{F}$, ${}^{Pc}|\!=_{TF}$, ${}^{Rc}|\!=_{T}$, ${}^{Rc}|\!=_{F}$, ${}^{Rc}|\!=_{TF}$. При цьому істотні особливості вносить нова композиція предикатного доповнення.

Нехай деяка $\Sigma \subseteq Fr$, нехай $\boldsymbol{J}$ – інтерпретація. Далі позначаємо:

$\bigcap_{\theta\in\Sigma} T(\theta_J)$ як $T^{\cap}(\Sigma_J)$, $\bigcap_{\theta\in\Sigma} F(\theta_J)$ як $F^{\cap}(\Sigma_J)$,
$\bigcap_{\theta\in\Sigma} \bot(\theta_J)$ як $\bot^{\cap}(\Sigma_J)$, $\bigcup_{\theta\in\Sigma} T(\theta_J)$ як $T^{\cup}(\Sigma_J)$,
$\bigcup_{\theta\in\Sigma} F(\theta_J)$ як $F^{\cup}(\Sigma_J)$, $\bigcup_{\theta\in\Sigma} \bot(\theta_J)$ як $\bot^{\cup}(\Sigma_J)$.

У випадку, коли $\Sigma = \varnothing$, маємо:

$$T^{\cup}(\Sigma_J) = F^{\cup}(\Sigma_J) = \bot^{\cup}(\Sigma_J) = \varnothing,$$

$$T^{\frown}(\Sigma_J) = F^{\frown}(\Sigma_J) = \bot^{\frown}(\Sigma_J) = {}^{V}A.$$

Нехай $\Gamma, \Delta \subseteq Fr$.

Δ є T-наслідком Γ при інтерпретації $\boldsymbol{J}$ (позн. $\Gamma_J|\!=_T \Delta$), якщо $T^{\frown}(\Gamma_J) \subseteq T^{\smile}(\Delta_J)$.

Δ є F-наслідком Γ при інтерпретації $\boldsymbol{J}$ (позн. $\Gamma_J|\!=_F \Delta$), якщо $F^{\frown}(\Delta_J) \subseteq F^{\smile}(\Gamma_J)$.

Δ є TF-наслідком Γ при інтерпретації $\boldsymbol{J}$ (позн. $\Gamma_J|\!=_{TF} \Delta$), якщо

$$\Gamma_J|\!=_T \Delta \text{ та } \Gamma_J|\!=_F \Delta.$$

Δ є IR-наслідком Γ при інтерпретації $\boldsymbol{J}$ (позн. $\Gamma_J|\!=_{IR} \Delta$), якщо

$$T^{\frown}(\Gamma_J) \cap F^{\frown}(\Delta_J) = \varnothing.$$

Відповідні відношення логічного τ-наслідку в семантиці α задаємо за схемою:

$\Phi\ {}^{\alpha}|\!=\tau\ \Psi$, якщо $\Phi_J|\!=\tau\ \Psi$ для кожної $\boldsymbol{J} \in \alpha$.

Приклад 1. Нехай $\vartheta \in Ps$. Візьмемо $\boldsymbol{J} \in \boldsymbol{R_C}$ таку: $\vartheta_J = \Upsilon$; тоді $T(\vartheta_J) \cap F(\vartheta_J) = {}^{V}A$, тому $\vartheta_J|\!\neq_{IR} \vartheta$, звідки $\vartheta\ {}^{Rc}|\!\neq_{IR} \vartheta$.

Приклад 1 узагальнимо так.

Твердження 5. Нехай формули в Γ та Δ не мають входжень $\sim$, тоді $\Gamma\ {}^{Rc}|\!\neq_{IR} \Delta$.

Справді, візьмемо $\boldsymbol{J} \in \boldsymbol{R_C}$ таку, що $\vartheta_J = \Upsilon$ для кожного $\vartheta \in \sigma(\Gamma \cup \Delta)$, тоді маємо $\Gamma_J|\!\neq_{IR} \Delta$, звідки $\Gamma\ {}^{Rc}|\!\neq_{IR} \Delta$.

Приклад 2. Для кожних $\Phi \in Fr$ та $\boldsymbol{J} \in \boldsymbol{R_C}$ маємо $F(\sim\Phi_J) = \varnothing$, $T(\neg\sim\Phi_J) = \varnothing$.

Звідси $\Gamma\ {}^{Rc}|\!=_F \sim\Phi$, $\neg\sim\Phi\ {}^{Rc}|\!=_T \Delta$, $\Gamma\ {}^{Rc}|\!=_{IR} \sim\Phi$, $\neg\sim\Phi\ {}^{Rc}|\!=_{IR} \Delta$.

Приклад 3. Для кожних $\Phi \in Fr$ та $\boldsymbol{J} \in \boldsymbol{R_C}$ маємо $T(\Phi_J) \not\subset T(\sim\Phi_J) = \bot(\Phi_J)$ та $F(\Phi_J) \not\subset F(\neg\sim\Phi_J) = \bot(\Phi_J)$.

Звідси $\Phi\ {}^{Pc}|\!\neq_T \sim\Phi$ та $\neg\sim\Phi\ {}^{Pc}|\!\neq_F \Phi$, тому й $\Phi\ {}^{Rc}|\!\neq_T \sim\Phi$ та $\neg\sim\Phi\ {}^{Rc}|\!\neq_F \Phi$.

Таким чином, в LC відношення ${}^{Rc}|\!=_{IR}$, ${}^{Rc}|\!=_T$, ${}^{Rc}|\!=_F$, ${}^{Rc}|\!=_{TF}$ різні, водночас в традиційній логіці квазіарних предикатів маємо ${}^{R}|\!=_{IR} = \varnothing$ та ${}^{R}|\!=_T = {}^{R}|\!=_F = {}^{R}|\!=_{TF}$. При цьому в LC ${}^{R}|\!=_{IR} \neq \varnothing$, проте має дуже виродженний характер: якщо в Γ та Δ немає входжень $\sim$, то $\Gamma\ {}^{Rc}|\!\neq_{IR} \Delta$.

Приклад 4. Нехай $\vartheta, \xi \in Ps$. Візьмемо $\boldsymbol{I}, \boldsymbol{J} \in \boldsymbol{R_C}$ такі: $\vartheta_I = \Upsilon$, $\xi_I = \bot$, $\vartheta_J = \bot$, $\xi_J = \Upsilon$; тоді маємо: $\vartheta, \neg\vartheta_I|\!\neq_T \xi, \neg\xi$ та $\vartheta, \neg\vartheta_J|\!\neq_F \xi, \neg\xi$.

Звідси маємо: $\Phi, \neg\Phi\ {}^{Rc}|\!\neq_T \Psi, \neg\Psi$; $\Phi, \neg\Phi\ {}^{Rc}|\!\neq_F \Psi, \neg\Psi$; $\Phi, \neg\Phi\ {}^{Rc}|\!\neq_{TF} \Psi, \neg\Psi$. Зокрема: $\Phi, \neg\Phi\ {}^{Rc}|\!\neq_T \Delta$ та $\Gamma\ {}^{Rc}|\!\neq_T \Psi, \neg\Psi$; $\Phi, \neg\Phi\ {}^{Rc}|\!\neq_F \Delta$ та $\Gamma\ {}^{Rc}|\!\neq_F \Psi, \neg\Psi$.

Приклад 5. Для кожних $\Phi, \Psi \in Fr$ та $\boldsymbol{J} \in \boldsymbol{P_C}$ маємо $\Phi, \neg\Phi_J|\!=_{TF} \Psi, \neg\Psi$.

Звідси $\Phi, \neg\Phi\ {}^{Pc}|\!=_{TF} \Psi, \neg\Psi$.

Приклад 6. Для кожних $\Phi, \Psi \in Fr$ та $\boldsymbol{J} \in \boldsymbol{P_C}$ маємо $\Phi, \neg\Phi_J|\!=_T \Delta$ та $\Gamma_J|\!=_F \Psi, \neg\Psi$. Звідси $\Phi, \neg\Phi\ {}^{Pc}|\!=_T \Delta$ та $\Gamma\ {}^{Pc}|\!=_F \Psi, \neg\Psi$.

Приклад 7. Нехай $\vartheta, \xi \in Ps$. Візьмемо $\boldsymbol{I}, \boldsymbol{J} \in \boldsymbol{P_C}$ такі: $\vartheta_I = \bot$, $\xi_I = \mathrm{F}$, $\vartheta_J = \mathrm{T}$, $\xi_J = \bot$; тоді $\vartheta, \neg\vartheta\ {}^{Pc}|\!\neq_F \xi$ та $\vartheta\ {}^{Pc}|\!\neq_T \xi, \neg\xi$. Звідси $\Phi, \neg\Phi\ {}^{Pc}|\!\neq_F \Delta$ та $\Gamma\ {}^{Pc}|\!\neq_T \Psi, \neg\Psi$.

Приклад 8. Для довільних $\Phi, \vartheta \in Fr$ маємо $\Phi \vee (\vartheta \& \neg\vartheta)\ {}^{Pc}|\!=_{IR} \Phi\ \&\ (\vartheta \vee \neg\vartheta)$.

Для кожної $\boldsymbol{J} \in \boldsymbol{P_C}$ маємо:

$$T(\Phi \vee (\vartheta \& \neg\vartheta)_J) = T(\Phi_J),$$
$$F(\Phi\ \&\ (\vartheta \vee \neg\vartheta)_J) = F(\Phi_J).$$

Звідси $T(\Phi \vee (\vartheta \& \neg\vartheta)_J) \cap F(\Phi\ \&\ (\vartheta \vee \neg\vartheta)_J) = T(\Phi_J) \cap F(\Phi_J) = \varnothing$.

Приклад 9. Маємо
$\Phi \vee (\vartheta \& \neg\vartheta)\ {}^{Pc}|\!\neq_T \Phi\ \&\ (\vartheta \vee \neg\vartheta)$ та
$\Phi \vee (\vartheta \& \neg\vartheta)\ {}^{Pc}|\!\neq_F \Phi\ \&\ (\vartheta \vee \neg\vartheta)$.

Візьмемо $\Phi, \vartheta \in Ps$ та $\boldsymbol{J} \in \boldsymbol{P_C}$ такі: $T(\vartheta_J) \cup F(\vartheta_J) \subset T(\Phi_J)$, $T(\vartheta_J) \cup F(\vartheta_J) \subset F(\Phi_J)$.

Наприклад, $\Phi_J = Ez$, $\vartheta_J = \bot$. Тоді:

$T(\Phi \vee (\vartheta \& \neg\vartheta)_J) = T(\Phi_J)$, $T(\Phi\ \&\ (\vartheta \vee \neg\vartheta)_J) = T(\Phi_J) \cap (T(\vartheta_J) \cup F(\vartheta_J)) \subset T(\Phi_J)$;

$F(\Phi\ \&\ (\vartheta \vee \neg\vartheta)_J) = F(\Phi_J)$, $F(\Phi \vee (\vartheta \& \neg\vartheta)_J) = F(\Phi_J) \cap (F(\vartheta_J) \cup T(\vartheta_J)) \subset F(\Phi_J)$. Звідси:

$T(\Phi \vee (\vartheta \& \neg\vartheta)_J) \not\subset T(\Phi\ \&\ (\vartheta \vee \neg\vartheta)_J)$ та
$F(\Phi \vee (\vartheta \& \neg\vartheta)_J) \not\subset F(\Phi\ \&\ (\vartheta \vee \neg\vartheta)_J)$.

Зведемо результати щодо наявності логічного наслідку в таблицю.

Таблиця. Наявність логічного наслідку

	φ_1	φ_2	φ_3	φ_4	φ_5	φ_6	φ_7
$\models_1$	+	+	+	+	+	+	+
$\models_2$	–	+	–	+	+	–	+
$\models_3$	–	–	+	+	+	+	–
$\models_4$	–	–	–	+	+	–	–
$\models_5$	–	–	–	–	–	+	+
$\models_6$	–	–	–	–	+	–	+
$\models_7$	–	–	–	–	+	+	–
$\models_8$	–	–	–	–	+	–	–

Тут введено скорочені позначення:

$\models_1$: ${}^{Pc}\models_{IR}$, $\models_2$: ${}^{Pc}\models_{T}$,

$\models_3$: ${}^{Pc}\models_{F}$, $\models_4$: ${}^{Pc}\models_{TF}$,

$\models_5$: ${}^{Rc}\models_{IR}$, $\models_6$: ${}^{Rc}\models_{T}$,

$\models_7$: ${}^{Rc}\models_{F}$, $\models_8$: ${}^{Rc}\models_{TF}$;

φ_1: $\Phi \vee (\vartheta \& \neg\vartheta) \models \Phi \,\&\, (\vartheta \vee \neg\vartheta)$,

φ_2: $\neg\Phi, \Phi, \Gamma \models \Delta$,

φ_3: $\Gamma \models \neg\Psi, \Psi, \Delta$,

φ_4: $\neg\Phi, \Phi, \Gamma \models \neg\Psi, \Psi, \Delta$,

φ_5: $\Phi, \Gamma \models \Phi, \Delta$,

φ_6: $\Gamma, \Phi \models {}^{\sim}\Phi, \Delta$,

φ_7: $\Gamma, \neg{}^{\sim}\Phi \models \Phi, \Delta$.

Теорема 1. Між розглянутими відношеннями логічного наслідку в LC маємо такі співвідношення:

${}^{Rc}\models_{TF} \subset {}^{Rc}\models_{T} \subset {}^{Pc}\models_{T}$; ${}^{Rc}\models_{TF} \subset {}^{Rc}\models_{F} \subset {}^{Pc}\models_{F}$;

${}^{Rc}\models_{TF} \subset {}^{Pc}\models_{TF} \subset {}^{Pc}\models_{F}$; ${}^{Rc}\models_{TF} \subset {}^{Pc}\models_{TF} \subset {}^{Pc}\models_{T}$;

${}^{Pc}\models_{TF} = {}^{Pc}\models_{T} \cap {}^{Pc}\models_{F}$; ${}^{Rc}\models_{TF} = {}^{Rc}\models_{T} \cap {}^{Rc}\models_{F}$;

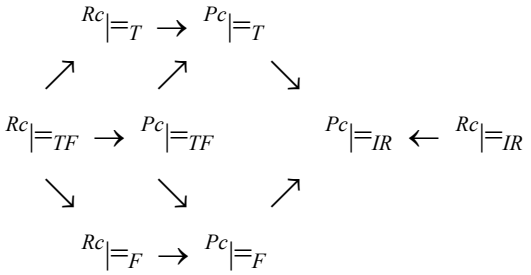
${}^{Pc}\models_{T} \subset {}^{Pc}\models_{IR}$; ${}^{Pc}\models_{F} \subset {}^{Pc}\models_{IR}$;

${}^{Rc}\models_{IR} \subset {}^{Pc}\models_{IR}$; ${}^{Rc}\models_{IR} \not\subset {}^{Pc}\models_{T}$; ${}^{Rc}\models_{IR} \not\subset {}^{Pc}\models_{F}$;

${}^{Rc}\models_{T} \not\subset {}^{Pc}\models_{F}$ та ${}^{Pc}\models_{F} \not\subset {}^{Rc}\models_{T}$;

${}^{Rc}\models_{F} \not\subset {}^{Pc}\models_{T}$ та ${}^{Pc}\models_{T} \not\subset {}^{Rc}\models_{F}$.

Графічно це можна подати так (тут замість $\subset$ вживаємо стрілку $\rightarrow$):



Вироджене відношення ${}^{Rc}\models_{IR}$ далі розглядати не будемо.

Надалі, якщо інше не зазначене окремо, будемо вживати такі позначення:

$\models_*$ – одне з відношень

${}^{Pc}\models_{IR}$, ${}^{Pc}\models_{T}$, ${}^{Pc}\models_{F}$, ${}^{Pc}\models_{TF}$, ${}^{Rc}\models_{T}$, ${}^{Rc}\models_{F}$, ${}^{Rc}\models_{TF}$;

${}_J\models_*$ – одне з ${}_J\models_{IR}$, ${}_J\models_{T}$, ${}_J\models_{F}$, ${}_J\models_{TF}$;

${}^{Pc}\models_*$ – одне з ${}^{Pc}\models_{IR}$, ${}^{Pc}\models_{T}$, ${}^{Pc}\models_{F}$, ${}^{Pc}\models_{TF}$;

${}^{Rc}\models_*$ – одне з ${}^{Rc}\models_{T}$, ${}^{Rc}\models_{F}$, ${}^{Rc}\models_{TF}$;

$\models_T$ – одне з ${}^{Pc}\models_{T}$, ${}^{Rc}\models_{T}$;

$\models_F$ – одне з ${}^{Pc}\models_{F}$, ${}^{Rc}\models_{F}$;

$\models_{TF}$ – одне з ${}^{Pc}\models_{TF}$, ${}^{Rc}\models_{TF}$.

Відношення логічного наслідку індукують відповідні відношення логічної еквівалентності між двома формулами.

Нехай $\Phi, \vartheta \in Fr$. Відношення еквівалентності при інтерпретації $\boldsymbol{J}$ задаємо за схемою: $\Phi\, {}_J\!\sim_* \Psi$, якщо ${}_J\Phi \models_* \Psi$ та ${}_J\Psi \models_* \Phi$.

Особливе значення має відношення ${}_J\!\sim_{TF}$ строгої еквівалентності:

Твердження 6. ${}_J\Phi \sim_{TF} \Psi \Leftrightarrow {}_J\Phi = {}_J\Psi$, тобто ${}_J\Phi$ та ${}_J\Psi$ – один і той же предикат.

Справді, ${}_J\Phi \sim_{TF} \Psi \Leftrightarrow T({}_J\Phi) = T({}_J\Psi)$ та $F({}_J\Phi) = F({}_J\Psi)$.

Задаємо відношення ${}^{Pc}\!\sim_{TF}$ та ${}^{Rc}\!\sim_{TF}$:

$\Phi\, {}^{Pc}\!\sim_{TF} \Psi$, якщо ${}_J\Phi \sim_{TF} \Psi$ для кожної $\boldsymbol{J} \in \boldsymbol{P_C}$;

$\Phi\, {}^{Rc}\!\sim_{TF} \Psi$, якщо ${}_J\Phi \sim_{TF} \Psi$ для кожної $\boldsymbol{J} \in \boldsymbol{R_C}$.

Згідно $\boldsymbol{P_C} \subset \boldsymbol{R_C}$ для всіх $\Phi, \Psi \in Fr$:

$$\Phi\, {}^{Rc}\!\sim_{TF} \Psi \Rightarrow \Phi\, {}^{Pc}\!\sim_{TF} \Psi.$$

Водночас (приклад 4) невірно

$$\Phi \overset{Pc}{\sim}_{TF} \Psi \Rightarrow \Phi \overset{Rc}{\sim}_{TF} \Psi.$$

Основою еквівалентних перетворень формул є теорема еквівалентності. Для LC та LCP вона формулюється так:

Теорема 2. Нехай Φ' отримана з Φ заміною деяких входжень $\Phi_1, ..., \Phi_n$ на $\Psi_1, ..., \Psi_n$.

Якщо $\Phi_1 \overset{Rc}{\sim}_{TF} \Psi_1$, ..., $\Phi_n \overset{Rc}{\sim}_{TF} \Psi_n$, то $\Phi \overset{Rc}{\sim}_{TF} \Phi'$;

якщо $\Phi_1 \overset{Pc}{\sim}_{TF} \Psi_1$, ..., $\Phi_n \overset{Pc}{\sim}_{TF} \Psi_n$, то $\Phi \overset{Pc}{\sim}_{TF} \Phi'$.

Теорема еквівалентності доводиться індукцією за побудовою формули подібно до відповідного доведення в традиційній логіці квазіарних предикатів [3, 4].

Теорема заміни еквівалентних для множин формул формулюється так.

Теорема 3. Нехай $\Phi \overset{Rc}{\sim}_{TF} \Psi$, тоді:

$$\Phi, \Gamma \models_* \Delta \Leftrightarrow \Psi, \Gamma \models_* \Delta;$$

$$\Gamma \models_* \Delta, \Phi \Leftrightarrow \Gamma \models_* \models \Delta, \Psi.$$

Нехай $\Phi \overset{Pc}{\sim}_{TF} \Psi$, тоді:

$$\Phi, \Gamma \overset{Pc}{\models}_* \Delta \Leftrightarrow \Psi, \Gamma \overset{Pc}{\models}_* \Delta;$$

$$\Gamma \overset{Pc}{\models}_* \Delta, \Phi \Leftrightarrow \Gamma \overset{Pc}{\models}_* \models \Delta, \Psi.$$

Дослідимо відношення $\overset{Pc}{\models}_{IR}$, $\overset{Pc}{\models}_T$, $\overset{Pc}{\models}_F$, $\overset{Pc}{\models}_{TF}$, $\overset{Rc}{\models}_T$, $\overset{Rc}{\models}_F$, $\overset{Rc}{\models}_{TF}$.

Властивості, в яких не фігурує композиція предикатного доповнення, в цілому аналогічні відповідним властивостям [3, 5] відношень $\overset{P}{\models}_{IR}$, $\overset{P}{\models}_T$, $\overset{P}{\models}_F$, $\overset{P}{\models}_{TF}$ $\overset{R}{\models}_{TF}$.

Властивості монотонності:

M) $\Gamma \subseteq \Lambda$ та $\Delta \subseteq \Sigma \Rightarrow (\Gamma \models_* \Delta \Rightarrow \Lambda \models_* \Sigma)$.

Властивості декомпозиції формул:

$\neg\neg_L$) $\neg\neg\Phi, \Gamma \models_* \Delta \Leftrightarrow \Phi, \Gamma \models_* \Delta$.

$\neg\neg_R$) $\Gamma \models_* \Delta, \neg\neg\Phi \Leftrightarrow \Gamma \models_* \Delta, \Phi$.

$\vee_L$) $\Phi \vee \Psi, \Gamma \models \Delta \Leftrightarrow \Phi, \Gamma \models \Delta$ та $\Psi, \Gamma \models \Delta$.

$\vee_R$) $\Gamma \models_* \Delta, \Phi \vee \Psi \Leftrightarrow \Gamma \models_* \Delta, \Phi, \Psi$.

$\neg\vee_L$) $\neg(\Phi \vee \Psi), \Gamma \models_* \Delta \Leftrightarrow \neg\Phi, \neg\Psi, \Gamma \models_* \Delta$.

$\neg\vee_R$) $\Gamma \models_* \Delta, \neg(\Phi \vee \Psi) \Leftrightarrow$

$\Leftrightarrow \Gamma \models_* \Delta, \neg\Phi$ та $\Gamma \models_* \Delta, \neg\Psi$.

Для $\overset{Pc}{\models}_{IR}$ додатково маємо:

$\neg_L$) $\neg\Phi, \Gamma \overset{Pc}{\models}_{IR} \Delta \Leftrightarrow \Gamma \overset{Pc}{\models}_{IR} \Delta, \Phi$.

$\neg_R$) $\Gamma \overset{Pc}{\models}_{IR} \Delta, \neg\Phi \Leftrightarrow \Phi, \Gamma \overset{Pc}{\models}_{IR} \Delta$.

Властивості $\neg_L$ та $\neg_R$ невірні для $\overset{Pc}{\models}_T$, $\overset{Pc}{\models}_F$, $\overset{Pc}{\models}_{TF}$, $\overset{Rc}{\models}_T$, $\overset{Rc}{\models}_F$, $\overset{Rc}{\models}_{TF}$.

Новими для відношень в LC є специфічні властивості декомпозиції в яких фігурує $\tilde{}\Phi$. Зокрема, стосовно $\overset{Pc}{\models}_{TF}$ та $\overset{Rc}{\models}_{TF}$ така декомпозиція можлива лише окремо для $\models_T$ та для $\models_F$; для $\overset{Pc}{\models}_{IR}$ коректно зробити декомпозицію $\tilde{}\Phi$ неможливо.

Теорема 4. Для відношення $\overset{Pc}{\models}_{IR}$ неможливо задати коректним способом умови декомпозиції формул вигляду $\tilde{}\Phi$.

Нехай $\Phi, \Psi, \vartheta \in Fr$; $\boldsymbol{J} \in \boldsymbol{P_C}$.

Маємо $\tilde{}\theta, \Phi_J \models_{IR} \Psi \Leftrightarrow$

$$\Leftrightarrow \perp(\tilde{}\theta_J) \cap T(\Phi_J) \cap F(\Psi_J) = \varnothing \Leftrightarrow$$

$$\Leftrightarrow \overline{T(\theta_J) \cup F(\theta_J)} \cap T(\Phi_J) \cap F(\Psi_J) = \varnothing.$$

Нехай предикат α утворено за допомогою $\vee$ та $\neg$ із деяких $\alpha_1, \ldots, \alpha_n$ Тоді $T(\alpha)$ та $F(\alpha)$ подаються через області істинності й хибності предикатів $\alpha_1, \ldots, \alpha_n$ за допомогою лише $\cap$ та $\cup$.

Шукаємо теоретико-множинну функцію $f(X, Y)$, будовану із $\cap$ та $\cup$ таку, що за умови $X \cap Y = \varnothing$ маємо $\overline{X \cup Y} \cap L = \varnothing \Leftrightarrow f(X, Y) \cap L = \varnothing$. Має виконуватись умова

$$\overline{T(\theta_J) \cup F(\theta_J)} \cap T(\Phi_J) \cap F(\Psi_J) = \varnothing \Leftrightarrow$$
$$\Leftrightarrow f(T(\theta_J), F(\theta_J)) \cap T(\Phi_J) \cap F(\Psi_J) = \varnothing.$$

Проте із X, Y таких, що $X \cap Y = \varnothing$, за допомогою $\cap$ та $\cup$ можна отримати лише 4 різних множини: $\varnothing$, X, Y, $X \cup Y$. Тому для $f(T(\theta_J), F(\theta_J))$ маємо лише 4 варіанти: $\varnothing$, $T(\vartheta_J)$, $F(\vartheta_J)$, $T(\vartheta_J) \cup F(\vartheta_J)$. Жоден з них не влаштовує зазначену умову.

Декомпозиція формул необхідна при побудові секвенційного числення, яке формалізує відповідне відношення логічного наслідку. Для $\overset{Pc}{\models}_{IR}$ така декомпозиція вимагає явного виділення області невизна-

ченості, адже умова для $_J|=_{IR}$ не дає змоги подати область невизначеності через області істинності та хибності за допомогою лише $\cap$ та $\cup$. Явне виділення області невизначеності означає перехід від $^{Pc}|=_{IR}$ до більш загального відношення $|=_{IR}^{\perp}$ *неспростовнісного логічного наслідку за умов невизначеності*, що зроблено в [10].

Відношення $^{Pc}|=_{IR}$ далі зазвичай розглядати не будемо.

В подальшому використаємо такі теоретико-множинні співвідношення:

$$A \cap \bar{B} \subseteq L \Leftrightarrow A \subseteq B \cup L \qquad \text{(Set1)}$$

$$A \cap B \subseteq L \Leftrightarrow A \subseteq \bar{B} \cup L \qquad \text{(Set2)}$$

Розглянемо властивості декомпозиції формул $\sim\Phi$ для $^{Pc}|=_T$, $^{Pc}|=_F$, $^{Rc}|=_T$, $^{Rc}|=_F$.

$\sim\Phi$ та $\neg\sim\Phi$ можуть бути в лівій частині та в правій частині відношення типу $|=_T$ та типу $|=_F$ – всього 8 комбінацій.

$\sim_{LT}$) $\sim\Phi, \Gamma |=_T \Delta \Leftrightarrow \Gamma |=_T \Delta, \Phi, \neg\Phi.$

Використаємо Set1 та умову $T(\sim P) = \overline{T(P) \cup F(P)} = \overline{T(P) \cup T(\neg P)}$.

Маємо $\sim\Phi, \Gamma |=_T \Delta \Leftrightarrow$ для кожної $\boldsymbol{J}$ $T^{\cap}(\Gamma_J) \cap T(\sim\Phi_J) \subseteq T^{\cup}(\Delta_J) \Leftrightarrow$ для кожної $\boldsymbol{J}$ $T^{\cap}(\Gamma_J) \cap \overline{T(\Phi_J) \cup T(\neg\Phi_J)} \subseteq T^{\cup}(\Delta_J) \Leftrightarrow$ для кожної $\boldsymbol{J}$ $T^{\cap}(\Gamma_J) \subseteq T^{\cup}(\Delta_J) \cup T(\Phi_J) \cup T(\neg\Phi_J)$ $\Leftrightarrow \Gamma |=_T \Delta, \Phi, \neg\Phi.$

$\sim_{RT}$) $\Gamma |=_T \Delta, \sim\Phi \Leftrightarrow$

$\Leftrightarrow \Phi, \Gamma |=_T \Delta$ та $\neg\Phi, \Gamma |=_T \Delta.$

Беремо до уваги Set2 та умову $T(\sim P) = \overline{T(P) \cup T(\neg P)}$.

Маємо $\Gamma |=_T \Delta, \sim\Phi \Leftrightarrow$ для кожної $\boldsymbol{J}$ $T^{\cap}(\Gamma_J) \subseteq T^{\cup}(\Delta_J) \cup T(\sim\Phi_J) \Leftrightarrow$ для кожної $\boldsymbol{J}$ $T^{\cap}(\Gamma_J) \subseteq T^{\cup}(\Delta_J) \cup \overline{T(\Phi_J) \cup T(\neg\Phi_J)} \Leftrightarrow$ для кожної $\boldsymbol{J}$ $T^{\cap}(\Gamma_J) \cap (T(\Phi_J) \cup T(\neg\Phi_J)) \subseteq T^{\cup}(\Delta_J)$ $\Leftrightarrow$ для кожної $\boldsymbol{J}$ $T^{\cap}(\Gamma_J) \cap T(\Phi_J) \subseteq T^{\cup}(\Delta_J)$ та $T^{\cap}(\Gamma_J) \cap T(\neg\Phi_J) \subseteq T^{\cup}(\Delta_J) \Leftrightarrow \Gamma, \Phi |=_T \Delta$ та $\Gamma, \neg\Phi |=_T \Delta.$

$\neg\sim_{RF}$) $\Gamma |=_F \Delta, \neg\sim\Phi \Leftrightarrow \Gamma, \Phi, \neg\Phi |=_F \Delta.$

Використаємо Set1 та умову $F(\neg\sim P) = T(\sim P) = \overline{F(P) \cup F(\neg P)}$.

Маємо $\Gamma |=_F^{\perp} \Delta, \neg\sim\Phi \Leftrightarrow$ для кожної $\boldsymbol{J}$ $F^{\cap}(\Delta_J) \cap F(\neg\sim\Phi) \subseteq F^{\cup}(\Gamma_J) \Leftrightarrow$ для кожної $\boldsymbol{J}$ $F^{\cap}(\Delta_J) \cap \overline{F(\Phi_J) \cup F(\neg\Phi_J)} \subseteq F^{\cup}(\Gamma_J)$ $\Leftrightarrow F^{\cap}(\Delta_J) \subseteq F^{\cup}(\Gamma_J) \cup F(\Phi_J) \cup F(\neg\Phi_J)$ для кожної $\boldsymbol{J}$ $\Leftrightarrow \Gamma, \Phi, \neg\Phi |=_F \Delta.$

$\neg\sim_{LF}$) $\neg\sim\Phi, \Gamma |=_F \Delta \Leftrightarrow$

$\Leftrightarrow \Gamma |=_F \Delta, \Phi$ та $\Gamma |=_F \Delta, \neg\Phi.$

Використаємо Set2 та умову $F(\neg\sim P) = \overline{F(P) \cup F(\neg P)}$.

Маємо $\neg\sim\Phi, \Gamma |=_F^{\perp} \Delta \Leftrightarrow$ для кожної $\boldsymbol{J}$ $F^{\cap}(\Delta_J) \subseteq F^{\cup}(\Gamma_J) \cup F(\neg\sim\Phi_J) \Leftrightarrow$ для кожної $\boldsymbol{J}$ $F^{\cap}(\Delta_J) \subseteq F^{\cup}(\Gamma_J) \cup \overline{F(\Phi_J) \cup F(\neg\Phi_J)}$ $\Leftrightarrow F^{\cap}(\Delta_J) \cap (F(\Phi_J) \cup F(\neg\Phi_J)) \subseteq F^{\cup}(\Gamma_J)$ для кожної $\boldsymbol{J}$ $\Leftrightarrow F^{\cap}(\Delta_J) \cap F(\Phi_J) \subseteq F^{\cup}(\Gamma_J)$ та $F^{\cap}(\Delta_J) \cap F(\neg\Phi_J) \subseteq F^{\cup}(\Gamma_J)$ для кожної $\boldsymbol{J}$ $\Leftrightarrow$ $\Gamma |=_F \Delta, \Phi$ та $\Gamma |=_F \Delta, \neg\Phi.$

Наступні властивості $\neg\sim_{El}$ та $\sim_{El}$ фактично є властивостями спрощення.

Для $\neg\sim_{El}$ та $\sim_{El}$ беремо до уваги $T(\neg\sim P) = F(\sim P) = \varnothing$.

$\neg\sim_{El}$) $\Gamma |=_T \Delta, \neg\sim\Phi \Leftrightarrow \Gamma |=_T \Delta.$

Маємо $\Gamma |=_T^{\perp} \Delta, \neg\sim\Phi \Leftrightarrow$ для кожної $\boldsymbol{J}$ $T^{\cap}(\Gamma_J) \subseteq T^{\cup}(\Delta_J) \cup T(\neg\sim\Phi_J) \Leftrightarrow$ для кожної $\boldsymbol{J}$ $T^{\cap}(\Gamma_J) \cap\subseteq T^{\cup}(\Delta_J) \cup \varnothing \Leftrightarrow \Gamma |=_T \Delta.$

$\sim_{El}$) $\sim\Phi, \Gamma |=_F \Delta \Leftrightarrow \Gamma |=_F \Delta.$

Маємо $\sim\Phi, \Gamma |=_F \Delta \Leftrightarrow$ для кожної $\boldsymbol{J}$ $F^{\cap}(\Delta_J) \subseteq F^{\cup}(\Gamma_J) \cup F(\sim\Phi_J) \Leftrightarrow$ для кожної $\boldsymbol{J}$ $F^{\cap}(\Delta_J) \subseteq F^{\cup}(\Gamma_J) \cup \varnothing \Leftrightarrow \Gamma |=_F \Delta.$

Залишились 2 комбінації, які дають властивості гарантованої наявності відношення логічного наслідку. Для всіх $\boldsymbol{J} \in \boldsymbol{R_C}$ маємо $F(\sim\Phi_J) = T(\neg\sim\Phi_J) = \varnothing$ звідки:

$C\sim)$ $\Gamma |=_F \sim\Phi, \Delta.$

$C\neg\sim)$ $\Gamma, \neg\sim\Phi |=_T \Delta.$

Зауважимо, що аналогічна до $C^{\sim}$ властивість для $|=_T$ – це властивість $^{\sim}_{\mathrm{LT}}$, аналогічна до $C\neg^{\sim}$ властивість для $|=_F$ – це властивість $\neg\,^{\sim}_{\mathrm{LF}}$.

На відміну від $_J|=_{IR}$, для $_J|=_T$ та $_J|=_F$ області невизначеності *можна* подати через області істинності та хибності за допомогою лише $\cap$ та $\cup$, тобто *можна* елімінувати області невизначеності. Це засвідчують $^{\sim}_{\mathrm{LT}}$, $^{\sim}_{\mathrm{RT}}$, $\neg\,^{\sim}_{\mathrm{RF}}$, $\neg\,^{\sim}_{\mathrm{LF}}$, $\neg\,^{\sim}_{\mathrm{El}}$, $^{\sim}_{\mathrm{El}}$.

Для відношень $^{Rc}|=_{TF}$ та $^{Pc}|=_{TF}$ ситуація дещо інша. Це зумовлено асиметричною поведінкою композиції $^{\sim}$ щодо областей істинності й хибності. Для відношень типу $|=_T$ та типу $|=_F$ маємо різні властивості декомпозиції формул вигляду $^{\sim}\Phi$, які не можна подати як спільну властивість для відношень типу $|=_{TF}$. Для них маємо:

$$^{\sim}\Phi,\Gamma\,|=_{TF}\Delta \quad\Leftrightarrow\quad {}^{\sim}\Phi,\Gamma\,|=_T\Delta \text{ та } {}^{\sim}\Phi,\Gamma\,|=_F\Delta \;\Leftrightarrow\; \Gamma\,|=_T\Delta,\Phi,\neg\Phi \text{ та } \Gamma\,|=_F\Delta.$$

$$\Gamma\,|=_{TF}\Delta,\neg^{\sim}\Phi \quad\Leftrightarrow\quad \Gamma\,|=_T\Delta,\neg^{\sim}\Phi \text{ та } \Gamma\,|=_F\Delta,\neg^{\sim}\Phi \;\Leftrightarrow\; \Gamma\,|=_T\Delta \text{ та } \Gamma,\Phi,\neg\Phi\,|=_F\Delta.$$

$$\Gamma\,|=_{TF}\Delta,{}^{\sim}\Phi \quad\Leftrightarrow\quad \Gamma\,|=_T\Delta,{}^{\sim}\Phi \text{ та } \Gamma\,|=_F\Delta,{}^{\sim}\Phi \;\Leftrightarrow\; \Phi,\Gamma\,|=_T\Delta \text{ та } \neg\Phi,\Gamma\,|=_T\Delta.$$

$$\neg^{\sim}\Phi,\Gamma\,|=_{TF}\Delta \quad\Leftrightarrow\quad \neg^{\sim}\Phi,\Gamma\,|=_T\Delta \text{ та } \neg^{\sim}\Phi,\Gamma\,|=_F\Delta \;\Leftrightarrow\; \Gamma\,|=_F\Delta,\Phi \text{ та } \Gamma\,|=_F\Delta,\neg\Phi.$$

Тому при побудові секвенційних числень, які формалізують $^{Rc}|=_{TF}$ та $^{Pc}|=_{TF}$, опираємось на базові визначення:

$$\Gamma\,{}^{Rc}|=_{TF}\Delta \;\Leftrightarrow\; \Gamma\,{}^{Rc}|=_T\Delta \text{ та } \Gamma\,{}^{Rc}|=_F\Delta;$$

$$\Gamma\,{}^{Pc}|=_{TF}\Delta \;\Leftrightarrow\; \Gamma\,{}^{Pc}|=_T\Delta \text{ та } \Gamma\,{}^{Pc}|=_F\Delta.$$

Таким чином, секвенційне числення для відношення $|=_{TF}$ має бути поєднанням числень для відношень $\Gamma\,|=_T\Delta$ та $\Gamma\,|=_F\Delta$: для встановлення $\Gamma\,|=_{TF}\Delta$ будуємо два виведення, перше виведення встановлює $\Gamma\,|=_T\Delta$, а друге – $\Gamma\,|=_F\Delta$.

Розглянемо властивості, які гарантують наявність відношення логічного наслідку. Для усіх розглянутих відношень логічного наслідку маємо властивість:

$$C)\ \Phi,\Gamma\,|=_*\Delta,\Phi.$$

Для $^{Pc}|=_{IR}$ C випливає з того, що $T(\Phi_J)\cap F(\Phi_J)=\varnothing$ для кожної $\boldsymbol{J}\in\boldsymbol{P_C}$.

Для інших відношень використовуємо співвідношення $T(\Phi_J)\cap \mathrm{L}\subseteq T(\Phi_J)\cup \mathrm{M}$ та $F(\Phi_J)\cap \mathrm{L}\subseteq F(\Phi_J)\cup \mathrm{M}$.

Додатково гарантують наявність відповідного відношення такі властивості.

$$CL)\ \Phi,\neg\Phi,\Gamma\,{}^{Pc}|=_T\Delta.$$

Випливає з того, що для кожної $\boldsymbol{J}\in\boldsymbol{P_C}$ $T(\Phi_J)\cap T(\neg\Phi_J)=T(\Phi_J)\cap F(\Phi_J)=\varnothing$.

$$CR)\ \Gamma\,{}^{Pc}|=_F\Delta,\Phi,\neg\Phi.$$

Випливає з того, що для кожної $\boldsymbol{J}\in\boldsymbol{P_C}$ $F(\Phi_J)\cap F(\neg\Phi_J)=F(\Phi_J)\cap T(\Phi_J)=\varnothing$.

$$CLR)\ \Phi,\neg\Phi,\Gamma\,{}^{Pc}|=_{TF}\Delta,\Psi,\neg\Psi.$$

Справді, для кожної $\boldsymbol{J}\in\boldsymbol{P_C}$ маємо $F(\Phi_J)\cap F(\neg\Phi_J)=F(\Psi_J)\cap F(\neg\Psi_J)=\varnothing$.

Маємо властивості гарантованої наявності відношення логічного наслідку:

– C, CL, $C\neg^{\sim}$ для $^{Pc}|=_T$;

– C, CR, $C^{\sim}$ для $^{Pc}|=_F$;

– C, CLR для $^{Pc}|=_{TF}$;

– C, $C\neg^{\sim}$ для $^{Rc}|=_T$;

– C, $C^{\sim}$ для $^{Rc}|=_F$;

– C для $^{Rc}|=_{TF}$.

Описані властивості засвідчують істотну відмінність LC від традиційної логіки квазіарних предикатів. Зокрема, $^{R}|=_T$, $^{R}|=_F$, $^{R}|=_{TF}$ – це єдине відношення, а в LC маємо *різні* відношення $^{Rc}|=_T$, $^{Rc}|=_F$, $^{R}|=_{TF}$. В LC відношення $^{Pc}|=_{IR}$ є некоректним.

Розглянемо пов'язані з реномінацією властивості еквівалентних перетворень для відношень логічного наслідку в LC. Їх доведення базується на теоремі 3. Кожна з наведених вище властивостей R, RI, RU, RR, $R\neg$, $R\vee$, $R^{\sim}$, $R\exists$, $R\exists s$ продукує 4 відповідні властивості для відношень $^{Pc}|=_T$, $^{Pc}|=_F$, $^{Rc}|=_T$, $^{Rc}|=_F$, коли виділена формула чи її заперечення знаходиться у лівій чи правій частині цього відношення. Ці властивості формулюються однотипно.

Наведемо тут властивості, індуковані $R^{\sim}$, інші властивості формулюються

аналогічно відповідним властивостям [5] для ${}^{P}|=_{T}$, ${}^{P}|=_{F}$, ${}^{R}|=_{TF}$.

Нехай $\Phi\in Fr$; $\Gamma,\Delta\subseteq Fr$; нехай $|=_{*}$ – одне з ${}^{Pc}|=_{T}$, ${}^{Pc}|=_{F}$, ${}^{Rc}|=_{T}$, ${}^{Rc}|=_{F}$.

$\mathrm{R}\tilde{}_{\mathrm{L}})\ R^{\bar{v}}_{\bar{x}}(\tilde{}\Phi),\Gamma|=_{*}\Delta \Leftrightarrow\ \tilde{}R^{\bar{v}}_{\bar{x}}(\Phi),\Gamma|=_{*}\Delta.$

$\mathrm{R}\tilde{}_{\mathrm{R}})\ \Gamma|=_{*}\Delta, R^{\bar{v}}_{\bar{x}}(\tilde{}\Phi) \Leftrightarrow\ \Gamma|=_{*}\Delta,\tilde{}R^{\bar{v}}_{\bar{x}}(\Phi).$

$\neg\mathrm{R}\tilde{}_{\mathrm{L}})\ \neg R^{\bar{v}}_{\bar{x}}(\tilde{}\Phi),\Gamma|=_{*}\Delta \Leftrightarrow$

$\Leftrightarrow\ \neg\tilde{}R^{\bar{v}}_{\bar{x}}(\Phi),\Gamma|=_{*}\Delta.$

$\neg\mathrm{R}\tilde{}_{\mathrm{R}})\ \Gamma|=_{*}\Delta,\neg R^{\bar{v}}_{\bar{x}}(\tilde{}\Phi) \Leftrightarrow$

$\Leftrightarrow\ \Gamma|=_{*}\Delta,\neg\tilde{}R^{\bar{v}}_{\bar{x}}(\Phi).$

Для відношень типу $|=_{TF}$ властивості еквівалентних перетворень теж вірні, проте їх задаємо *опосередковано*, через відповідні властивості для відношень типу $|=_{T}$ та типу $|=_{F}$. При цьому враховуємо:

$$\Gamma|=_{TF}\Delta \Leftrightarrow \Gamma|=_{T}\Delta \text{ та } \Gamma|=_{F}\Delta.$$

Властивості елімінації кванторів, *E*-розподілу та первісного означення для відношень ${}^{Pc}|=_{T}$, ${}^{Rc}|=_{T}$, ${}^{Pc}|=_{F}$, ${}^{Rc}|=_{F}$ формулюються та доводяться аналогічно відповідним властивостям [5] для відношень ${}^{P}|=_{T}$, ${}^{P}|=_{F}$, ${}^{R}|=_{TF}$.

Теорема 5. Нехай $\Phi\in Fr$; $\Gamma,\Delta\subseteq Fr$; тоді:

$\exists_{\mathrm{L}})$ за умови $z\in fu(\Gamma,\Delta,\exists x\Phi))$ маємо $\exists x\Phi,\Gamma|=_{*}\Delta \Leftrightarrow\ R^{x}_{z}(\Phi),Ez,\Gamma|=_{*}\Delta;$

$\exists\mathrm{R}_{\mathrm{L}})$ за умови $z\in fu(\Gamma,\Delta,R^{\bar{u}}_{\bar{v}}(\exists x\Phi))$ $R^{\bar{u}}_{\bar{v}}(\exists x\Phi),\Gamma|=_{*}\Delta \Leftrightarrow\ R^{\bar{u},x}_{\bar{v},z}(\Phi),Ez,\Gamma|=_{*}\Delta;$

$\neg\exists_{\mathrm{R}})$ за умови $z\in fu(\Gamma,\Delta,\exists x\Phi))$ маємо $\Gamma|=_{*}\neg\exists x\Phi,\Delta \Leftrightarrow\ \Gamma,Ez|=_{*}\neg R^{x}_{z}(\Phi),\Delta$

$\neg\exists\mathrm{R}_{\mathrm{R}})$ за умови $z\in fu(\Gamma,\Delta,R^{\bar{u}}_{\bar{v}}(\exists x\Phi))$ $\Gamma|=_{*}\neg R^{\bar{u}}_{\bar{v}}(\exists x\Phi),\Delta \Leftrightarrow\ \Gamma,Ez|=_{*}\neg R^{\bar{u},x}_{\bar{v},z}(\Phi),\Delta$

$\exists\mathrm{v}_{\mathrm{R}})\quad \Gamma,Ey|=_{*}\exists x\Phi,\Delta \Leftrightarrow$
$\Leftrightarrow\ \Gamma,Ey|=_{*}\exists x\Phi,R^{x}_{y}(\Phi),\Delta.$

$\exists\mathrm{Rv}_{\mathrm{R}})\quad \Gamma,Ey|=_{*}\Delta,R^{\bar{u}}_{\bar{v}}(\exists x\Phi) \Leftrightarrow$
$\Leftrightarrow\ \Gamma,Ey|=_{*}\Delta,R^{\bar{u}}_{\bar{v}}(\exists x\Phi),R^{\bar{u},x}_{\bar{v},y}(\Phi).$

$\neg\exists\mathrm{v}_{\mathrm{L}})\quad \neg\exists x\Phi,Ey,\Gamma|=_{*}\Delta \Leftrightarrow$
$\Leftrightarrow\ \neg\exists x\Phi,\neg R^{x}_{y}(\Phi),Ey,\Gamma|=_{*}\Delta.$

$\neg\exists\mathrm{Rv}_{\mathrm{L}})\quad \neg R^{\bar{u}}_{\bar{v}}(\exists x\Phi),Ey,\Gamma|=_{*}\Delta \Leftrightarrow$
$\Leftrightarrow\ \neg R^{\bar{u}}_{\bar{v}}(\exists x\Phi),\neg R^{\bar{u},x}_{\bar{v},y}(\Phi),Ey,\Gamma|=_{*}\Delta.$

Теорема 6. Для $\Gamma,\Delta\subseteq Fr$ маємо:

Ed) $\Gamma|=_{*}\Delta \Leftrightarrow\ \Gamma|=_{*}\Delta,Ey$ та $Ey,\Gamma|=_{*}\Delta.$

Ev) $\Gamma|=_{*}\Delta \Leftrightarrow Ez,\Gamma|=_{*}\Delta$, де $z\in fu(\Gamma,\Delta)$.

3. Відношення логічного наслідку за умов невизначеності

Наявність в LC нової композиції предикатного доповнення мотивує до розгляду нових відношень логічного наслідку, які враховують особливості цієї композиції. Такими є відношення *логічного наслідку за умови невизначеності*.

Узагальненням відношення ${}^{P}|=_{IR}$ є досліджене в [10] відношення $|=_{IR}{}^{\perp}$ неспростовнісного логічного наслідку за умов невизначеності. Нехай $\Gamma,\mathrm{U},\Delta\subseteq Fr$.

Δ є неспростовнісним наслідком Γ за умов невизначеності U при інтерпретації $\boldsymbol{J}$, що позначимо $\mathrm{U}/\Gamma_{J}|=_{IR}{}^{\perp}\Delta$, якщо $T^{\frown}(\Gamma_{J})\cap\perp^{\frown}(\mathrm{U}_{J})\cap F^{\frown}(\Delta_{J})=\varnothing$.

Δ є неспростовнісним логічним наслідком Γ за умов невизначеності U, що позначимо $\mathrm{U}/\Gamma|=_{IR}{}^{\perp}\Delta$, якщо $\mathrm{U}/\Gamma_{J}|=_{IR}{}^{\perp}\Delta$ для кожної інтерпретації $\boldsymbol{J}\in\boldsymbol{P_{C}}$.

Якщо $\mathrm{U}=\varnothing$, то отримуємо визначення відношення ${}^{Pc}|=_{IR}$.

Подібне вироджене відношення ${}^{R}|=_{IR}{}^{\perp}$, яке відповідає виродженому відношенню ${}^{Rc}|=_{IR}$, тут розглядати не будемо.

Узагальненнями відомих [3, 5] відношень типу $|=_{T}$, $|=_{F}$ та $|=_{TF}$ для традиційних логік квазіарних предикатів є пропоновані в цій роботі відношення істиннісного хибнісного та строгого логічного наслідку за умов невизначеності. Ми вводимо:

– відношення ${}^{P}|=_{T}{}^{\perp}$ логічного *T*-наслідку в LPC за умов невизначеності;

– відношення ${}^{P}|=_{F}{}^{\perp}$ логічного *F*-наслідку в LPC за умов невизначеності;

– відношення $^{P}|=_{TF}^{\perp}$ логічного *TF*-наслідку в LPC за умов невизначеності;

– відношення $^{R}|=_{T}^{\perp}$ логічного *T*-наслідку в LC за умов невизначеності;

– відношення $^{R}|=_{F}^{\perp}$ логічного *F*-наслідку в LC за умов невизначеності;

– відношення $^{R}|=_{TF}^{\perp}$ логічного *FT*-наслідку в LC за умов невизначеності.

Неформально те, що Δ є *T*-наслідком Γ за умови невизначеності U при інтерпретації $\boldsymbol{J}$, має означати: "для кожного $d\in^{V}A$ якщо $d\in\perp^{\frown}(\mathrm{U}_J)$, то $(d\in T^{\frown}(\Gamma_J) \Rightarrow d\in T^{\smile}(\Delta_J))$". Це рівносильне наступному:

"для кожного $d\in^{V}A$ якщо $d\in\perp^{\frown}(\mathrm{U}_J)$ та $d\in T^{\frown}(\Gamma_J)$ то $d\in T^{\smile}(\Delta_J)$".

Те, що Δ є *F*-наслідком Γ за умови невизначеності U при інтерпретації $\boldsymbol{J}$, має означати: "для кожного $d\in^{V}A$ якщо $d\in\perp^{\frown}(\mathrm{U}_J)$, то $(d\in F^{\frown}(\Delta_J) \Rightarrow d\in F^{\smile}(\Gamma_J))$".

Це рівносильне: "для кожного $d\in^{V}A$ якщо $d\in\perp^{\frown}(\mathrm{U}_J)$ та $d\in F^{\frown}(\Delta_J)$ то $d\in F^{\smile}(\Gamma_J)$".

Те, що Δ є *TF*-наслідком Γ за умов невизначеності U при інтерпретації $\boldsymbol{J}$, має означати: "для кожного $d\in^{V}A$ якщо $d\in\perp^{\frown}(\mathrm{U}_J)$, то $(d\in T^{\frown}(\Gamma_J) \Rightarrow d\in T^{\smile}(\Delta_J))$ та $(d\in F^{\frown}\Delta_J) \Rightarrow d\in F^{\smile}(\Gamma_J))$".

Це рівносильне: "для кожного $d\in^{V}A$ якщо $d\in\perp^{\frown}(\mathrm{U}_J)$ та $d\in T^{\frown}(\Gamma_J)$ то $d\in T^{\smile}(\Delta_J)$ та якщо $d\in\perp^{\frown}(\mathrm{U}_J)$ та $d\in F^{\frown}(\Delta_J)$ то $d\in F^{\smile}(\Gamma_J))$".

Приходимо до наступних визначень.

Δ є *T*-наслідком Γ за умови невизначеності U при інтерпретації $\boldsymbol{J}$, якщо $T^{\frown}(\Gamma_J)\cap\perp^{\frown}(\mathrm{U}_J)\subseteq T^{\smile}(\Delta_J)$.

Це позначимо $\mathrm{U}/\Gamma_J|=_{T}^{\perp}\Delta$.

Δ є *F*-наслідком Γ за умови невизначеності U при інтерпретації $\boldsymbol{J}$, якщо $F^{\frown}(\Delta_J)\cap\perp^{\frown}(\mathrm{U}_J)\subseteq F^{\smile}(\Gamma_J)$.

Це позначимо $\mathrm{U}/\Gamma_J|=_{F}^{\perp}\Delta$.

Δ є *TF*-наслідком Γ за умови невизначеності U при інтерпретації $\boldsymbol{J}$, якщо $\mathrm{U}/\Gamma_J|=_{T}^{\perp}\Delta$ та $\mathrm{U}/\Gamma_J|=_{F}^{\perp}\Delta$.

Це позначимо $\mathrm{U}/\Gamma_J|=_{TF}^{\perp}\Delta$.

Якщо $\mathrm{U}=\varnothing$, то маємо визначення відношень $\Gamma_J|=_{T}\Delta$, $\Gamma_J|=_{F}\Delta$, $\Gamma_J|=_{TF}\Delta$ в LC.

Далі визначаємо традиційно.

Δ є логічним *T*-наслідком Γ за умови невизначеності U в семантиці $\boldsymbol{P_C}$, що позначимо $\mathrm{U}/\Gamma\,^{P}|=_{T}^{\perp}\Delta$, якщо $\mathrm{U}/\Gamma_J|=_{T}^{\perp}\Delta$ для кожної $\boldsymbol{J}\in\boldsymbol{P_C}$.

Δ є логічним *F*-наслідком Γ за умови невизначеності U в семантиці $\boldsymbol{P_C}$, що позначимо $\mathrm{U}/\Gamma\,^{P}|=_{F}^{\perp}\Delta$, якщо $\mathrm{U}/\Gamma_J|=_{F}^{\perp}\Delta$ для кожної $\boldsymbol{J}\in\boldsymbol{P_C}$.

Δ є логічним *TF*-наслідком Γ за умови невизначеності U в семантиці $\boldsymbol{P_C}$, що позначимо $\mathrm{U}/\Gamma\,^{P}|=_{TF}^{\perp}\Delta$, якщо $\mathrm{U}/\Gamma_J|=_{TF}^{\perp}\Delta$ для кожної $\boldsymbol{J}\in\boldsymbol{P_C}$.

Δ є логічним *T*-наслідком Γ за умови невизначеності U в семантиці $\boldsymbol{R_C}$, що позначимо $\mathrm{U}/\Gamma\,^{R}|=_{T}^{\perp}\Delta$, якщо $\mathrm{U}/\Gamma_J|=_{T}^{\perp}\Delta$ для кожної $\boldsymbol{J}\in\boldsymbol{R_C}$.

Δ є логічним *F*-наслідком Γ за умови невизначеності U в семантиці $\boldsymbol{R_C}$, що позначимо $\Gamma\,^{R}|=_{F}^{\perp}\Delta$, якщо $\mathrm{U}/\Gamma_J|=_{F}^{\perp}\Delta$ для кожної $\boldsymbol{J}\in\boldsymbol{R_C}$.

Δ є логічним *TF*-наслідком Γ за умови невизначеності U в семантиці $\boldsymbol{R_C}$, що позначимо $\mathrm{U}/\Gamma\,^{R}|=_{TF}^{\perp}\Delta$, якщо $\mathrm{U}/\Gamma_J|=_{TF}^{\perp}\Delta$ для кожної $\boldsymbol{J}\in\boldsymbol{R_C}$.

Якщо $\mathrm{U}=\varnothing$, то маємо визначення $^{Pc}|=_{T}$, $^{Pc}|=_{F}$, $^{Pc}|=_{TF}$, $^{Rc}|=_{T}$, $^{Rc}|=_{F}$, $^{Rc}|=_{TF}$.

Твердження 7. $\mathrm{U}/\Gamma\,^{P}|=_{TF}^{\perp}\Delta \Leftrightarrow \mathrm{U}/\Gamma\,^{P}|=_{T}^{\perp}\Delta$ та $\mathrm{U}/\Gamma\,^{P}|=_{F}^{\perp}\Delta$;

$\mathrm{U}/\Gamma\,^{R}|=_{TF}^{\perp}\Delta \Leftrightarrow \mathrm{U}/\Gamma\,^{R}|=_{T}^{\perp}\Delta$ та $\mathrm{U}/\Gamma\,^{R}|=_{F}^{\perp}\Delta$.

Твердження 8. Для $\mathrm{S}\subseteq Fr$ та $\boldsymbol{J}\in\boldsymbol{P_C}$ $T^{\smile}(\mathrm{S}_J)\cap F^{\frown}(\mathrm{S}_J)=\varnothing$ та $F^{\smile}(\mathrm{S}_J)\cap T^{\frown}(\mathrm{S}_J)=\varnothing$.

Звідси як наслідок отримуємо

Твердження 9. Для довільних $\Gamma, \mathrm{U}, \Delta\subseteq Fr$ та $\boldsymbol{J}\in\boldsymbol{P_C}$ маємо:

$\mathrm{U}/\Gamma_J|=_{T}^{\perp}\Delta \Rightarrow \mathrm{U}/\Gamma_J|=_{IR}^{\perp}\Delta$;

$\mathrm{U}/\Gamma_J|=_{F}^{\perp}\Delta \Rightarrow \mathrm{U}/\Gamma_J|=_{IR}^{\perp}\Delta$.

Справді, згідно твердження 8:

$$T^{\cap}(\Gamma_J)\cap\perp^{\cap}(\mathrm{U}_J)\subseteq T^{\cup}(\Delta_J)\Rightarrow$$
$$\Rightarrow\ T^{\cap}(\Gamma_J)\cap\perp^{\cap}(\mathrm{U}_J)\cap F^{\cap}(\Delta_J)=\varnothing;$$

$$F^{\cap}(\Delta_J)\cap\perp^{\cap}(\mathrm{U}_J)\subseteq F^{\cup}(\Gamma_J)\Rightarrow$$
$$\Rightarrow\ T^{\cap}(\Gamma_J)\cap\perp^{\cap}(\mathrm{U}_J)\cap F^{\cap}(\Delta_J)=\varnothing.$$

Твердження 10. Беручи до уваги приклади 4–7, маємо наступні властивості:

1) $\mathrm{U}/\Gamma\ |=_{IR}^{\perp}\Psi,\neg\Psi$ та $\mathrm{U}/\Phi,\neg\Phi\ |=_{IR}^{\perp}\Delta$;

2) $\Gamma\ {}^{Pc}|\neq_T\Psi,\neg\Psi$ та $\Phi,\neg\Phi\ {}^{Pc}|\neq_F\Delta$, тому
$\mathrm{U}/\Gamma\ {}^{P}|\neq_T^{\perp}\Psi,\neg\Psi$ та $\mathrm{U}/\Phi,\neg\Phi\ {}^{P}|\neq_F^{\perp}\Delta$;

3) $\mathrm{U}/\Phi,\neg\Phi\ {}^{P}|=_T^{\perp}\Psi,\neg\Psi$ та
$\mathrm{U}/\Phi,\neg\Phi\ {}^{P}|=_F^{\perp}\Psi,\neg\Psi$, тому
$\mathrm{U}/\Phi,\neg\Phi\ {}^{P}|=_{TF}^{\perp}\Psi,\neg\Psi$;

4) $\mathrm{U}/\Phi,\neg\Phi\ {}^{R}|\neq_T\Psi,\neg\Psi$;
$\mathrm{U}/\Phi,\neg\Phi\ {}^{R}|\neq_F^{\perp}\Psi,\neg\Psi$;
$\mathrm{U}/\Phi,\neg\Phi\ {}^{R}|\neq_{TF}^{\perp}\Psi,\neg\Psi$.

Теорема 7. Маємо співвідношення:

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

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

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

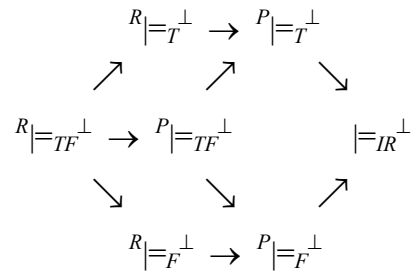
${}^{R}|=_{TF}^{\perp}\subset{}^{P}|=_{TF}^{\perp}\subset{}^{P}|=_T^{\perp}$;

${}^{P}|=_T^{\perp}\subset|=_{IR}^{\perp}$; ${}^{P}|=_F^{\perp}\subset|=_{IR}^{\perp}$;

${}^{R}|=_T^{\perp}\not\subset{}^{P}|=_F^{\perp}$ та ${}^{P}|=_F^{\perp}\not\subset{}^{R}|=_T^{\perp}$;

${}^{R}|=_F^{\perp}\not\subset{}^{P}|=_T^{\perp}$ та ${}^{P}|=_T^{\perp}\not\subset{}^{R}|=_F^{\perp}$.

Графічно це можна подати так (тут замість $\subset$ вживаємо стрілку $\rightarrow$):



Таким чином, співвідношення між відношеннями $|=_{IR}^{\perp}$, ${}^{P}|=_T^{\perp}$, ${}^{P}|=_F^{\perp}$, ${}^{P}|=_{TF}^{\perp}$, ${}^{R}|=_T^{\perp}$, ${}^{R}|=_F^{\perp}$, ${}^{R}|=_{TF}^{\perp}$ ідентичні співвідношенням між відношеннями ${}^{Pc}|=_{IR}$, ${}^{Pc}|=_T$, ${}^{Pc}|=_F$, ${}^{Pc}|=_{TF}$, ${}^{Rc}|=_T$, ${}^{Rc}|=_F$, ${}^{Rc}|=_{TF}$.

Сформулюємо теорему заміни еквівалентних для відношень $|=_T^{\perp}$, $|=_F^{\perp}$, $|=_{TF}^{\perp}$.

Теорема 8. 1) Нехай $\Phi\ {}^{P}\!\sim_{TF}\Psi$, тоді маємо (${}^{P}|=_*$ – це ${}^{P}|=_T^{\perp}$, ${}^{P}|=_F^{\perp}$ чи ${}^{P}|=_{TF}^{\perp}$):

$$\mathrm{U}/\Phi,\Gamma\ {}^{P}|=_*\Delta\Leftrightarrow\mathrm{U}/\Psi,\Gamma\ {}^{P}|=_*\Delta;$$

$$\mathrm{U}/\Gamma\ {}^{P}|=_*\Delta,\Phi\Leftrightarrow\mathrm{U}/\Gamma\ {}^{P}|=_*\Delta,\Psi;$$

$$\mathrm{U},\Phi/\Gamma\ {}^{P}|=_*\Delta\Leftrightarrow\mathrm{U},\Psi/\Gamma\ {}^{P}|=_*\Delta.$$

2) Нехай $\Phi\ {}^{R}\!\sim_{TF}\Psi$, тоді маємо (${}^{R}|=_*$ – це ${}^{R}|=_T^{\perp}$, ${}^{R}|=_F^{\perp}$ чи ${}^{R}|=_{TF}^{\perp}$):

$$\mathrm{U}/\Phi,\Gamma\ {}^{R}|=_*\Delta\Leftrightarrow\mathrm{U}/\Psi,\Gamma\ {}^{R}|=_*\Delta;$$

$$\mathrm{U}/\Gamma\ {}^{R}|=_*\Delta,\Phi\Leftrightarrow\mathrm{U}/\Gamma\ {}^{R}|=_*\Delta,\Psi;$$

$$\mathrm{U},\Phi/\Gamma\ {}^{R}|=_*\Delta\Leftrightarrow\mathrm{U},\Psi/\Gamma\ {}^{R}|=_*\Delta.$$

Дослідимо властивості відношень ${}^{P}|=_T^{\perp}$, ${}^{P}|=_F^{\perp}$, ${}^{P}|=_{TF}^{\perp}$, ${}^{R}|=_T^{\perp}$, ${}^{R}|=_F^{\perp}$, ${}^{R}|=_{TF}^{\perp}$.

Особливості цих відношень проявляються вже на пропозиційному рівні, тому в першу чергу розглядаємо властивості пропозиційного рівня.

Для множини формул $\Sigma\subseteq Fr$ використаємо позначення $\neg\Sigma=\{\neg\Phi\mid\Phi\in\Sigma\}$.

Позначимо $\bigcap\limits_{\theta\in\Sigma}T(\neg\theta_J)$ як $T^{\cap}(\neg\Sigma_J)$, $\bigcap\limits_{\theta\in\Sigma}F(\neg\theta_J)$ як $F^{\cap}(\neg\Sigma_J)$, $\bigcap\limits_{\theta\in\Sigma}\perp(\neg\theta_J)$ як $\perp^{\cap}(\neg\Sigma_J)$, $\bigcup\limits_{\theta\in\Sigma}T(\neg\theta_J)$ як $T^{\cup}(\neg\Sigma_J)$, $\bigcup\limits_{\theta\in\Sigma}F(\neg\theta_J)$ як $F^{\cup}(\neg\Sigma_J)$, $\bigcup\limits_{\theta\in\Sigma}\perp(\neg\theta_J)$ як $\perp^{\cup}(\neg\Sigma_J)$.

Маємо $\perp(S)=\overline{T(S)}\cap\overline{F(S)}$. Звідси

$$\perp^{\cap}(\mathrm{U}_J)=\bigcap_{\theta\in\mathrm{U}}\perp(\theta_J)=\bigcap_{\theta\in\mathrm{U}}(\overline{T(\theta_J)}\cap\overline{F(\theta_J)})=$$
$$=\bigcap_{\theta\in\mathrm{U}}\overline{T(\theta_J)}\cap\bigcap_{\theta\in\mathrm{U}}\overline{F(\theta_J)}=\overline{\bigcup_{\theta\in\mathrm{U}}T(\theta_J)}\cap\overline{\bigcup_{\theta\in\mathrm{U}}F(\theta_J)}.$$

Таким чином, для R-предикатів:

$$\perp^{\cap}(\mathrm{U}_J)=\overline{\bigcup_{\theta\in\mathrm{U}}T(\theta_J)}\cap\overline{\bigcup_{\theta\in\mathrm{U}}T(\neg\theta_J)}=$$
$$=\overline{\bigcup_{\theta\in\mathrm{U}}F(\theta_J)}\cap\overline{\bigcup_{\theta\in\mathrm{U}}F(\neg\theta_J)}.$$

Звідси визначальне твердження:

Теорема 9. Відношення $_J|=_T^{\perp}$ можна звести до $_J|=_T$; відношення $_J|=_F^{\perp}$ можна звести до $_J|=_F$.

Для довільних $\Gamma, U, \Delta \subseteq Fr$ та $\boldsymbol{J}$ маємо (тут $\Leftrightarrow_s$ означає "$\Leftrightarrow$ згідно Set1"):

$$U/\Gamma_J|=_T^{\perp}\Delta \Leftrightarrow T^{\cap}(\Gamma_J)\cap\perp^{\cap}(U_J)\subseteq T^{\cup}(\Delta_J)\Leftrightarrow$$

$$T^{\cap}(\Gamma_J)\cap\overline{\bigcup_{\theta\in U}T(\theta_J)}\cap\overline{\bigcup_{\theta\in U}T(\neg\theta_J)}\subseteq T^{\cup}(\Delta_J)\Leftrightarrow_s$$

$$T^{\cap}(\Gamma_J)\subseteq T^{\cup}(\Delta_J)\cup\bigcup_{\theta\in U}T(\theta_J)\cup\bigcup_{\theta\in U}T(\neg\theta_J)\Leftrightarrow$$

$$T^{\cap}(\Gamma_J)\subseteq T^{\cup}(\Delta_J)\cup T^{\cup}(U_J)\cup T^{\cup}(\neg U_J)\Leftrightarrow$$

$$\Leftrightarrow \Gamma_J|=_T\Delta, U, \neg U;$$

$$U/\Gamma_J|=_F^{\perp}\Delta \Leftrightarrow F^{\cap}(\Delta_J)\cap\perp^{\cap}(U_J)\subseteq F^{\cup}(\Gamma_J)\Leftrightarrow$$

$$F^{\cap}(\Delta_J)\cap\overline{\bigcup_{\theta\in U}F(\theta_J)}\cap\overline{\bigcup_{\theta\in U}F(\neg\theta_J)}\subseteq F^{\cup}(\Gamma_J)\Leftrightarrow_s$$

$$F^{\cap}(\Delta_J)\subseteq F^{\cup}(\Gamma_J)\cup\bigcup_{\theta\in U}F(\theta_J)\cup\bigcup_{\theta\in U}F(\neg\theta_J)\Leftrightarrow$$

$$F^{\cap}(\Delta_J)\subseteq F^{\cup}(\Gamma_J)\cup F^{\cup}(U_J)\cup F^{\cup}(\neg U_J)\Leftrightarrow$$

$$\Leftrightarrow \Gamma, U, \neg U_J|=_F\Delta.$$

Подібним способом отримуємо загальніші твердження:

$$W, U/\Gamma_J|=_T^{\perp}\Delta \Leftrightarrow W/\Gamma_J|=_T^{\perp}\Delta, U, \neg U;$$

$$W, U/\Gamma_J|=_F^{\perp}\Delta \Leftrightarrow W/\Gamma, U, \neg U_J|=_F^{\perp}\Delta.$$

Ми отримали фундаментальну теорему про елімінацію умов невизначеності:

Теорема 10.

$$W, U/\Gamma\,^{P}|=_T^{\perp}\Delta \Leftrightarrow W/\Gamma\,^{P}|=_T^{\perp}\Delta, U, \neg U;$$

$$W, U/\Gamma\,^{R}|=_T^{\perp}\Delta \Leftrightarrow W/\Gamma\,^{R}|=_T^{\perp}\Delta, U, \neg U;$$

$$W, U/\Gamma\,^{P}|=_F^{\perp}\Delta \Leftrightarrow W/\Gamma, U, \neg U\,^{P}|=_F^{\perp}\Delta;$$

$$W, U/\Gamma\,^{R}|=_F^{\perp}\Delta \Leftrightarrow W/\Gamma, U, \neg U\,^{R}|=_F^{\perp}\Delta.$$

Як наслідок отримуємо: відношення $^{P}|=_T^{\perp}$, $^{P}|=_F^{\perp}$, $^{R}|=_T^{\perp}$, $^{R}|=_F^{\perp}$ можна цілком звести до відношень $^{Pc}|=_T$, $^{Pc}|=_F$, $^{Rc}|=_T$, $^{Rc}|=_F$.

Наслідок 1.

$$U/\Gamma\,^{P}|=_T^{\perp}\Delta \Leftrightarrow \Gamma\,^{Pc}|=_T\Delta, U, \neg U;$$

$$U/\Gamma\,^{R}|=_T^{\perp}\Delta \Leftrightarrow \Gamma\,^{Rc}|=_T\Delta, U, \neg U;$$

$$U/\Gamma\,^{P}|=_F^{\perp}\Delta \Leftrightarrow \Gamma, U, \neg U\,^{Pc}|=_F\Delta;$$

$$U/\Gamma\,^{R}|=_F^{\perp}\Delta \Leftrightarrow \Gamma, U, \neg U\,^{Rc}|=_F\Delta.$$

Теорема 10 дає змогу сформулювати властивості елімінації умов невизначеності у відношеннях $^{P}|=_T^{\perp}$, $^{P}|=_F^{\perp}$, $^{R}|=_T^{\perp}$, $^{R}|=_F^{\perp}$:

ElU$_T$) $W,U/\Gamma\,^{P}|=_T^{\perp}\Delta \Leftrightarrow W/\Gamma\,^{P}|=_T^{\perp}\Delta, U, \neg U;$

$W, U/\Gamma\,^{R}|=_T^{\perp}\Delta \Leftrightarrow W/\Gamma\,^{R}|=_T^{\perp}\Delta, U, \neg U;$

ElU$_F$) $W,U/\Gamma\,^{P}|=_F^{\perp}\Delta \Leftrightarrow W/\Gamma, U, \neg U\,^{P}|=_F^{\perp}\Delta;$

$W, U/\Gamma\,^{R}|=_F^{\perp}\Delta \Leftrightarrow W/\Gamma, U, \neg U\,^{R}|=_F^{\perp}\Delta.$

Таким чином, від відношень $^{P}|=_T^{\perp}$, $^{P}|=_F^{\perp}$, $^{R}|=_T^{\perp}$, $^{R}|=_F^{\perp}$ можна перейти до відношень $^{Pc}|=_T$, $^{Pc}|=_F$, $^{Rc}|=_T$, $^{Rc}|=_F$.

Дещо складніша ситуація із відношеннями $^{P}|=_{TF}^{\perp}$ та $^{R}|=_{TF}^{\perp}$. Маємо:

$U/\Gamma|=_{TF}^{\perp}\Delta \Leftrightarrow U/\Gamma|=_T^{\perp}\Delta$ та $\Gamma|=_F\Delta \Leftrightarrow$

$\Leftrightarrow \Gamma|=_T\Delta, U, \neg U$ та $\Gamma, U, \neg U|=_F^{\perp}\Delta$.

Звідси:

Наслідок 2. $U/\Gamma\,^{P}|=_{TF}^{\perp}\Delta \Leftrightarrow$

$\Leftrightarrow \Gamma\,^{Pc}|=_T\Delta, U, \neg U$ та $\Gamma, U, \neg U\,^{Pc}|=_F^{\perp}\Delta.$

$U/\Gamma\,^{R}|=_{TF}^{\perp}\Delta \Leftrightarrow$

$\Leftrightarrow \Gamma\,^{Rc}|=_T\Delta, U, \neg U$ та $\Gamma, U, \neg U\,^{Rc}|=_F^{\perp}\Delta.$

Отже, відношення типу $|=_{TF}^{\perp}$ можна звести лише до сукупності відношень типу $|=_T$ та типу $|=_F$, а не до єдиного відношення типу $|=_{TF}$. Те ж саме було для $^{Pc}|=_{TF}$ та $^{Rc}|=_{TF}$: проведення декомпозиції формул вимагає подання $^{Rc}|=_{TF}$ через $^{Rc}|=_T$ та $^{Rc}|=_F$, а $^{Pc}|=_{TF}$ – через $^{Pc}|=_T$ та $^{Pc}|=_F$.

Підсумовуючи, отримуємо:

Теорема 11. Відношення $^{P}|=_T^{\perp}$, $^{P}|=_F^{\perp}$, $^{R}|=_T^{\perp}$, $^{R}|=_F^{\perp}$ та $^{P}|=_{TF}^{\perp}$, $^{R}|=_{TF}^{\perp}$ можна промоделювати за допомогою $^{Pc}|=_T$, $^{Pc}|=_F$, $^{Rc}|=_T$, $^{Rc}|=_F$, усунувши умови невизначеності.

Водночас для відношення $|=_{IR}^{\perp}$ *неможливо* таким способом усунути умови невизначеності і тим самим промоделювати $|=_{IR}^{\perp}$ за допомогою $^{Pc}|=_{IR}$. Це фактично вже випливає з теореми 4: для відношення $^{Pc}|=_{IR}$ неможливо задати коректним способом умови декомпозиції формул вигляду $\tilde{}\Phi$. Доведення неможливості моделювання $|=_{IR}^{\perp}$ за допомогою $^{Pc}|=_{IR}$ подібне до доведення теореми 4:

Нехай $\Phi, \Psi, \vartheta \in Fr$; $\boldsymbol{J} \in \boldsymbol{P_C}$. Маємо $\vartheta / \Phi_J \models_{IR}^{\perp} \Psi \Leftrightarrow T(\Phi_J) \cap F(\Psi_J) \cap \perp(\vartheta_J) = \varnothing \Leftrightarrow T(\Phi_J) \cap F(\Psi_J) \cap \overline{T(\theta_J) \cup F(\theta_J)} = \varnothing$

Нехай предикат α утворено за допомогою $\vee$ та $\neg$ із деяких $\alpha_1,\ldots, \alpha_n$ Тоді $T(\alpha)$ та $F(\alpha)$ подаються через області істинності й хибності предикатів $\alpha_1,\ldots, \alpha_n$ за допомогою лише $\cap$ та $\cup$. Проте із множин X, Y таких, що $X \cap Y = \varnothing$, за допомогою $\cap$ та $\cup$ можна отримати лише 4 різних множини: $\varnothing$, X, Y, $X \cup Y$. Нехай $f(X, Y)$ – будована із $\cap$ та $\cup$ така теоретико-множинна функція, що за умови $X \cap Y = \varnothing$ маємо $L \cap \overline{X \cup Y} = \varnothing \Leftrightarrow L \cap f(X, Y) = \varnothing$. Має виконуватись наступна умова: $\vartheta / \Phi_J \models_{IR}^{\perp} \Psi \Leftrightarrow T(\Phi_J) \cap F(\Psi_J) \cap \overline{T(\theta_J) \cup F(\theta_J)} = \varnothing \Leftrightarrow T(\Phi_J) \cap F(\Psi_J) \cap f(T(\theta_J), F(\theta_J)) = \varnothing$. Проте для $f(T(\vartheta_J), F(\vartheta_J))$ маємо лише 4 варіанти: $\varnothing$, $T(\vartheta_J)$, $F(\vartheta_J)$, $T(\vartheta_J) \cup F(\vartheta_J)$. Жоден з них не влаштовує нашу умову. Таким чином:

Теорема 12. В LC відношення $\models_{IR}^{\perp}$ не можна звести до відношення $^{Pc}\models_{IR}$.

Укажемо основні властивості відношень $^{P}\models_{T}^{\perp}$, $^{P}\models_{F}^{\perp}$, $^{R}\models_{T}^{\perp}$, $^{R}\models_{F}^{\perp}$, $^{P}\models_{TF}^{\perp}$, $^{R}\models_{TF}^{\perp}$. Надалі, якщо інше не зазначене окремо:

$\models_{*}^{\perp}$ – одне з $^{P}\models_{T}^{\perp}$, $^{P}\models_{F}^{\perp}$, $^{R}\models_{T}^{\perp}$, $^{R}\models_{F}^{\perp}$;

$\models_{T}^{\perp}$ – одне з $^{P}\models_{T}^{\perp}$, $^{R}\models_{T}^{\perp}$;

$\models_{F}^{\perp}$ – одне з $^{P}\models_{F}^{\perp}$, $^{R}\models_{F}^{\perp}$;

$\models_{TF}^{\perp}$ – одне з $^{P}\models_{TF}^{\perp}$, $^{R}\models_{TF}^{\perp}$.

Із визначень отримуємо властивості монотонності M:

M) Нехай $\Gamma \subseteq \Lambda$, $\mathrm{U} \subseteq \mathrm{W}$, та $\Delta \subseteq \Sigma$; тоді $\mathrm{U} / \Gamma \models_{*}^{\perp} \Delta \Rightarrow \mathrm{W} / \Lambda \models_{*}^{\perp} \Sigma$.

Маємо такі властивості гарантованої наявності логічного наслідку.

C) $\mathrm{U} / \Phi, \Gamma \models_{*}^{\perp} \Delta, \Phi$.

$C\neg^{\sim}$) $\mathrm{U} / \neg^{\sim}\Phi, \Gamma \models_{T}^{\perp} \Delta$.

$C^{\sim}$) $\mathrm{U} / \Gamma \models_{F}^{\perp} \Delta, {\sim}\Phi$.

Властивість *C* випливає з того що $T(\Phi_J) \cap \mathrm{L} \subseteq T(\Phi_J) \cup \mathrm{M}$, $F(\Phi_J) \cap \mathrm{L} \subseteq F(\Phi_J) \cup \mathrm{M}$.

В силу $T(\neg^{\sim}\Phi_J) = F({\sim}\Phi_J) = \varnothing$ умова $T(\neg^{\sim}\Phi_J) \cap T^{\cap}(\Gamma_J) \cap \perp^{\cap}(\mathrm{U}_J) \subseteq T^{\cup}(\Delta_J)$ гарантована, звідки випливає $C\neg^{\sim}$.

Властивість $C^{\sim}$ випливає з того, що $F({\sim}\Phi_J) = \varnothing$, звідки маємо гарантоване $F^{\cap}(\Delta_J) \cap \perp^{\cap}(\mathrm{U}_J) \cap F({\sim}\Phi_J) \subseteq F^{\cup}(\Gamma_J)$.

Додатково гарантують наявність відповідного відношення такі властивості.

CL) $\mathrm{U} / \Phi, \neg\Phi, \Gamma \models_{T}^{\perp} \Delta$; це рівносильне такому: $\Phi, \neg\Phi, \Gamma \models_{T}^{Pc} \Delta, \mathrm{U}, {\neg}\mathrm{U}$;

CR) $\mathrm{U} / \Gamma \models_{F}^{\perp} \Delta, \Phi, \neg\Phi$; це рівносильне такому: $\Gamma, \mathrm{U}, {\neg}\mathrm{U} \models_{F}^{Pc} \Delta, \Phi, \neg\Phi$.

Властивості *C*, $C^{\sim}$, $C\neg^{\sim}$, *CL*, *CR* вже були сформульовані для відношень логічного наслідку в LC, тут вони узагальнені для відношень логічного наслідку за умов невизначеності.

Властивості декомпозиції логічних зв'язок аналогічні відповідним властивостям для відношень $^{P}\models_{T}$, $^{P}\models_{F}$, $^{P}\models_{TF}$, $^{R}\models_{TF}$ традиційної логіки квазіарних предикатів та наведеним вище відповідним властивостям для $^{Pc}\models_{T}$, $^{Pc}\models_{F}$, $^{Pc}\models_{TF}$, $^{Rc}\models_{T}$, $^{Rc}\models_{F}$, $^{Rc}\models_{TF}$; вони виконуються для усіх відношень $^{P}\models_{T}^{\perp}$, $^{P}\models_{F}^{\perp}$, $^{R}\models_{T}^{\perp}$, $^{R}\models_{F}^{\perp}$, $^{P}\models_{TF}^{\perp}$, $^{R}\models_{TF}^{\perp}$.

$\neg\neg_{\mathrm{L}}$) $\mathrm{U} / \neg\neg\Phi, \Gamma \models_{*}^{\perp} \Delta \Leftrightarrow \mathrm{U} / \Phi, \Gamma \models_{*}^{\perp} \Delta$.

$\neg\neg_{\mathrm{R}}$) $\mathrm{U} / \Gamma \models_{*}^{\perp} \Delta, \neg\neg\Phi \Leftrightarrow \mathrm{U} / \Gamma \models_{*}^{\perp} \Delta, \Phi$.

$\vee_{\mathrm{L}}$) $\mathrm{U} / \Phi \vee \Psi, \Gamma \models_{*}^{\perp} \Delta \Leftrightarrow$

$\Leftrightarrow \mathrm{U} / \Phi, \Gamma \models_{*}^{\perp} \Delta$ та $\mathrm{U} / \Psi, \Gamma \models_{*}^{\perp} \Delta$.

$\vee_{\mathrm{R}}$) $\mathrm{U} / \Gamma \models_{*}^{\perp} \Delta, \Phi \vee \Psi \Leftrightarrow \mathrm{U} / \Gamma \models_{*}^{\perp} \Delta, \Phi, \Psi$.

$\neg\vee_{\mathrm{L}}$) $\mathrm{U} / \neg(\Phi \vee \Psi), \Gamma \models_{*}^{\perp} \Delta \Leftrightarrow$

$\Leftrightarrow \mathrm{U} / \neg\Phi, \neg\Psi, \Gamma \models_{*}^{\perp} \Delta$.

$\neg\vee_{\mathrm{R}}$) $\mathrm{U} / \Gamma \models_{*}^{\perp} \Delta, \neg(\Phi \vee \Psi) \Leftrightarrow$

$\Leftrightarrow \mathrm{U} / \Gamma \models_{*}^{\perp} \Delta, \neg\Phi$ та $\mathrm{U} / \Gamma \models_{*}^{\perp} \Delta, \neg\Psi$.

Для $\models_{IR}^{\perp}$ додатково маємо [10] властивості $\neg_{\mathrm{L}}$ та $\neg_{\mathrm{R}}$. Ці властивості невірні

для ${}^{Pc}|\!=_T$, ${}^{Pc}|\!=_F$, ${}^{Pc}|\!=_{TF}$, ${}^{Rc}|\!=_T$, ${}^{Rc}|\!=_F$, ${}^{Rc}|\!=_{TF}$ та для ${}^{P}|\!=_T{}^{\perp}$, ${}^{P}|\!=_F{}^{\perp}$, ${}^{R}|\!=_T{}^{\perp}$, ${}^{R}|\!=_F{}^{\perp}$, ${}^{P}|\!=_{TF}{}^{\perp}$, ${}^{R}|\!=_{TF}{}^{\perp}$.

Для ${}^{P}|\!=_T{}^{\perp}$, ${}^{P}|\!=_F{}^{\perp}$, ${}^{R}|\!=_T{}^{\perp}$, ${}^{R}|\!=_F{}^{\perp}$ властивості декомпозиції, в яких фігурує $\sim\Phi$, аналогічні наведеним вище відповідним властивостям для ${}^{Pc}|\!=_T$, ${}^{Pc}|\!=_F$, ${}^{Rc}|\!=_T$, ${}^{Rc}|\!=_F$.

$\sim_{\mathrm{LT}}$) $U/\sim\Phi,\Gamma \mid\!=_T{}^{\perp}\Delta \Leftrightarrow U/\Gamma \mid\!=_T{}^{\perp}\Delta,\Phi,\neg\Phi$;

$\sim_{\mathrm{RT}}$) $U/\Gamma \mid\!=_T{}^{\perp}\Delta,\sim\Phi \Leftrightarrow$

$\Leftrightarrow U/\Phi,\Gamma \mid\!=_T{}^{\perp}\Delta$ та $U/\neg\Phi,\Gamma \mid\!=_T{}^{\perp}\Delta$;

$\neg\sim_{\mathrm{RF}}$) $U/\Gamma \mid\!=_F{}^{\perp}\Delta,\neg\sim\Phi \Leftrightarrow$

$\Leftrightarrow U/\Gamma,\Phi,\neg\Phi \mid\!=_F{}^{\perp}\Delta$;

$\neg\sim_{\mathrm{LF}}$) $U/\neg\sim\Phi,\Gamma \mid\!=_F{}^{\perp}\Delta \Leftrightarrow$

$\Leftrightarrow U/\Gamma \mid\!=_F{}^{\perp}\Delta,\Phi$ та $U/\Gamma \mid\!=_F{}^{\perp}\Delta,\neg\Phi$;

$\neg\sim_{\mathrm{El}}$) $U/\Gamma \mid\!=_T{}^{\perp}\Delta,\neg\sim\Phi \Leftrightarrow U/\Gamma \mid\!=_T{}^{\perp}\Delta$;

$\sim_{\mathrm{El}}$) $U/\sim\Phi,\Gamma \mid\!=_F{}^{\perp}\Delta \Leftrightarrow U/\Gamma \mid\!=_F{}^{\perp}\Delta$.

Властивості $\neg\sim_{\mathrm{El}}$ та $\sim_{\mathrm{El}}$ – це властивості спрощення, елімінації $\sim$.

Доведення цих властивостей зводиться до елімінації умов невизначеності згідно теореми 10, застосування однойменних властивостей для відношень логічного наслідку в LC та застосування теореми 10 в зворотному порядку. Наведемо для прикладу доведення $\sim_{\mathrm{LT}}$ та $\neg\sim_{\mathrm{LF}}$.

Маємо $U/\sim\Phi,\Gamma \mid\!=_T{}^{\perp}\Delta \Leftrightarrow$ (теорема 10) $\sim\Phi,\Gamma \mid\!=_T\Delta,U,\neg U \Leftrightarrow$ ($\sim_{\mathrm{LT}}$ в LC) $\Leftrightarrow \Gamma \mid\!=_T\Delta,U,\neg U,\Phi,\neg\Phi \Leftrightarrow$ (теорема 10) $U/\Gamma \mid\!=_T{}^{\perp}\Delta,\Phi,\neg\Phi$.

Маємо $U/\neg\sim\Phi,\Gamma \mid\!=_F{}^{\perp}\Delta \Leftrightarrow$ (теорема 10) $\neg\sim\Phi,\Gamma,U,\neg U \mid\!=_F\Delta \Leftrightarrow$ ($\neg\sim_{\mathrm{LF}}$ в LC) $\Leftrightarrow \Gamma,U,\neg U \mid\!=_F\Delta,\Phi$ та $\Gamma,U,\neg U \mid\!=_F\Delta,\neg\Phi$ $\Leftrightarrow$ (теорема 10) $U/\Gamma \mid\!=_F{}^{\perp}\Delta,\Phi$ та $U/\Gamma \mid\!=_F{}^{\perp}\Delta,\neg\Phi$.

Для всіх відношень ${}^{P}|\!=_T{}^{\perp}$, ${}^{P}|\!=_F{}^{\perp}$, ${}^{R}|\!=_T{}^{\perp}$, ${}^{R}|\!=_F{}^{\perp}$ можна сформулювати:

$C_{\sim *}$) $U,\Phi/\Gamma \mid\!=_*{}^{\perp}\Delta,\sim\Phi$;

$C_{\neg\sim *}$) $U,\Phi/\neg\sim\Phi,\Gamma \mid\!=_*{}^{\perp}\Delta$.

Для $|\!=_F{}^{\perp}$ властивість $C_{\sim *}$ – окремий випадок $C\sim$; для $|\!=_T{}^{\perp}$ $C_{\sim *}$ є похідною від $\mathrm{ElU_T}$, $\sim_{\mathrm{RT}}$, C. Для $|\!=_T{}^{\perp}$ властивість $C_{\neg\sim *}$ – окремий випадок $C\neg\sim$; для $|\!=_F{}^{\perp}$ ця властивість є похідною від $\mathrm{ElU_F}$, $\neg\sim_{\mathrm{LF}}$, C.

Властивості декомпозиції формул вигляду $\sim\Phi$ *різні* для відношень типу $|\!=_T{}^{\perp}$ та типу $|\!=_F{}^{\perp}$, тому для відношень типу $|\!=_{TF}{}^{\perp}$ властивості декомпозиції $\sim\Phi$ не можна подати як спільну властивість для відношень типу $|\!=_{TF}$.

Проте $\Gamma \mid\!=_{TF}{}^{\perp}\Delta \Leftrightarrow \Gamma \mid\!=_T{}^{\perp}\Delta$ та $\Gamma \mid\!=_F{}^{\perp}$, тому властивості декомпозиції та гарантованої наявності логічного наслідку для відношень типу $|\!=_{TF}{}^{\perp}$ можна задати опосередковано, через відповідні властивості для відношень типу $|\!=_T{}^{\perp}$ та типу $|\!=_F{}^{\perp}$.

Властивості, в яких фігурує $\sim\Phi$, *істотно різні* для $|\!=_{IR}{}^{\perp}$, ${}^{P}|\!=_T{}^{\perp}$, ${}^{P}|\!=_F{}^{\perp}$, ${}^{R}|\!=_T{}^{\perp}$, ${}^{R}|\!=_F{}^{\perp}$, ${}^{P}|\!=_{TF}{}^{\perp}$, ${}^{R}|\!=_{TF}{}^{\perp}$. Це ще раз засвідчує відмінність усіх зазначених відношень.

Пов'язані з реномінацією властивості еквівалентних перетворень для відношень ${}^{P}|\!=_T{}^{\perp}$, ${}^{R}|\!=_T{}^{\perp}$, ${}^{P}|\!=_F{}^{\perp}$, ${}^{R}|\!=_F{}^{\perp}$ подібні до відповідних властивостей еквівалентних перетворень для відношень ${}^{Pc}|\!=_T$, ${}^{Pc}|\!=_F$, ${}^{Rc}|\!=_T$, ${}^{Rc}|\!=_F$. Їх доведення базується на теоремі 8. Кожна з властивостей R, RI, RU, RR, $R\neg$, $R\vee, R\sim$, $R\exists$, $R\exists s$ дає 6 відповідних властивостей для ${}^{P}|\!=_T{}^{\perp}$, ${}^{P}|\!=_F{}^{\perp}$, ${}^{R}|\!=_T{}^{\perp}$, ${}^{R}|\!=_F{}^{\perp}$, коли виділена формула чи її заперечення знаходиться у лівій чи правій частині цього відношення або входить до умови невизначеності. Враховуючи властивості $\mathrm{ElU_T}$ та $\mathrm{ElU_F}$ елімінації умови невизначеності, явно виписувати 2 випадки, коли виділена формула чи її заперечення входить до умови невизначеності, немає потреби, тому залишаються 4 випадки. Наведемо для прикладу властивості, індуковані, $R\sim$.

Нехай $\Phi,\Psi\in Fr$; $\Gamma,U,\Delta\subseteq Fr$; $|\!=_*{}^{\perp}$ – одне з ${}^{P}|\!=_T{}^{\perp}$, ${}^{P}|\!=_F{}^{\perp}$, ${}^{R}|\!=_T{}^{\perp}$, ${}^{R}|\!=_F{}^{\perp}$.

$\mathrm{R}\sim_{\mathrm{L}}$) $U/R_{\bar{x}}^{\bar{v}}(\sim\Phi),\Gamma \mid\!=_*{}^{\perp}\Delta \Leftrightarrow$

$\Leftrightarrow\ \mathrm{U}/{\sim}R^{\bar{v}}_{\bar{x}}(\Phi),\Gamma\models_*^{\perp}\Delta;$

$\mathrm{R}{\sim}_{\mathrm{R}})\ \mathrm{U}/\Gamma\models_*^{\perp}\Delta, R^{\bar{v}}_{\bar{x}}({\sim}\Phi)\ \Leftrightarrow$

$\Leftrightarrow\ \mathrm{U}/\Gamma\models_*^{\perp}\Delta, {\sim}R^{\bar{v}}_{\bar{x}}(\Phi);$

$\neg\,\mathrm{R}{\sim}_{\mathrm{L}})\ \mathrm{U}/\neg R^{\bar{v}}_{\bar{x}}({\sim}\Phi),\Gamma\models_*^{\perp}\Delta\ \Leftrightarrow$

$\Leftrightarrow\ \mathrm{U}/\neg{\sim}R^{\bar{v}}_{\bar{x}}(\Phi),\Gamma\models_*^{\perp}\Delta;$

$\neg\,\mathrm{R}{\sim}_{\mathrm{R}})\ \mathrm{U}/\Gamma\models_*^{\perp}\Delta, \neg R^{\bar{v}}_{\bar{x}}({\sim}\Phi)\ \Leftrightarrow$

$\Leftrightarrow\ \mathrm{U}/\Gamma\models_*^{\perp}\Delta, \neg{\sim}R^{\bar{v}}_{\bar{x}}(\Phi).$

Для відношень типу $\models_{TF}^{\perp}$ властивості еквівалентних перетворень можна задати опосередковано, через відповідні властивості для відношень типу $\models_T^{\perp}$ та $\models_F^{\perp}$.

Властивості елімінації кванторів, *E*-розподілу та первісного означення для відношень ${}^{P}\models_T^{\perp}$, ${}^{R}\models_T^{\perp}$, ${}^{P}\models_F^{\perp}$, ${}^{R}\models_F^{\perp}$ подібні до відповідних властивостей для відношень ${}^{Pc}\models_T$, ${}^{Rc}\models_T$, ${}^{Pc}\models_F$, ${}^{Rc}\models_F$, сформульованих в теоремах 5 та 6.

Наведемо для прикладу властивості $\exists_{\mathrm{L}}$ та $\exists\mathrm{Rv}_{\mathrm{R}}$.

$\exists_{\mathrm{L}})$ за умови $z\in fu(\mathrm{U},\Gamma,\Delta,\exists x\Phi))$

$\mathrm{U}/\exists x\Phi,\Gamma\models_*^{\perp}\Delta \Leftrightarrow \mathrm{U}/R^{x}_{\mathrm{z}}(\Phi), Ez,\Gamma\models_*^{\perp}\Delta;$

$\exists\mathrm{Rv}_{\mathrm{R}})\ \mathrm{U}/\Gamma, Ey\models_*^{\perp}\Delta, R^{\bar{u}}_{\bar{v}}(\exists x\Phi)\ \Leftrightarrow$

$\Leftrightarrow\ \mathrm{U}/\Gamma, Ey\models_*^{\perp}\Delta, R^{\bar{u}}_{\bar{v}}(\exists x\Phi), R^{\bar{u},x}_{\bar{v},y}(\Phi).$

Доведення властивостей елімінації кванторів та побудова для LC першопорядкових числень секвенційного типу будуть проведені в наступних статтях.

Висновки

В роботі досліджено першопорядкові LC – логіки часткових предикатів з композицією (операцією) предикатного доповнення ~ Подібні операції використовуються в різних варіантах логік Флойда-Хоара з частковими перед- та після-умовами. Описано композиційні алгебри і мови LC. Запропоновано низку відношень логічного наслідку в LC (типів $\models_T$, $\models_F$, $\models_{TF}$, а також відношення ${}^{Pc}\models_{IR}$) та відношень логічного наслідку за умови невизначеності (типів $\models_T^{\perp}$, $\models_F^{\perp}$, $\models_{TF}^{\perp}$). Досліджено властивості цих відношень, встановлено співвідношення між ними. Для запропонованих відношень описано умови їх гарантованої наявності, наведено властивості декомпозиції формул та елімінації кванторів.

Для відношень типів $\models_T$ та $\models_F$ доведено теорему про елімінацію умов невизначеності, що дає змогу звести відношення ${}^{P}\models_T^{\perp}$, ${}^{P}\models_F^{\perp}$, ${}^{R}\models_T^{\perp}$, ${}^{R}\models_F^{\perp}$ до відношень ${}^{Pc}\models_T$, ${}^{Pc}\models_F$, ${}^{Rc}\models_T$, ${}^{Rc}\models_F$. Водночас $\models_{IR}^{\perp}$ неможливо звести до відношення ${}^{Pc}\models_{IR}$. Понад те, для ${}^{Pc}\models_{IR}$ неможливо коректно задати умови декомпозиції формул ${\sim}\Phi$.

Встановлені властивості LC засвідчують істотну її відмінність від традиційної логіки квазіарних предикатів.

Для пропонованих відношень логічного наслідку в LC та відношень логічного наслідку за умов невизначеності в наступних статтях планується побудова першопорядкових числень секвенційного типу.

Література

1. Handbook of Logic in Computer Science. Edited by S. Abramsky, Dov M. Gabbay and T. S. E. Maibaum. Oxford University Press, Vol. 1–5. 1993–2000.
2. S.C. Kleene. Mathematical Logic. New York, 1967.
3. Нікітченко М.С., Шкільняк С.С. Прикладна логіка. К.: ВПЦ Київський університет, 2013. 278 с.
4. Нікітченко М.С., Шкільняк С.С. Математична логіка та теорія алгоритмів. К.: ВПЦ Київський університет, 2008. 528 с.
5. Нікітченко М.С., Шкільняк О.С., Шкільняк С.С. Чисті першопорядкові логіки квазіарних предикатів. *Проблеми програмування*. 2016. № 2–3. С. 73–86.
6. Mykola S. Nikitchenko and Stepan S. Shkilniak. Algebras and logics of partial quasiary predicates. Algebra and Discrete Mathematics, Vol. 23 (2017). N 2. P. 263–278.
7. Hoare C. An axiomatic basis for computer programming, Comm. 1969. ACM. 12(10). P. 576–580.
8. Ivanov I., Nikitchenko M. On the sequence rule for the Floyd-Hoare logic with partial pre- and post-conditions. In Proceedings of the 14th International Conference on ICT.

Vol. 2104 of CEUR Workshop Proc. 2018. P. 716–724.
9. Ivanov I., Nikitchenko M. Inference Rules for the Partial Floyd-Hoare Logic Based on Composition of Predicate Complement, Comm. in Computer and Information Science. 2019. Vol. 1007. Springer, Cham. P. 71–88.
10. Нікітченко М.С., Шкільняк О.С., Шкільняк С.С., Мамедов Т.А. Пропозиційні логіки часткових предикатів з композицією предикатного доповнення. *Проблеми програмування.* 2019. № 1. С. 3–13.

References

1. ABRAMSKY, S., GABBAY, D. and MAIBAUM, T. (editors). (1993–2000). *Handbook of Logic in Computer Science* Oxford University Press, Vol. 1–5.
2. KLEENE, S. (1967) *Mathematical Logic*. New York.
3. NIKITCHENKO, M. and SHKILNIAK, S. (2013). *Applied logic*. Kyiv: VPC Kyivskyi Universytet (in ukr).
4. NIKITCHENKO, M. and SHKILNIAK, S. (2008). *Mathematical logic and theory of algorithms*. Kyiv: VPC Kyivskyi Universytet (in ukr).
5. NIKITCHENKO, M., SHKILNIAK, O. and SHKILNIAK, S. (2016). Pure first-order logics of quasiary predicates. In *Problems in Progamming*. No 2–3. P. 73–86 (in ukr).
6. NIKITCHENKO, M. and SHKILNIAK, S. (2017). Algebras and logics of partial quasiary predicates. In *Algebra and Discrete Mathematics*. Vol. 23. No 2. P. 263–278.
7. HOARE, C. (1969). An axiomatic basis for computer programming, Comm. ACM 12(10), 576–580, 1969.
8. IVANOV, I. and NIKITCHENKO, M. (2018). On the sequence rule for the Floyd-Hoare logic with partial pre- and post-conditions. *In Proceedings of the 14th International Conference on ICT.* Vol 2104 of CEUR Workshop Proc., P. 716–724.
9. IVANOV, I. and NIKITCHENKO, M. (2019). Inference Rules for the Partial Floyd-Hoare Logic Based on Composition of Predicate Complement. *In Communication. in Computer and Information Science.* Vol. 1007. Springer, Cham, P. 71–88.
10. NIKITCHENKO, M., SHKILNIAK, O., SHKILNIAK, S. and MAMEDOV, T. (2019). Propositional logics of partial predicates with composition of predicate complement. In *Problems in Progamming*. No 1. P. 3–13 (in ukr).

Одержано 21.05.2019

Про автора:

Шкільняк Оксана Степанівна,
кандидат фізико-математичних наук,
доцент, доцент кафедри
інформаційних систем.
Кількість наукових публікацій в
українських виданнях – понад 95,
у тому числі у фахових виданнях – 36.
Кількість наукових публікацій в
зарубіжних виданнях – 13.
Scopus Author ID: 57190873266
h-індекс (Google Scholar): 5 (4 з 2014)
http://orcid.org/0000-0003-4139-2525.

***Місце роботи автора*:**

Київський національний університет
імені Тараса Шевченка,
01601, Київ, вул. Володимирська, 60.
Тел.: (044) 259 05 19.
E-mail: me.oksana@gmail.com

УДК 004.42:510.69 https://doi.org/10.15407/pp2019.03.028

С.С. Шкільняк

ПЕРШОПОРЯДКОВІ КОМПОЗИЦІЙНО-НОМІНАТИВНІ ЛОГІКИ З ПРЕДИКАТАМИ СЛАБКОЇ ТА СТРОГОЇ РІВНОСТІ

Досліджено нові програмно-орієнтовані логіки часткових предикатів з операцією (композицією) предикатного доповнення, такі логіки названо LC. Для першопорядкових LC запропоновано низку відношень логічного наслідку та відношень логічного наслідку за умови невизначеності. Досліджено властивості цих відношень, встановлено співвідношення між ними. Для відношень типів $|=_T$ та $|=_F$ доведено теорему про елімінацію умов невизначеності. Для запропонованих відношень описано умови їх гарантованої наявності, наведено властивості декомпозиції формул та елімінації кванторів.

Ключові слова: логіка, частковий предикат, композиційна алгебра, логічний наслідок.

Вступ

Поняття і методи математичної логіки засвідчують високу ефективність при розв'язанні широкого спектра задач інформатики й програмування (див., напр., [1]). Особливе місце серед них посідають задачі, пов’язані з розробкою надійного програмного забезпечення, найперше, із побудовою систем специфікації та верифікації програм. До найпоширеніших логічних формалізмів, які успішно використовуються в системах верифікації, належать логіки Флойда-Хоара [2, 3]. Такі логіки базуються на класичній логіці предикатів, яка в недостатній мірі враховує неповноту, частковість інформації про предметну область. Традиційні логіки Флойда-Хоара використовують тотальні перед- та після-умови (предикати), тому було запропоновано (див., напр., [4, 5]) їх узагальнення на випадок часткових предикатів. Важливим напрямком такого узагальнення стало введення спеціальної немонотонної операції (композиції) предикатного доповнення [5]. Композиційно-номінативні логіки часткових квазіарних предикатів, розширені композицією предикатного доповнення, названо LC. Пропозиційні LC детально описано в [6], чисті першопорядкові LC в розглянуто [7], відношення логічного наслідку в LC досліджено в [8].

В даній роботі запропоновано розширити композицією предикатного доповнення першопорядкові композиційно-номінативні логіки з предикатами рівності. Такі нові логіки названо LCE.

Спеціальні предикати рівності дають змогу ототожнювати й розрізняти значення предметних імен. Композиційно-номінативні логіки з предикатами рівності вивчалися в [9]. Можна виділити два різновиди цих предикатів: слабкої рівності та строгої рівності. Такі предикати можна розглядати на реномінативному рівні та на чистому першопорядковому (кванторному) рівні. Це дає принаймі 4 різновиди LCE: з предикатами слабкої рівності та з предикатами строгої рівності, які розглядаємо на реномінативному рівні та на чистому першопорядковому рівні.

В роботі описано композиційні алгебри та мови цих різновидів LCE. Основну увагу зосереджено на вивченні властивостей, пов’язаних з предикатами слабкої рівності й строгої рівності та з композицією предикатного доповнення. Введено та досліджено низку відношень логічного наслідку в першопорядкових LCE.

Поняття, які в роботі не визначаються, тлумачимо в сенсі [6, 9, 10].

1. Композиційні алгебри логік з предикатним доповненням та рівністю

Розглядаємо квазіарні предикати реляційного типу, або *R*-предикати [10].

V-A-квазіарний *R*-предикат – це часткова неоднозначна функція вигляду $Q : {}^{V}A \to \{T, F\}$, де $\{T, F\}$ – множина істиннісних значень, ${}^{V}A$ – множина всіх *V-A*-

іменних множин. Тут V та A – множини предметних імен та базових значень.

Позначаємо $Q[d]$ множину всіх значень, які R-предикат Q може приймати на аргументі $d \in {}^{V}A$. Така $Q[d]$ може бути однією з множин $\varnothing$, $\{T\}$, $\{F\}$, $\{T,F\}$. Отже, кожний R-предикат Q можна задати за допомогою множин $T(Q) = \{d \mid T \in Q[d]\}$ та $F(Q) = \{d \mid F \in Q[d]\}$, які називають областю істинності та областю хибності предиката Q. Для R-предикатів область невизначеності визначається множинами $T(Q)$ та $F(Q)$: $\perp(Q) = \overline{T(Q) \cup F(Q)} = \overline{T(Q)} \cap \overline{F(Q)}$.

R-предикат Q монотонний, якщо

$$d_1 \subseteq d_2 \Rightarrow Q[d_1] \subseteq Q[d_2].$$

R-предикат Q антитонний, якщо

$$d_1 \subseteq d_2 \Rightarrow Q[d_1] \supseteq Q[d_2].$$

R-предикат Q частковий однозначний або P-предикат, якщо $T(Q) \cap F(Q) = \varnothing$.

R-предикат Q тотальний, або T-предикат, якщо $T(Q) \cup F(Q) = {}^{V}A$.

Для P-предиката Q запис $Q(d)\downarrow$ означає, що $Q(d)$ визначене, запис $Q(d)\uparrow$ означає, що $Q(d)$ невизначене.

Можна виділити [10] 4 константні R-предикати T, F, $\curlywedge$, $\curlyvee$:

$$T(\mathrm{T}) = F(\mathrm{F}) = {}^{V}A, \quad T(\mathrm{F}) = F(\mathrm{T}) = \varnothing,$$
$$T(\curlywedge) = F(\curlywedge) = \varnothing, \; T(\curlyvee) = F(\curlyvee) = {}^{V}A.$$

Предикати T, F, $\curlywedge$, $\curlyvee$ відповідають множинам значень $\{T\}$, $\{F\}$, $\varnothing$, $\{T,F\}$, які R-предикат може прийняти на вхідному даному.

Предикати T, F, $\curlywedge$, $\curlyvee$ монотонні й антитонні, при цьому T, F, $\curlywedge$ однозначні.

Класи V-A-квазіарних R-предикатів та P-предикатів позначимо Pr^{V-A} та PrP^{V-A}.

Характерною особливістю LC є наявність спеціальної немонотонної 1-арної композиції предикатного доповнення $\tilde{}$ (див. [6]). Для R-предикатів $\tilde{}$ задамо так:

$$T(\tilde{}Q) = \perp(Q) = \overline{T(Q) \cup F(Q)},$$

$$F(\tilde{}Q) = \varnothing.$$

Звідси $\perp(\tilde{}Q) = T(Q) \cup F(Q)$.

Для P-предикатів композицію $\tilde{}$ можна задати [6] так:

$$(\tilde{}Q)(d) = \begin{cases} T, & \text{якщо } Q(d)\uparrow, \\ \text{невизначене}, & \text{якщо } Q(d)\downarrow \end{cases}$$

Із визначень випливає, що $\tilde{}$ зберігає однозначність R-предикатів

Таким чином, класи P-предикатів та R-предикатів замкнені щодо композиції $\tilde{}$.

Водночас композиція $\tilde{}$ не зберігає тотальність R-предикатів.

Твердження 1. Для кожного предиката $Q \in Pr^{V-A}$ маємо $\tilde{}Q \in PrP^{V-A}$;

для кожного $Q \in PrT^{V-A}$ маємо $\tilde{}Q = \curlywedge$.

Звідси випливає, що клас T-предикатів незамкнений щодо $\tilde{}$.

Це означає, що для T-предикатів LC не мають смислу.

Таким чином, можна розглядати LC R-предикатів та LC P-предикатів.

LC R-предикатів реномінативного рівня далі називаємо L^{RC}; LC P-предикатів реномінативного рівня називаємо L^{RCP};

LC R-предикатів чистого першопорядкового (кванторного) рівня називаємо L^{QC}; LC P-предикатів чистого першопорядкового рівня називаємо L^{QCP}.

В логіках квазіарних предикатів ототожнювати й розрізняти значення предметних імен можна за допомогою спеціальних 0-арних композицій – параметризованих за іменами предикатів рівності. Розглядають [9] два різновиди цих предикатів:

– слабкої (з точністю до визначеності) рівності $=_{xy}$;

– строгої (точної) рівності $\equiv_{xy}$.

Предикати $=_{xy}$ та $\equiv_{xy}$ задаємо так:

$$T(=_{xy}) = \{d \mid d(x)\downarrow, d(y)\downarrow \text{ та } d(x) = d(y)\},$$

$$F(=_{xy}) = \{d \mid d(x)\downarrow, d(y)\downarrow \text{ та } d(x) \neq d(y)\};$$

$$T(\equiv_{xy}) = \{d \mid d(x)\downarrow, d(y)\downarrow \text{ та } d(x) = d(y)\} \cup$$
$$\cup \{d \mid d(x)\uparrow \text{ та } d(y)\uparrow\},$$

$F(\equiv_{xy}) = \{d \mid d(x)\downarrow, d(y)\downarrow, d(x) \neq d(y)\} \cup$

$\cup \{d \mid d(x)\downarrow, d(y)\uparrow$ або $d(x)\uparrow, d(y)\downarrow\}$.

Предикати $=_{xy}$ та $\equiv_{xy}$ традиційно також позначають $x = y$ та $x \equiv y$.

Предикати $\equiv_{xy}$ тотальні однозначні, вони [9] немонотонні й неантитонні.

Водночас предикати $=_{xy}$ часткові однозначні, монотонні й еквітонні.

Неістотним для предикатів $=_{xy}$ та $\equiv_{xy}$ є кожне $z \in V \setminus \{x, y\}$.

КНЛ з предикатами рівності назвемо LE. Зокрема, LE з предикатами слабкої рівності назвемо LEw, LE з предикатами строгої рівності назвемо LEs.

На реномінативному й першопорядковому рівнях можна виділити [9] низку різновидів LE. Зокрема, для *R*-предикатів та *P*-предикатів маємо такі різновиди.

На реномінативному рівні:

– L^{RE} – LEw *R*-предикатів;

– L^{REP} – LEw *P*-предикатів;

– L^{REs} – LEs *R*-предикатів;

– L^{REsP} – LEs *P*-предикатів.

На чистому першопорядковому (кванторному) рівні:

– L^{QE} – LEw *R*-предикатів;

– L^{QEP} – LEw *P*-предикатів;

– L^{QEs} – LEs *R*-предикатів;

– L^{QEsP} – LEs *P*-предикатів.

LC з предикатами рівності назвемо LCE. Зокрема, LCE з предикатами слабкої рівності назвемо LCEw, LCE з предикатами строгої рівності назвемо LCEs.

Маємо такі різновиди LCE.

На реномінативному рівні:

– L^{RCE} – LCEw *R*-предикатів;

– L^{RCEP} – LCEw *P*-предикатів;

– L^{RCEs} – LCEs *R*-предикатів;

– L^{RCEsP} – LCEs *P*-предикатів.

На чистому першопорядковому рівні:

– L^{QCE} – LCEw *R*-предикатів;

– L^{QCEP} – LCEw *P*-предикатів;

– L^{QCEs} – LCEs *R*-предикатів;

– L^{QCEsP} – LCEs *P*-предикатів.

Спільними базовими композиціями зазначених вище різновидів КНЛ є логічні зв'язки $\neg$ та $\vee$, композиції реномінації $\mathrm{R}^{\bar{v}}_{\bar{x}}$, композиції квантифікації $\exists x$ для першопорядкових логік. Для КНЛ зі слабкою рівністю до них додаються предикати слабкої рівності $=_{xy}$, а для КНЛ зі строгою рівністю – предикати строгої рівності $=_{xy}$. Для LC та LCE додаємо композицію предикатного доповнення $\tilde{}$.

Стисло нагадаємо (див. [6]) визначення композицій $\neg$, $\vee$, $\mathrm{R}^{\bar{v}}_{\bar{x}}$, $\exists x$.

Задамо $\neg$, $\vee$, $\exists x$ через області істинності й хибності відповідних предикатів:

$$T(\neg P) = F(P);$$

$$F(\neg P) = T(P);$$

$$T(P \vee Q) = T(P) \cup T(Q);$$

$$F(P \vee Q) = F(P) \cap F(Q);$$

$T(\exists xP) = \{d \mid d\nabla x \mapsto a \in T(P)$ для деякого $a \in A\}$;

$F(\exists xP) = \{d \mid d\nabla x \mapsto a \in F(P)$ для всіх $a \in A\}$.

Композицію $\mathrm{R}^{\bar{v}}_{\bar{x}}$ задаємо умовою:

$$\mathrm{R}^{\bar{v}}_{\bar{x}}(P)(d) = P(\mathrm{r}^{\bar{v}}_{\bar{x}}(d)) \text{ для всіх } d \in {}^{V}A.$$

Традиційним є використання похідних композицій $\rightarrow$, &, $\forall x$, вони задаються так:

$$P \rightarrow Q = \neg P \vee Q;\ P \& Q = \neg(\neg P \vee \neg Q);$$

$$\forall xP = \neg\exists x \neg P.$$

Підсумовуючи, для зазначених вище різновидів КНЛ маємо такі множини базових композицій.

$C_{RE} = \{\neg, \vee, \mathrm{R}^{\bar{v}}_{\bar{x}}, =_{xy}\}$ – для L^{RE} та L^{REP};

$C_{REs} = \{\neg, \vee, \mathrm{R}^{\bar{v}}_{\bar{x}}, \equiv_{xy}\}$ – для L^{REs} та L^{REsP};

$C_{QE} = \{\neg, \vee, \mathrm{R}^{\bar{v}}_{\bar{x}}, \exists x, =_{xy}\}$ – для L^{QE} та L^{QEP};

$C_{QEs} = \{\neg, \vee, \mathrm{R}^{\bar{v}}_{\bar{x}}, \exists x, \equiv_{xy}\}$ – для L^{QEs} та L^{QEsP};

$C_{RC} = \{\neg, \vee, \mathrm{R}^{\bar{v}}_{\bar{x}}, \sim\}$ – для L^{RC} та L^{RCP};

$C_{QC} = \{\neg, \vee, \mathrm{R}^{\bar{v}}_{\bar{x}}, \exists x, \sim\}$ – для L^{QC} та L^{QCP};

$C_{RCE} = \{\neg, \vee, \mathrm{R}^{\bar{v}}_{\bar{x}}, \sim, =_{xy}\}$ – для L^{RCE} та L^{RCEP};

$C_{RCEs} = \{\neg, \vee, \mathrm{R}^{\bar{v}}_{\bar{x}}, \sim, \equiv_{xy}\}$ – для L^{RCEs} та L^{RCEsP};

$C_{QCE} = \{\neg, \vee, \mathrm{R}^{\bar{v}}_{\bar{x}}, \exists x, \sim, =_{xy}\}$ – для L^{QCE} та L^{QCEP};

$C_{QCEs} = \{\neg, \vee, \mathrm{R}^{\bar{v}}_{\bar{x}}, \exists x, \sim, \equiv_{xy}\}$ – для L^{QCEs} та L^{QCEsP}.

Семантичною основою КНЛ є композиційні предикатні системи вигляду $({}^{V}A, Pr^{V-A}, C_{LC})$, де C_B – множина базових композицій. Така композиційна система задає дві алгебри: алгебру (алгебраїчну систему) даних $({}^{V}A, Pr^{V-A})$ та композиційну алгебру предикатів $({}^{V}A, Pr^{V-A}, C_B)$.

Композиційну систему $({}^{V}A, Pr^{V-A}, C_{QE})$ назвемо *чистою першопорядковою композиційною системою R*-предикатів зі слабкою рівністю.

Композиційну алгебру $A^{QE} = (Pr^{V-A}, C_{QE})$ назвемо *чистою першопорядковою композиційною алгеброю R*-предикатів зі слабкою рівністю.

Композиційну систему $({}^{V}A, Pr^{V-A}, C_{QEs})$ назвемо *чистою першопорядковою композиційною системою R*-предикатів зі строгою рівністю.

Композиційну алгебру $A^{QEs} = (Pr^{V-A}, C_{QEs})$ назвемо *чистою першопорядковою композиційною алгеброю R*-предикатів зі строгою рівністю.

Композиційну систему $({}^{V}A, Pr^{V-A}, C_{QC})$, назвемо *чистою першопорядковою композиційною системою R*-предикатів з предикатним доповненням.

Композиційну алгебру $A^{QC} = (Pr^{V-A}, C_{QC})$ назвемо *чистою першопорядковою композиційною алгеброю R*-предикатів з предикатним доповненням.

Композиційну систему $({}^{V}A, Pr^{V-A}, C_{QCE})$ назвемо *чистою першопорядковою композиційною системою R*-предикатів зі слабкою рівністю та предикатним доповненням.

Композиційну алгебру $A^{QCE} = (Pr^{V-A}, C_{QCE})$ назвемо *чистою першопорядковою композиційною алгеброю R*-предикатів зі слабкою рівністю та предикатним доповненням.

Композиційну систему $({}^{V}A, Pr^{V-A}, C_{QCEs})$ назвемо *чистою першопорядковою композиційною системою R*-предикатів зі строгою рівністю та предикатним доповненням.

Композиційну алгебру $A^{QCEs} = (Pr^{V-A}, C_{QCEs})$ назвемо *чистою першопорядковою композиційною алгеброю R*-предикатів зі строгою рівністю та предикатним доповненням.

Композиційні алгебри зі слабкою рівністю також називатимемо композиційними *E*-алгебрами, зі строгою рівністю – композиційними *Es*-алгебрами, з композицією предикатного доповнення – композиційними *C*-алгебрами, з композицією предикатного доповнення та слабкою рівністю – композиційними *CE*-алгебрами, з композицією предикатного доповнення та строгою рівністю – композиційними *CEs*-алгебрами.

Наприклад, A^{QCEs} – композиційна *CEs*-алгебра *R*-предикатів.

Клас *P*-предикатів замкнений щодо базових композицій C_{QE}, C_{QEs}, C_{QC}, C_{QCE}, C_{QCEs}.

Таким чином, в алгебрах A^{QE}, A^{QEs}, A^{QC}, A^{QCE}, A^{QCEs} виділено підалгебри:

– $A^{QEP} = (PrP^{V-A}, C_{QE})$ – чиста першопорядкова композиційна *E*-алгебра *P*-предикатів;

– $A^{QEsP} = (PrP^{V-A}, C_{QEs})$ – чиста першопорядкова композиційна *Es*-алгебра *P*-предикатів;

– $A^{QCP} = (PrP^{V-A}, C_{QC})$ – чиста першопорядкова композиційна *C*-алгебра *P*-предикатів;

– $A^{QCEP} = (PrP^{V-A}, C_{QCE})$ – чиста першопорядкова композиційна *CE*-алгебра *P*-предикатів;

– $A^{QCEsP} = (PrP^{V-A}, C_{QCEs})$ – чиста першопорядкова композиційна *CEs*-алгебра *P*-предикатів.

Подібним чином отримуємо композиційні предикатні системи та композиційні алгебри КНЛ на реномінативних рівнях. Маємо такі композиційні алгебри:

$A^{RE} = (Pr^{V-A}, C_{RE})$,

$A^{REP} = (PrP^{V-A}, C_{RE})$;

$A^{REs} = (Pr^{V-A}, C_{REs})$,

$A^{REsP} = (PrP^{V-A}, C_{REs})$;

$A^{RC} = (Pr^{V-A}, C_{RC})$,

$A^{RCP} = (PrP^{V-A}, C_{RC})$;

$A^{RCE} = (Pr^{V-A}, C_{RCE})$,

$A^{RCEP} = (PrP^{V-A}, C_{RCE})$;

$A^{RCEs} = (Pr^{V-A}, C_{RCEs})$,

$A^{RCEsP} = (PrP^{V-A}, C_{RCEs})$.

2. Властивості композицій

Властивості пропозиційних композицій в LE, LC та LCE аналогічні властивостям пропозиційних композицій класичної логіки, вони наведені в [6].

В LE, LC та LCE основні властивості композицій квантифікації, не пов'язані з реномінацією, в цілому аналогічні відповідним властивостям кванторів класичної логіки. Зокрема, це

– комутативність однотипних кванторів;

– закони де Моргана для кванторів;

– неістотність квантифікованих імен:

Властивості, пов’язані з композиціями реномінації та квантифікації в LE, LC та LCE, аналогічні відповідним властивостям традиційної логіки квазіарних предикатів (див. [10]). Зокрема, це властивості R, RI, RU, R¬, R∨, RR, Ren, R∃s, R∃.

Для опису в перщопорядкових логіках властивостей елімінації кванторів додатково використовують спеціальні 0-арні композиції – предикати-індикатори *Ez*, які визначають наявність у вхідних даних компоненти з відповідним іменем $z \in V$.

Предикати *Ez* задаються [5] так:

$T(Ez) = \{d \mid d(z)\downarrow\}$; $F(Ez) = \{d \mid d(z)\uparrow\}$.

Предикати-індикатори *Ez* тотальні, однозначні, немонотонні, неантитонні. Кожне $x \in V$ таке, що $x \neq z$, неістотне для *Ez*.

Твердження 2. Маємо $\equiv_{xx} = \mathrm{T}$ та $\equiv_{xy} \vee \neg\equiv_{xy} = \mathrm{T}$.

Звідси $\neg\equiv_{xx} = \mathrm{F}$ та $\equiv_{xy} \& \neg\equiv_{xy} = \mathrm{F}$.

Отже, носій кожної підалгебри алгебр A^{QEs}, A^{QEsP}, A^{REs}, A^{REsP}, A^{QCEs}, A^{QCEsP}, A^{RCEs}, A^{RCEsP} містить константні предикати T та F.

Позначимо $[\mathrm{Pr}]_{\mathrm{Cm}}$ замикання множини предикатів *Pr* щодо множини композицій *Cm*

$\{\mathrm{T}, \mathrm{F}\}_{\equiv} = [\{\mathrm{T}, \mathrm{F}\} \cup \{\equiv_{xy} \mid x, y \in V]_{CQEs}$ – це найменша множина *R*-предикатів, замкнена щодо базових композицій C_{QEs} та C_{REs}. Таким чином, в LEs можна виділити сингулярні композиційні алгебри предикатів $(\{\mathrm{T}, \mathrm{F}\}_{\equiv}, C_{QEs})$ та $(\{\mathrm{T}, \mathrm{F}\}_{\equiv}, C_{REs})$.

Подібним чином задаємо множини з константними предикатами $\{\lambda, \mathrm{T}, \mathrm{F}\}_{\equiv}$, $\{\Upsilon, \mathrm{T}, \mathrm{F}\}_{\equiv}$, $\{\lambda, \Upsilon, \mathrm{T}, \mathrm{F}\}_{\equiv}$, вони замкнені щодо базових композицій C_{QEs} та C_{REs}. Тому:

Твердження 3. 1) В першопорядкових LEs *R*-предикатів виділяємо сингулярні композиційні алгебри $(\{\mathrm{T}, \mathrm{F}\}_{\equiv}, C_{QEs})$ $(\{\lambda, \mathrm{T}, \mathrm{F}\}_{\equiv}, C_{QEs})$, $(\{\Upsilon, \mathrm{T}, \mathrm{F}\}_{\equiv}, C_{QEs})$ та $(\{\lambda, \Upsilon, \mathrm{T}, \mathrm{F}\}_{\equiv}, C_{QEs})$.

2) В першопорядкових LEs *P*-предикатів виділяємо сингулярні композиційні алгебри $(\{\mathrm{T}, \mathrm{F}\}_{\equiv}, C_{QEs})$, $(\{\lambda, \mathrm{T}, \mathrm{F}\}_{\equiv}, C_{QEs})$.

Подібні сингулярні композиційні алгебри виділяємо в реномінативних LEs.

$[\mathrm{E}]_{=} = [\{=_{xy} \mid x, y \in V\}]_{CRE}$ замкнена щодо C_{RE}; $\{\lambda\}_{=} = [\{\lambda\} \cup \{=_{xy} \mid x, y \in V\}]_{CQE}$ вже замкнена щодо C_{QE}. Аналогічно задамо замкнені щодо C_{QE} множини з констан-

тними предикатами $\{\curlyvee\}_=$, $\{\curlywedge, \curlyvee\}_=$, $\{\mathrm{T}, \mathrm{F}\}_=$, $\{\curlywedge, \mathrm{T}, \mathrm{F}\}_=$, $\{\curlyvee, \mathrm{T}, \mathrm{F}\}_=$, $\{\curlywedge, \curlyvee, \mathrm{T}, \mathrm{F}\}_=$. Звідси:

Твердження 4. 1) В першопорядкових LEw *R*-предикатів можна виділити сингулярні композиційні алгебри $(\{\curlywedge\}_=, C_{QE})$, $(\{\curlyvee\}_=, C_{QE})$, $(\{\curlywedge, \curlyvee\}_=, C_{QE})$, $(\{\mathrm{T}, \mathrm{F}\}_=, C_{QE})$, $(\{\curlywedge, \mathrm{T}, \mathrm{F}\}_=, C_{QE})$, $(\{\curlyvee, \mathrm{T}, \mathrm{F}\}_=, C_{QE})$, $(\{\curlywedge, \curlyvee, \mathrm{T}, \mathrm{F}\}_=, C_{QE})$.

2) В першопорядкових LEw *P*-предикатів залишаються сингулярні алгебри $(\{\curlywedge\}_=, C_{QE})$, $(\{\mathrm{T}, \mathrm{F}\}_=, C_{QE})$, $(\{\curlywedge, \mathrm{T}, \mathrm{F}\}_=, C_{QE})$.

Подібні сингулярні композиційні алгебри виділяємо в реномінативних LEw.

Згідно твердження 1 $\sim\mathrm{T} = \sim\mathrm{F} = \curlywedge$.

З іншого боку, маємо

Твердження 5. $\sim\curlywedge = \mathrm{T}$ та $\sim\curlyvee = \curlywedge$.

Справді, маємо $F(\sim\curlywedge) = \varnothing$,

$$T(\sim\curlywedge) = \perp(\curlywedge) = \overline{T(\curlywedge) \cup F(\curlywedge)} = \overline{\varnothing} = {}^{V}A;$$

далі маємо $F(\sim\curlyvee) = \varnothing$,

$$T(\sim\curlyvee) = \perp(\curlyvee) = \overline{T(\curlyvee) \cup F(\curlyvee)} = \varnothing.$$

Отже, носій кожної підалгебри алгебр A^{QC}, A^{QCP}, A^{RC}, A^{RCP}, A^{QCEs}, A^{QCEsP}, A^{RCEs}, A^{RCEsP}, A^{QCE}, A^{QCEP}, A^{RCE}, A^{RCEP} містить константні предикати $\curlywedge, \mathrm{T}, \mathrm{F}$; носій кожної підалгебри алгебр A^{QC}, A^{RC}, A^{QCEs}, A^{RCEs}, A^{QCE}, A^{RCE} містить константні предикати $\curlywedge, \curlyvee, \mathrm{T}, \mathrm{F}$. Звідси маємо таке.

Множини $\{\curlywedge, \mathrm{T}, \mathrm{F}\}$ та $\{\curlywedge, \curlyvee, \mathrm{T}, \mathrm{F}\}$ замкнені щодо C_{QC} та C_{RC}. Множини $\curlywedge\mathrm{TF}_{\equiv} = [\{\curlywedge, \mathrm{T}, \mathrm{F}\} \cup \{\equiv_{xy} \mid x, y \in V\}]_{CQCEs}$ та $\curlywedge\curlyvee\mathrm{TF}_{\equiv} = [\{\curlywedge, \curlyvee, \mathrm{T}, \mathrm{F}\} \cup \{\equiv_{xy} \mid x, y \in V\}]_{CQCEs}$ замкнені щодо C_{QCEs}. Подібним чином задаємо замкнені щодо C_{QCE} множини $\curlywedge\mathrm{TF}_=$ та $\curlywedge\curlyvee\mathrm{TF}_=$. Таким чином:

Твердження 6. 1) В першопорядкових LC, в першопорядкових LCEw та в першопорядкових LCEs можна відповідно виділити сингулярні композиційні алгебри $(\{\curlywedge, \mathrm{T}, \mathrm{F}\}, C_{QC})$, $(\curlywedge\mathrm{TF}_=, C_{QCE})$ та $(\curlywedge\mathrm{TF}_{\equiv}, C_{QCEs})$.

2) В першопорядкових LC *R*-предикатів, в першопорядкових LCEw та в першопорядкових LCEs *R*-предикатів також можна відповідно виділити сингулярні композиційні алгебри $(\{\curlywedge, \curlyvee, \mathrm{T}, \mathrm{F}\}, C_{QC})$, $(\curlywedge\curlyvee\mathrm{TF}_=, C_{QCE})$ та $(\curlywedge\curlyvee\mathrm{TF}_{\equiv}, C_{QCEs})$.

Аналогічні сингулярні композиційні алгебри можна виділити в реномінативних LC та LCE.

Розглянемо тепер властивості композиції $\sim$.

Теорема 1. Для кожного *R*-предиката *Q* маємо $Q \vee \neg Q \vee \sim Q = \mathrm{T}$.

Справді, для кожного $Q \in Pr^{V-A}$

$$T(Q \vee \neg Q \vee \sim Q) = T(Q) \cup T(\neg Q) \cup T(\sim Q) =$$

$$= T(Q) \cup F(Q) \cup \overline{T(Q) \cup F(Q)} = {}^{V}A;$$

$$F(Q \vee \neg Q \vee \sim Q) = F(Q) \cap F(\neg Q) \cap F(\sim Q) =$$

$$= F(Q) \cap T(Q) \cap \varnothing = \varnothing.$$

Таким чином, $Q \vee \neg Q \vee \sim Q = \mathrm{T}$.

Наслідок 1. Для кожного *R*-предиката *Q* маємо

$$\neg(Q \vee \neg Q \vee \sim Q) = \mathrm{F};$$

$$\sim(Q \vee \neg Q \vee \sim Q) = \curlywedge.$$

Твердження 7 (див. також [6]).

1) Для кожного $Q \in Pr^{V-A}$ маємо:

$$F(\neg\sim P) = T(\sim P) = \perp(P) = \overline{T(P)} \cap \overline{F(P)};$$

$$T(\neg\sim P) = F(\sim P) = \varnothing;$$

$$\perp(\neg\sim P) = \perp(\sim P) = T(P) \cup F(P).$$

2) Для кожного $Q \in Pr^{V-A}$ маємо:

$$\sim\neg Q = \sim Q; \quad \sim\sim\sim Q = \sim Q; \quad \sim\sim\sim\sim Q = \sim\sim Q.$$

3) Для кожного $Q \in PrP^{V-A}$ додатково маємо: $\sim\sim Q = Q \vee \neg Q$.

Розглянемо зв'язок між композиціями предикатного доповнення, реномінації та квантифікації. Згідно визначень маємо:

$$T(\mathrm{R}^{\bar{v}}_{\bar{x}}(\sim P)) = T(\sim(\mathrm{R}^{\bar{v}}_{\bar{x}}(P)));$$

$$F(\mathrm{R}^{\bar{v}}_{\bar{x}}(\sim P)) = F(\sim(\mathrm{R}^{\bar{v}}_{\bar{x}}(P))) = \varnothing;$$

$$\perp(\mathrm{R}^{\bar{v}}_{\bar{x}}(\tilde{}P)) = \perp(\tilde{}(\mathrm{R}^{\bar{v}}_{\bar{x}}(P))).$$

Звідси отримуємо властивість R$\tilde{}$-дистрибутивності:

$$\mathrm{R}\tilde{}) \ \mathrm{R}^{\bar{v}}_{\bar{x}}(\tilde{}P) = \tilde{}\mathrm{R}^{\bar{v}}_{\bar{x}}(P).$$

Із визначень отримуємо:

$$T(\tilde{}\exists xP) = \perp(\exists xP); \ \ F(\tilde{}\exists xP) = \varnothing;$$

$$\perp(\tilde{}\exists xP) = T(\exists xP) \cup F(\exists xP).$$

Розглянемо дію композиції $\tilde{}$ на спеціальні предикати-індикатори та предикати слабкої й строгої рівності.

Предикати $\equiv_{xy}$ та Ez є тотальними. Враховуючи твердження 1, отримуємо:

Твердження 8.

$$\tilde{}\equiv_{xy} = \text{⅄}; \ \ \tilde{}Ez = \text{⅄}.$$

Твердження 9. Для предикатів слабкої рівності маємо таке подання:

$$\tilde{}=_{xy} = \neg Ex \vee \neg Ey \vee \text{⅄} = \neg Ex \vee \neg Ey \vee \tilde{}\equiv_{xy}.$$

Справді, для кожного $d \in {}^{V}A$ маємо:

$$\tilde{}=_{xy}(d) = \begin{cases} T, \ \text{якщо} \ =_{xy}(d)\uparrow, \\ \text{невизначене, якщо} \ =_{xy}(d)\downarrow \end{cases} =$$

$$= \begin{cases} T, \ \text{якщо} \ \neg Ex(d) \vee \neg Ey(d) = T, \\ \text{невизначене, якщо} \ Ex(d) \,\&\, Ey(d) = T \end{cases} =$$

$$= \neg Ex(d) \vee \neg Ey(d) \vee \text{⅄}(d).$$

Нагадаємо (див. [9]) властивості, пов'язані з предикатами рівності.

Твердження 10. Маємо

$$\equiv_{xy} = (=_{xy} \,\&\, Ex \,\&\, Ey) \vee (\neg Ex \,\&\, \neg Ey).$$

Отже, предикати $\equiv_{xy}$ можна подати через предикати $=_{xy}$ та Ez.

Наведемо властивості реномінації предикатів рівності.

RD) Маємо

$$\mathrm{R}^{\bar{u},x,y}_{\bar{v},z,w}(=_{xy}) = =_{zw} \text{ та } \mathrm{R}^{\bar{u},x,y}_{\bar{v},z,w}(\equiv_{xy}) = \equiv_{zw};$$

за умови $y \notin \{\bar{u}\}$ маємо

$$\mathrm{R}^{\bar{u},x}_{\bar{v},z}(=_{xy}) = =_{zy} \text{ та } \mathrm{R}^{\bar{u},x}_{\bar{v},z}(\equiv_{xy}) = \equiv_{zy};$$

за умови $x, y \notin \{\bar{u}\}$ маємо

$$\mathrm{R}^{\bar{u}}_{\bar{v}}(=_{xy}) = =_{xy} \text{ та } \mathrm{R}^{\bar{u}}_{\bar{v}}(\equiv_{xy}) = \equiv_{xy}.$$

Для $=_{xy}$ та $\equiv_{xy}$ маємо властивості рефлективності, симетричності, транзитивності.

RfP) Кожний предикат вигляду $=_{xx}$ є неспростовним;

кожний предикат вигляду $\equiv_{xx}$ є тотожно істинним, тобто $\equiv_{xx} = \mathrm{T}$.

SmP) Для кожного $d \in {}^{V}A$ маємо

$=_{xy}(d) = =_{yx}(d)$ та $\equiv_{xy}(d) = \equiv_{yx}(d)$.

TrP) Для кожного $d \in {}^{V}A$ маємо:

$$=_{xy}(d) = T \text{ та } =_{yz}(d) = T \Rightarrow =_{xz}(d) = T;$$

$$\equiv_{xy}(d) = T \text{ та } \equiv_{yz}(d) = T \Rightarrow \equiv_{xz}(d) = T.$$

Звідси отримуємо:

Наслідок 2. Кожний предикат вигляду $=_{xy} \,\&\, =_{yz} \to =_{xz}$ неспростовний;

кожний предикат вигляду $\equiv_{xy} \,\&\, \equiv_{yz} \to \equiv_{xz}$ тотожно істинний, тобто $\equiv_{xy} \,\&\, \equiv_{yz} \to \equiv_{xz} = \mathrm{T}$.

Для $=_{xy}$ та $\equiv_{xy}$ маємо властивість заміни рівних.

ER) Для кожних $P \in Pr^{A}$ та $d \in {}^{V}A$ маємо:

$$=_{xy}(d) = T \Rightarrow \mathrm{R}^{\bar{u},v}_{\bar{z},x}(P)(d) = \mathrm{R}^{\bar{u},v}_{\bar{z},y}(P)(d);$$

$$\equiv_{xy}(d) = T \Rightarrow \mathrm{R}^{\bar{u},v}_{\bar{z},x}(P)(d) = \mathrm{R}^{\bar{u},v}_{\bar{z},y}(P)(d).$$

Квантифікація предикатів строгої рівності та слабкої рівності має (див. також [9]) певні особливості.

Теорема 2. 1) $\exists x \equiv_{xy} = Ey$;

2) якщо $|A| \geq 2$, то $\exists x \neg \equiv_{xy} = \mathrm{T}$; якщо $|A| = 1$, то $\exists x \neg \equiv_{xy} = \neg Ey$.

Доводимо п.1. $\exists x \equiv_{xy}(d) = T \Leftrightarrow$ існує $a \in A$ таке, що $d(y) = a \Leftrightarrow d(y)\downarrow \Leftrightarrow Ey(d) = T$.

$\exists x \equiv_{xy}(d) = F \Leftrightarrow$ для кожного $a \in A$ невірно, що $d(y) = a \Leftrightarrow d(y)\uparrow \Leftrightarrow Ey(d) = F$.

Доводимо п.2. $\exists x \neg \equiv_{xy}(d) = T \Leftrightarrow$ існує $a \in A$ таке, що невірно $d(y) = a$. Якщо $|A| \geq 2$, то це так; якщо $|A| = 1$, то це вимагає $d(y)\uparrow$, тобто $\neg Ey(d) = T$;

$\exists x \neg \equiv_{xy}(d) = F \Leftrightarrow$ для кожного $a \in A$ маємо $d(y) = a$. Якщо $|A| \geq 2$, то це не так; якщо $|A| = 1$, то це вимагає $d(y)\downarrow$, тобто $Ey(d) = T$, звідки $\neg Ey(d) = F$.

Наслідок 3. В L^{QCEs} предикати-індикатори є похідними.

Теорема 3. 1) $\exists x =_{xy} = Ey \vee \lambda$;

2) якщо $|A| \geq 2$, то $\exists x \neg =_{xy} = Ey \vee \lambda$; якщо $|A| = 1$, то $\exists x \neg =_{xy} = \neg Ey \,\&\, \lambda$.

Доводимо п.1. $\exists x (=_{xy})(d) = T \Leftrightarrow$ існує $a \in A$ таке, що $d(y) = a \Leftrightarrow d(y)\downarrow \Leftrightarrow Ey(d) = T$;

$\exists x =_{xy}(d) = F \Leftrightarrow d(y)\downarrow$ та для кожного $a \in A$ невірно, що $d(y) = a$, а це неможливо;

$\exists x =_{xy}(d)\uparrow \Leftrightarrow d(y)\uparrow$. Таким чином, $\exists x =_{xy} = Ey \vee \lambda$.

Доводимо п.2. $\exists x \neg =_{xy} = F \Leftrightarrow$ для кожного $a \in A$ маємо $d(y)\downarrow = a$. Якщо $|A| \geq 2$, то це неможливо; якщо $|A| = 1$, то це означає $d(y)\downarrow$, тобто $Ey(d) = T$, звідки $\neg Ey(d) = F$.

$\exists x \neg =_{xy}(d) = T \Leftrightarrow$ існує $a \in A$ таке, що $d(y)\downarrow$ та $d(y) \neq a$. Якщо $|A| \geq 2$, то це так за умови $d(y)\downarrow$; якщо $|A| = 1$, то це неможливо.

$\exists x \neg =_{xy}(d)\uparrow \Leftrightarrow$ для кожного $a \in A$ маємо $d(y) = a$ або $d(y)\uparrow$, причому неможливо $d(y)\downarrow = a$ для кожного $a \in A$; якщо $|A| \geq 2$, то це буде при $d(y)\uparrow$; якщо $|A| = 1$, то це буде при $d(y)\uparrow$.

Отже, $\exists x \neg =_{xy}(d)\uparrow \Leftrightarrow d(y)\uparrow$.

Таким чином:

– якщо $|A| \geq 2$, то $\exists x \neg =_{xy}(d) = T$ при $d(y)\downarrow$ та $\exists x \neg =_{xy}(d)\uparrow$ при $d(y)\uparrow$; звідси $\exists x \neg =_{xy} = Ey \vee \lambda$.

– якщо $|A| = 1$, то $\exists x \neg =_{xy}(d) = F$ при $d(y)\downarrow$ та $\exists x \neg =_{xy}(d)\uparrow$ при $d(y)\uparrow$; звідси $\exists x \neg =_{xy} = \neg Ey \,\&\, \lambda$.

3. Мови LCE

Мови LE та мови LC єокремими випадками мов LCE. Мови першопорядкових LE детально описано в [9]. Мови пропозиційних LC досліджено в [6], мови першопорядкових LC введено в [7]. Тому в даній роботі в першу чергу зосередимося на дослідженні мов чистих першопорядкових LCE.

Детально розглянемо мову L^{QCEs}.

Алфавіт мови:

– множина V предметних імен (змінних),

– множина Ps предикатних символів,

– множина $Cs_{QLCEs} = \{\neg, \vee, R_{\bar{x}}^{\bar{v}}, \exists x, \sim, =_{xy}\}$ символів базових композицій.

Дамо індуктивне визначення множини Fr формул мови:

– кожний $p \in Ps$ та кожний $\equiv_{xy}$ є формулою; такі формули назвемо атомарними;

– нехай $\Phi, \Psi \in Fr$; тоді $\neg\Phi \in Fr$, $\vee\Phi\Psi \in Fr$, $R_{\bar{x}}^{\bar{v}}\Phi \in Fr$, $\exists x\Phi \in Fr$, $\sim\Phi \in Fr$

Для L^{QCEs} задамо множину $V_T \subseteq V$ імен, неістотних для всіх $p \in Ps$ – множину тотально неістотних [5] імен

Для визначення множин гарантовано неістотних для формул імен таку ν продовжуємо до $\nu : Fr \to 2^V$ таким чином:

$$\nu(\neg\Phi) = \nu(\Phi);$$

$$\nu(\vee\Phi\Psi) = \nu(\Phi) \cap \nu(\Psi);$$

$$\nu(R_{x_1,\ldots,x_n}^{v_1,\ldots,v_n}\Phi) =$$

$$= (\nu(\Phi) \cup \{v_1,\ldots,v_n\}) \setminus \{x_i \mid v_i \notin \nu(\Phi),\ i \in \overline{1,n}\};$$

$$\nu(\exists x\Phi) = \nu(\Phi) \cup \{x\};$$

$$\nu(\sim\Phi) = \nu(\Phi);$$

$$\nu(\equiv_{xy}) = V \setminus \{x, y\}.$$

Розширена сигнатура мови L^{QCEs} – це $\Sigma = (V, V_T, Cs_{QLCEs}, Ps)$.

Якщо $x \in \nu(\Phi)$, то (див. [3]) ім'я x неістотне для формули Φ.

Для довільної $\Gamma \subseteq Fr$ вводимо такі позначення:

– $\sigma(\Gamma)$ – це множина всіх $p \in Ps$, які входять до складу $\Phi \in \Gamma$;

– $nm(\Gamma)$ – це множина всіх $x \in V$, які фігурують у формулах $\Phi \in \Gamma$;

$$– \nu(\Gamma) = \bigcap_{\Phi \in \Gamma} \nu(\Phi), fu(\Gamma) = V_T \setminus nm(\Gamma).$$

Інтерпретуємо мову L^{QCEs} на композиційних системах $CS = ({}^{V}A, Pr^{V-A}, C_{QCEs})$.

Задаємо тотальне однозначне $I : Ps \to Pr^{V-A}$, яке продовжимо до відображення інтерпретації формул $I : Fr \to Pr^{V-A}$:

$$I(\neg\Phi) = \neg(I(\Phi)),$$

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

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

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

$$I(\tilde{}\Phi) = \tilde{}(I(\Phi)),$$

$$I(\equiv_{xy}) = \equiv_{xy}.$$

Трійку $\boldsymbol{J} = (CS, \Sigma, I)$ назвемо інтерпретацією мови L^{QCEs}. Скорочено інтерпретації мови позначаємо як (A, I).

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

Предметне ім'я $x \in V$ неістотне для формули Φ, якщо при кожній інтерпретації $\boldsymbol{J}$ ім'я x неістотне для предиката Φ_J.

Мова L^{QCE} визначається аналогічно мові L^{QCEs} лише замість символів $\equiv_{xy}$ пишемо символи $=_{xy}$.

Подібним чином описуємо мови реномінативних логік L^{RCEs} та L^{RCE}, опускаючи пункти для $\exists x$.

При визначенні мов L^{QC} та L^{RC} опускаємо пункти, де фігурують символи предикатів рівності.

При визначенні мов L^{QEs}, L^{QE}, L^{REs}, L^{RE} опускаємо пункти, де фігурують символи композиції предикатного доповнення.

Виділення в LCE підалгебр P-предикатів виділяє загальний клас R-інтерпретацій та підклас P-інтерпретацій. Такі класи інтерпретацій називають семантиками, їх позначаємо $\boldsymbol{R}$ та $\boldsymbol{P}$.

Нехай α – деякий клас інтерпретацій (семантика).

Формула Φ неспростовна (частково істинна) при інтерпретації $\boldsymbol{J}$ (позн. $\boldsymbol{J} \models \Phi$), якщо предикат Φ_J – неспростовний.

Формула Φ неспростовна (частково істинна) в α, що позначаємо ${}^{\alpha}\models \Phi$, якщо $\boldsymbol{J} \models \Phi$ при кожній $\boldsymbol{J} \in \alpha$.

Формула Φ тотожно істинна при інтерпретації $\boldsymbol{J}$ (позн. $\boldsymbol{J} \models_{id} \Phi$), якщо предикат Φ_J – тотожно істинний.

Формула Φ тотожно істинна в α (позн. ${}^{\alpha}\models_{id} \Phi$), якщо $\boldsymbol{J} \models_{id} \Phi$ при кожній $\boldsymbol{J} \in \alpha$.

Якщо семантика α зафіксована, то замість ${}^{\alpha}\models$, ${}^{\alpha}\models_{id}$ пишемо $\models$, $\models_{id}$.

Формула Φ виконувана при інтерпретації $\boldsymbol{J}$, якщо предикат Φ_J – виконуваний.

Формула Φ виконувана в α, якщо Φ виконувана при деякій $\boldsymbol{J} \in \alpha$.

Твердження 11. $\boldsymbol{J} \models_{id} \Phi \;\Rightarrow\; \boldsymbol{J} \models \Phi$; $\models_{id} \Phi \Rightarrow \models \Phi$.

Приклад 2. Маємо ${}^{\alpha}\models x = x$ та ${}^{\alpha}\models x = y \leftrightarrow y = x$ (тут α – R чи P).

Приклад 3. Маємо ${}^{\alpha}\models_{id} x \equiv x$ та ${}^{\alpha}\models_{id} x \equiv y \leftrightarrow y \equiv x$ (тут α – R чи P).

Із твердження 8 отримуємо:

Твердження 12. Для кожної інтерпретації $\boldsymbol{J}$ маємо

$$(\tilde{} \equiv_{xy})_J = \text{人};$$

$$(\tilde{} Ez)_J = \text{人}.$$

Із теореми 1 отримуємо:

Твердження 13. Для кожних $\Phi \in Fr$ та інтерпретації $\boldsymbol{J}$ маємо

$$(\Phi \vee \neg\Phi \vee \tilde{}\Phi)_J = \mathrm{T};$$

$$\neg(\Phi \vee \neg\Phi \vee \tilde{}\Phi)_J = \mathrm{F};$$

$$\tilde{}(\Phi \vee \neg\Phi \vee \tilde{}\Phi)_J = \text{人}.$$

Наслідок 4. Для кожної $\Phi \in Fr$

${}^{\alpha}\models_{id} \Phi \vee \neg\Phi \vee \tilde{}\Phi$ (тут α – R чи P).

Отже, в LC та LCE в семантиках ***R*** та ***P*** множини тотожно істинних формул та неспростовних формул непорожні.

4. Відношення логічного наслідку

Для формалізації фундаментального поняття логічного наслідку в логіках квазіарних предикатів запропоновано (див. [10, 9]) низку відповідних відношень.

Спочатку задаємо відношення наслідку для двох формул при фіксованій інтерпретації ***J***.

1) Істиннісний, або *T*-наслідок $_J|=_T$: $\Phi\ _J|=_T \Psi \Leftrightarrow T(\Phi_J) \subseteq T(\Psi_J)$.

2) Хибнісний, або *F*-наслідок $_J|=_F$: $\Phi\ _J|=_F \Psi \Leftrightarrow F(\Psi_J) \subseteq F(\Phi_J)$.

3) Сильний, або *TF*-наслідок $_J|=_{TF}$: $\Phi\ _J|=_{TF} \Psi \Leftrightarrow \Phi\ _J|=_T \Psi$ та $\Phi\ _J|=_F \Psi$.

4) Неспростовнісний, або *IR*-наслідок $_J|=_{IR}$: $\Phi\ _J|=_{IR} \Psi \Leftrightarrow T(\Phi_J) \cap F(\Psi_J) = \varnothing$.

5) Дуальний до *IR*, або *DI*-наслідок $_J|=_{DI}$: $\Phi\ _J|=_{DI} \Psi \Leftrightarrow F(\Phi_J) \cup T(\Psi_J) = {}^V A$.

Відповідні відношення логічного наслідку в семантиці α задаємо за схемою: $\Phi\ ^{\alpha}|=_* \Psi$, якщо $\Phi\ _J|=_* \Psi$ для кожної $\boldsymbol{J} \in \alpha$.

В загальному випадку логік квазіарних предикатів (реляційного типу) та LE можна розглядати (див. [10, 9]) семантики ***R***, ***P***, ***T***, ***TS***; при цьому семантики ***P*** та ***T*** дуальні, семантики ***R*** та ***TS*** автодуальні.

Для зазначених відношень маємо [10, 9] такі властивості:

– в *TS*-семантиці усі ці відношення збігаються і стають єдиним відношенням $TS|=$;

– $^P|=_{DI} = {}^T|=_{IR} = {}^R|=_{IR} = {}^R|=_{DI} = \varnothing$;

– $^P|=_{IR} = {}^T|=_{DI} = {}^{TS}|=$;

– $^P|=_T = {}^T|=_F$; $^P|=_F = {}^T|=_T$;

– $^P|=_{TF} = {}^T|=_{TF}$; $^R|=_T = {}^R|=_F = {}^R|=_{TF}$.

Серед цих відношень маємо лише 5 різних:

$$^P|=_{IR},\ ^P|=_T,\ ^P|=_F,\ ^P|=_{TF},\ ^R|=_{TF}.$$

При цьому:

– $^P|=_T \not\subset {}^P|=_F$ та $^P|=_F \not\subset {}^P|=_T$;

– $^R|=_{TF} \subset {}^P|=_{TF} = {}^P|=_T \cap {}^P|=_F$;

– $^P|=_T \subset {}^P|=_{IR}$ та $^P|=_F \subset {}^P|=_{IR}$.

Відношення логічного наслідку індукують відношення логічної еквівалентності.

Відношення еквівалентності при інтерпретації ***J*** $_{J\sim T}$, $_{J\sim F}$, $_{J\sim TF}$, $_{J\sim IR}$ визначаємо за такою схемою:

$\Phi\ _{J\sim *} \Psi$, якщо $\Phi\ _J|=_* \Psi$ та $\Psi\ _J|=_* \Phi$.

Відношення логічної еквівалентності $^P\sim_{IR}$, $^P\sim_T$, $^P\sim_F$, $^P\sim_{TF}$, $^R\sim_{TF}$ визначаємо так:

$\Phi\ ^{\alpha}\sim_* \Psi$, якщо $\Phi\ ^{\alpha}|=_* \Psi$ та $\Psi\ ^{\alpha}|=_* \Phi$.

При цьому:

$\Phi\ ^{\alpha}\sim_* \Psi \Leftrightarrow \Phi\ _{J\sim *} \Psi$ для кожної $\boldsymbol{J} \in \alpha$.

Особливе значення має відношення $_{J\sim TF}$. Це випливає з наступного:

$$\Phi\ _{J\sim TF} \Psi \Leftrightarrow T(\Phi_J) = T(\Psi_J) \text{ та } F(\Phi_J) = F(\Psi_J).$$

Отже, $\Phi\ _{J\sim TF} \Psi$ означає:

Φ_J та Ψ_J – це один і той же предикат.

Основою еквівалентних перетворень формул є [10] теорема еквівалентності. Вона формулюється для відношень $^R\sim_{TF}$, $^P\sim_{TF}$, $^P\sim_{IR}$. Для $^P\sim_T$ та $^P\sim_F$ теорема невірна.

Теорема 4. Нехай Φ' отримано з формули Φ заміною деяких входжень $\Phi_1, ..., \Phi_n$ на $\Psi_1, ..., \Psi_n$. Якщо $\Phi_1\ ^{\alpha}\sim_* \Psi_1$, ..., $\Phi_n\ ^{\alpha}\sim_* \Psi_n$, то $\Phi\ ^{\alpha}\sim_* \Phi'$.

Тут $^{\alpha}\sim_*$ одне з $^R\sim_{TF}$, $^P\sim_{TF}$, $^P\sim_{IR}$.

Відношення логічного наслідку поширюються на пари множин формул.

Нехай $\Gamma, \Delta \subseteq Fr$, ***J*** – інтерпретація.

Далі позначаємо

$$\bigcap_{\Phi\in\Gamma} T(\Phi_J) \text{ як } T^{\cap}(\Gamma_J),\ \bigcap_{\Psi\in\Delta} F(\Psi_J) \text{ як } F^{\cap}\Delta_J),$$

$$\bigcup_{\Psi\in\Delta} T(\Psi_J) \text{ як } T^{\cup}(\Delta_J),\ \bigcup_{\Phi\in\Gamma} F(\Phi_J) \text{ як } F^{\cup}(\Gamma_J).$$

Δ є *IR*-наслідком Γ при ***J*** (позн. $\Gamma\ _J|=_{IR} \Delta$),

якщо $T^{\cap}(\Gamma_J) \cap F^{\cap}(\Delta_J) = \varnothing$.

Δ є *T*-наслідком Γ при ***J*** (позн. $\Gamma\ _J|=_T \Delta$),

якщо $T^{\cap}(\Gamma_J) \subseteq T^{\cup}(\Delta_J)$.

Δ є *F*-наслідком Γ при $\boldsymbol{J}$ (позн. $\Gamma_J|=_F \Delta$),

якщо $F^{\cap}(\Delta_J) \subseteq F^{\cup}(\Gamma_J)$.

Δ є *TF*-наслідком Γ при $\boldsymbol{J}$ (позн. $\Gamma_J|=_{TF} \Delta$),

якщо $\Gamma_J|=_T \Delta$ та $\Gamma_J|=_F \Delta$.

Відповідні відношення логічного наслідку визначаємо за схемою:

$\Gamma\ ^{\alpha}|=_* \Delta$, якщо $\Gamma_J|=_* \Delta$ для кожної $\boldsymbol{J}\in\boldsymbol{\alpha}$.

Аналогом теореми 4 для множин формул є теорема заміни еквівалентних.

Теорема 5. Нехай $\Phi \sim \Psi$, тоді:

$$\Phi, \Gamma |= \Delta \Leftrightarrow \Psi, \Gamma |= \Delta;$$

$$\Gamma |= \Delta, \Phi \Leftrightarrow \Gamma |= \Delta, \Psi.$$

Тут $\sim$ – одне з ${}^{R}\sim_{TF}$, ${}^{P}\sim_{TF}$, ${}^{P}\sim_{IR}$; $|=$ – одне з відношень ${}^{R}|=_{TF}$, ${}^{P}|=_{TF}$, ${}^{P}|=_{IR}$.

Нагадаємо (див. [10, 9]) основні властивості відношень логічного наслідку для множин формул. Надалі, якщо інше не зазначене, $|=$ – це одне з ${}^{R}|=_{TF}$, ${}^{P}|=_{TF}$, ${}^{P}|=_{T}$, ${}^{P}|=_{F}$, ${}^{P}|=_{IR}$.

M) Нехай $\Gamma \subseteq \Lambda$ та $\Delta \subseteq \Sigma$, тоді

$\Gamma |= \Delta \Rightarrow \Lambda |= \Sigma$ (монотонність).

Декомпозиція формул:

$\neg\neg_{\mathrm{L}}$) $\neg\neg\Phi, \Gamma |= \Delta \Leftrightarrow \Phi, \Gamma |= \Delta$.

$\neg\neg_{\mathrm{R}}$) $\Gamma |= \Delta, \neg\neg\Phi \Leftrightarrow \Gamma |= \Delta, \Phi$.

$\vee_{\mathrm{L}}$) $\Phi\vee\Psi, \Gamma |= \Delta \Leftrightarrow \Phi, \Gamma |= \Delta$ та $\Psi, \Gamma |= \Delta$.

$\vee_{\mathrm{R}}$) $\Gamma |= \Delta, \Phi\vee\Psi \Leftrightarrow \Gamma |= \Delta, \Phi, \Psi$.

$\neg\vee_{\mathrm{L}}$) $\neg(\Phi\vee\Psi), \Gamma |= \Delta \Leftrightarrow \neg\Phi, \neg\Psi, \Gamma |= \Delta$.

$\neg\vee_{\mathrm{R}}$) $\Gamma |= \Delta, \neg(\Phi\vee\Psi) \Leftrightarrow \Gamma |= \Delta, \neg\Phi$ та $\Gamma |= \Delta, \neg\Psi$.

Для ${}^{P}|=_{IR}$ також маємо (це невірно [10] для ${}^{R}|=_{TF}$, ${}^{P}|=_{TF}$, ${}^{P}|=_{T}$, ${}^{P}|=_{F}$):

$\neg_{\mathrm{L}}$) $\neg\Phi, \Gamma\ {}^{P}|=_{IR} \Delta \Leftrightarrow \Gamma\ {}^{P}|=_{IR} \Delta, \Phi$.

$\neg_{\mathrm{R}}$) $\Gamma\ {}^{P}|=_{IR} \Delta, \neg\Phi \Leftrightarrow \Phi, \Gamma\ {}^{P}|=_{IR} \Delta$.

Властивості еквівалентних перетворень, пов'язані з реномінацією, отримуються на основі властивостей R, RI, RU, RR, R$\neg$, R$\vee$ (див. [10]). Кожна з них продукує 4 відповідні властивості для відношення логічного наслідку, коли виділена формула чи її заперечення знаходиться у лівій чи правій частині цього відношення.

Для першопорядкових логік маємо [10] властивості елімінації кванторів, первісного означення та *E*-розподілу; вони справджуються також для LE та LCE.

Властивості елімінації кванторів:

$\exists_{\mathrm{L}}$) $\exists x\Phi, \Gamma |= \Delta \Leftrightarrow R_z^x(\Phi), Ez, \Gamma |= \Delta$
за умови $z \in fu(\Gamma, \Delta, \exists x\Phi)$;

$\exists \mathrm{R}_{\mathrm{L}}$) $R_{\bar{v}}^{\bar{u}}(\exists x\Phi), \Gamma |= \Delta \Leftrightarrow R_{\bar{v},z}^{\bar{u},x}(\Phi), Ez, \Gamma |= \Delta$
за умови $z \in fu(\Gamma, \Delta, R_{\bar{v}}^{\bar{u}}(\exists x\Phi))$;

$\neg\exists_{\mathrm{R}}$) $\Gamma |= \neg\exists x\Phi, \Delta \Leftrightarrow \Gamma, Ez |= \neg R_z^x(\Phi), \Delta$
за умови $z \in fu(\Gamma, \Delta, \exists x\Phi)$;

$\neg\exists \mathrm{R}_{\mathrm{R}}$) $\Gamma |= \neg R_{\bar{v}}^{\bar{u}}(\exists x\Phi), \Delta \Leftrightarrow$
$\Leftrightarrow \Gamma, Ez |= \neg R_{\bar{v},z}^{\bar{u},x}(\Phi), \Delta$
за умови $z \in fu(\Gamma, \Delta, R_{\bar{v}}^{\bar{u}}(\exists x\Phi))$;

$\exists \mathrm{v}_{\mathrm{R}}$) $\Gamma, Ey |= \exists x\Phi, \Delta \Leftrightarrow$
$\Leftrightarrow \Gamma, Ey |= \exists x\Phi, R_y^x(\Phi), \Delta$;

$\exists \mathrm{Rv}_{\mathrm{R}}$) $\Gamma, Ey |= \Delta, R_{\bar{v}}^{\bar{u}}(\exists x\Phi) \Leftrightarrow$
$\Leftrightarrow \Gamma, Ey |= \Delta, R_{\bar{v}}^{\bar{u}}(\exists x\Phi), R_{\bar{v},y}^{\bar{u},x}(\Phi)$;

$\neg\exists \mathrm{v}_{\mathrm{L}}$) $\neg\exists x\Phi, Ey, \Gamma |= \Delta \Leftrightarrow$
$\Leftrightarrow \neg\exists x\Phi, \neg R_y^x(\Phi), Ey, \Gamma |= \Delta$;

$\neg\exists \mathrm{Rv}_{\mathrm{L}}$) $\neg R_{\bar{v}}^{\bar{u}}(\exists x\Phi), Ey, \Gamma |= \Delta \Leftrightarrow$
$\Leftrightarrow \neg R_{\bar{v}}^{\bar{u}}(\exists x\Phi), \neg R_{\bar{v},y}^{\bar{u},x}(\Phi), Ey, \Gamma |= \Delta$.

Властивості *E*-розподілу та первісного означення:

Ed) $\Gamma |= \Delta \Leftrightarrow \Gamma |= \Delta, Ey$ та $Ey, \Gamma |= \Delta$;

Ev) $\Gamma |= \Delta \Leftrightarrow Ez, \Gamma |= \Delta$, де $z \in fu(\Gamma, \Delta)$.

Гарантують (див. [10]) наявність того чи іншого відношення логічного наслідку властивості *C*, *CL*, *CR*, *CLR*.

Для усіх відношень маємо:

C) $\Phi, \Gamma |= \Delta, \Phi$.

Додатково гарантують наявність відповідного відношення такі властивості:

CL) $\Phi, \neg\Phi, \Gamma\ {}^{P}|=_{T} \Delta$;

CR) $\Gamma\ {}^{P}|=_{F} \Delta, \Phi, \neg\Phi$;

CLR) $\Phi, \neg\Phi, \Gamma \,{}^{P}\!\models_{TF} \Delta, \Psi, \neg\Psi$.

Властивості *C*, *CL*, *CR*, *CLR* справджуються для LE та LCE.

Зауважимо, що відношення, які ми позначаємо одним символом (напр., ${}^{P}\!\models_{T}$), *різні* в LE, LC, LCE, але з контексту буде зрозуміло, про яке відношення йде мова.

5. Особливості відношень логічного наслідку в LE та в LCE

Розглянемо особливості відношень логічного наслідку в LE, пов'язані з наявністю предикатів рівності.

Опишемо властивості LEw, пов'язані з предикатами слабкої рівності.

На основі Sm маємо властивості:

$\mathrm{Sm_L}$) $x = y, \Gamma \models \Delta \Leftrightarrow x = y, y = x, \Gamma \models \Delta$;

$\mathrm{Sm_R}$) $\Gamma \models \Delta, \neg x = y \Leftrightarrow \Gamma \models \Delta, \neg x = y, \neg y = x$.

Tr індукує властивості:

$\mathrm{Tr_L}$) $x = y, y = z, \Gamma \models \Delta \Leftrightarrow$
$\Leftrightarrow x = y, y = z, x = z, \Gamma \models \Delta$
(тут $\models$ – це ${}^{P}\!\models_{IR}$ чи ${}^{P}\!\models_{T}$);

$\mathrm{Tr_R}$) $\Gamma \models \Delta, \neg x = y, \neg y = z \Leftrightarrow$
$\Leftrightarrow \Gamma \models \Delta, \neg x = y, \neg y = z, \neg x = z$
(тут $\models$ – це ${}^{P}\!\models_{IR}$ чи ${}^{P}\!\models_{F}$).

Справді, $T(x = y) \cap T(y = z) =$
$= T(x = y) \cap T(y = z) \cap T(x = z)$.

Теорема 6. $\mathrm{Tr_L}$ невірна для $\models_F$; $\mathrm{Tr_R}$ невірна для $\models_T$.

Справді, для $d = [x \mapsto a, z \mapsto b]$
$d \notin F(x = y) \cup F(y = z) = F(x = y \,\&\, y = z)$;
проте $d \in F(x = z)$, тому

$d \in F(x = z) \cup F(x = y) \cup F(y = z) =$
$= F(x = y \,\&\, y = z \,\&\, x = z)$.

Звідси $x = y \,\&\, y = z \,{}^{P}\!\models_{F} x = y \,\&\, y = z \,\&\, x = z$.
Проте $\Phi \,{}^{P}\!\models_{F} \Psi \Leftrightarrow \neg\Psi \,{}^{P}\!\models_{T} \neg\Phi$, тому

$\neg x = y \vee \neg y = z \vee \neg x = z \,{}^{P}\!\not\models_{T} \neg x = y \vee \neg y = z$.

Водночас $x = y \,\&\, y = z \,{}^{P}\!\models_{T} x = y \,\&\, y = z \,\&\, x = z$,

$\neg x = y \vee \neg y = z \vee \neg x = z \,{}^{P}\!\models_{F} \neg x = y \vee \neg y = z$.

Таким чином, транзитивність слабкої рівності порушується для відношень ${}^{P}\!\models_{F}$ та ${}^{P}\!\models_{T}$, тому й для ${}^{P}\!\models_{TF}$ та ${}^{R}\!\models_{TF}$. Тому в LEw ці відношення некоректні.

Наслідок 5. Для LEw коректним залишається лише відношення ${}^{P}\!\models_{IR}$.

Властивість RD індукує властивості реномінації рівності для відношення ${}^{P}\!\models_{IR}$.

$\mathrm{RD_L}$) $R^{\bar{u}}_{\bar{v}}(x = y), \Gamma \,{}^{P}\!\models_{IR} \Delta \Leftrightarrow x = y, \Gamma \,{}^{P}\!\models_{IR} \Delta$
за умови $x, y \notin \{\bar{u}\}$;

$R^{\bar{u},x}_{\bar{v},z}(x = y), \Gamma \,{}^{P}\!\models_{IR} \Delta \Leftrightarrow z = y, \Gamma \,{}^{P}\!\models_{IR} \Delta$
за умови $y \notin \{\bar{u}\}$;

$R^{\bar{u},x,y}_{\bar{v},z,u}(x = y), \Gamma \,{}^{P}\!\models_{IR} \Delta \Leftrightarrow z = u, \Gamma \,{}^{P}\!\models_{IR} \Delta$.

$\mathrm{RD_R}$) $\Gamma \,{}^{P}\!\models_{IR} R^{\bar{u}}_{\bar{v}}(x = y), \Delta \Leftrightarrow$
$\Leftrightarrow \Gamma \,{}^{P}\!\models_{IR} \Delta, x = y$ за умови $x, y \notin \{\bar{u}\}$;

$\Gamma \,{}^{P}\!\models_{IR} R^{\bar{u},x}_{\bar{v},z}(x = y), \Delta \Leftrightarrow \Gamma \,{}^{P}\!\models_{IR} \Delta, z = y$
за умови $y \notin \{\bar{u}\}$;

$\Gamma \,{}^{P}\!\models_{IR} R^{\bar{u},x,y}_{\bar{v},z,u}(x = y), \Delta \Leftrightarrow \Gamma \,{}^{P}\!\models_{IR} \Delta, z = u$.

Властивість ER індукує властивості заміни рівних для відношення ${}^{P}\!\models_{IR}$:

$\mathrm{ER_L}$) $x = y, R^{\bar{u},z}_{\bar{v},x}(\Phi), \Gamma \,{}^{P}\!\models_{IR} \Delta \Leftrightarrow$
$\Leftrightarrow x = y, R^{\bar{u},z}_{\bar{v},x}(\Phi), R^{\bar{u},z}_{\bar{v},y}(\Phi), \Gamma \,{}^{P}\!\models_{IR} \Delta$;

$\mathrm{ER_R}$) $x = y, \Gamma \,{}^{P}\!\models_{IR} R^{\bar{u},z}_{\bar{v},x}(\Phi), \Delta \Leftrightarrow$
$\Leftrightarrow x = y, \Gamma \,{}^{P}\!\models_{IR} R^{\bar{u},z}_{\bar{v},x}(\Phi), R^{\bar{u},z}_{\bar{v},y}(\Phi), \Delta$.

Властивість Rf індукує спеціальну умову наявності відношення ${}^{P}\!\models_{IR}$:

C_{Rf}) $\Gamma \,{}^{P}\!\models_{IR} x = x, \Delta$.

Опишемо властивості LEs, пов'язані з предикатами строгої рівності. Далі $\models$ – це одне з ${}^{R}\!\models_{TF}$, ${}^{P}\!\models_{TF}$, ${}^{P}\!\models_{T}$, ${}^{P}\!\models_{F}$, ${}^{P}\!\models_{IR}$.

Для відношень типу *T*, *F*, *TF* знімати заперечення, переносячи $\neg\Phi$ з лівої частини відношення у праву як Φ і навпаки, взагалі кажучи, не можна. Проте це можна робити для предикатів строгої рівності:

$\mathrm{E_{LR}}$) $\Gamma \models \neg x \equiv y, \Delta \Leftrightarrow x \equiv y, \Gamma \models \Delta$ та
$\neg x \equiv y, \Gamma \models \Delta \Leftrightarrow \Gamma \models x \equiv y, \Delta$;

$\mathrm{ER_{LR}}$) $\Gamma \models \neg R^{\bar{u}}_{\bar{v}}(x \equiv y), \Delta \Leftrightarrow$

$\Leftrightarrow R^{\bar{u}}_{\bar{v}}(x \equiv y), \Gamma \models \Delta$ та

$\neg R^{\bar{u}}_{\bar{v}}(x \equiv y), \Gamma \models \Delta \Leftrightarrow \Gamma \models R^{\bar{u}}_{\bar{v}}(x \equiv y), \Delta$.

Властивості, індуковані Sm та Tr:

SmS) $x \equiv y, \Gamma \models \Delta \Leftrightarrow x \equiv y, y \equiv x, \Gamma \models \Delta$;

TrS) $x = y, y = z, \Gamma \models \Delta \Leftrightarrow$
$\Leftrightarrow x = y, y = z, x = z, \Gamma \models \Delta$.

Властивість RD індукує властивості реномінації рівності.

$\text{RDS}_\text{L})$ $R^{\bar{u}}_{\bar{v}}(x \equiv y), \Gamma \models \Delta \Leftrightarrow$
$\Leftrightarrow x \equiv y, \Gamma \models \Delta$ за умови $x, y \notin \{\bar{u}\}$;

$R^{\bar{u},x}_{\bar{v},z}(x \equiv y), \Gamma \models \Delta \Leftrightarrow z \equiv y, \Gamma \models \Delta$
за умови $y \notin \{\bar{u}\}$;

$R^{\bar{u},x,y}_{\bar{v},z,u}(x \equiv y), \Gamma \models \Delta \Leftrightarrow z \equiv u, \Gamma \models \Delta$;

$\text{RDS}_\text{R})$ $\Gamma \models R^{\bar{u}}_{\bar{v}}(x \equiv y), \Delta \Leftrightarrow \Gamma \models x \equiv y, \Delta$
за умови $x, y \notin \{\bar{u}\}$;

$\Gamma \models R^{\bar{u},x}_{\bar{v},z}(x \equiv y), \Delta \Leftrightarrow \Gamma \models z \equiv y, \Delta$
за умови $y \notin \{\bar{u}\}$;

$\Gamma \models R^{\bar{u},x,y}_{\bar{v},z,u}(x \equiv y), \Delta \Leftrightarrow \Gamma \models z \equiv u, \Delta$.

Властивість ER індукує такі властивості заміни рівних:

$\text{ERS}_\text{L})$ $\Gamma, x \equiv y, R^{\bar{u},z}_{\bar{v},x}(\Phi) \models \Delta \Leftrightarrow$
$\Leftrightarrow \Gamma, x \equiv y, R^{\bar{u},z}_{\bar{v},x}(\Phi), R^{\bar{u},z}_{\bar{v},y}(\Phi) \models \Delta$;

$\text{ERS}_\text{R})$ $\Gamma, x \equiv y \models R^{\bar{u},z}_{\bar{v},x}(\Phi), \Delta \Leftrightarrow$
$\Leftrightarrow \Gamma, x \equiv y \models R^{\bar{u},z}_{\bar{v},x}(\Phi), R^{\bar{u},z}_{\bar{v},y}(\Phi), \Delta$.

Властивість Rf індукує спеціальну умову наявності кожного з відношень ${}^{R}\models_{TF}$, ${}^{P}\models_{TF}$, ${}^{P}\models_{T}$, ${}^{P}\models_{F}$, ${}^{P}\models_{IR}$:

$C_\text{RfS})$ $\Gamma \models x \equiv x, \Delta$.

Підсумовуючи, в LEw маємо такі властивості відношення ${}^{P}\models_{IR}$:

– декомпозиції $\neg_\text{L}$, $\neg_\text{R}$, $\vee_\text{L}$, $\vee_\text{R}$;

– еквівалентних перетворень на основі R, RI, RU, RR, R$\neg$, R$\vee$;

– елімінації кванторів $\exists_\text{L}$, $\exists\text{R}_\text{L}$, $\exists\text{v}_\text{R}$, $\exists\text{Rv}_\text{R}$;

– E-розподілу Ed та первісного означення Ev;

– пов'язані з предикатами слабкої рівності Sm_L, Tr_L, RD_L, RD_R, ER_L, ER_R;

– властивості C та C_Rf гарантованої наявності відношення ${}^{P}\models_{IR}$.

Подібні властивості відношення ${}^{P}\models_{IR}$ маємо в LEs; відмінність тільки в заміні символів $=_{xy}$ на символи $\equiv_{xy}$, що дає такі властивості:

– пов'язані з предикатами строгої рівності SmS, TrS, RDS_L, RDS_R, ERS_L, ERS_R;

– властивості C та C_RfS гарантованої наявності відношення ${}^{P}\models_{IR}$.

Властивості відношень ${}^{R}\models_{TF}$, ${}^{P}\models_{TF}$, ${}^{P}\models_{T}$, ${}^{P}\models_{F}$ в LEs:

– декомпозиції формул $\neg\neg_\text{L}$, $\neg\neg_\text{R}$, $\vee_\text{L}$, $\vee_\text{R}$, $\neg\vee_\text{L}$, $\neg\vee_\text{R}$;

– еквівалентних перетворень на основі R, RI, RU, RR, R$\neg$, R$\vee$;

– елімінації кванторів $\exists_\text{L}$, $\exists\text{R}_\text{L}$, $\neg\exists_\text{R}$, $\neg\exists\text{R}_\text{R}$, $\exists\text{v}_\text{R}$, $\exists\text{Rv}_\text{R}$, $\neg\exists\text{v}_\text{L}$, $\neg\exists\text{Rv}_\text{L}$;

– E-розподілу і первісного означення;

– пов'язані з предикатами строгої рівності властивості E_LR, ER_LR, SmS, TrS, RDS_L, RDS_R, ERS_L, ERS_R;

– властивості C, CL, CR, CLR, C_RfS гарантованої наявності відношення логічного наслідку.

Детальніше щодо виконання цих властивостей щодо різних відношень:

– C, CL, C_RfS для ${}^{P}\models_{T}$;

– C, CR, C_RfS для ${}^{P}\models_{F}$;

– C, CLR, C_RfS для ${}^{P}\models_{TF}$;

– C, C_RfS для ${}^{R}\models_{TF}$.

В LC та в LCE відношення логічного наслідку вводимо так, як описано вище. Водночас композиція предикатного доповнення має специфічні особливості, що відображається на властивостях відношень логічного наслідку.

В LC, тому і в LCE, змістовними є лише *R*-семантика та *P*-семантика, тому малозмістовними є відношення типу *DI*.

Для відношення $^{R}|=_{IR}$ рефлексивність порушується як в традиційних логіках квазіарних предикатів, так і в LC, LE, LCE. Справді, для довільного $p \in Ps$ маємо $p\ ^{Rc}|\neq_{IR} p$ (беремо $\boldsymbol{J} \in \boldsymbol{R}$ таку, що $p_J = \Upsilon$). Тому вироджене відношення $^{R}|=_{IR}$; далі не розглядаємо.

Для LEw відношення типів *T, F, TF* для пар формул нетранзитивні (теорема 6), нетранзитивними вони залишаться і для LCEw. Тому для LCEw коректним може бути лише відношення $^{P}|=_{IR}$.

Відношення логічного наслідку в LC детально досліджено в [8]. Зокрема, показано, що відношення $^{P}|=_{IR}$ в LC коректно ввести неможливо. Причиною цього є неможливість коректно задати умови декомпозиції формул вигляду $\sim\Phi$. Зумовлено це тим, що декомпозиція формули $\sim\Phi$ вимагає подання області невизначеності формули Φ через її області істинності та хибності за допомогою лише $\cap$ та $\cup$. Як показано в [8], це неможливо.

Таким чином, для LC відношення $^{P}|=_{IR}$ неадекватне. Це мотивує перехід від відношення $^{P}|=_{IR}$ до загальнішого відношення $|=_{IR}^{\perp}$ неспростовнісного логічного наслідку за умов невизначеності. Відношення $|=_{IR}^{\perp}$ в LC досліджено в [6, 7].

Неадекватність відношення $^{P}|=_{IR}$ в LC робить його і неадекватним і в LCE.

Отже, $^{P}|=_{IR}$ буде неадекватним в LCEw та LCEs. Враховуючи, що в LCEw відношення типів *T, F, TF* теж неадекватні, LCEw до певної міри вироджена, для неї може працювати лише відношення $|=_{IR}^{\perp}$.

Стисло опишемо відношення типів *T* та *F* в LC та LCEs. Це відношення $^{P}|=_{T}$ і $^{R}|=_{T}$, які також будемо позначати $|=_{T}$, та відношення $^{P}|=_{F}$ і $^{R}|=_{F}$, які також позначатимемо $|=_{F}$.

Для цих відношень в LC та LCEs справджуються відповідні властивості декомпозиції формул, еквівалентних перетворень, елімінації кванторів, *E*-розподілу та первісного означення, які мають місце для традиційних логік квазіарних предикатів (див. [10]). Справджуються також наведені в [10] умови, які гарантують наявність того чи іншого відношення логічного наслідку. Водночас в LC з'являються (див. [8]) нові властивості, пов'язані з композицією предикатного доповнення.

Для відношень $|=_{T}$ додатково маємо наступні властивості.

Умова гарантованої наявності $|=_{T}$:

$C\neg\sim)\ \ \neg\sim\Phi, \Gamma\ |=_{T} \Delta.$

Декомпозиція формул типу $\sim\Phi$:

$\sim_{\mathrm{LT}})\ \sim\Phi, \Gamma\ |=_{T} \Delta \ \Leftrightarrow\ \Gamma\ |=_{T} \Delta, \Phi, \neg\Phi.$

$\sim_{\mathrm{RT}})\ \Gamma\ |=_{T} \Delta, \sim\Phi \ \Leftrightarrow$

$\Leftrightarrow \Phi, \Gamma\ |=_{T} \Delta$ та $\neg\Phi, \Gamma\ |=_{T} \Delta.$

$\neg\sim_{\mathrm{El}})\ \Gamma\ |=_{T} \Delta, \neg\sim\Phi \ \Leftrightarrow\ \Gamma\ |=_{T} \Delta.$

Властивість $\neg\sim_{\mathrm{El}}$ – це фактично елімінація $\sim$.

Для відношень $|=_{F}$ додатково маємо наступні властивості.

Умова гарантованої наявності $|=_{F}$:

$C\sim)\ \ \Gamma\ |=_{F} \sim\Phi, \Delta.$

Декомпозиція формул типу $\sim\Phi$:

$\neg\sim_{\mathrm{RF}})\ \Gamma\ |=_{F} \Delta, \neg\sim\Phi \ \Leftrightarrow\ \Gamma, \Phi, \neg\Phi\ |=_{F} \Delta.$

$\neg\sim_{\mathrm{LF}})\ \neg\sim\Phi, \Gamma\ |=_{F} \Delta \ \Leftrightarrow$

$\Leftrightarrow \Gamma\ |=_{F} \Delta, \Phi$ та $\Gamma\ |=_{F} \Delta, \neg\Phi.$

$\sim_{\mathrm{El}})\ \sim\Phi, \Gamma\ |=_{F} \Delta \ \Leftrightarrow\ \Gamma\ |=_{F} \Delta.$

Властивість $\sim_{\mathrm{El}}$ – це фактично елімінація $\sim$.

Як випливає з цих співвідношень, для відношень типу $|=_{T}$ та типу $|=_{F}$ маємо *різні* властивості декомпозиції формул вигляду $\sim\Phi$, їх *не можна* подати як спільну властивість для відношень типу $|=_{TF}$.

Це означає, що в LC відношення $^{R}|=_{T}$, $^{R}|=_{F}$, $^{R}|=_{TF}$ *різні*, водночас в традиційній логіці квазіарних предикатів маємо $^{R}|=_{T} = {}^{R}|=_{F} = {}^{R}|=_{TF}$.

Властивості декомпозиції формул вигляду $\sim\Phi$ для відношень типу $|=_{TF}$ отримуємо, поєднуючи наведені вище відповідні властивості для $|=_{T}$ та $|=_{F}$: $\sim_{\mathrm{LT}}$ та $\sim_{\mathrm{El}}$,

$\sim_{RT}$ та $C\sim$, $C\neg\sim$ та $\neg\sim_{LF}$, $\neg\sim_{El}$ та $\neg\sim_{RF}$:

$\sim_{LTF}$) $\sim\Phi, \Gamma \models_{TF} \Delta \Leftrightarrow$
$\Leftrightarrow \Gamma \models_T \Delta, \Phi, \neg\Phi$ та $\Gamma \models_F \Delta$;

$\sim_{RTF}$) $\Gamma \models_{TF} \Delta, \sim\Phi \Leftrightarrow$
$\Leftrightarrow \Phi, \Gamma \models_T \Delta$ та $\neg\Phi, \Gamma \models_T \Delta$;

$\neg\sim_{LTF}$) $\neg\sim\Phi, \Gamma \models_{TF} \Delta \Leftrightarrow$
$\Leftrightarrow \Gamma \models_F \Delta, \Phi$ та $\Gamma \models_F \Delta, \neg\Phi$;

$\neg\sim_{RTF}$) $\Gamma \models_{TF} \Delta, \neg\sim\Phi \Leftrightarrow$
$\Leftrightarrow \Gamma \models_T \Delta$ та $\Gamma, \Phi, \neg\Phi \models_F \Delta$.

Аналогічні властивості відношень типів $\models_T$ та $\models_F$, пов'язані з композицією предикатного доповнення, маємо в LCEs.

Звідси, зокрема, випливає, що в в LCEs усі відношення ${}^P\models_T$, ${}^P\models_F$, ${}^P\models_{TF}$, ${}^R\models_T$, ${}^R\models_F$, ${}^R\models_{TF}$ є різними.

В LCEs також виконуються наведені вище властивості LEs, пов'язані з предикатами строгої рівності. Новими є відношення, які пов'язують $\sim$ та предикати строгої рівності.

Згідно твердження 12 для кожної інтерпретації $\boldsymbol{J}$ маємо

$$T((\sim x \equiv y)_J) = F((\sim x \equiv y)_J) = \varnothing.$$

Звідси умови гарантованої наявності $\models_T$ та гарантованої наявності $\models_F$:

$C_{\sim\equiv L}$) $\sim x \equiv y, \Gamma \models_T \Delta$;

$C_{\sim\equiv R}$) $\Gamma \models_F \Delta, \sim x \equiv y$.

При взаємній заміні $\models_T$ та $\models_F$ маємо властивості спрощення – елімінації $\sim\equiv_{xy}$:

$El_{\sim\equiv L}$) $\sim x \equiv y, \Gamma \models_F \Delta \Leftrightarrow \Gamma \models_F \Delta$;

$El_{\sim\equiv R}$) $\Gamma \models_T \Delta, \sim x \equiv y \Leftrightarrow \Gamma \models_T \Delta$.

Підсумовуючи, в LCEs маємо такі властивості відношень ${}^R\models_T$, ${}^R\models_F$, ${}^P\models_T$, ${}^P\models_F$.

Спільні для ${}^R\models_T$, ${}^R\models_F$, ${}^P\models_T$, ${}^P\models_F$ властивості:

– декомпозиції формул $\neg\neg_L$, $\neg\neg_R$, $\vee_L$, $\vee_R$, $\neg\vee_L$, $\neg\vee_R$;

– еквівалентних перетворень на основі R, RI, RU, RR, R$\neg$, R$\vee$, R$\sim$;

– елімінації кванторів $\exists_L$, $\exists R_L$, $\neg\exists_R$, $\neg\exists R_R$, $\exists v_R$, $\exists Rv_R$, $\neg\exists v_L$, $\neg\exists Rv_L$;

– пов'язані з предикатами строгої рівності властивості E_{LR}, ER_{LR}, SmS, TrS, RDS_L, RDS_R, ERS_L, ERS_R.

Для відношень $\models_T$ додатково маємо:

– властивості $\sim_{LT}$, $\sim_{RT}$, $\neg\sim_{El}$ декомпозиції формул типу $\sim\Phi$;

– властивість $El_{\sim\equiv R}$ елімінації $\sim\equiv_{xy}$.

Властивості гарантованої наявності відношення ${}^P\models_T$: C, CL, C_{RfS}, $C\neg\sim$, $C_{\sim\equiv L}$.

Властивості гарантованої наявності відношення ${}^R\models_T$: C, C_{RfS}, $C\neg\sim$, $C_{\sim\equiv L}$.

Для відношень $\models_F$ додатково маємо:

– властивості $\neg\sim_{RF}$, $\neg\sim_{LF}$, $\sim_{El}$ декомпозиції формул типу $\sim\Phi$;

– властивість $El_{\sim\equiv L}$ елімінації $\sim\equiv_{xy}$.

Властивості гарантованої наявності відношення ${}^P\models_F$: C, CR, C_{RfS}, $C\sim$, $C_{\sim\equiv R}$.

Властивості гарантованої наявності відношення ${}^R\models_F$: C, C_{RfS}, $C\sim$, $C_{\sim\equiv R}$.

Такі ж властивості відношень ${}^R\models_T$, ${}^R\models_F$, ${}^P\models_T$, ${}^P\models_F$ справджуються в LC, але при цьому ми опускаємо властивості, в яких фігурують символи предикатів строгої рівності.

Наявність в LC та в LCE композиції предикатного доповнення дає змогу визначити відношення логічного наслідку за умови невизначеності. Для LC такі відношення вивчались в [6–8]. Для LCE дослідження відношень логічного наслідку за умови невизначеності буде зроблено в наступних статтях.

Висновки

В роботі досліджено нові програмно-орієнтовані логічні формалізми – композиційно-номінативні логіки з предикатами рівності та композицією предикатного доповнення, такі логіки названо LCE. Можна виділити предикати слабкої рівності та строгої рівності, це дає LCE з предикатами слабкої рівності, які названо LCEw,

та з предикатами строгої рівності, які названо LCEs. LCE можна розглядати на першопорядковому рівні та реномінативному рівні. Розглянуто композиційні алгебри LCE, досліджено властивості їх композицій, описано першопорядкові мови цих логік. Основну увагу зосереджено на вивченні властивостей, пов'язаних з предикатами рівності та композицією предикатного доповнення. Введено та досліджено низку відношень логічного наслідку в першопорядкових LCE, розглянуто їх особливості. Зокрема, встановлено певну виродженість LCEw, для якої коректним є лише відношення неспростовнісного логічного наслідку за умов невизначеності. Детальне дослідження відношень логічного наслідку за умови невизначеності в LCE буде зроблено в наступних роботах.

Властивості відношень логічного наслідку є семантичною основою побудови в LCE відповідних першопорядкових числень секвенційного типу. Таку побудову теж планується здійснити робити в наступних роботах.

Література

1. Handbook of Logic in Computer Science / Edited by S. Abramsky, Dov M. Gabbay and T. S. E. Maibaum. – Oxford University Press, Vol. 1–5. 1993–2000.
2. Hoare C. An axiomatic basis for computer programming, Comm. 1969. ACM. 12(10). P. 576–580.
3. Apt K. Ten years of Hoare's logic: a survey – part I, ACM Trans. Program. Lang. Syst. 3(4) (1981). P. 431–483.
4. Ivanov I., Nikitchenko M. On the sequence rule for the Floyd-Hoare logic with partial pre- and post-conditions. In Proceedings of the 14th International Conference on ICT. 2018. Vol. 2104. of CEUR Workshop Proc. P. 716–724.
5. Ivanov I., Nikitchenko M. Inference Rules for the Partial Floyd-Hoare Logic Based on Composition of Predicate Complement, Comm. in Computer and Information Science, 2019. Vol. 1007. Springer, Cham, P. 71–88.
6. Нікітченко М.С., Шкільняк О.С., Шкільняк С.С., Мамедов Т.А. Пропозиційні логіки часткових предикатів з композицією предикатного доповнення. *Проблеми програмування.* 2019. № 1. С. 3–13.
7. Нікітченко М.С., Шкільняк О.С., Шкільняк С.С. Чисті першопорядкові логіки квазіарних предикатів з композицією предикатного доповнення. Dynamical systems modelling and stability investigation: international conference: proceeding. К., 2019. С. 371–373.
8. Шкільняк О.С. Відношення логічного наслідку в логіках часткових предикатів з композицією предикатного доповнення. *Проблеми програмування.* 2019. № 3. С. 11–27.
9. Нікітченко М.С., Шкільняк С.С. Чисті першопорядкові квазіарні логіки з предикатами рівності. *Проблеми програмування.* 2017. № 2. С. 3–23.
10. Нікітченко М.С., Шкільняк О.С., Шкільняк С.С. Чисті першопорядкові логіки квазіарних предикатів. *Проблеми програмування*. 2016. № 2–3. С. 73–86.

References

1. ABRAMSKY, S., GABBAY, D. and MAIBAUM, T. (editors). (1993–2000). *Handbook of Logic in Computer Science* Oxford University Press, Vol. 1–5.
2. HOARE, C. (1969). An axiomatic basis for computer programming, Comm. ACM 12(10), 1969. P. 576–580,
3. APT, K. (1981). Ten years of Hoare's logic: a survey – part I, ACM Trans. Program. Lang. Syst. 3(4). P. 431–483.
4. IVANOV, I. and NIKITCHENKO, M. (2018). On the sequence rule for the Floyd-Hoare logic with partial pre- and post-conditions. *In Proceedings of the 14th International Conference on ICT.* Vol 2104 of CEUR Workshop Proc. P. 716–724.
5. IVANOV, I. and NIKITCHENKO, M. (2019). Inference Rules for the Partial Floyd-Hoare Logic Based on Composition of Predicate Complement. *In Communication. in Computer and Information Science*. Vol. 1007. Springer, Cham. P. 71–88.
6. NIKITCHENKO, M., SHKILNIAK, O., SHKILNIAK, S. and MAMEDOV, T. (2019). Propositional logics of partial predicates with composition of predicate complement. In *Problems in Progamming.* N 1. P. 3–13 (in ukr).

7. NIKITCHENKO, M., SHKILNIAK, O. and SHKILNIAK, S. (2019). Pure first-order logics of quasiary predicates with the composition of predicate complement. In Proceedings of the XIX^{h} International Conference "Dynamical systems modelling and stability investigation". P. 371–373 (in ukr).
8. SHKILNIAK, O. (2019). Relations of logical consequence in logics of partial predicates with composition of predicate complement. In *Problems in Progamming*. N 3 (in ukr).
9. NIKITCHENKO, M. and SHKILNIAK, S. (2017). Pure first-order quasiary logics with equality predicates. In *Problems in Progamming*. N 2. P. 3–23 (in ukr).
10. NIKITCHENKO, M., SHKILNIAK, O. and SHKILNIAK, S. (2016). Pure first-order logics of quasiary predicates. In *Problems in Progamming*. N 2–3. P. 73–86 (in ukr).

Одержано 10.06.2019

Про автора:

Шкільняк Степан Степанович,
доктор фізико-математичних наук,
професор, професор кафедри Теорії
та технології програмування.
Кількість наукових публікацій в
українських виданнях – понад 240,
у тому числі у фахових виданнях –
понад 100.
Кількість наукових публікацій в
зарубіжних виданнях – 22.
Scopus Author ID: 36646762300
h-індекс (Google Scholar): 7 (5 з 2014).
http://orcid.org/0000-0001-8624-5778.

***Місце роботи автора*:**

Київський національний університет
імені Тараса Шевченка,
01601, Київ, вул. Володимирська, 60.
Тел.: (044) 259 05 19.

УДК 502:004.45 (075.8) https://doi.org/10.15407/pp2019.03.045

Н.А. Сидоров

ОСНОВЫ ПРОГРАММИРОВАНИЯ В КОНТЕКСТЕ ИНЖЕНЕРИИ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

Рассматривается применение конструктивного подхода к построению текстов программ, который систематически культивируется в инженерии программного обеспечения и стало возможным благодаря ряду фундаментальных результатов, полученных в теории программирования. Во-первых, на основе известной структурной теоремы, аргументировано отказались от использования оператора *go to* и предложили метод структурного программирования, что обеспечило реальный путь к созданию понятных программ. Во-вторых, понятие подпрограммы, хотя и использовалось только для уменьшения рутинной работы в процессе программирования, стало первым средством модульного представления программ. Позднее блок и подпрограмма составили основу блок-ориентированных (процедурных, подпрограммных) языков и метода процедурного (подпрограммного) программирования. В-третьих, для ответа на вопросы, относящихся к определению границ, размеров и устройства модуля ввели понятия связывания частей, составляющих модуль и соединения между модулями; создали конкретные критерии модуляризации; предложили устройство модуля на основе понятия сокрытия информации. Модуль был реализован в языке программирования *Modula*, а позже *Modula-2*, в которых использовался на основе метода модульного (композиционного) программирования. При разработке языка *Simula 67*, были заложены основы объектно-ориентированных языков, которые получили развитие благодаря работам по концепциям наследования, позднего связывания и ссылкам и основы были завершены разработкой объектно-ориентированных языков и методом объектно-ориентированного (классификационного) программирования. Таким образом, была создана база для повторного, многократного использования и компонентной разработки программного обеспечения. Сейчас эти работы развиваются в направлении исследования и создания программного обеспечения как системы систем (system of systems), используя связь системного анализа и инженерии программного обеспечения, и развивая системную инженерию программного обеспечения. В статье, для обучения основам программирования, как средство, которое позволяет уточнить понятие программной конструкции, использовано классификацию, а как классификационный признак - уровень инкапсуляции, который строится на основе принципов инженерии программного обеспечения - инкапсуляции и многоуровневого представления, Применяя принцип инкапсуляции на разных уровнях представления структуры программы, соответствующие различным степеням абстракции программного обеспечения, получено понятие уровня инкапсуляции. Воспользовавшись этим понятием, можно выяснить типы программных конструкций и соответствующие методы программирования (конструирования) программ. На основе конструктивного подхода, и введенных понятий к построению программы, автором создана дидактика основ программирования, которая внедрена путем использования в лекциях для студентов специальности «Инженерия программного обеспечения» (121) и при написании автором учебного пособия для студентов и аспирантов, указанной специальности по основам программирования.

Ключевые слова: обучение, дидактика, основы программирования, языки программирования, инженерия программного обеспечения.

Введение

Функционирование своей машины Ч. Бэббидж основал на принципе программного управления, и вследствие этого появилась необходимость писать программы, процесс – программирование и первые программисты. Впоследствии, с появлением электронных вычислительных машин и их широким распространением, этот процесс стал массовым, а, принимая во внимание его сложность и стоимость, стоимость труда писателей и исполнителей программ еще и дорогим. В первое время к созданию программного обеспечения относились как к созданию «железа», но к 60-ым годам пришло понимание, что программное обеспечение принципиально отличается от «железа». Во-первых, программное обеспечение легко модифицировалось и копировалось. Во-вторых, программное обеспечение не изнашивалось, а его сопровождение отличалось от сопровождения «железа». Программное обеспе-

чение было невидимо, не имело веса, но при этом было очень дорогим. Разработка программного обеспечения была практически неуправляемой и требовала огромного количества спецификаций (в 50 раз больше чем для «железа») [1]. В-третьих, с распространением машин в программировании появилась кадровая проблема – нехватка программистов, которую решали путем тренинга гуманитариев, филологов, социологов и специалистов из подобных отраслей. Изменение отношения отразилось и на программировании, которое стали квалифицировать как процесс относящийся к искусству. Для решения проблем была предложена инженерия программного обеспечения (*Software engineering)* – приложение систематического, дисциплинированного, измеримого подхода к разработке, функционированию и сопровождению программного обеспечения, а также исследованию этого подхода; приложение дисциплины инженерии к программному обеспечению (*ISO/IEC/IEEE 24765-2010*) [2]. Поскольку основным процессом в инженерии программного обеспечения попрежнему оставалось программирование, то первые результаты были получены в этой области. Во-первых, на основе известной структурной теоремы, аргументировано отказались от использования оператора *go to* и предложили метод структурного программирования, что обеспечило реальный путь к созданию понятных программ. Во-вторых, понятие подпрограммы, хотя и использовалось только для уменьшения рутинной работы в процессе программирования, стало первым средством модульного представления программ. Позднее блок и подпрограмма составили основу блок-ориентированных (процедурных, подпрограммных) языков и метода процедурного (подпрограммного) программирования. В-третьих, для ответа на вопросы, относящиеся к определению границ, размеров и устройства модуля ввели понятия связывания (c*ohesion)* частей, составляющих модуль и соединения (*coupling)* между модулями [3]. Затем ввели конкретные критерии модуляризации и предложили устройство модуля на основе понятия сокрытия информации. В 1980 году реализовали язык программирования *Modula*, а позже *Modula-2*, в которых использовалось понятие модуля и модульного (композиционного) программирования [4]. В 1984 году был создан язык программирования *Ada*, в котором такое же понятие реализовано в форме пакета [5]. В 1967 году, используя блок и сокрытие информации при разработке языка *Simula 67*, были заложены основы объектно-ориентированных языков [6]. Эти основы получили развитие благодаря работам по концепциям наследования, позднего связывания и ссылкам [7]. Завершены эти работы были разработкой объектно-ориентированных языков и методом объектно-ориентированного (классификационного) программирования [8]. Таким образом, в основном были завершены работы в области модуляризации, как для композиционных, так и классификационных языков и создана основа для повторного, многократного использования и компонентной разработки программного обеспечения. Сейчас эти работы развиваются в направлении исследования и создания программного обеспечения как системы систем (system of systems), используя связь системного анализа и инженерии программного обеспечения, и развивая системную инженерию программного обеспечения (system software engineering) [9].

Конструктивный подход к устройству программы в контексте инженерии программного обеспечения

В контексте инженерии программного обеспечения, рассматривая программу метафорически с точки зрения теории машин, будем иметь в виду три точки зрения на машину, а именно технологическую, кинематическую и конструктивную [10].

В статье будет применяется последняя точка зрения, конструктивная. Конструктивный подход к рассмотрению программы, систематически культивируется в инженерии программного обеспечения и стал возможным благодаря ряду фундаментальных результатов, полученных в теории программирования, о которых

кратко говорится во введении статьи. Конструктивная теория машин рассматривает их с точки зрения форм и частей целого, статически, разделяя машину на отдельные части и подчеркивая ее конструкцию. В XVIII веке утверждалось, что «многообразные механизмы движения, которыми пользуются для устройства рабочих машин, не должны заново изобретаться каждый раз. Это было необходимо, когда были изобретены паровые и прядильные машины, так как тогда были известны лишь немногие механизмы для преобразования движений. Теперь же известно очень много разнообразных механизмов и всегда можно отыскать такой, который подходит для частного случая. Таким образом, лишь для совершенно необычных условий движения действительно необходимы новые изобретения, и очень ясное и полное знание изобретенных до настоящего времени передаточных механизмов, служащих для устройства рабочих машин, является необычайно важным» [11]. В работе [12], B. Boehm делает аналогичное предположение относительно программного обеспечения, аргументируя повторное использование (reuse и systematic reuse) программного обеспечения как наиболее перспективный подход для повышения продуктивности создания и сопровождения программного обеспечения на ближайшие десятилетия. С годами, это предположение подтвердилось. Очевидно, что продуктивность этого подхода зависит от реализации конструктивной точки зрения на программы. При этом, такой взгляд имеет важное значение не только при проектировании, но и при программировании. К сожалению, компонентный (конструктивный) подход дается студентам поздно, только в дисциплинах проектирования программного обеспечения и совсем не применяется в раннем обучении основам программирования. Студенты, в контексте обучения по специальности инженерия программного обеспечения должны не только учиться писать работающие программы, но программы, которые отвечают ряду дополнительных требований, связанных с реализацией компонентного подхода. Не владея конструктивным взглядом на программы невозможно удовлетворять эти требования.

При обучении основам программирования, как средство, которое позволяет уточнить понятие программной конструкции, предложено использовать классификацию, а как классификационный признак – уровень инкапсуляции, который строится на основе принципов инженерии программного обеспечения – инкапсуляции и многоуровневого представления [13].

Инкапсуляция в целом – это процесс создания оболочки вокруг тех или иных веществ. Оболочка называется капсулой, а вещества в капсуле инкапсулированными и характеризуются высокими внутренними (cohesion) и низкими внешними (coupling) связями. Инкапсуляция в программировании – это процесс, который агрегирует «вещество» программ, кодированного в том или ином языке программирования, образуя капсулы. Они могут использоваться в программировании как единственные в определенном смысле законченные части программ. Этот процесс, как увидим дальше, регламентирован.

В программе роль инкапсулированного «вещества» играют ее компоненты, например, операторы, подпрограммы. Образование капсулы (конструкции) вокруг компонентов программы позволяет достичь следующих, присущих конструктивному взгляду, целей [13]:

- манипулировать при написании, отладке и понимании программ капсулами, рассматривая их как законченные части;
- указать метод программирования, который регламентирует использование капсул;
- указать значения, обрабатываемые капсулой;
- ограничивать допуск к компонентам, размещенным в капсуле;
- скрывать детали реализации компонентов, размещенных в капсуле;
- использовать капсулы для построения других капсул.

Капсула строится агрегированием необходимых частей программы и созданием оболочки и интерфейса, что обеспечивает правильное применение капсулы. При создании или повторном использова-

нии капсул перед программистом возникают три вопроса. Во-первых, в каком случае (для реализации каких проектных решений – действий) может применяться та или иная капсула (какова концепция капсулы). Во-вторых, какой должна быть капсула (какое устройство, конструкция капсулы). В-третьих, как связывается капсула с окружением (контекст капсулы). Программист, создавая или повторно, многократно используя капсулы, применяет их в соответствии с методом программирования и реализует проектные решения, полученные в предыдущих фазах жизненного цикла программы, что составляет часть процессов конструирования программ. Капсулу, которая применяется в соответствии с методом программирования можно называть программной конструкцией. Чтобы охарактеризовать известные ныне программные конструкции, а также выяснить, какие значения они обрабатывают и в чем заключаются методы конструирования (программирования) программ из них, воспользуемся еще одним принципом инженерии программного обеспечения – многоуровневым представлением.

Применяя принцип инкапсуляции на разных уровнях представления структуры программы, соответствующие различным степеням абстракции программного обеспечения, получено понятие уровня инкапсуляции [13]. Воспользовавшись этим понятием, можно выяснить типы программных конструкций и соответствующие методы программирования (конструирования) программ. Сейчас можно выделить шесть уровней инкапсуляции и столько же типов программных конструкций (таблица).

Таблица. Уровни инкапсуляции

Уровень инкапсуляции	**Инкапсулируемые части программы**	**Создаваемая конструкция (капсула)**	**Метод программирования (применения)**	**Средства и механизмы**	**Метафора**
Лексический	Символы алфавита языка программирования	Лексема	Правила создания лексем и выражений	Оператор присваивания, операции	Звено
Операторный	Лексемы и выражения из них	Структурный оператор	Структурное программирование	Структурный операторный базис языка программирования	Цепь
Подпрограммный	Описание данных, структурные операторы	Подпрограмма (макрос, процедура, функция)	Процедурное (подпрограммное) программирование	Независимая компиляция	Механізм
Модульный	Описание данных, подпрограммы	Модуль (пакет)	Модульное (композиционное) программирование	Механизмы сокрытия и видимости, раздельная компиляция	Механізм
Классный	Описание данных, подпрограммы	Класс	Объектно-ориентированное (классификационное) программирование	Наследование, полиморфизм, связывание	Механізм
Мегамодульный (система систем)	Онтология, модули, классы	Мегамодуль (система)	Мегапрограммирование	Операционная независимость, независимое управление и поведение, географическая распределенность, эволюционное и адаптивное развитие	Машина

Степень абстрагирования и понимания программы повышается от лексического уровня к мегамодульному. Каждому уровню инкапсуляции соответствует свой тип капсул (программная конструкция), правила образования и дисциплина их использования в конструировании программ (метод программирования).

Программные конструкции. В аспекте конструктивной точки зрения на программу, программная конструкция – это фундаментальное понятие языка программирования. С точки зрения инженерии программного обеспечения каждая часть программы – это программная конструкция, если в ней инкапсулированы другие части программы (нижнего уровня инкапсуляции), она характеризуется конструктивными (системными) свойствами и имеет собственный способ применения [13].

Простейшая программная конструкция, это лексема. Являясь обозначением, она играет важную роль в построении таких конструкций как литерал, константа, переменная. Более сложными программными конструкциями являются выражение, оператор, подпрограмма, модуль, класс, мегамодуль (система систем).

Лексемы. Символы алфавита не играют в языке самостоятельной роли, однако они используются для построения лексем. С одной стороны, лексемы – это простые программные конструкции, которые составляют словарный запас языка. А с другой стороны, лексемы – это капсулы, которые инкапсулируют символы алфавита. «Оболочки» капсул образуют специальные символы, или символы других лексем. Капсулу-лексему можно подать размеченной цепочкой: $S_i*l_1\ l_2...l_n\ *S_j...$, где S_i, S_j, $l_k \in V$; V – алфавит языка; S_i, S_j – пробельные символы или символы других лексем; l_k, ($k = 1, ..., n$) – символы лексемы, при этом l_1 – первый символ, а l_n – последний символ лексемы (если S_i, S_j символы других лексем). В лексеме-капсуле можно установить порядок (последовательность) символов, который дает понятие границ капсулы (первый и последний символы капсулы) и используется лексическим анализатором – программой, входящей в состав транслятора. Этот порядок позволяет говорить о входе в капсулу и выходе из нее. Все лексемы в тексте программы являются обозначениями. Это означает, что лексемы обозначают другие конструкции программы, которые строятся и используются в процессах трансляции и выполнения программы. На лексическом уровне инкапсуляции текст программы конструктивно состоит из последовательности лексем. Когда программа транслируется или выполняется, тогда каждой лексеме ставится в соответствие некоторое значение. Соответствие между лексемой и значением может устанавливать программист или транслятор, или это соответствие может заранее определяться стандартным окружением языка программирования. На рис. 1 показана роль лексемы в конструкции переменной.

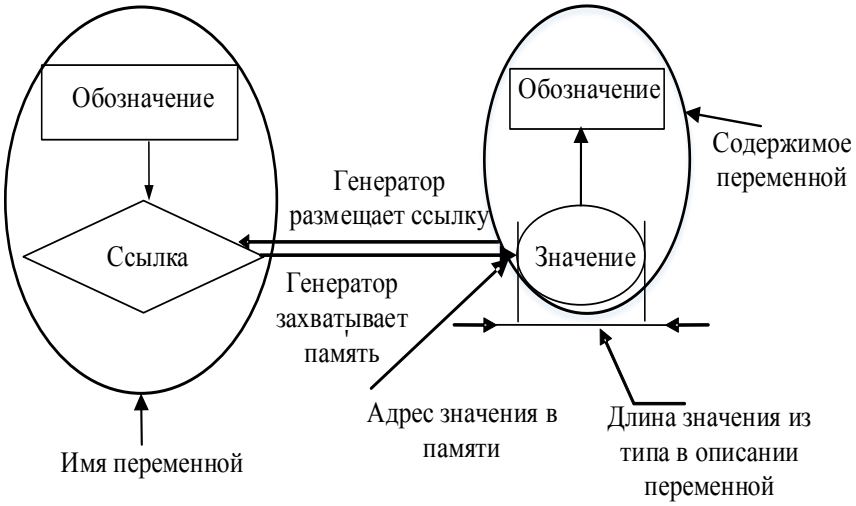


Рис. 1. Конструкция простой переменной

Установление соответствия между лексемой и значением или между обозначением и значением, или еще шире - между двумя объектами программы обеспечивается механизмом связывания (binding).

Выражение описывает правило обработки значений, содержащихся в программных объектах, входящих в его состав. В результате выполнения правила образуется значение, которое является значением выражения. Таким образом, текст выражения – это обозначение, и после выполнения выражения – это обозначение связывается с значением. Поэтому, выражение является программной конструкцией наряду с литералом, константой и переменной.

Операторы. Структурные операторы составляют структурный операторный базис языков программирования, которого достаточно для записи любой программы. Именно структурные операторы и структурное программирование – метод их применения – в свое время стали фундаментальными концепциями в языках программирования. Лексемы выступают как части, которые образуют структурную капсулу, а оболочка капсулы и интерфейс реализуются через организационные ограничения. Во-первых, что касается оболочки, запрещается произвольный доступ к конструкциям внутри капсулы, и произвольный выход из капсулы (рис. 2). Это запрет применения операторов go to, break, exit или continuous. Во-вторых, что касается интерфейса, указанные ограничения технически можно преодолеть, но если их учитывать, то в любой структурной капсуле предполагается только один вход и один выход (рис. 2), с помощью которых она соединяется с другими капсулами, образуя пары (звенья цепи) и цепи (таблица).

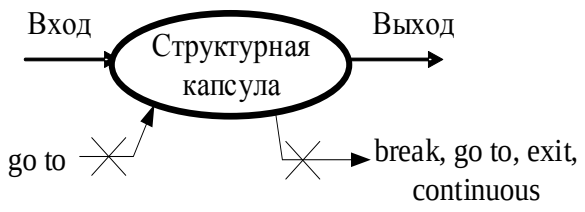


Рис. 2. Структурная капсула

Применение структурного программирования ведет к созданию программных текстов, которые легко читаются и модифицируются. В рамках структурного программирования рассматриваются преобразования программ, которые могут выполняться вручную или автоматически.

Рассмотрим простой пример. В технологическом аспекте необходимо создать подпрограмму («механизм»), которая осуществляет обмен значений целого типа, swap (int a, int b). Имеем следующую конструкцию (рис. 3, 4):

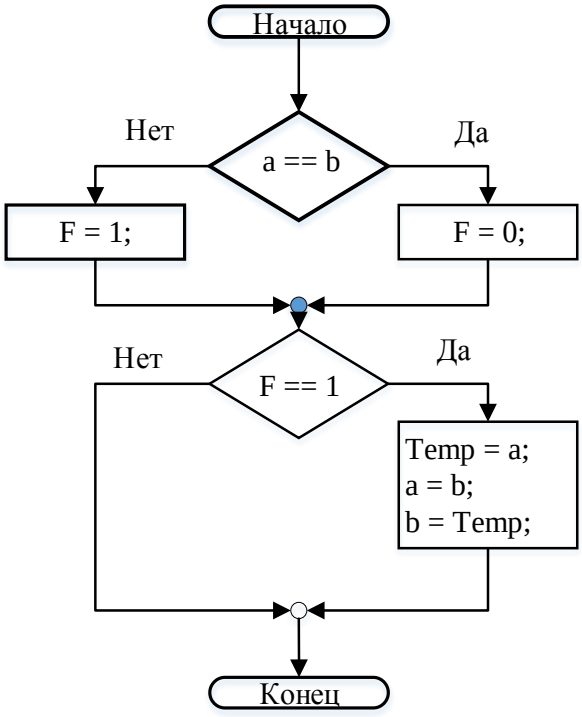


Рис. 3. Конструкция программы

```
void swap(int &a, int &b)
{
 int F;
 int Temp;
 if a == b
    F = 0;
 else
    F = 1;
 if F == 1
    {
     Temp = a;
     a = b;
     b = Temp;
    }
}
```

Рис. 4. Текст программы

– две «цепочки» типа условный оператор в форме альтернативы и выбора;

– три «звена» типа преобразовательного оператора (оператор присваивания);

– один механизм, полученный путем последовательного соединения «цепочек».

Таким образом, процесс конструирования подпрограммы, это использование конструкций-«звеньев», сочетание «звеньев» (построение «цепочек») и построение «механизма» путем последовательного соединения «цепочек». «Механизм» – это «замкнутая» последовательность «цепочек» в смысле выполняемой функции, которая, как известно у модуля должна быть одна.

Очевидно, что полученная конструкция проста для понимания и модификации, что является требованиями компонентного подхода.

Подпрограммы. Программная конструкция на подпрограммном уровне инкапсуляции состоит из операторов и называется подпрограммой. Из подпрограмм формируются библиотеки подпрограмм, которые когда-то стали основой систем программирования. Сначала подпрограммы рассматривались как повторяющиеся части программ, которые предварительно описываются, хранятся отдельно от программы или непосредственно в ней и многократно используются путем встраивания вызовов подпрограмм в программу. Таким образом, подпрограммы служили средством сокращения текстов программ. Сейчас благодаря процедурной абстракции и абстракции управления подпрограмма играет очень важную роль в модульной организации программ и их проектировании и является фундаментальной концепцией любого языка программирования, средством абстрагирования и проектирования. Подпрограмма — это повторяющаяся часть программы, которая важна с двух точек зрения: во-первых, как средство сокращения затрат на запись программы, во-вторых, как средство проектирования программ. На подпрограммном уровне инкапсуляции содержимое капсулы, составленное из структурных операторов методом структурного программирования, не интересует программиста, поэтому основной интерес представляет оболочка капсулы и, в частности ее интерфейс. Оболочка капсулы открытой подпрограммы не существует после ее вызова, поэтому она «непрочная», а содержимое капсулы, видно программисту. Оболочка закрытой подпрограммной капсулы существует и после вызова, хотя ее содержимое также видно программисту, но она более «прочная», чем оболочка капсулы структурного оператора. Степень «прочности» определяется оператором go to. Если для капсулы структурного оператора переход в нее и выход из нее оператором go to запрещены организационно, то в подпрограммной капсуле (закрытая подпрограмма) это невозможно уже конструктивно. Поэтому часть оболочки подпрограммной капсулы должна играть роль интерфейса (рис. 5).

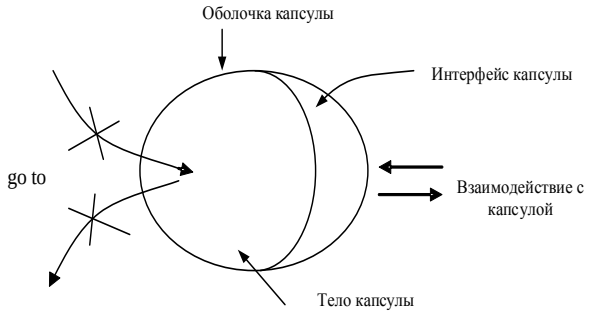


Рис. 5. Капсула подпрограммы

Интерфейс обеспечивает связь внешнего окружения капсулы с компонентами, размещенными в середине капсулы. По определению инкапсуляции подпрограммного уровня следует, что подпрограммная капсула – это последовательность структурных операторов, «помещенных» в оболочку. Порядок размещения операторов в капсуле соответствует алгоритму решаемой капсулой задачи. Отражая сущность процедурной абстракции, понятие подпрограммной капсулы позволяет программисту-читателю абстрагироваться от того, как решает задачу капсула, и сосредоточиться на том, какую задачу она решает.

Подпрограммную капсулу можно рассматривать как «черный ящик», на вход которого передается управление, а на выходе управление возвращается. В результате передачи управления и выполнения операторов капсулы можно получить полезный эффект, который найдет отражение в векторе состояния программы, в точке после вызова подпрограммы. Такой взгляд на подпрограмму составляет сущность абстракции управления. Таким образом, закрытая подпрограмма – это не столько средство сокращения записи текста программы (как считалось ранее), сколько средство разложения (декомпозиции) программы на логически связанные, законченные и, в определенном смысле, закрытые компоненты. Перед началом разработки программы программист имеет более или менее точную формулировку задачи в терминах некоторого домена и инструмент – язык программирования. Говорят, что между сформулированной задачей и языком программирования существует концептуальноя расстояние, которое преодолевается с помощью программирования [14]. Абстрактные типы данных – это фундаментальная концепция программирования, которая направляет как замыслы разработчиков языков программирования, так и повседневные действия программистов. Для первых, абстрактный тип данных – это та концепция, которую должен эффективно реализовать язык программирования, а для вторых – это, тот продукт, который должен создавать программист в любом домене. Подпрограммный уровень инкапсуляции позволяет реализовывать абстрактные типы данных, но очень неэффективно. Именно это является причиной того, что требуется применение возможностей классного или модульного уровней инкапсуляции. Именно на эффективную реализацию абстрактных типов данных и направлены возможности этих уровней инкапсуляции. Класс и модуль – компоненты, которые обычно представляют абстрактный тип данных.

Модуль. На сегодня известно две схемы реализации пост подпрограммного уровня инкапсуляции – композиционная и классификационная. Обе существуют одновременно. Первая используется на модульном уровне инкапсуляции, а вторая - на классном. Обе схемы строятся на основе понятия отношения между компонентами схемы. Различия проявляются в разной интерпретации понятия отношение в схемах как при организации сред (библиотек, коллекций) из программных компонентов (модулей или классов), так и при их использовании. При разработке программ, которые состоят из большого количества частей, основное требование разложения программы на составные части приобретает первостепенное значение. Разложив большую задачу на части, можно получить задачи-компоненты с таким количеством деталей, когда ее можно реализовать, а каждый компонент можно реализовывать отдельно и независимо. Такое разложение, которое реализуется на основе независимой компиляции, абсолютно необходимо, если в разработке программы участвует более одного программиста. Образующиеся в результате разложения задачи реализуются частями программы, которые называются в процедурном программировании – подпрограммы, в композиционном программировании – модули или пакеты. Модуль – это капсула, которая инкапсулирует данные и подпрограммы. В ограниченном виде модуль ввел Н. Вирт в языке Pascal, а наибольшего развития он приобрел в языках Modula-2, Modula-3 и Ada. В языке Ada модуль называется пакетом.

Другое требование – сокрытие деталей реализации модулей приводит к тому, что модуль обычно подается двумя конструкциями. В первой – интерфейсе – описываются ресурсы (сервисы), которые представляет модуль (в зависимости от типа модуля). Во второй – описываются реализации ресурсов, поданных интерфейсом модуля. Таким образом, любая капсула-модуль состоит из интерфейса и инкапсулированных конструкций (рис. 6). Механизм экспорта и импорта действует через перечень экспортируемых ресурсов (типы, объекты и подпрограммы), описанных в модуле определения и перечень импортируемых ресурсов, объявленных в модуле реализации. Обычно, при обучении основам программирования различают модули,

которые поставляют с языком программирования и реализуют, например, ввод/вывод.

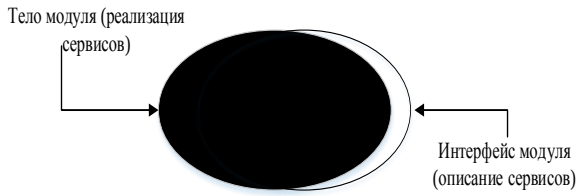


Рис. 6. Капсула модуля

Модульное программирование – это метод создания программ из программных конструкций модулей, который применяется на модульном уровне инкапсуляции. Он основывается на механизме раздельной компиляции, который наряду с указанным устройством модуля обеспечивает полное сокрытие информации о реализации модуля. Суть механизма заключается в том, что обе части модуля (интерфейс и реализация) компилируются раздельно. Благодаря этому в случае изменения реализации ресурсов (с сохранением ее интерфейсов) в перекомпиляции нуждаются только модули реализации и не требует программа, в которой применены соответствующие модули определения. Модульное программирование называется еще композиционным программированием. Альтернативой ему является классификационное программирования, реализацией которого является объектно-ориентированное программирование. Выделение программного модуля влияет на программирование в трех аспектах:

- понятность – программа понятна, если она состоит из модулей;
- эффективность – программа работает более эффективно, поскольку модуль позволяет рационально организовать структуру текста;
- тестируемость – программа легче тестируется, если она модульная.

Классы. Концепция объектно-ориентированного программирования исторически тесно связана с концепцией универсального языка программирования, которая интенсивно разрабатывалась в 70х. Основными принципами объектно-ориентированного программирования являются наследование и полиморфизм. Для реализации полиморфизма в объектно-ориентированном программировании кроме раннего связывания – механизма, который обеспечивает установление связи между интерфейсом программной конструкции и одной из реализаций (форм) конструкции, предусмотренных для нее, во время компиляции, вводится позднее связывание – установление указанной связи во время выполнения программы. Кроме основных могут применяться дополнительные принципы: параметризация, многократное и повторное использование. Многократное использование (systematic reuse) – принцип, обеспечивающий создание, так называемых типовых программных конструкций (компонентов), которые можно использовать многократно. Этим принципом в модульном и объектно-ориентированном программировании руководствуются при проектировании модулей и классов. Повторное использование (reuse) – принцип, который обеспечивает использование наследуемого программного обеспечения в новых разработках. Любой класс как программная конструкция в целом, также как модуль состоит из интерфейса и тела (рис. 7). Интерфейс описывает ресурсы, которые предоставляются классом. Тело состоит из конструкций, описывающих реализацию ресурсов, которые предоставляет класс. В зависимости от конкретного языка программирования строение класса может отличаться.

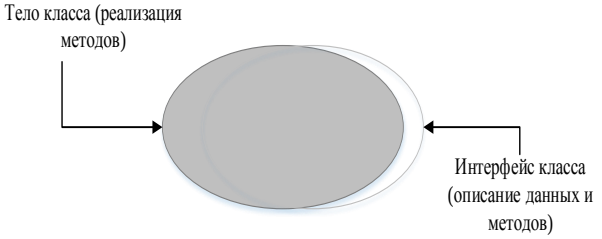


Рис. 7. Капсула класса

Поскольку класс, как правило – это абстрактный тип данных, то тело класса и также, как и модуля должно содержать описание следующих компонентов:

- значений, предоставляемых абстрактным типом данных (классом)
- подпрограмм, обеспечивающих обработку значений абстрактного типа данных (класса).

Важно, что класс, в отличии от модуля не реализует важный принцип модуляризации – сокрытие информации. Он реализуется либо схемой, имитирующей раздельную компиляцию (подобно модульному сокрытию), либо путем введения дополнительных понятий в язык программирования.

Мегамодули. Системы систем. Понятие мегамодуля введено в работе [15], и в отличие от модуля инкапсулирует не только данные и операции, но и поведение и доменные знания, отражающие, в частности, языки, культуры и традиции. При этом, несколько мегамодулей могут входить в мегапрограмму, которая пишется на языке мегапрограммирования. Позднее, было введено понятие системы систем (system of systems), которое по сути является мегапрограммой [16]. Так как программирование системы систем значительно сложнее основ программирования, этот класс модулей и составление из них систем должен изучаться в отдельной дисциплине, например, при подготовке магистров.

Дидактика основ программирования

В 2006 г. постановлением Кабинета Министров Украины было открыто новое направление обучения 6.050103 «Программная инженерия». Сейчас, это 121 специальность «Инженерия программного обеспечения». Был разработан соответствующий стандарт для подготовки бакалавров [17]. Новое направление обучения, очевидно требует собственного учебно-методического обеспечения, поскольку, например, от направления «Компьютерные науки» его отличает четкая инженерная направленность. Дисциплины учебного плана этого направления должны как можно больше учитывать указанную направленность. К сожалению, в основном из-за нехватки грамотных, в смысле инженерии программного обеспечения, педагогических кадров, в университетах сложилась ситуация, которая характеризуется тем, что не только не выдерживается соответствующий стандарт «в большом», но и «в малом», особенно в дисциплинах, которые перешли, например, из Компьютерных наук. Это относится и к дисциплине «Основы программирования», которая преподается как правило на первом курсе университета. Поэтому не удивительно то, что отрасль по-прежнему недовольна уровнем подготовки студентов, и как следствие несет огромные затраты по «доводке» студентов до требуемого уровня [18]. Принимая во внимание, то, что в отрасли катастрофически не хватает грамотных программистов, дидактика дисциплины «Основы программирования» приобретает особое значение.

На основе конструктивного подхода к программе, изложенного в статье построена дидактика основ программирования, которая реализована в учебном пособии автора [19].

Известно два подхода к преподаванию основ программирования [20]:

- императивный (imperative-first) – традиционный подход, в котором изложение основ осуществляется, начиная с императивных аспектов языка программирования: выражения, управляющие структуры, процедуры и функции и другие основные элементы процедурных языков программирования. Объектно-ориентированное программирование рассматривается позже в другой дисциплине;
- объектно-ориентированный (objects-first) – с самого начала обучения главное внимание уделяется объектно-ориентированному программированию, объясняя понятие класса, объекта и наследования, затем предлагают студентам написать объектно-ориентированные программы. Позже вводятся императивные аспекты объектно-ориентированного языка программирования.

Традиционно, в Украине используется только первый подход при обучении основам программирования. Это ведет к тому, что, когда начинается обучение объектно-ориентированному программи-

рованию студенты с трудом усваивают такие концепции как класс, объект, наследование, полиморфизм, виртуальные функции, и еще труднее даётся понимание специальных конструкций языков (интерфейсы, делегаты), ориентированных на реализацию модульности.

Автор делал попытку, применения второго подхода при обучении. Опыт показал, что такая подача конструкций программ также затрудняет обучение. Небольшой начальный опыт в объектно-ориентированном программировании с последующим переключением на процедурное программирование только запутывает студента. Вероятно, проблем бы не было или было меньше, если изучать «чистый» объектно-ориентированный язык подобный *SmallTalk,* но это не привлекает студентов в Украине, в перспективе трудоустройства.

Многолетний опыт автора в использовании изложенного в статье конструктивного подхода показал, что он понятен студентам и позволяет на общей основе осваивать как сложные конструкции императивной части – переменная, ссылка, указатель, подпрограммы, и механизмы – разыменование, приведение типов, так и конструкции модульной части – модули, классы, объекты, и механизмы – сокрытие, полиморфизм, наследование. Конструктивный подход знакомит студентов не только с традиционно распространённым в Украине объектно-ориентированном программировании, но с менее известным у нас – модульным программированием. Кроме этого, студенты с первых занятий осознанно настраиваются на применение повторного и многократного использования, понимая цели и задачи компонентной инженерии программного обеспечения.

Дидактика основ программирования, основанная на конструктивном подходе, базируется на понятии конструкции и структурирует изложение учебного материала в соответствии с уровнями инкапсуляции (табл.). Для объяснения устройства конструкций широко используются графические схемы подобные тем, что были введены в работах [21, 22] (например, рис. 1).

Выводы

Известно два подхода к преподаванию основ программирования:

- императивный – традиционный подход, в котором изложение основ осуществляется, начиная с императивных аспектов языка программирования;
- объектно-ориентированный – с самого начала обучения главное внимание уделяется объектно-ориентированному программированию.

В контексте инженерии программного обеспечения для обучения студентов основам программирования целесообразно использовать конструктивный подход, который подготовит студентов к созданию и сопровождению программного обеспечения методами многократного и повторного использования в рамках парадигмы компонентной разработки.

Литература

1. Boehm B., 2006, A View of 20th and 21st Century Software engineering [Text]. ICSE'06. May 20–28. China. 2006. P. 12–29.
2. Report on a conference sponsored by the NATO science committee, Garmisch, Germany, 7th to 11th October 1968, Editors: Peter Naur and Brian Randell.
3. Segmentation and Design Strategies for Modular Programming." In T. O. Barnett and L. L. Constantine (eds.), Modular Programming: Proceedings of a National Symposium. Cambridge, Mass.: Information & Systems Press, 1968.
4. Wirth N. Programming in Modula-2. Springer-Verlag, Heidelberg, New York, 1982.
5. Jean Ichbiah (October 1984). «Ada: Past, Present, Future – An Interview with Jean Ichbiah, the Principal Designer of Ada». *Communications of the ACM*. **27** (10). P. 990–997. doi:10.1145/358274.358278.
6. Dahl O.-J., Myhrhaug B., Nygaard K. SIMULA67, Common base language/-Oslo. 1968. 96 p.
7. Hoare C.A.R. An axiomatic basis for computer programming, Comm. Of ACM, 12 (1969). P. 576–580.

8. Goldberg A., Robson D. SmallTalk 80 The language and its implementation, Addison-Wesley, New-York, 1983.
9. Maier M.W. Architecting principels for systems-of-systems, Systems engineering, 1, 4(1998). P. 267–284.
10. Энгельмейер П.К. Философия техники, М. 1912.
11. Redtenbacher F., Der Maschinbau, Mannheim, 1862.
12. Boehm B.W. Improving Software Productivity. Computer. 1987. Vol. 20, N 9. P. 43–57.
13. Сидоров Н.А. Применение принципов программной инженерии в преподавании основ программирования. *Управляющие системы и машины*. 1998. № 4. С. 50–59.
14. Дал У., Дейкстра Э., Хоор К. Структурное программирование = Structured Programming. 1-е изд. М.: Мир, 1975. 247 с.
15. Widerhold G., Wegner P., Ceri S., Toward megaprogramming. *Communication of the ACM*. 1992. Vol. 35, N 11. P. 89–99.
16. System of systems engineering, ed.by Jamshidi M., John Wiley&Sons, 2009, 591 p.
17. Сидоров Н.А. Инженерия программного обеспечения – учебная дисциплина или подготовка бакалавра? *Управляющие системы и машины*. 2006. № 2. С. 25–34.
18. It Ukraine from a to z, http://www.uadn.net/files/ua_hightech.pdf
19. Сидоров М. Основи програмування. Київ, 2018, 435 с.
20. Bennedsen J., Teaching and Learning Introductory Programming, – A Model-Based Approach. 327 p.
21. Линдси Ч., ван дер Мюйлен С., Нефор-мальное введение в Алгол 68, М., Мир, 1973.
22. Баррон Д., Введение в языки программирования, М., Мир, 1980. 189 с.

References

1. Boehm B., 2006, A View of 20th and 21st Century Software engineering[Text]. ICSE'06.-May 20–28 China. 2006. P. 12–29.
2. Report on a conference sponsored by the NATO science committee, Garmisch, Germany, 7-th to 11-th October 1968, Editors: Peter Naur and Brian Randell.
3. Segmentation and Design Strategies for Modular Programming." In T. O. Barnett and L. L. Constantine (eds.), Modular Programming: Proceedings of a National Symposium. Cambridge, Mass.: Information & Systems Press, 1968.
4. Wirth N. Programming in Modula-2. Springer-Verlag, Heidelberg, New York, 1982.
5. Jean Ichbiah (October 1984). «Ada: Past, Present, Future — An Interview with Jean Ichbiah, the Principal Designer of Ada». *Communications of the ACM* **27** (10): 990–997. DOI:10.1145/358274.358278
6. Dahl O.-J., Myhrhaug B., Nygaard K. SIMULA67, Common base language/-Oslo, 1968, 96 p.
7. Hoare C.A.R. An axiomatic basis for computer programming, Comm. Of ACM, 12 (1969). P. 576–580.
8. Goldberg A., Robson D. SmallTalk 80 The language and its implementation, Addison-Wesley, New-York, 1983.
9. M.W.Maier Architecting principels for systems-of-systems, Systems engineering, 1, 4(1998). P. 267–284.
10. Engelmeyer. P. C. Philosophy of technology. M., 1912.
11. Redtenbacher F., Der Maschinbau, Mannheim, 1862.
12. Boehm B.W. Improving Software Productivity. Computer. 1987. Vol. 20, N 9. P. 43–57.
13. Sydorov M. Using the software engineering principals in basics programming education, USIM. 1998. N 4. P. 50–59.
14. Dijkstra E.W. Structured Programming. 1975. 247 p.
15. Widerhold G., Wegner P., Ceri S. Toward megaprogramming. *Communication of the ACM*. 1992. Vol. 35, N 11. P. 89–99.
16. System of systems engineering, ed.by Jamshidi M., John Wiley&Sons, 2009, 591 p.
17. Sydorov M. Is the software engineering education subject or postgraduate, USIM. 2006. N 2. P. 25–34.
18. It Ukraine from a to z, http://www.uadn.net/files/ua_hightech.pdf
19. Sydorov M., Basics of programming, Kiev 2018, 435 p.
20. Bennedsen J. Teaching and Learning Introductory Programming, – A Model-Based Approach. 327 p.

21. Lindsey C., van der Meulen S., Informal introduction to ALGOL 68, London, 1971.
22. Barron D.W. An introduction to the study of programming languages, Cambridge university press, London, 1977.

Получено 18.07.2019

Об авторе:

Сидоров Николай Александрович,
доктор технических наук,
профессор.
Количество научных публикаций в украинских изданиях – 118.
Количество научных публикаций в зарубежных изданиях – 12.
http://orcid.org/0000-0002-3794-780X

Место работы автора:

Национальный Технический Университет Украины «Киевский политехнический институт имени Игоря Сикорского», факультет информатики и вычислительной техники, кафедра автоматизированных систем обработки информации и управления, АСОИУ, профессор.
02000, Киев,
ул. Политехническая, 41.
Моб. тел.: 067 7980361.
Тел.: 044 2343600.
E-mail: nikolay.sidorov@livenau.net,
sna@nau.edu.ua

УДК 004.855:519.216 https://doi.org/10.15407/pp2019.03.058

О.С. Балабанов

ЗАДАЧІ ТА МЕТОДИ АНАЛІЗУ ВЕЛИКИХ ДАНИХ (ОГЛЯД)

Розглянуто основні задачі та методи глибокого аналізу великих даних. У викладі зроблено акцент на «фізичному» сенсі задач і методів, без математичних деталей. Спектр аналізу й використання великих даних охоплює чотири концептуальні класи завдань: «інтелектуальний» пошук інформації; масовану (конвеєрну) переробку даних; індукцію моделі об'єкту (середовища) та екстракцію знань з даних (відкриття закономірностей). Висвітлено суть типових класів задач великої аналітики: групування випадків (кластеризація даних); виведення ціле-визначених моделей (класифікація, регресія); виведення генеративних моделей; відкриття структур і закономірностей. Розглянуто ключові методи кластеризації, регресії та класифікації (включаючи глибоке навчання), а також виведення генеративних моделей. Методи розв'язання ціле-визначених задач поділяються на ті, що виводять модель у явному вигляді (модель «відокремлюється» від даних) та методи, «прив'язані до даних». Охарактеризовано особливості задач аналізу темпоральних даних (сегментація, виявлення точок зміни і т. д.). Детальніше викладено індуктивне виведення каузальних мереж методами, основаними на незалежності. Вказано особливості виведення динамічних каузальних мереж. Окремо підсумовано загальні особливості застосування статистичних методів у аналізі великих даних.

Ключові слова: великі дані, аналіз даних, виведення генеративної моделі, статистичні методи, кластеризація, регресія, прогноз, виявлення закономірностей, темпоральні дані, каузальні мережі.

Вступ

Стаття є другою частиною огляду аналітики великих даних. В першій частині [1] було розглянуто сфери застосувань великих даних, режими їх використання, основні напрямки, принципи, задачі глибокого аналізу великих даних та організацію циклу робіт з аналізу даних. У другій частині детальніше викладено задачі й методи глибокого аналізу даних.

Великі дані (ВД) характеризуються як масовані (сьогодні – це порядку 10^{21} байтів), різноманітні («строкаті»), неоднорідні, неструктуровані (чи погано структуровані), мінливі та «швидкі» (тобто такі, що швидко оновлюються чи поповнюються) [2–9]. Про глибокий аналіз доречно говорити тільки коли даних багато і вони багатовимірні. Основними сферами застосування аналітики великих даних є бізнес та наукові дослідження. Застосування у державному секторі подібні до бізнесових або наукових. Застосування у бізнесі зосереджуються переважно на задачах предикції (прогнозування), розпізнавання, виявлення трендів (тенденцій) та аномалій, сумарізації даних тощо. Застосування у наукових дослідженнях в першу чергу спрямовані на інтегративну (синтетичну) переробку експериментальних даних, виявлення закономірностей та зв’язків, генерацію та перевірку гіпотез, виведення моделей, які допомагають зрозуміти об’єкт дослідження. Образно кажучи, науковці шукають пояснень, узагальнень та знань, а бізнес цікавить, що відбувається, що буде і «що станеться, якщо ми зробимо так».

Основний режим використання ВД – глибокий аналіз даних, коли величезний масив сирої інформації «перетравлюється» і перетворюється на концентровану й цінну інформацію кінцевого споживання. Як кажуть, з даних «висмоктується» (екстрагується) їх цінний сенс. Впровадження великих даних неодмінно і безальтернативно («автоматично») передбачає застосування великої аналітики. Оскільки сфера великих даних та сфера великої аналітики взаємно доповнюють одна другу, доречно вести мову про формування єдиного річища, до якого зіллються дослідження і розробки, що охоплюють цикл діяльності від збору даних до вироблення інформаційного продукту кінцевого споживання. Тобто формується «наука» (в широкому розумінні) «Великі дані плюс Велика Аналітика» [3, 7–20].

В роботі [1] виділено наступні типи (концептуальні режими) використання ВД:

1) «інтелектуальний» пошук потрібної інформації (фактів) або запису, файлу, що містить ту інформацію;

2) масована (конвеєрна) переробка даних («відпрацювання» даних за один-два сканування);

3) індукція моделі об'єкту (джерела даних);

4) екстракція знань з даних (відкриття закономірностей та структур).

В процесі «інтелектуального» пошуку також застосовується аналіз даних, але результат пошуку по рівню абстракції є той самий, у якому дані зберігаються. (По-суті, результат є компіляцією фрагментів даних.) Гіпотетичний приклад запиту на «інтелектуальний» пошук інформації наведено в [1]. Режими використання даних «3» та «4» єдині в тому, що вони здійснюють узагальнення даних, тобто результатом є концентрована («кристалізована») інформація. Далі розглядаються методи розв'язання саме задач глибокого аналізу даних. Вважається, що дані вже підготовлені для аналізу. Сучасні методи аналізу даних та їх застосування викладено в [4, 10, 21–29]. Багато стандартних методів аналізу даних імплементовано у середовищах програмування **R**, Python, SAS, Matlab, Apache Mahout, Apache Spark і т. д.

Спектр задач аналізу даних та відкриття знань

Як аргументовано в [1], на виході інформаційної технології можна отримати тільки те, що містилося в масі даних перед переробкою (в «розчиненій», розпорошеній формі), а також було задано як апріорна інформація. Коли даних багато, внесок даних переважає. Успішність аналізу ВД визначається гармонійним взаємним доповненням даних, апріорної інформації та вдалою специфікацією завдання. Для розв'язання задач аналізу використовуються переважно статистичні методи. Протягом останніх кількох десятиліть карколомне зростання швидкодії комп'ютерів та обчислювальних систем стимулювало бурхливий розвиток статистичних методів обробки даних [23–29]. На передній план висунулися обчислювально-інтенсивні та комбінаторні методи – непараметричні, ітеративні та апроксимаційні, а також методи, вільні від припущень і форм розподілень. Широко застосовується техніка бутстрепінгу, згладжування кернелом, генерація вибірки за Гіббсом, максимізація очікування і методи MCMC [25–28]. Великі дані спричинили наступний поштовх розвитку статистичних методів.

Загальна спрощена система задач аналізу ВД показана на рис. 1. (Деякі з задач, представлених на рис. 1, можуть виступати етапом вирішення інших задач.) Типовий і лаконічний спосіб визначити акцент й мету завдання аналізу – вказати цільову змінну (характеристику, атрибут) y. В такому разі отримуємо ціле-визначену («націлену») задачу. Виведення ціле-визначених моделей – один з найпоширеніших класів задач аналізу даних. Задачі, в яких не задано цільової змінної, часто називають unsupervised learning [25, 26]. До таких задач відносять виведення генеративних моделей, кластеризацію і багато іншого. Коли інтерес аналітика не акцентовано на певній змінній, підходяща «загальна» задача – вивести генеративну модель для певного набору змінних X, тобто модель, яка відображає сумісне розподілення ймовірностей $p(X)$. Опис сукупності даних називаємо моделлю тоді, коли він компактний, поданий у зручній (наочній) формі й відображає головний вміст даних (відкидаючи випадкове і несуттєве). Подібні моделі $p(X)$ є генеративними моделями в слабкому сенсі. Іноді більш компактний і аналітичний опис розподілення даних $p(X)$ можна отримати через гіпотетичні змінні Z, так що $p(X) = \Omega(p_Z(Z))$, де $\Omega(\cdot)$ – певне стандартне перетворення. Подібну репрезентацію надають факторний аналіз, аналіз головних компонент тощо.

Серед методів виведення ціле-визначених моделей доцільно розрізняти методи, призначені безпосередньо для прогнозування (оцінки) значення цільової змінної, і методи, призначені для побудови самої моделі як синтетичного змістовного результату. Крім того, аналітика може цікавити навіть не вся модель, а тільки

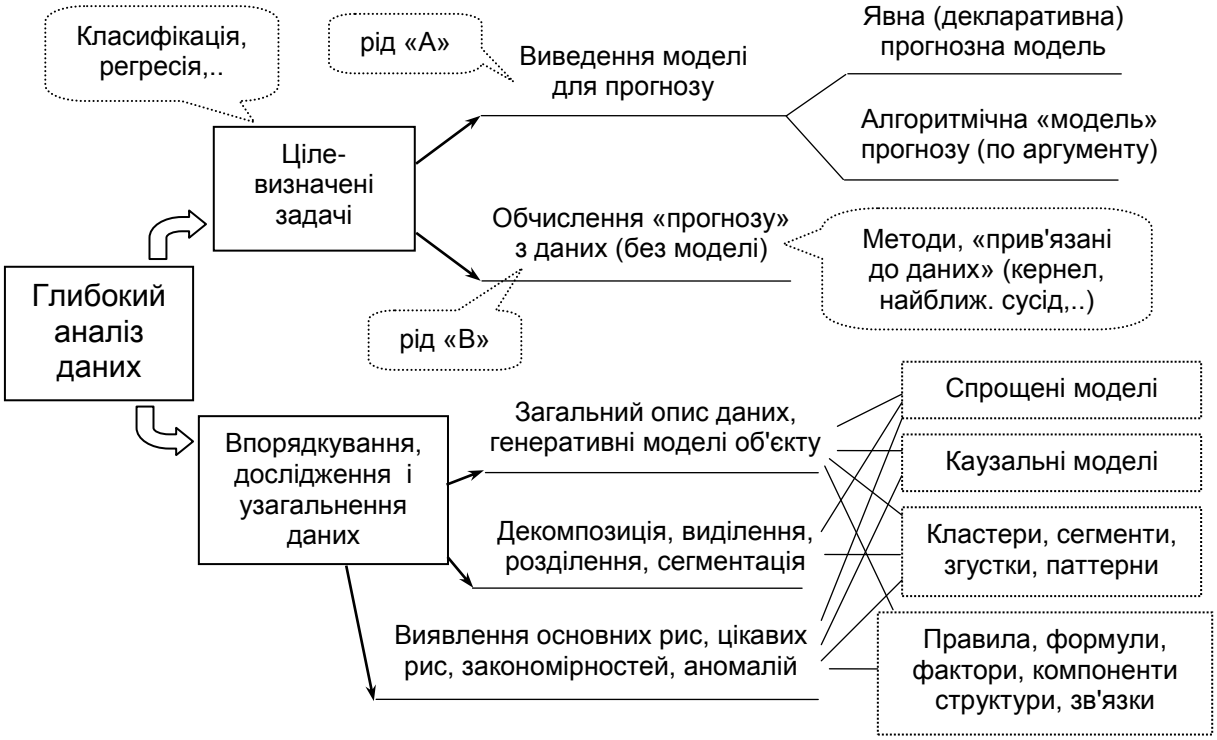


Рис. 1. Система задач великої аналітики

головні фактори (причини), які об'єктивно визначають значення вказаної змінної. Тут пролягає нечітка межа між задачами виведення моделей та задачами відкриття знань в даних. Якщо разом з отриманим описом моделі $p(X)$ виявлено цікаві, несподівані й статистично значущі «риси» моделі, доречно казати про відкриття знань. Навіть результати кластерного аналізу іноді можна кваліфікувати як відкриття знань, але за умови, що виявлені кластери є статистично значущими та чітко визначеними.

В результаті неакцентованого аналізу (дослідження) даних можна отримати закономірності в таких формах: послідовності, що повторюються (motifs); часто повторювані набори елементів (асоціації); структури залежностей; імплікативні зв'язки подій (можливо, нечіткі); періодичність коливань індикаторів у часі; інваріанти на сукупності значень характеристик (сталі співвідношення) тощо.

Традиційно на вхід статистичних методів подається статистична вибірка $\mathbf{X}$, тобто плаский масив, утворений зі записів-випадків (прецедентів) однакового формату. Зазвичай постулюється, що ці дані є статистичною вибіркою за схемою I.I.D. Але реальні дані можуть походити з різних «моделей». Розділити компоненти суміші в загальній ситуації важко. Класичним підходом до розділення (групування) випадків (записів) є кластерний аналіз. Сучасні практичні задачі стикаються з масивами даних, які не є класичною вибіркою. В багатьох ситуаціях між окремими записами даних є залежності у часі («післядія»). «Сирі» темпоральні дані з природного об'єкту не є I.I.D.-вибіркою. (Неможливо повернутися в минуле і повторити вимірювання.) Аналіз темпоральних даних охоплює зв'язки між різними записами даних.

Методи кластеризації

Кластерний аналіз має давню історію і продовжує розвиватися. Мета кластеризації – розбити множину прикладів (точок, записів даних) на кілька кластерів (груп, підмножин) так, щоб точки одного й того самого кластеру були значно подібніші (ближче) одна до одної, ніж точки, належні різним кластерам [23, 25, 26, 30]. (Зрозуміло, що розглядаються нетривіальні ситуації, коли задача не розв'язується сортуванням даних за значеннями одної

змінної чи за простим критерієм. Неодмінно треба розглядати одночасно кілька (багато) змінних, між якими існують залежності невідомого характеру.) Накопичено багатий арсенал методів кластеризації. Серед розмаїття цих методів можна виділити три підходи, або принципи: 1) кластеризація, основана на сукупній близькості прикладів; 2) кластеризація за принципом локальної близькості і множинного сусідства (зв'язності); 3) кластеризація на основі статистичної моделі розподілення даних [31].

Модель даних – це, зазвичай, суміш компонент, де кожна компонента описана параметрично-заданим розподіленням щільності ймовірності. Як правило, використовується нормальне розподілення. Критерієм вибору моделі є правдоподібність моделі. Відомий метод кластеризації, оснований на суміші моделей – AutoClass. Варіант систематизації методів кластеризації показано на рис. 2. Іншу (розгорнуту) класифікацію методів можна знайти в [30].

Традиційні методи кластеризації спираються на «середню» (або сукупну) відстань між точками. (Якщо точки розташовано у єдиному просторі, сукупна відстань обчислюється відносно центру кластеру.) Відстань є оберненою мірою щодо близькості; аналогічно, розбіжність обернена до подібності. У випадку категорних змінних аналітик задає матрицю відстані (розбіжності) для пар точок. Мабуть найбільш популярним методом, базованим на середній відстані, є відомий K-means. Цей метод, отримавши «ззовні» кількість кластерів, ітеративно повторює корекцію кластерів, чергуючи два кроки: 1) кожна точка даних присвоюється (приписується) тому кластеру, центр якого розташований найближче до точки; 2) для кожного кластеру обчислюється новий центр, використовуючи сукупність точок цього кластеру. Робота завершується, коли припиняється перерозподіл точок між кластерами. Достоїнство методу K-means –економічність обчислень (не треба обчислювати відстані для пар точок даних). Проте цей метод не дає задовільного результату у складних ситуаціях. Можна узагальнити принцип роботи методу K-means і розширити сферу застосування, відмовившись від обчислення метричних центрів кластерів, а замість центру використовувати один з членів кластеру (так, як це робиться у методі K-medoids). Тоді можна взагалі працювати без метричного простору прикладів і спиратися на попарно задану величину розбіжності прикладів.

Одні методи потребують, щоб кількість кластерів була задана на вході. Інші, більш гнучкі методи мають розбити дані на стільки кластерів, скільки їх «об'єктивно викристалізовується» згідно розташування точок. Зрозуміло, що аналітику ціка-

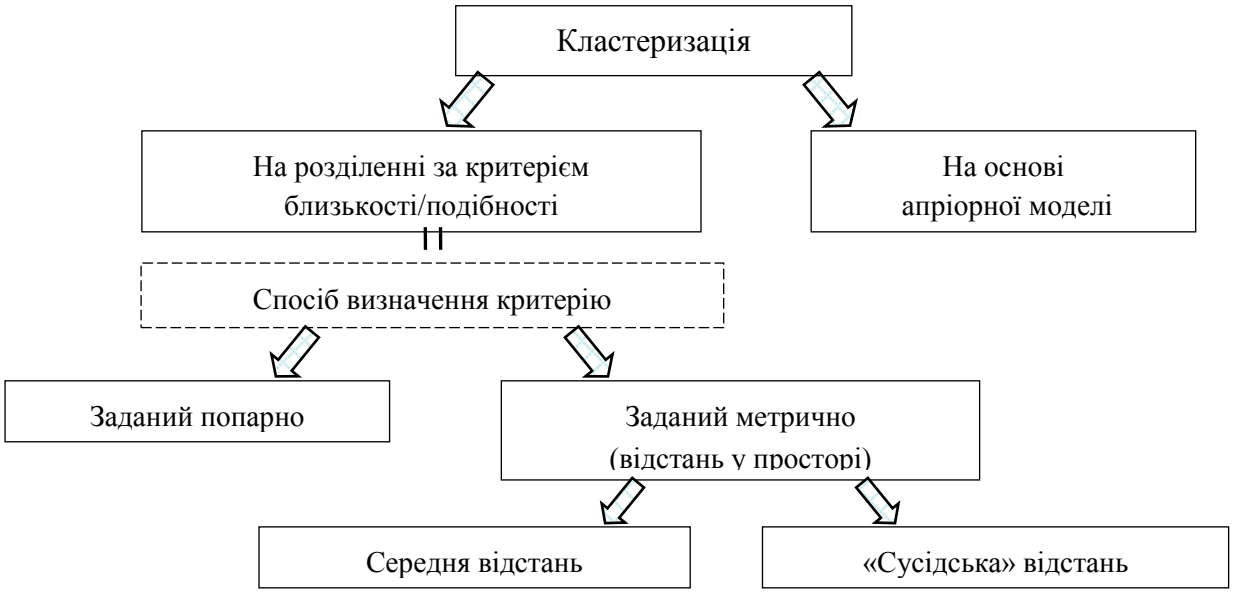


Рис. 2. Принципи кластеризації

ві такі кластери, які відображають об'єктивне групування (стратифікацію) прикладів, тобто відображають «контури» джерел даних (об'єктів). Відтак, аналітик зацікавлений у методах, які автоматично й обґрунтовано визначають кількість кластерів в даних. Можна уникнути довільного визначення кількості кластерів, звернувшись до ієрархічної кластеризації. Тотальне застосування ієрархічного принципу («до кінця») породжує дендрограму даних, де на нижньому «поверсі» кожний кластер представлено однією точкою, а кожний наступний поверх утворений злиттям (об'єднанням) двох найближчих кластерів.

Для вибору найкращого варіанту кластеризації потрібно визначити критерій якості. Універсального ефективного критерію не створено. Оцінка валідності результатів кластеризації залишається відкритим питанням. Хоча задача кластеризації інтуїтивно зрозуміла, в загальній постановці ця задача неоднозначна і не має єдиної теорії. Без додаткових уточнень і обмежень задача кластеризації погано поставлена. Наприклад, невизначеність задачі випливає з невизначеності міри близькості (відстані), коли різні змінні мають різні одиниці виміру або навіть відносяться до різних типів. На результати кластеризації критично впливає вибір (зміна) масштабів для змінних. Якщо маємо змінні різних типів, то взагалі немає єдиного простору змінних. Задавати величину відстані (розбіжності) для всіх пар точок може бути практично неприйнятно (особливо для великих даних).

Класичні алгоритми кластеризації стикаються з труднощами у випадках, коли кластери не є опуклими і не розмежовуються лінійно (тим більше – коли одні кластери оточують інші). В таких ситуаціях потрібні методи, основані на локальній близькості і зв'язності. Особливо складна ситуація – коли кластери перетинаються (частково суміщені). Слабкість традиційних методів в таких ситуаціях випливає з використання сферичної або еліптичної метрики. Один з підходів, що позбавлений цих обмежень – спектральна кластеризація. Суть цього підходу полягає в тому, що ті й тільки ті пари точок, які достатньо подібні (близькі) одна до одної, поєднуються ребром. Тоді задача ставиться як розбиття графу. Для розв'язання прикладних задач (головним чином для ситуацій з кластерами нестандартних форм) фахівці розробили багато евристичних алгоритмів кластеризації. Більшість тих методів спирається на відношення сусідства. Критерій об'єднання точок у кластер спирається на колективну локальну близькість та нерозривність (гущину) кожного кластера.

Достатню гнучкість у важких умовах (викривлених, не-опуклих кластерів) показав метод само-організуючих відображень, або мап (self-organizing map) [26]. Ідея само-організуючих мап (СОМ) полягає в тому, що у багатовимірному просторі, утвореному даними, будується двовимірна «різноманітність», яка апроксимує дані. Ця різноманітність формується як грати (у відповідних координатах) з $k \times m$ точок-прототипів. Дані скануються послідовно, для кожної точки даних знаходять найближчий прототип і зсувають його, щоб наблизити до точки даних. Але зсув стримується, щоб зберігати просторову гладкість двовимірної різноманітності і відношення сусідства прототипів. Для цього в СОМ підтримується певна кількість сусідів кожного прототипу. Сусідство визначається умовою, що відстань (у координатах гратів) не перевищує заданого порогу.

Для того, щоб результати кластеризації можна було сприймати як цікаві знахідки та знання, потрібно компактно описати кластери і показати статистичну значущість виділення саме таких кластерів. Кластеризацію можна розглядати як засіб розділення суміші даних на компоненти. Але зробити це буває важко або неможливо. Функції щільності ймовірностей компонент можуть значною мірою перетинатися. Тоді для ідентифікації компонентів (кластерів) потрібні методи, основані на моделі. Але ці методи спираються на апріорні відомості про характер компонент. Навіть після вірної ідентифікації компонент суміші однозначно розділити самі дані неможливо. Тобто кожний запис (приклад) можна віднести (приписати) до кількох компонент (з різною ймовірністю).

Кластеризація на основі суміші моделей нагадує виведення моделей з прихованими (латентними) змінними. Для кластеризації даних з пропусками застосовуються техніка максимізації очікування (EM). Коли обсяг даних надто зростає, відомі методи стають практично неприйнятними. Пом'якшити проблему можна, наприклад, за рахунок фокусування ітеративного процесу обчислень, уникаючи сканування частини даних [32].

Виведення ціле-визначених моделей

В задачах цього типу аналітик вказує одну цільову змінну y (відгук) серед змінних обраного масиву. В задачах регресії цільова змінна неперервна, а в задачах класифікації та розпізнавання – дискретна. Ціле-визначена задача виводить результат (модель) у формі $y = \Phi(X)$. (Строго кажучи, оскільки маємо $y \in X$, треба писати $y = \Phi(Z)$, де $Z \subseteq X \setminus \{y\}$.) Коли цільова змінна y дискретна, модель вигляду $y = \Phi(Z)$ називають «дискримінативною» (на противагу «генеративній» моделі $p(X)$). В разі дискретної цільової змінної модель може мати форму $p(y \mid X)$. Тоді зазвичай додається ще правило рішення $\hat{y} = \arg\max\{p(y)\}$.

Якщо метою є тільки вироблення прогнозу значення цільової змінної, то опис $\Phi(\cdot)$ може бути алгоритмом чи процедурою (і не мати декларативної або аналітичної форми). Результат у вигляді $y = \Phi(Z)$ або $p(y \mid Z)$ є предиктивною моделлю по формі, тобто у слабкому сенсі. Натомість предиктивними моделями у строгому сенсі є такі, які добре узагальнюють залежності і зберігають адекватність «на відстані» від точок оброблених даних, тобто достатньо точно екстраполюють залежності. Вимога до таких моделей – точні прогнози в усьому просторі (на всьому діапазоні) застосування.

Оскільки йдеться про аналіз великих даних, то є сенс систематизувати методи виведення ціле-визначених моделей згідно режиму використання даних. З цієї точки зору всі методи розділяємо на два роди: 1) рід «А» – «модельні» методи; 2) рід «Б» – «відкриті процедури» (методи, «прив'язані до даних»). Методи роду «А» надають завершений компактний загальний опис (модель у явному вигляді) $y = \Phi(Z)$. Загальна модель виводиться з даних один раз і надалі зберігається та застосовується «окремо» від тих даних. Для отримання прогнозу за допомогою такої моделі потрібно задати на вхід опису $\hat{y} = \Phi(Z)$ тільки значення «аргументу» (значення предикторів) Z_0. Натомість для отримання прогнозу із застосуванням методу роду «Б» використовується процедура вигляду $\hat{y} = \Phi(\mathbf{yZ}, Z_0)$, і потрібно задавати не тільки значення «аргументу» Z_0, але й кожний раз використовувати всі дані $\mathbf{yZ}$, на які спирається метод. Кожний раз дані обробляються заново. Методи роду «Б» не продукують моделі, а тільки обчислюють окремі значення $\hat{y}$ прямо з даних.

Методи роду «А» поділяються на ті, що виводять явну компактну модель $y = \mathrm{F}(Z)$ у декларативній або аналітичній формі (родина «А1»), та на ті, що надають лише алгоритмічний засіб $\hat{y} := \Phi(Z)$ для обчислення прогнозу, виходячи з значення «аргументу» (родина «А2»). Між методами родин «А1» і «А2» позиціонуються проміжні варіанти, коли маємо кілька явних моделей відповідно для кількох секторів (сегментів) простору значень «аргументу». В свою чергу, методи роду «Б» можуть використовувати спільну процедуру «налаштування», яка виконується один раз, так що її результати (параметри) потім застосовуються багатократно для обчислення прогнозу. Спільна процедура «налаштування» також може включати відбір необхідних (значущих) факторів (предикторів, ознак). Відбір значущих предикторів (регресорів) виконується майже всіма методами. Для багатьох прикладних задач (класифікації, розпізнавання) перед власне побудовою моделі формуються «ознаки», тобто нові змінні (підвищеного рівня порівняно з заданими на вході).

Створено великий арсенал методів виведення ціле-визначених моделей (зокрема, класифікації та регресії) [215-27].

Багато методів регресійного аналізу намагаються відтворити функцію $y = \mathrm{F}(X)$, незважаючи на те, що емпірична залежність не є однозначною функцією (внаслідок недоступності деяких факторів та завдяки домішку «гамору» в даних). Втім, іноді можна було би точно описати дані однозначною функцією $y = \mathrm{F}(X)$, але це означало би відтворювати випадковий гамір. Така модель не буде адекватною. Історично першим методом була лінійна регресія, яка виводить модель вигляду $\mathrm{E}(y \mid X) = b_0 + \mathbf{B}^{\mathrm{T}} \cdot \mathbf{X}$. Критерієм якості моделі (для налаштування коефіцієнтів $\mathbf{B}$) є мінімум суми квадратів відхилень (помилок). Зрозуміло, що у більшості практичних задач лінійна модель не дасть задовільного результату. Було пропоноване багато варіантів нелінійної регресії. Залучення все більш складних і гнучких форм залежності (наприклад, поліномів високого ступеня) дозволяє мінімізувати відхилення даних від прогнозу моделі. Але висока гнучкість (адаптивність) моделі веде до синдрому гіпер-специфікації, тобто «натяжки» (overfitting), з яким треба боротися. Наприклад, нехай для відгуку y маємо лише один регресор x. В такому разі будь-які дані **xy** (коли немає двох записів з однаковими значеннями змінної x) можна абсолютно точно описати моделлю $y = = c \cdot \sin(a + bx)$. Для цього налаштовуються три параметри a, b, c (завважте, значення b може бути великим). Але ясно, що така модель (будучи «точною» з точки зору оброблених даних) буде катастрофічно неадекватною для нових даних, навіть з того самого джерела. (Цей приклад також показує, що номінальна кількість параметрів далеко не завжди вірно характеризує складність моделі.)

Найбільш популярними способами стримування гіпер-специфікації є кросвалідація, регуляризація та процедури на основі бутстрепінгу [26, 27]. З точки зору аналітики критичним питанням виведення ціле-визначених моделей є підбір значущих предикторів (коваріат). Мабуть найбільш робастна процедура підбору предикторів (серед традиційних) – це двохетапна процедура: на першому етапі предиктори послідовно включаються в модель, а на другому – виключаються. Проте повного розв'язання ця задача знаходить тільки в апараті каузальних мереж, оскільки предиктори часто є умовно-інформативними [1]. Найбільш популярними критеріями якості моделі (для підбору складу предикторів) є *AIC* (або C_p) та *BIC* [27–29]. Інший підхід полягає у тому, щоб обмежувати величину коефіцієнтів регресії («стягувати» їх до нуля). Гребенева регресія та «Лассо» [26, 27, 29] включають в постановку задачі оптимізації моделі штрафи на розмір коефіцієнтів. Але гребенева регресія має тенденцію зменшувати коефіцієнти, але не видаляти терми з моделі. Натомість постановка задачі «Лассо» стимулює жорсткий відбір (відсів) підмножини предикторів. Тому «Лассо» дає простішу модель.

Зазвичай немає підстав розраховувати на те, що «істинна» модель має якусь аналітичну форму. Взагалі, апріорне визначення класу моделі часто виглядає волюнтаристським й непереконливим. Тому цілком закономірно виникла ідея іншого підходу до розв'язання ціле-визначених задач. Замість побудови єдиного математичного опису $\hat{y} = \mathrm{F}(X)$ відтворюють локальну залежність в межах ареалів (сегментів, ділянок) простору значень предикторів. До таких методів (родина «A1-Loc»), належать сплайни регресії. Аби локальні функції регресії з'єднувалися без розривів і зламів, достатньо застосувати кубічні сплайни [26, 29].

Протягом останніх 20-30-ти років було розроблено багато адаптивних методів відтворення ціле-визначених моделей, які вдаються до нових способів формування моделі. Виникли методи, що структурують (сегментують) простір предикторів, тобто розбивають простір змінних на ареали (квазі-прямокутники, квазі-паралепіпеди) відповідно до локальної поведінки залежності. Популярним способом адаптивної сегментації простору стала побудова дерев (дерева регресії, методи MARS, boosting tree, bagging, випадкові дерева) [26, 29]. Ці методи ієрархічно та ітеративно обирають оптимальне розбиття

підмножини даних на кожному кроці так, щоб сегменти простору охоплювали дані з близькими значеннями цільової змінної. Такі прості методи, як CART, C4.5, C5.0, утворюють розгалуження дерева, обравши чергову змінну та її певне значення. В кожному листі дерева змінна y визначається просто (наприклад, це відповідна константа). Для задачі регресії таке рішення породжує проблему – відсутність гладкості. Ця проблема дерев долається більш витонченими методами, наприклад, MARS (багатовимірні адаптивні регресійні сплайни). Метод MARS формує модель з «однобічних» лінійних базових функцій, які мають вигляд $b \cdot \max\{0,(X_j - t)\}$ та $b \cdot \max\{0,(t - X_j)\}$. Модель виводиться як сума обраних базових функцій з налаштованими параметрами t та b. Навіть такі прості базові функції дозволяють відтворювати нелінійні залежності. В разі недостатньої точності в модель додаються добутки базових функцій. За такою процедурою можна отримати дуже складну модель, тому на заключному етапі виведення виконується зворотній процес «підрізання» (спрощення) дерева. Деякі терми моделі видаляються, але так, щоб відхилення моделі від даних зростало на мінімальну величину. Оптимальний «розмір» моделі визначається за допомогою узагальненої крос-валідації [26, 27, 29]. Вдосконаленням методу CART є «ієрархічна суміш експертів». Додаткова адаптивність досягається за рахунок того, що розгалуження дерева утворюють згідно (лінійних) комбінацій змінних (тому «нарізка» простору – не прямокутна).

Хоча виведення дерева не породжує єдиної компактної моделі (у традиційному сенсі), але обґрунтована сегментація простору змінних придатна для інтерпретації та візуалізації і надає певне знання.

Якщо пріоритетом аналітика є точність предикції чи класифікації, то застосовують ансамблі моделей. Виводять набір «моделей-дерев» з квазі-вибірок, отриманих за допомогою бутстрепінгу. Для розгалужень в кожному дереві черговий предиктор обирають стохастично (щоб зменшити кореляцію та ухил). Прогноз обчислюється як середнє значення для всіх дерев (в задачах регресії) або обирається більшістю «голосів» (в задачах класифікації). Це дозволяє знизити дисперсію прогнозу. Але перехід від однієї моделі до ансамблю означає втрату наочності та «зрозумілості».

Протягом останніх 15 років серед методів класифікації значної популярності набув напрямок support vector machine (SVM) [25, 29]. Ці методи є розвитком ідеї класифікатора з максимальними полями. Такі методи спрямовані на побудову подільної (дискримінативної, сепараторної) поверхні або лінії, такої, щоб вона вірно розділяла всі точки на класи і, крім того, щоб відстань від сепараторної поверхні (лінії) до найближчих точок була якнайбільшою. Обчислювальна перевага методів виведення класифікаторів за критерієм максимальних полів випливає з того, що в задачі оптимізації враховуються тільки точки, що лежать на полях (а не всі). Для моделі support vector classifier вимога коректної класифікації всіх точок (даних) послаблюється (тому їх звуть класифікаторами з «м'якими» полями). Тобто достатньо вірно класифікувати не всі, а лише переважну більшість точок, але натомість треба додатково забезпечити робастність класифікації шляхом збільшення полів та спрощення (згладжування) поверхні сепарації. Точність класифікації моделями SVM підвищується за рахунок побудови нелінійної сепараторної поверхні з використанням кернелу (ядра).

Переглянемо методи, «прив'язані до даних», тобто методи роду «Б». В літературі їх іноді називають «базованими на пам'яті» (“memory-based”). Зрозуміло, що прогноз має спиратися на ближчі точки даних. Найпростіший спосіб використання принципу локальності – метод найближчого сусіда: для оцінки відгуку $\hat{y}$ в заданій точці X_0 береться значення y з того запису, у якому значення X є найближчим до заданого ($X \approx X_0$). Завважимо, що хоча цей метод не передбачає обчислень, у разі використання великих обсягів даних пошук ближчих сусідів може потребувати значних витрат. Уявімо, що

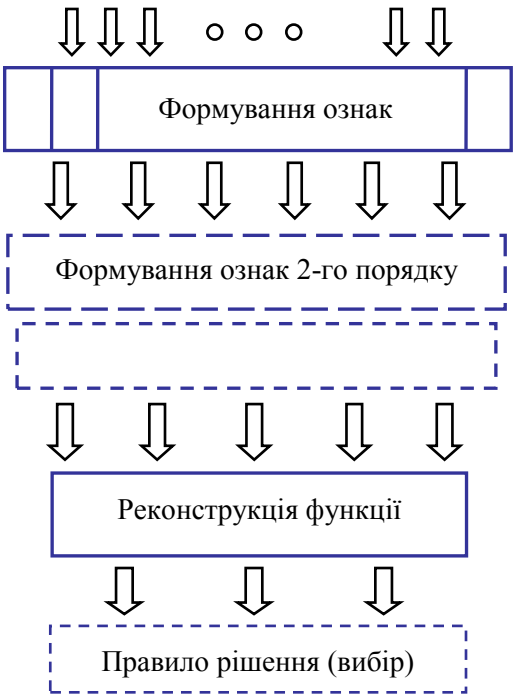


Рис. 3. Схема нейронної мережі для глибокого навчання

користувача цікавить прогноз в точці X_0, для якої є відповідний (i–й) запис даних (тобто маємо $X_i = X_0$). З огляду на загамованість даних значення цільової змінної з i–го запису даних далеко не завжди є кращим прогнозом. Вдосконалити метод найближчого сусіда (підвищити робастність) можна за рахунок того, що прогноз обчислюється як середнє значення y на основі кількох ближчих точок даних (з вагами). Ще гнучкіший метод – непараметрична регресія, – використовує згладжування залежності навколо X_0 за допомогою кернела. Явну модель не виводять (але на попередньому етапі виконується настройка параметрів, зокрема, «вікна» кернелу). Кожна чергова предикція потребує значних обчислень на основ даних (модель існує віртуально).

Локальна лінійна регресія поєднує традиційну регресію з принципом локального непараметричного згладжування. При цьому замість єдиної моделі настроюється функція для кожної цільової точки. Задача ставиться наступним чином [27]:

$$\hat{\beta}(x_0) = \underset{\beta}{\operatorname{argmin}} \sum_{i=1}^{n} K(x_0, x_i)(y_i - x_i\beta)^2 ,$$

де $K(\cdot,\cdot)$ – функція кернелу, а n – кількість записів даних. Хоча така постановка схожа на звичайну регресію, але модель (в строгому сенсі) не будується. Техніка лінійної регресії використовується тут для того, щоб уникнути зміщення (викривлення) прогнозів на краях діапазону даних. (Таке зміщення є типовим синдромом звичайного непараметричного згладжування.) Для того, щоб обчислення прогнозу не вимагало використання всієї вибірки даних **yZ**, можна використати функцію кернелу без «хвостів» або попередньо настроювати параметри «вікна» кернелу для окремих секторів простору. Але ефект такого вдосконалення може бути незначний, бо все одно треба переглядати всі дані, щоб відібрати частину даних, яка буде оброблена.

Техніку локальної лінійної регресії можна застосувати також в режимі виведення явної аналітичної моделі, але в локальному варіанті (родина «A1-Loc»). Тоді можна не повторювати обробку всіх даних кожний раз, а натомість зберігати набір локальний (частинних) моделей. В кожному ареалі предикторів обирається «представницька» точка, і модель, виведена для цієї точки, застосовується для всього ареалу. Для втілення такого режиму потрібно виконати розбивку простору предикторів на ареали (сегменти) і підтримувати засіб вибору відповідної локальної моделі.

Для розв'язання ціле-визначених задач, призначених, перш за все, для розпізнавання та класифікації, спільнота комп'ютерників давно розвиває техніку так званих нейронних мереж. Методи «навчання» нейронних мереж спочатку розробляли радше евристично, за аналогією до функціонування мозкових структур, яким його уявляли комп'ютерники. Тому ці методи традиційно позиціонували як один з напрямків «штучного інтелекту». Логічно віднести цей розділ досліджень і розробок до напрямку «самонавчання алгоритмів» [1]. Навчання нейронних мереж належить до методів родини «A2» (див. вище). Нейронні мережі продукують радше вміння, а не знання. Нейронні мережі є багато-параметричними статистичними

моделями, які мають кілька рівнів, де застосовуються лінійні перетворення, нелінійні функції та порогові функції [25–27]. З вхідних змінних формують інформативніші (синтетичні) «ознаки», з яких вже будується модель. Модель може включати багато рівнів. По-суті, техніка навчання нейронної мережі є варіантом виконання градієнтної оптимізації. Параметри конструкції «моделі» (кількість рівнів, блоків, ознак,..) іноді задаються, але можуть налаштовуватися за допомогою кросвалідації або підбиратися за допомогою експериментів.

Схематично конструкція нейронної мережі показана на рис. 3 (але треба мати на увазі, що сусідні рівні поєднані перехресними зв'язками). Кожний наступний рівень конструкції підвищує рівень абстракції (узагальнення) інформації. Навчання здійснюється як налаштування багатьох (до мільйонів) вагових коефіцієнтів моделі.

Варіант нейронної мережі може будуватися наступним чином. Кожна ознака формується за допомогою функції вигляду

$$z_i = \sigma(\alpha_{i,0} + \alpha_{i,1} \cdot x_1 + ... + \alpha_{i,m} \cdot x_m),$$

де $x_1, x_2, ..$ – вхідні сигнали, $\sigma(\cdot)$ – функція активації, наприклад, порогова або сигмоїдна (згладжена версія).

Вихідний результат виробляється за допомогою функції вигляду

$$y_k = g_k(\beta_{k,0} + \beta_{k,1} \cdot z_1 + \beta_{k,2} \cdot z_2 + ...),$$

де $g_k(\cdot)$– нелінійна функція, наприклад, подібна до $\sigma(\cdot)$, але з вільними параметрами, які налаштовуються. Для моделей класифікації в ролі $g_k(\cdot)$ може використовуватися функція softmax.

Коли мережа має кілька виходів, додається ще правило рішення (обрання). Навчання полягає у підборі коефіцієнтів $\alpha_{i,j}$ та $\beta_{k,j}$ з метою мінімізувати середньоквадратичне відхилення прогнозу від фактичних значень y_i.

Впродовж останнього десятиріччя у руслі нейронних мереж виокремилися нова хвиля розробок під назвою «глибоке навчання» [25–27, 33]. Ці засоби відрізняються від попередніх поколінь нейронних мереж більшою кількості рівнів конструкції та ускладненням форм перетворень. На старті процесу глибокого навчання задано «каркас» моделі, який розрахований на певний клас прикладних задач. Розрізняють кілька різновидів моделей глибокого навчання. «Конволюційні» мережі пристосовані для обробки образів, відео і мовлення, а «рекурентні» мережі – для послідовних даних [33]. В «конволюційних» мережах чергуються шари (рівні) згортки та злиття [27].

Фактори, що правдоподібно забезпечують успішність глибокого навчання (у відповідних впровадженнях), названо в [1]. В задачах глибокого навчання кількість вхідних змінних (предикторів) велика, але вони «дрібні». Сусідні змінні взаємно тісно корельовані.

Аналіз темпоральних даних

Зібрання даних з темпоральною прив'язкою прикладів та характеристик інтенсивно накопичуються. Спектр застосувань методів обробки й аналізу темпоральних та просторово-темпоральних типів даних надзвичайно широкий. Він охоплює такі різноманітні сфери, як біологія, медицина, астрофізика, економетрика, інтелектуалізація роботів і багато іншого. В формі темпоральних даних фіксуються фінансові та біржові індекси, аудіо- та відеозаписи, транзакції через систему тощо. Розв'язання прикладних проблем може потребувати цілого комплексу методів і технологій. Методи аналізу темпоральних даних все більше диференціюються й спеціалізуються відповідно до прикладних задач. Тут доцільно розглянути тільки базові задачі й методи.

Взагалі, в багатьох ситуаціях аналітик стикається з даними, об'єктивно вбудованими в певну просторово-часову структуру. Дані мають невід'ємні координати виміру. Спеціальним (одновимірним) випадком просторової структури є послідовність. Відмінність впорядкованості у часі від просторової послідовності можна пояснити за допомогою понять інерції (післядії) та спрямованості в одну сторону (у майбутнє). Завдяки часовому виміру ста-

ють практично важливими такі поняття, як форма сигналу, швидкість зміни, періодичність, спектр і т. д. З'являються вагомі підстави розділяти «корисний сигнал» та «гамір» у єдиному потоці даних.

Статистична методологія вже давно озброєна процедурами роботи з даними у формі часових рядів. (Відомі трансформації дозволяються привести такі дані до схеми I.I.D.-вибірки.) Дискретизація (у часі) неперервних процесів відбувається вже на етапі вимірювання. Багато методів розв'язання типових задач аналізу, розроблених для традиційних даних (зокрема, класифікація, кластеризація), були поширені на темпоральні дані за допомогою трансформацій та спрощення даних. Наприклад, виділяють фрагменти часового ряду і трактують їх як ознаки в традиційних задачах. Традиційні підходи до аналізу сигналів та неперервних процесів використовують різні варіанти згортки і фільтрації. В епоху великих даних попередня обробка темпоральних даних залучає широкий набір простих й практичних процедур – «нарізка» послідовності на фрагменти, масштабування, агрегація (спрощення, зменшення точності і деталізації). «Нарізка» є типовим засобом сформувати вибірку випадків з єдиного потоку даних. Іноді вибір варіанта нарізки на фрагменти має вирішальний вплив на результати подальшого аналізу, причому невідомо, як треба «вірно» нарізати дані. Задачі аналізу часових рядів залучають спеціальну відповідну техніку, зокрема, авторегресію (AR), moving average (MA), приховані марковські ланцюги (HMM), DTW [25, 34] і т. д.

Задача сегментації темпоральних даних та часових рядів широко розповсюджена і має варіанти з дуже різними постановками відповідно до прикладних цілей. Якщо йдеться про аналіз об'єктивного процесу (фізичного, економічного і т. д.), то мета може полягати в тому, щоб знайти фрагменти (попередньо невідомої довжини), які характеризуються стаціонарністю або певною динамікою зміни форми сигналу у часі. В роботі [34] головною метою сегментації часових рядів названо апроксимацію й зниження розмірності. Тому сегменти мають бути типовими (а їх номенклатура – мінімальною), щоб кожний тип сегменту описувався простою моделлю. Якщо маємо задачу розпізнавання аудіо мови, то такі показники, як швидкість, інерція, стаціонарність і тренд будуть несуттєвими. Але спрямованість послідовності даних у часі не можна відкинути. Для розпізнавання аудіо мови потрібно розділити ряд даних на фрагменти, які відповідають словам. Тоді ряд даних трактується як ланцюжок варіантів стандартних паттернів (фрагменти є репрезентантами класів еквівалентності). Така задача сегментації даних переплітається з класифікацією. Натомість якщо поставити мету розшифрувати аудіозапис невідомої мови, то спочатку, мабуть, треба застосувати кластеризацію та пошук motifs.

Взагалі, можна розрізняти типи темпоральних даних згідно природи генераторів даних. Типовими генераторами даних є: 1) природні процеси, 2) ергатичні системи (або цілеспрямовані об'єкти), 3) бібліотеки «паттернів» та записів. Деякі генератори даних останнього типу породжують такі послідовності символів, що для них вибір початку і кінця послідовності є питанням домовленості. Такі дані можна трактувати як одновимірний варіант просторових даних. Методи аналізу символьних послідовностей і виявлення аномалій в них оглянуто в [35].

Відомо кілька постановок задачі виявлення аномалій в послідовностях. Перша: виявити аномальний фрагмент в довгій суцільній послідовності. Друга: серед багатьох послідовностей виявити таку, яка значно відрізняється від решти. Третя: знайти послідовність, що містить деякий фрагмент (паттерн), який повторюється в цій послідовності значно частіше від очікуваного («нормального» чи середнього) рівня. В разі, коли розглядаються дані, генеровані природними процесами чи ергатичними системами, актуальною є задача прогнозу, тобто передбачення кількох найближчих майбутніх значень. Для цього застосовуються такі традиційні методи, як авторегресія, приховані марковські ланцюги тощо.

Задача виявлення точок змін є характерною саме для аналізу темпоральних даних (перш за все аналізу даних з природних процесів). Ця задача є різновидом задачі сегментації, сформульованим в інших поняттях. Точка зміни трактується як грубий, раптовий, різкий (швидкий), крутий перехід процесу від одної поведінки до іншої. Наприклад, зміну можна розуміти як перехід від одного стаціонарного режиму до іншого («переключення»). Створено багато методів виділення точок зміни [36–39]. В економетричному аналізі так звані структурні зміни («злами») розуміються як моменти, коли різко змінюються коефіцієнти лінійної залежності цільового часового ряду від інших (паралельних) часових рядів. Проблему виявлення точок зміни іноді формулюють як задачу вибору кращої статистичної моделі даних [36]. При цьому кожний фрагмент даних між двома точками зміни має бути достатньо однорідним, щоб добре описуватися простою локальною моделлю. Мета цієї задачі – оптимізація «сукупної» моделі за трьома вимогами: максимізація точності відтворення даних, спрощення всіх локальних моделей для фрагментів даних і зменшення кількості точок зміни.

Виявлення точок зміни є однією з задач відкриття знань. На перший погляд може здатися, що результат у формі точок зміни відображає лише окремі (поодинокі) події, тобто не має характеру регулярності, повторюваності, закономірності і узагальнення, що зазвичай асоціюється з поняттям знань. Насправді ця начебто суперечливість є оманливою, бо точка зміни – це демаркатор (кордон) між двома виявленими й локалізованими регулярними послідовностями (режимами). Тобто за «точкою» стоять дві регулярності (закономірності).

Дещо спрощуючи, можна поділити базові методи аналізу темпоральних даних на «феноменологічний» аналіз та «пізнавальний» аналіз. «Феноменологічний» аналіз виявляє, розпізнає, порівнює, підраховує, групує та типізує різні паттерни в даних. Такий аналіз оперує з великими зібраннями окремих послідовностей. Виявлені паттерни можуть бути використані як ознаки для задач класифікації, кластеризації, розпізнавання та пошуку. Задача виявлення типових фрагментів (motifs) полягає в тому, щоб в довгій послідовності знайти короткі послідовності, які найчастіше повторюються. Якщо аналізуються дані у формі суцільного ряду спостережень за об'єктом (середовищем), до поверхового аналізу можна віднести такі задачі, як виявлення періодичності, трендів, динамічних аномалій тощо [34, 40]. До «пізнавального» аналізу відносимо задачі та методи, які допомагають аналітику зрозуміти механізм розвитку процесу, знайти «неявні» об'єктивні зв'язки і закони, що керують процесом. Для цього треба аналізувати процес разом з його оточенням, тобто системно аналізувати багатовимірні ряди даних. Підходящим апаратом для цього, зокрема, є динамічні каузальні моделі. Методи їх виведення з даних узагальнюють такі традиційні підходи, як авторегресія. Виявлення каузальної структури зв'язків дає змогу прогнозувати наслідки втручання в об'єкт (ефект керування). Якщо є підстави вважати, що кілька рядів даних формуються на основі гармонічного сигналу з єдиного джерела, то перед тим, як приступити до виявлення зв'язків між рядами, бажано виділити ту спільну гармоніку й «очистити дані». Припустимо, частота (або період) гіпотетичної гармоніки відома, але невідомий зсув фази. Тоді для кожного ряду можна ідентифікувати модель за допомогою лінійної регресії, включивши в набір предикторів дві фіктивні змінні – $\sin(t \cdot x)$ та $\cos(t \cdot x)$. Отримавши коефіцієнти регресії, можна перетворити дві такі гармоніки на одну (з відповідним зсувом фази).

Із задачами й методами масованої переробки просторово-часових даних можна ознайомитися в [41]. Розширюється практика використання даних з просторовою структурою. Аналіз тривимірних та двовимірних даних застосовується у біології (протеоміка), медичній діагностиці, геофізиці тощо. Для деяких задач більш зручно працювати з не-метричними структурами (графовими, топологічними).

Виведення генеративних моделей

До цього роду задач і методів відносимо виведення (з даних) таких моделей, конструкцій і описів, які компактно описують сукупність даних, а іноді описують механізм породження (генерації) цих даних. Для такої задачі аналітик визначає набір змінних, без акценту на жодній з них. Виведення генеративних моделей охоплює вельми різноманітний арсенал методів і форм моделей. Результати (моделі) бувають різної точності та різного ступеня спрощення. Механізм генерації даних можна розуміти як зручний алгоритмічний опис процесу породження даних, еквівалентних (за статистичними характеристиками) фактичним даним, поданим на вхід методу. Натомість виведення генеративної моделі у сильному сенсі має більш амбітну мету – вивести таку модель, яка відображає структуру реального механізму генерації даних в об'єкті. До таких «структурно-адекватних» моделей належать каузальні мережі.

До методів виведення генеративних моделей (в їх широкому розумінні) можна віднести аналіз головних компонент (PCA), аналіз незалежних компонент (ICA), факторний аналіз, виявлення асоціацій і навіть (з деякими умовами) кластеризацію. (До речі, задачу характеристики сумісного розподілення ймовірностей $p(X)$ іноді редукують до пошуку «піків» й інтервалів великих значень ймовірності $p(X)$, а це нагадує кластеризацію.) В той час як PCA та класичний факторний аналіз обмежуються припущеннями лінійності та нормальності [25, 26, 29], метод ICA працює з прихованими факторами, розподіленими довільно. Відомий також нелінійний варіант ICA [42]. Не обмежені припущенням лінійності методи принципових кривих та принципових поверхонь, які шукають апроксимацію для розподілу даних у просторі [26]. Принципова крива проходить через «середні» точки прилеглих скупчень точок даних. Таких кривих можна знайти безліч, але задача розв'язується завдяки вимогам гладкості кривої. Розв'язок шукається ітеративною процедурою, яка уточнює криву та перерозподіляє точки даних між «відповідальними» за них точками на кривій. Аналогічно, мета побудови принципової поверхні – знайти двовимірну апроксимацію для даних. Побудова принципової поверхні дуже подібна до само-організуючих мап (COM). Різниця в тому, що метод COM знаходить невелику кількість точок-прототипів, а метод принципових поверхонь використовує окремий прототип для кожної точки даних [26].

Задача виявлення правил асоціації для дискретних даних була відповіддю на практичні потреби (виявлення типових кошиків покупок). Найвідоміший метод – алгоритм Apriori; він застосовує ідею покрокового збільшення довжини k асоціацій [15, 23, 26]. На старті список пропозицій складається з окремих змінних ($k=1$), а далі ітеративна процедура працює за наступною схемою:

– зібрати пропозиції довжиною $k+1$, такі, що кожний фрагмент пропозиції входить у список знайдених (прийнятих) асоціацій.

– відібрати ті пропозиції, які повторюються в даних частіше заданого порогу, і додати їх списку асоціацій.

З метою розширити можливості аналізу даних та підвищити рівень експресивності результатів аналізу дослідники шукають варіанти інтеграції статистичних методів з логікою [43–45]. Один з нових варіантів поєднання ймовірнісних і логічних моделей з каузальною семантикою – реляційна логістична регресія [45]. Моделі цього виду, подібно до каузальних мереж, використовують орієнтовані зв'язки.

На роль генеративних моделей в сильному сенсі претендують каузальні мережі [1, 46–49]. Каузальні мережі структурно (ізоморфно) описують процес генерації змінних моделі, відображаючи процес розгортання відповідних характеристик в об'єкті. Якщо цей опис дійсно адекватний, то каузальна мережа придатна для прогнозування наслідків планованих дій (наприклад, виконання рішень менеджера). Водночас ці моделі є багатоцільовими, оскільки дозволяють оперативно обирати цільову змінну і адаптуватися до будь-якого формату запиту без потреби повторно виводити модель.

Каузальна мережа (КМ) описується як пара (G, Θ), де G – граф, що специфікує структуру моделі, Θ – кількісні параметри, прив'язані до G. Граф структури G зазвичай є ациклонним орграфом. Якщо модель формується з одно-орієнтованих ребер, то параметри моделі специфікуються у формі $p(y|X)$ або $y = \mathrm{F}(X)$, де X – набір безпосередніх причин для y. Згідно форми параметризації моделі виділяються кілька різновидів КМ. Баєсові мережі побудовані на основі ординарних орграфів та дискретних (категорних) змінних. (Залежності в них описуються таблицями.) Каузальні Гаусові мережі використовують лінійні залежності та нормальні розподілення змінних. Гібридні мережі використовують неперервні та дискретні змінні в рамках одної моделі. Такі мережі покликані практичними потребами аналізу біологічних та медико-біологічних даних (і взагалі, великих даних). В разі гібридних мереж необхідно обробляти суміш категорних та неперервних змінних [50, 51].

Каузальні мережі підтримують два відмінні типи задач прогнозування – «пасивну» предикцію та каузальний прогноз («активну предикцію») [46, 52]. «Пасивна» предикції – це задача, подібна до тої, яку розв'язують задачі регресії та класифікації, вона формулюється як $p(C|a,b,..)$. (Але, на відміну від моделей регресії та класифікації, змінні $C, A, B, ..$ можна оперативно обирати, причому вони можуть займати будь-які позиції в моделі.) Каузальний прогноз виражається як $p(C|do(a),b,d..)$ і дає оцінку ефекту після втручання на змінну A (тобто ефект при мусового присвоєння $A = a$). Такий прогноз обчислюється на основі локально зміненої моделі [46, 48, 53].

Для опису таких процесів у часі розроблено динамічні каузальні мережі.Особливість динамічних КМ полягає в тому, що замість кожної окремої змінної в моделі представлено серію однойменних змінних (рис. 4), які репрезентують послідовні стани відповідного показника (характеристики). Функціонування динамічної каузальної мережі (генерація даних) може продовжуватися необмежено довго, послідовно просуваючись у часі дискретними інтервалами. Тобто черговий крок функціонування динамічної КМ (просування у часі) виконується таким чином, що, наприклад, вектору змінних з індексом часу $i-2$ присвоюється значення вектору змінних з індексом часу $i-1$. Тобто всі вектора значень пересуваються на один інтервал «назад» (у минуле). Значення вектору змінних з найбільшим індексом (тобто для поточного часу) генерується згідно залежностей моделі, як у звичайних каузальних мережах.

Динамічна каузальна мережа представляє регулярну структуру залежностей багатовимірного процесу у часі (рис. 4). Такі моделі описують генерацію багатовимірних часових рядів. Динамічна каузальна мережа передбачає стаціонарність

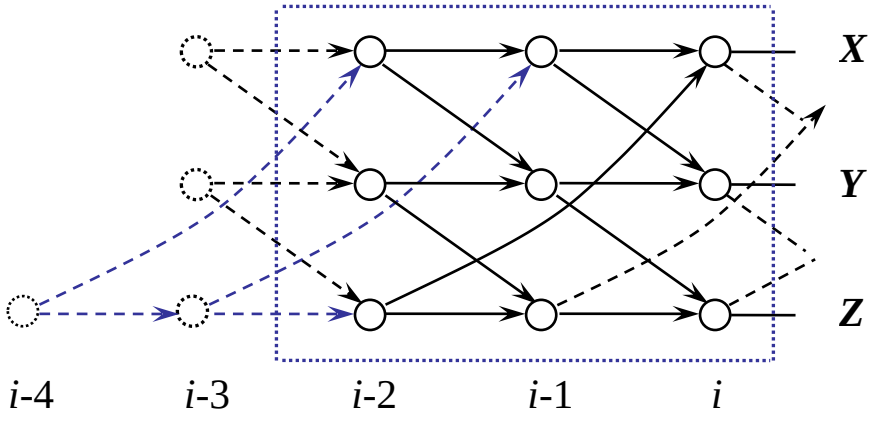


Рис. 4. Формування динамічної каузальної мережі

процесу і використовує форму векторної авторегресійної моделі [54, 55]. Модель виглядає як фрагмент часового ряду з довжиною, достатньою для явного відображення «найдовших» зв'язків. (Тобто в моделі мусить бути описано впливи з найбільшим лагом). На відміну від звичайної («статичної») каузальної мережі, в динамічний моделі структура виглядає як «клонована», оскільки зв'язки (ребра), як правило, повторюються на кожному кроці часового ряду.

Виведення каузальних мереж з емпіричних даних

Методи індуктивного виведення каузальних мереж націлені на виявлення каузальних зв'язків в ситуації, коли дані було зібрано як пасивні спостереження. (Більш того, зазвичай на вході не задається апріорних знань про структуру моделі.) Багато сучасних методів розраховані на існування прихованих (не присутніх в даних) спільних причин двох (чи більше) змінних. Далеко не для всіх зв'язків можливо розпізнати автентичний напрямок впливу. Через вказані обставини модель ідентифікується тільки як клас еквівалентності моделей, де використовуються ребра (зв'язки) різних типів, в тому числі з невизначеною орієнтацією (напрямком впливу). В типовій ситуації виведена модель використовує безпосередні зв'язки (ребра) чотирьох типів. Каузальне ребро (дуга) вигляду $X \to Y$ відображає каузальний вплив X на Y. Асоціативне ребро $U \leftrightarrow W$ позначає існування прихованої змінної (причини), що впливає рівночасно (паралельно) на U та W. Субкаузальне ребро $V \circ\!\!\to Z$ відображає два можливих варіанти: каузальний вплив або існування прихованої змінної (спільної причини). Неорієнтоване ребро $Q \circ\!\!-\!\!\circ R$ означає, що каузальний характер цього зв'язку зовсім не визначений. Кінці орієнтованого ребро $X \to Y$ називають «хвіст» та «вістря» відповідно.

Існуючі методи відтворення КМ з даних являються за своєю природою статистичними методами. Тому із зростанням довжини вибірки даних має підвищуватися точність виведеної моделі (за умови коректності методу), але ускладнюються обчислення. Збільшення ширини (багатовимірності) даних по-своєму ускладнює аналіз, але водночас створює можливість підвищити адекватність та точність моделі.

Стисло оглянемо та охарактеризуймо методи й алгоритми відтворення КМ з даних. За принципом роботи методи поділяються на «оптимізаційні» та «базовані на обмеженнях» (constraint-based) [47, 48, 50, 56]. «Оптимізаційні» методи спираються на показник «якості» моделі, який характеризує точність апроксимації даних (або предикції значень змінних) із штрафом за складність (розмірність) моделі. Зазвичай показником точності є правдоподібність моделі. Найбільш відомим «оптимізаційним» алгоритмом є GES. (Він використовує критерій BIC.) Серед методів, «базованих на обмеженнях», провідне місце посідають методи, що спираються на виявлення фактів умовної незалежності змінних (коротко – основані на незалежності). Методи, основані на незалежності, більш адаптовані до роботи з прихованими змінними, часто виграють у швидкості та мають тенденцію задовольнятися статистиками меншого формату. Відомі також гібридні методи відтворення моделей. Якщо дані розсічені («розщеплені») на окремі вибірки із скороченою номенклатурою змінних, класичні методи відтворення моделі стають не результативними. В таких ситуаціях можуть допомогти спеціальні співвідношення показників залежностей, які характерні для певних структур моделі. Приклади таких співвідношень можна знайти в [57].

Методи, основані на незалежності, виходять з припущення, що умовна незалежність (з відповідною умовою) буде чинна в даних тоді і тільки тоді, коли ця незалежність випливає зі структури генеративної моделі. В такому разі знайдена умовна незалежність для пари змінних X, Y свідчить про відсутність ребра між X та Y. Схема роботи цих методів показана на рис. 5. Найбільш витратний етап – ідентифікація безпосередніх зв'язків. Технічно, на цьому етапі для кожної пари змінних шукається умова («сепаратор»), яка забезпечує умовну взаємну незалеж-

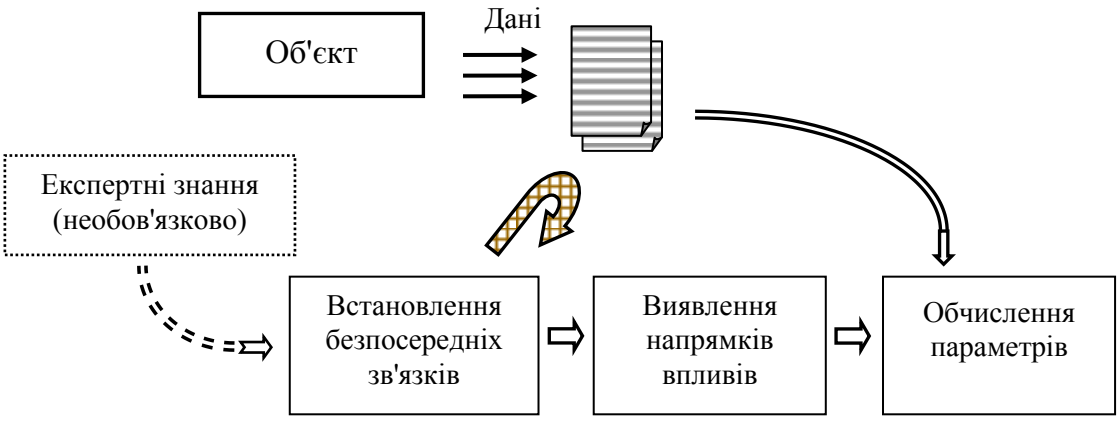


Рис. 5. Схема виведення каузальних мереж з даних

ність змінних цієї пари. Кількість тестів умовної незалежності, що виконуються на цьому етапі, може сягати десятків чи сотень тисяч.

Вказані методи здатні ідентифікувати каузальне відношення з даних тільки за наявності достатніх свідчень. Зрозуміло, що сама змінна ефекту (наслідку) та її причина мусять бути присутні в даних. Остаточно каузальний зв'язок (ребро) ідентифікується на підставі того, що знайдено квазі-інструментальну змінну, а також супутню «автономну» змінну, яка допомагає розпізнати першу змінну як квазі-інструментальну [46, 47, 58]. Тому зростання «широти» даних має сприяти виявленню каузальних зв'язків. Якщо структура мережі наближається до насиченої, то каузальні зв'язки не ідентифікуються.

Еталонними інструментами цього напрямку стали алгоритми PC та FCI [47, 50, 59]. Протягом останніх років було запропоновано низку вдосконалень цих алгоритмів та інструментів. Алгоритм FCI-Stable [60] наразі постає стандартом замість FCI. Цей алгоритм – більш робастний, в ньому усунута чутливість до порядку змінних. В алгоритмі FCI-Stable ребра видаляються з кістяка моделі тільки після виконання всіх тестів незалежності поточного рангу. Алгоритм залучає техніку тестування для мішаних даних й також використовує розпаралелювання. Показано, що в стратегію роботи алгоритмів відтворення (таких, як PC) доцільно залучити принцип пошуку, оснований на максимумі ймовірності. Перший варіант алгоритму з цим принципом (не розрахований на латентні змінні) – PC-Max. Нова версія (FCI-MAX) залучає пошук за максимумом ймовірності для орієнтації ребер. У [51] окреслено чотири нові стратегії для відтворення каузальних моделей (з допуском латентних змінних), на основі мішаних даних: FCI-MAX, MGM-FCI-MAX, MGM-FCI та MGM-CFCI. Версія FCI-MAX для орієнтації ребер перевіряє цілий набір підходящих сепараторів. Техніка тестування незалежності основана на лінійній та логістичній регресії. Найбільш універсальний підхід до тестування умовної незалежності спирається на техніку репродуктивного кернелу [25, 26, 61]. Оскільки ця техніка потребує дуже витратних обчислень, багато досліджень присвячено її вдосконаленню.

Процес виведення моделі часто стикається з обчислювальними проблемами, бо кількість можливих структур моделі є факторіально (експоненційно) великою, і кількість можливих сепараторів – також. З метою стримування комбінаторної складності алгоритмів виведення моделі були знайдені можливості фокусування пошуку сепараторів. Для цього було створено апарат локально-мінімальної сепарації та розроблено набір резолюцій індуктивного виведення [49, 62, 63]. (Ці засоби мають логічний характер і придатні для будь-яких форм параметризації моделі.) Тим самим, запропоновано систематичний підхід до пошуку сепараторів, базований на використанні імплікацій марковських властивос-

тей каузальних мереж. Теоретичний ґрунт новацій – необхідні вимоги до членів локально-мінімального d-сепаратора. Розроблені засоби вбудовано в алгоритми серії Razor і дозволяють адаптивно оптимізувати і звужувати пошук складних мінімальних сепараторів, виходячи з знання вже знайдених «сусідніх» простих сепараторів та паттернів залежностей. Відсікаються цілі сектори простору пошуку сепараторів. Результати випробувань показали перевагу алгоритмів Razor над базовим аналогом PC за кількістю тестів (а відтак, за швидкодією) і за адекватністю відтворення каузальних зв'язків [49, 63].

Для демонстрації можливостей відтворення каузальних мереж методами, основаними на незалежності, наведемо приклад аналізу реальних даних соціального характеру (ці дані не є великими). Мета – ідентифікація факторів, які визначають вік матері при народженні першої дитини в США (за даними 70-х років 20-го століття). Спеціалісти зібрали вибірку даних з наступним набором змінних (можливих факторів): 1) професія батьків; 2) раса; 3) відсутність братів (сестер); 4) жила (чи ні) матір на фермі; 5) регіон США; 6) наявність двох дорослих у родині, де росла матір; 7) релігія; 8) паління сигарет; 9) був чи ні викидень; 10) освітній рівень матері (на час виходу заміж); 11) вік матері при народженні першої дитини. Ця задача раніше була розв'язана за допомогою алгоритму PC [47]. Тепер для цієї задачі застосовано розроблений автором алгоритм Razor-1.3. Подібно до постановки в [47], також було гіпотезовано лінійність залежностей. Але на відміну від вищезгаданої постановки не було задано жодних апріорних обмежень на структуру (і не вказувалась роль змінних). Алгоритм Razor-1.3 виводить модель в більш загальному класі – класі не-рекурсивних каузальних мереж, де розрізняються каузальні, суб-каузальні, асоціативні та неорієнтовані ребра (зв'язки) [47, 49, 58, 63]. Результат роботи (структура моделі) показано на рис. 6. «Кільця» позначають невизначений кінець ребра (він може виявитися «вістрям» або «хвостом»). Для візуальної наочності подвійними лініями зображено ті ребра, які ідентифіковано як каузальні зв'язки (тобто такі, що відображають спрямований вплив одної змінної на іншу). Отже, виявлено п'ять каузальних зв'язків. Зокрема, на вік матері при народженні першої дитини впливають освітній рівень матері та регіон проживання (‘$5 \to 11$’ та ‘$10 \to 11$’). Більшість виведених ребер є суб-каузальними та неорієнтованими, наприклад, ‘$2 \circ \to 11$’ та ‘$2 \circ - \circ 10$’ відповідно. Для отримання інформативніших результатів потрібно залучити до аналізу ширший комплект релевантних характеристик.

Завдяки гіпотезі лінійності залежностей тестування незалежності виконується просто (через частинні кореляції, які обчислюються з матриці парних коваріацій.) Але поза лінійністю (наприклад, коли змінні – номінального типу) кожний тест незалежності потребує нового сканування даних, що призводить до зростання тривалості виведення моделі.

Стисло охарактеризуємо методи відтворення динамічних каузальних мереж. Ці методи ґрунтуються на припущенні, що задані на вході часові ряди даних відображають стаціонарний векторний авторегресійний процес. Виведення каузальних моделей з багатовимірних часових рядів має тривалу передісторію. Подібні за характером моделі й методи розроблялися в руслі економетричних досліджень. Концепція каузальності для часових рядів була запропонована К. Грейнжером (Granger C.W.J.) [64, 65]. Сучасну постановку ця проблема отримала в термінах каузальних мереж [54, 56, 66]. Адаптація звичайних (статичних) методів відтворення каузальних мереж до ситуації динамічних даних ґрунтується на припущенні стаціонарності та на наступних домовленостях. Структура зв'язків повторюється на кожному інтервалі вимірювання даних. Модель виглядає як фрагмент багатовимірного ряду, але компактно презентує увесь довгий ряд (в згорнутому вигляді). Особливістю цієї задачі є регулярна присутність авторегресійних зв'язків, а також заданий темпоральний порядок в даних. Розмір (часова глибина) моделі має визначатися довжиною (глибиною) безпосереднього зв'язку з

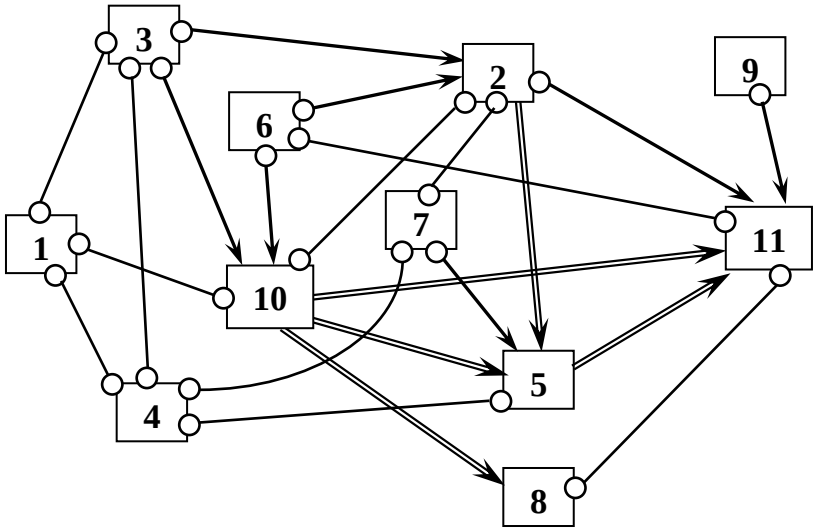


Рис. 6. Модель взаємодії гіпотетичних факторів, що впливають на вік матері при народженні першої дитини

найбільшим часовим лагом (післядією). В динамічних каузальних мережах завжди присутні додаткові «неявні» залежності, які є наслідком впливів з минулого. Наприклад, на рис. 4 ілюструється ситуація, коли найбільший лаг зв'язку дорівнює двом (зв'язок $Z_{i-2} \to X_i$). Модель охоплює три послідовні стани (обведено пунктирною рамкою) і включає зв'язки, показані суцільними дугами. Стоїть питання, як відображати впливи й залежності «з минулого». Вони не вкладаються в рамки моделі і тому показані на рис. 4 пунктирними дугами. Такі залежності розповсюджуються далі через ребра моделі і проявляються як транзитні залежності. Якщо названі впливи не відображати в моделі, то, згідно буквальної інтерпретації структури моделі, наприклад, змінні X_{i-2} та Y_{i-2} мають бути взаємонезалежні (а це не так). Якщо ж замість зв'язків, показаних на рис. 4 пунктиром, ввести в моделі додаткові дуги (ребра), які замінять впливи з минулого, то структура моделі стане нерегулярною (більш насиченою зліва), що суперечить стандартним припущенням. Але ці незручності мають синтаксичний характер. Можливі й більш суттєві проблеми. Існування прихованого часового ряду може створити феномен «нескінченно-довгого» часового лагу [54]. Це робить неможливим тестування марковських властивостей в ході виведення моделі.

Регулярність структури та відомий темпоральний порядок змінних сприяють спрощенню пошуку сепараторів і орієнтації ребер в ході виведення динамічних каузальних мереж (порівняно з звичайними мережами). Але задача ускладнюється через те, що зазвичай аналітик на знає довжини найбільшого лагу впливів.

Алгоритми SVAR-FCI та SVAR-GFCI [54] є результатом адаптації методу FCI до даних у формі багатовимірних часових рядів з прихованими змінними. Ці алгоритми припускають існування «швидких» зв'язків, тобто зв'язків між змінними в межах одного інтервалу вимірювання даних, але без утворення циклонів. Циклони не утворюються, якщо впливи протягом одного інтервалу вимірювання не встигають оббігти цикл й повернутися до стартової змінної. Якщо дані часових рядів фіксувалися з недостатньою частотою (тобто інтервал між вимірюваннями надто великий), то відтворити адекватну каузальну структуру проблематично [66]. Відомою проблемою аналізу часових рядів даних є нестаціонарність. Тоді вимірювання і інтерпретація залежностей стають суперечливими. В [67] пропонується метод виведення динамічних каузальних мереж в умовах

однієї з форм нестаціонарності (за присутності тренду).

В принципі, припущення про суцільну регулярність системи зв'язків крізь всі інтервали часу не є обов'язковим. Можна виводити модель навіть якщо зв'язки для сусідніх інтервалів відрізняються. Але структура зв'язків має бути періодичною, тобто повторюватися з зсувом на кожні k інтервалів. (Така ситуація нетипова.) Якщо величина періоду k невідома, виявити її важко. В такому разі статистики для тестування обчислюються інакше (по-суті, треба паралельно виводити кілька моделей).

Особливості застосування методів аналізу до великих даних

Достаток і рясніта доступних даних можуть спровокувати надмірний оптимізм і нехтування науковим підходом та настановами статистичного аналізу. Під враженням успіхів перших (поверхових) прикладів аналізу даних з Інтернету програмісти припустили легковажність у підході до аналізу даних і у тлумаченні результатів [68]. Номінально величезний обсяг даних не завжди означає подолання проблем скінченних вибірок даних. Дані можуть бути деформовані внаслідок селекції (особливо якщо вони зібрані за допомогою пошуку в Інтернеті). Якими би «повними» не були (не здавалися) дані, не припустимо плутати кореляцію з каузальністю.

Поява ВД впливає на вибір методів аналізу і стимулює їх розвиток [21, 22, 27, 28, 69]. З огляду на великий обсяг даних преференції надаються швидким методам (навіть якщо вони менш точні). Підсилюються стимули застосовувати розпаралелювання обчислень (особливо коли розпаралелювання узгоджується з схемою розміщення даних). ВД характерні також тим, що включають дані (змінні, атрибути) різних типів (дійсні, дискретні, ординальні, категорні), що ускладнює техніку обробки і побудови моделей. В методах, що спираються на оптимізацію квазіправдоподібності моделі, обчислювальна складність стає неприйнятною. Тому пропонують відмовитися від обчислення повного градієнту і послідовно просуватися вздовж окремих координат [12]. Коли йдеться про ВД, обчислювальну складність методів аналізу треба оцінювати дещо інакше. На відміну від традиційних застосувань, значним фактором складності стає кількість (кратність) сканувань даних. (Іноді це навіть важливіше за кількість абстрактних обчислювальних операцій.)

Уявімо, що дані є статистичною вибіркою у формі плаского масиву. Зростання тільки довжини даних (збільшення розміру вибірки) забезпечує один ефект – зростання надійності й точності результату статистичного аналізу, звуження довірчих інтервалів. Зрозуміло, що аналіз великих даних має значно амбітнішу мету – отримати повніші й змістовніші результати, глибше «зазирнути» в суть об'єкту. Для цього потрібні дані більшого формату. Зростання довжини («вишини») даних має супроводжуватися зростанням їх «ширини», тобто їх вимірності. (Дані мусять бути великими одночасно в обох «вимірах».) Отже, одна з важливих ознак великих даних (для глибокого аналізу) – багатовимірність. Багатовимірні дані, з одного боку, надають нові можливості й шанси, а з іншого – породжують проблеми для аналізу [12, 70]. Наприклад, для збереження ефективності методів найближчого сусіда необхідно, аби довжини даних зростала експоненційно швидше за вимірність даних [26, 27, 70]. Від зростання вимірності даних також потерпають методи, що спираються на техніку кернелу. Небажані ефекти особливо загострюються, коли ширина даних перевищує довжину (це характерно для даних експресії генів і взагалі біоінформатики). Коли багато змінних одночасно використовуються як ознаки, виникає синдром накопичення гамору для задач класифікації. Між «істинними» факторами та деякими не-релевантними змінними можуть випадково виникати обманні асоціації.

Коли дані вертикально- секціоновані («розщеплені») і не вдається їх синтезувати (ототожнити випадки), то аналітику залишається послаблений варіант багатовимірного аналізу. Замість аналізу на рівні випадків (прецедентів) доведеться перейти до аналізу на рівні нечітких «класів еквівалентності» прецедентів (які формуються

на підставі подібності властивостей). Зрозуміло, що зв'язки між такими класами еквівалентності напевно будуть нечіткими, «розмазаними» і малоінформативними.

Із зростанням довжини вибірки стає критичним питання вибору критерію якості моделі, бо збільшується розходження (відмінність) між вибором за критерієм BIC та критерієм AIC, а також результатом традиційного тестування гіпотез [27]. Штраф за складність моделі в BIC має додатковий (порівняно з AIC) множник $\log(n)$. Тому BIC обирає меншу (простішу) модель, ніж AIC.

Великі дані відкривають перспективи розробки нових методів. Покращуються умови для розв'язання «витончених» задач «не-прямими» методами, такими, як «навчання на чужому досвіді» [27, 28]. Ідею цього підходу можна пояснити наступним чином. Уявімо, що класичної I.I.D.-вибірки даних немає. Лише мала частка даних строго релевантна для задачі оцінки (прогнозу) характеристики. Тоді «пряму» оцінку характеристики (отриману з релевантних даних) беруть як орієнтовну, початкову. Потім вносять поправку згідно загальної закономірності, попри те, що закономірність була отримана з інших вибірок даних. Популярна нині стратегія масового тестування гіпотез за критерієм FDR також може розглядатися як втілення ідеї «навчання на чужому досвіді».

Кількість випадків (довжину масиву) не завжди припустимо тлумачити як розмір статистичної вибірки. Часто зростання масиву даних досягається шляхом об'єднання даних з різних джерел. Утворена таким чином «вибірка» даних буде неоднорідною і порушує стандартні припущення. З дещо послабленим варіантом неоднорідності даних аналітик стикається, якщо дані збиралися протягом тривалого часу, так що об'єкт змінювався (еволюціонував). Для ситуацій з можливою різнорідністю даних пропонується [21] ввести припущення інваріантності для підмножини коваріат. Цьому припущенню задовольняють каузальні коваріати. Різнорідність даних призводить до погіршення стабільності та якості результатів аналізу. Водночас різнорідність даних надає можливість вирішити корисну задачу виокремлення різних джерел даних (коли вони апріорі невідомі аналітику). Внаслідок різнорідності даних стандартна крос-валідація стає поганим інструментом оцінки предиктивної адекватності моделі (і адекватності каузального виведення). В такій ситуації краще звернутися до використання синтетичних даних та симуляції (але потрібні предметні знання).

Взагалі, якщо для розробки традиційних статистичних методів можна було прийняти зручні припущення про вибірку даних, то тепер, маючі справу з великими даними, необхідно сприймати як факт реальні механізми збору даних і відповідно обирати чи розробляти методи аналізу. Треба створювати нові методи аналізу, які будуть повніше враховувати механізм збору даних (і навіть «виправляли» і компенсували викривлення даних, наскільки це можливо). Методи аналізу даних мають вибиратися (налаштовуватися) у відповідності з особливостями збору даних. Наприклад, якщо дані були піддані селекційному зміщенню, то в залежності від обставин можливі такі рішення [53, 71]: 1) компенсувати (виправити) селекційне зміщення; 2) інтерпретувати результати аналізу інакше; 3) констатувати, що задачу неможливо розв'язати. Принципову можливість отримати коректний результат на основ «викривлених» даних можна проілюструвати на прикладі методу адекватної оцінки каузального ефекту на основі даних, що пройшли селекцію [71]. Налаштування методів аналізу на характер збору даних буде свідчити про інтеграцію комплексу досліджень «Великі дані плюс Велика Аналітика» у єдину науку.

Алгоритми й методи аналізу необхідно піддавати всебічному випробуванню у різних сценаріях, включно з «крайніми». Результати глибокого аналізу приймаються як основа для наукових узагальнень та практичних висновків тільки після кількісної оцінки невизначеності, статистичних помилок, стабільності і реплікативності результатів. (Реплікативність означає, що альтернативні аналітичні дослідження обраного об'єкту дають близькі результати [21, 69].)

Підсумки

Інтенсивний збір даних та накопичення великих даних у комп'ютерних сховищах стимулює зміни інформаційних технологій. На передній план висувається індуктивно-емпірична методологія та технології глибокого аналізу даних. Виник попит на методи, які швидко й результативно перетворять величезний масив «сирих» даних на цінну інформацію кінцевого споживання. Необхідна передумова глибокого аналізу даних – їх багатовимірність та великий обсяг. Оскільки зібрані дані зазвичай «сирі» й погано структуровані, перед власне застосуванням аналітичних методів необхідно виконати підготовку даних.

Увесь арсенал задач аналізу великих даних можна розбити на такі групи: 1) впорядкування даних (зокрема, кластеризацію); 2) виведення ціле-визначених (предиктивних) моделей; 3) дослідження та узагальнення даних (в тому числі відкриття структур, зв'язків, паттернів і закономірностей).

Спектр методів кластерного аналізу за принципом роботи можна розподілити на три підходи: кластеризація на основі сукупної близькості; кластеризація на основі локальної близькості та множинному сусідстві; кластеризація на основі статистичної моделі даних. Оскільки універсального критерію якості кластеризації не запропоновано, виділення кластерів часто є евристичним і залежить від вибору аналітика. Для здатності виділяти кластери нестандартних (не-опуклих) форм розроблено багато евристичних алгоритмів, які спираються на відношення сусідства, на колективну локальну близькість точок.

До ціле-визначених задач відноситься виведення предиктивних (дискримінативних) моделей (зокрема, регресії та класифікації), які описують цільову змінну через інші змінні (предиктори або ознаки). Використовуючи гнучкі й адаптивні форми залежності, можна мінімізувати відхилення моделі від вхідних даних. Але узгодженість моделі з вхідними даними далеко не завжди означає адекватність і здатність прогнозувати значення у впровадженнях. Втрата адекватності моделі відбувається через синдром гіпер-специфікації (over-fitting), до якого призводить зависока адаптивність і складність обраного класу моделей. Типовими засобами захисту від гіпер-специфікації є крос-валідація та регуляризація. Адекватність ціле-визначених моделей суттєво залежить від підбору значущих предикторів. Популярна тенденція – побудова моделі за допомогою дерева (дерева регресії, методи MARS, випадкові дерева). Висока адаптивність цих методів досягається завдяки тому, що простір предикторів раціонально розбивається на сегменти (ареали) відповідно до локальної поведінки залежності.

Задачі «глибокого навчання» створюють високо-адаптивні моделі для розпізнавання. Успішність «глибокого навчання» пояснюється тим, що задача високоспеціалізована, результат – лаконічний, а вхідні змінні – «пасивні» і взаємозамінні. Складність отриманих моделей виправдана тим, вхідні змінні «дрібні», а «сусідні» змінні тісно корельовані. Натомість задачі глибокого аналізу даних та відкриття знань (на відміну від «глибокого навчання») характеризуються більш невизначеною ситуацією на вході. Виявлені закономірності та «знання» є результатом «кристалізації» неявних (прихованих) відносин між різноплановими змінними. Ці два напрями різняться також характером переробки даних: перший має характер тренування, «підгонки» й оптимізації; другий має пошуково-дослідницький характер. Модель з високою предиктивною ефективністю не завжди корисна для розуміння (пояснення) об'єкту.

Розв'язання багатьох прикладних задач аналізу ВД все частіше потребує враховувати часову прив'язку записів даних та автокореляцію. Аналізуються зв'язки не тільки в межах записів, але й поміж записами. Задачі аналізу темпоральних даних вирізняються великим розмаїттям, потребують спеціалізації й комплексування методів та їх ієрархічного застосування. Ці задачі включають сегментацію, виявлення точок змін, виявлення трендів, аномалій, періодичності, спектру і т. д.

Традиційні ціле-визначені моделі прив'язані не тільки до цільової змінної,

але й до фіксованого набору предикторів. Тому неясно, як застосовувати модель, коли відомі не всі предиктори. Це питання знаходить коректне розв'язання в апараті каузальних моделей. Каузальні моделі (мережі) поєднують у собі переваги генеративних, ціле-визначених та багатоцільових моделей. Каузальні мережі здатні адекватно описати процес генерації даних в об'єкті. Каузальну мережу можна розглядати як інтегровану систему предиктивних та дискримінативних моделей. Головна перевага каузальних моделей над традиційними – вони підтримують прогнозування наслідків втручання в об'єкт (ефект керування).

Методи виведення каузальних мереж з даних за принципом роботи поділяються на «оптимізаційні» та основані на незалежності. Останні спираються на виявлення фактів умовної незалежності змінних. Ці методи більш адаптовані до роботи з латентними змінними, часто виграють у швидкості та потребують статистик меншого формату. Найбільш універсальна техніка тестування умовної незалежності – репродуктивний кернел (ядро); її недоліком є високі обчислювальні витрати. Зниження обчислювальних витрат в ході виведення каузальних мереж можна досягти, зокрема, обмежуючи комбінаторну складність виведення завдяки фокусуванню пошуку сепараторів. Для цього розроблено набір резолюцій, які дозволяють розв’язувати питання щодо присутності ребер раніше, які обґрунтовані на графовому рівні (через поняття локально-мінімального d-сепаратора та необхідні вимоги до членів локально-мінімального сепаратора).

Методи виведення каузальних мереж з емпіричних даних (які були зібрано як пасивні спостереження) мають обмеження у можливостях вичерпно розпізнавати каузальні зв'язки. Результатом виведення каузальної мережі з даних зазвичай є неповністю визначена модель, де характер (спрямування) багатьох зв'язків неоднозначний.

Можливості методів відтворення каузальних мереж продемонстровано на прикладі. Виведено модель взаємодії гіпотетичних факторів, що впливають на вік матері при народженні першої дитини. Більшість виведених зв'язків – суб-каузальні та неорієнтовані (тільки декілька каузальних). Для отримання більш інформативних результатів потрібні дані з розширеним набором релевантних характеристик.

Застосування методів аналізу до великих даних має низку особливостей. Важливим фактором обчислювальної складності методів стає кількість (кратність) сканування даних. Поява великих даних впливає на вибір методів аналізу (включаючи статистичні), і стимулює їх розвиток. Не припустимо нехтувати науковим підходом та застереженнями статистичного аналізу. Зокрема, не можна плутати кореляцію з каузальністю. Дані, зібрані за допомогою пошуку в Інтернеті, можуть бути деформовані внаслідок селекції. Серед засобів оцінки коректності методів та адекватності результатів зростає роль симуляції з використанням синтетичних даних, а також випробування у різних сценаріях. Щоб прийняти результати аналізу даних як основу для узагальнень та висновків, необхідно оцінити їх невизначеність, стабільність, реплікативність та статистичні помилки.

Єдина методологія для всього циклу життя великих даних дозволить вибирати і налаштовувати методи аналізу у відповідності з особливостями збору даних. Технології збору даних, збереження, менеджменту та пошуку будуть інтегруватися з методами глибокого аналізу даних, утворюючи нову науково-технологічну сферу «Великі дані плюс Велика Аналітика».

Література

1. Балабанов О.С. Аналітика великих даних: принципи, напрямки і задачі. *Проблеми програмування*. 2019. № 2. С. 47–68.
2. Bühlmann P., Drineas P., Kane M., van der Laan M. (eds.) Handbook of Big Data. Taylor and Francis, 2016. 456 p.
3. Mayer-Schönberger V., Cukier K. Big Data: A revolution that will transform how we live, work, and think. Boston, MA: Houghton Mifflin Harcourt, 2013. 256 p.
4. Chen C.L.P. and Zhang C.-Y. Data-intensive applications, challenges, techniques and

technologies: A survey on Big Data. *Information Sciences*. 2014. Vol. 275. P. 314–347.
5. Chen M., Mao S. and Liu Y. Big Data: A Survey. *Mobile Networks and Applications*. 2014. Vol. 19, Issue 2. P. 171–209.
6. Bhadani A. and Jothimani D. Big Data: Challenges, opportunities and realities / In.: M.K. Singh and D.G. Kumar (eds.). Effective Big Data management and opportunities for implementation. – IGI Global, Pennsylvania, USA, 2016. – [Електронний ресурс] Доступ: https://arxiv.org/pdf/1705.04928.
7. Oussous A., Benjelloun F.-Z., Lahcen A.A. and Belfkih S. Big Data technologies: A survey. *Journal of King Saud University. Computer and Information Sciences*. 2018. Vol. 30, Issue 4. P. 431–448.
8. Cao L. Data science: a comprehensive overview. *ACM Computing Surveys*. 2017. Vol. 50, N 3, Article 43, 42 p.
9. Gandomi A. and Haider M. Beyond the hype: Big data concepts, methods, and analytics. *Intern. Jour. of Information Management*. 2015. Vol. 35, N 2. P. 137–144.
10. Tsai C.-W., Lai C.-F., Chao H.-C. and Vasilakos A.V. Big data analytics: a survey. *Journal of Big Data*. 2015. Vol. 2, N 1. P. 1–32.
11. Watson H.J. Tutorial: Big Data analytics: Concepts, technologies, and applications. *Comm. of the Association for Information Systems*. 2014. Vol. 34, Article 65. P. 1247–1268.
12. Fan J., Han F. and Liu H. Challenges of Big Data analysis. *Nat. Scient. Rev*. 2014., Vol. 1, N 2. P. 293–314.
13. Franke B., Plante J.-F., Roscher R., Lee E.A., Smyth C., Hatefi A., Chen F., Gil E., Schwing A.G., Selvitella A., Hoffman M.M., Grosse R., Hendricks D. and Reid N. Statistical inference, learning and models in Big Data. *Intern. Statistical Review*. 2016. Vol. 84, N. 3. P. 371–389.
14. Zafarani R., Abbasi M.A. and Liu H. Social media mining. An introduction. Cambridge University Press, 2019. 380 p.
15. Андон Ф.И., Балабанов А.С. Выявление знаний и изыскания в базах данных: подходы, модели, методы и системы (обзор). *Проблемы программирования*. 2000. № 1–2, С. 513–526.
16. Балабанов А.С. Выделение знаний из баз данных – передовые компьютерные технологии интеллектуального анализа данных. *Математичні машини і системи*. 2001. № 1–2. С. 40–54.
17. Azzalini A. and Scarpa B. Data analysis and Data Mining: An introduction. – N.Y.: Oxford University Press, 2012. 288 p.
18. Swanson N.R. and Xiong W. Big Data analytics in economics: What have we learned so far, and where should we go from here? *Canadian J. of Economics*. 2018, Vol. 51, Issue 3. P. 695–746.
19. Graham E. and Timmermann A. Forecasting in Economics and Finance. *Annual Review of Economics*. (2016). Vol. 8. P. 81–110.
20. Weihs C. and Ickstadt K. Data Science: the impact of statistics. *Intern. J. of Data Science and Analytics*. 2018. Vol. 6. P. 189–194.
21. The role of statistics in the era of big data. Special issue of the journal: *Statistics and Probability Letters*. May 2018. Vol. 136.
22. Secchi P. On the role of statistics in the era of big data: A call for a debate. *Statistics and Probability Letters*. 2018. Vol. 136. P. 10–14.
23. Witten I.H., Eibe F., Hall M.A. (3rd ed.).Data mining: practical machine learning tools and techniques. Morgan Kaufmann, 2011. 629 p.
24. Maimon O., Rokach L. (Eds.) Data Mining and Knowledge Discovery Handbook. 2nd ed., Springer-Verlag New-York Inc., 2010. 1285 p.
25. Murphy K.P. Machine learning: a probabilistic perspective. MIT Press, Cambridge, Massachusetts, 2012. 1055 p.
26. Hastie T., Tibshirani R. and Friedman J. The elements of statistical learning. (2nd ed.). Springer. 2009. 745 p.
27. Efron B. and Hastie T. Computer age statistical inference. Cambridge University Press, 2016. 475 p.
28. Efron B. Large-scale inference. Stanford University Press, 2010. 263 p.
29. James G., Witten D., Hastie T. and Tibshirani R. An introduction to statistical learning with applications in R. Springer, N.Y., 2013. 426 p.
30. Berkhin P. A survey of clustering data mining techniques. In: Kogan J., Nicholas C., Teboulle M. (eds.). *Grouping multidimensional data*. Springer-Verlag: Berlin-Heidelberg, 2006. P. 25–71.
31. Bouveyron C., Brunet-Saumard C. Model-based clustering of high-dimensional data: A review. *Computational Statistics and Data Analysis*. 2014. Vol. 71. P. 52–78.
32. Kurban H., Jenne M. and Dalkilic M.M. Using data to build a better EM: EM* for big data. *Intern. J. of Data Science and Analytics*. 2017. Vol. 4, Issue 2. P. 83–97.
33. LeCun Y., Bengio Y., Hinton G. Deep learning. *Nature*. 2015. Vol. 521, P.436–444.

34. Esling P. and Agón C. Time-series data mining. *ACM Computing Surveys*. 2012. Vol. 45, Issue 1. P. 12–34.
35. Chandola V., Banerjee A. and Kumar V. Anomaly detection for discrete sequences: a survey. *IEEE Trans. on Knowledge and Data Eng*. (TKDE). 2012. Vol. 24, N 5. P. 823–839.
36. Truong C., Oudre L. and Vayatis N. Selective review of offline change point detection methods. [Electronic resource] URL: https://arxiv.org/abs/1801.00718.
37. Aminikhanghahi S. and Cook D.J. A survey of methods for time series change point detection. *Knowledge and Information Systems*. 2017. Vol. 51, Issue 2. P. 339–367.
38. Frick K., Munk A. and Sieling H. Multiscale change point inference. *J. Roy. Statist. Soc., ser. B*. 2014. Vol. 76, Pt. 3. P. 495–580.
39. Wang T. and Samworth R.J. High dimensional change point estimation via sparse projection. *J. Roy. Statist. Soc., ser. B*. 2018. Vol. 80, Pt. 1. P. 57–83.
40. Liao T.W. Clustering of time series data – a survey. *Pattern Recognition*. 2005. Vol. 38. P. 1857–1874.
41. Atluri G., Karpatne A. and Kumar V. Spatio-temporal Data Mining: a survey of problems and methods. *ACM Computing Surveys*. 2018. Vol. 51, Issue 4, Article N 83.
42. Lee T.-W., Girolami M., Bell A.J., Sejnowski T.J. A unifying information-theoretic framework for Independent Component Analysis. *Intern. J. Computers and Mathematics with Applications*. 2000. Vol. 39. P. 1–21.
43. Neville J. and Jensen D. Relational Dependency Networks. *Jour. of Machine Learning Res*. 2007. Vol. 8. P. 653–692.
44. De Raedt L., Kersting K., Natarajan S. and Poole D. Statistical relational artificial intelligence: Logic, probability, and computation. *Synthesis Lectures on Artificial Intelligence and Machine Learning*. 2016. Vol. 10, N 2. P.1–89.
45. Kazemi S.M., Buchman D., Kersting K., Natarajan S. and Poole D. Relational logistic regression: The directed analog of Markov logic networks. *Workshops at the Twenty-Eighth AAAI Conf. on Artificial Intelligence*. 2014. P. 41–43.
46. Pearl J. Causality: models, reasoning, and inference. Cambridge: Cambridge Univ. Press, 2000. 526 p.
47. Spirtes P., Glymour C. and Scheines R. Causation, prediction and search. New York: MIT Press, 2001. 543 p.
48. Peters J., Janzing D. and Schölkopf B. Elements of Causal Inference. Foundations and Learning Algorithms. MIT Press, Cambridge, MA, USA, 2017. 265 p.
49. Балабанов О.С. Відкриття знань в даних та каузальні моделі в аналітичних інформаційних технологіях. *Проблеми програмування*. 2017. № 3. С. 96–112.
50. Raghu V.K., Ramsey J.D., Morris A., Manatakis D.V., Sprites P., Chrysanthis P.K., Glymour C., Benos P.V. Comparison of strategies for scalable causal discovery of latent variable models from mixed data. *Intern. Jour. of Data Science and Analytics*. 2018. Vol. 6, Issue 1. P. 33–45.
51. Tsagris M., Borboudakis G., Lagani V., Tsamardinos I. Constraint-based causal discovery with mixed data. *Intern. Jour. of Data Science and Analytics*. 2018. Vol. 6, Issue 1. P. 19–30.
52. Pearl J. The seven tools of causal inference, with reflections on machine learning. *Communications of the ACM*. 2019. Vol. 62, Issue 3. P. 54–60.
53. Pearl J. and Bareinboim E. External validity: From do-calculus to transportability across populations. *Statistical Science*. 2014. Vol. 29, N 4. P. 579–595.
54. Malinsky D. and Spirtes P. Causal structure learning from multivariate time series in settings with unmeasured confounding. *Proc. of 2018 ACM SIGKDD Workshop on Causal Discovery*, August 2018, London, UK. PMLR, Vol. 92. P. 23–47.
55. Entner D. and Hoyer P.O. On causal discovery from time series data using FCI. *Proc. of the 5th European Workshop on Probabilistic graphical models*. 2010, Helsinki, Finland. P. 121–128.
56. Runge J. Causal network reconstruction from timeseries: From theoretical assumptions to practical estimation. *Chaos*. 2018. Vol. 28, paper 075310. 20 p.
57. Балабанов А.С. Верхняя граница для суммы корреляций трех индикаторов в отсутствии общего фактора. *Кибернетика и системный анализ*. 2019. № 2. С. 10–21.
58. Балабанов О.С. Від коваріацій до каузальності. Відкриття структур залежностей в даних. *Системні дослідження та інформаційні технології*. 2011. № 4. С. 104–118.
59. Colombo D., Maathuis M.H., Kalisch M. and Richardson T.S. Learning high-dimensional directed acyclic graphs with latent and selection variables. *Annals of Statistics*. 2012. Vol. 40, Issue 1. P. 294–321.

60. Colombo D., Maathuis M.H. Order-independent constraint-based causal structure learning. *Jour. of Machine Learning Research.* 2014. Vol.15. P. 3921−3962.
61. Kernel-based conditional independence test and application in causal discovery / K.Zhang, J. Peters, D. Janzing, B. Schölkopf. / *Proc. of the 27th Conf. on Uncertainty in Artificial Intelligence*, (UAI-2011). Corvallis, Oregon: AUAI Press, 2011. P. 804–813.
62. Балабанов А.С. Минимальные сепараторы в структурах зависимостей. Свойства и идентификация. *Кибернетика и системный анализ*. 2008. № 6. Р. 17–32.
63. Балабанов О.С. Відтворення каузальних мереж на основі аналізу марковських властивостей. *Математичні машини та системи*. 2016. № 1. С. 16–26.
64. Granger C.W.J. Investigating causal relations by econometric models and cross-spectral methods. *Econometrica*. 1969. Vol. 37. P. 424–459.
65. Swanson N.R. and Granger C.W.J. Impulse response functions based on a causal approach to residual orthogonalization in vector autoregressions. *J. of the American Statistical Association*. 1997. Vol. 92, N 437, P. 357–367.
66. Gong M., Zhang K., Schölkopf B., Tao D. and Geiger P. Discovering temporal causal relations from subsampled data. *Proc. of the 32nd Intern. Conf. on Machine Learning*, 2015. P. 1898–1906.
67. Malinsky D. and Spirtes P. Learning the structure of a nonstationary vector autoregression. The 22nd Intern. Conf. on Artificial Intelligence and Statistics. *Proc. of Machine Learning Research*, *PMLR*, 2019, Vol. 89. P. 2986–2994.
68. Harford T. Big data: A big mistake? *Significance*. 2014. Vol. 11, N 5. P. 14–19.
69. Bühlmann P. and van de Geer S. Statistics for high-dimensional data: Methods, theory and applications. Springer, 2011. 556 p.
70. Donoho D.L. High-dimensional data analysis: the curses and blessings of dimensionality – In: American Mathematical Society Conf. "*Math Challenges of the 21st Century*", 2000, Los Angeles. P. 1–32.
71. Bareinboim E., Tian J., Pearl J. Recovering from selection bias in causal and statistical inference. *Proc. of the 28th AAAI Conf. on Artificial Intelligence*. 2014. P. 2419–2416. (July 27–31, 2014, Québec Convention Center, Québec City, Québec, Canada).

References

1. Balabanov O.S. Big Data Analytics: principles, trends and tasks (a survey). *Problems in programming*. 2019. N 2. P. 47–68. (ISSN 1727–4907) [In Ukrainian].
2. Bühlmann P., Drineas P., Kane M., van der Laan M. (eds.) Handbook of Big Data. Taylor and Francis, 2016. 456 p.
3. Mayer-Schönberger V., Cukier K. Big Data: A revolution that will transform how we live, work, and think. Boston, MA: Houghton Mifflin Harcourt, 2013. 256 p.
4. Chen C.L.P. and Zhang C.-Y. Data-intensive applications, challenges, techniques and technologies: A survey on Big Data. *Information Sciences*. 2014. Vol. 275. P. 314–347.
5. Chen M., Mao S. and Liu Y. Big Data: A Survey. *Mobile Networks and Applications*. 2014. Vol. 19, Issue 2. P. 171–209.
6. Bhadani A. and Jothimani D. Big Data: Challenges, opportunities and realities / In.: M.K. Singh and D.G. Kumar (eds.). Effective Big Data management and opportunities for implementation. – IGI Global, Pennsylvania, USA, 2016. – [Електронний ресурс] Доступ: https://arxiv.org/pdf/1705.04928.
7. Oussous A., Benjelloun F.-Z., Lahcen A.A. and Belfkih S. Big Data technologies: A survey. *Journal of King Saud University. Computer and Information Sciences*. 2018. Vol. 30, Issue 4. P. 431–448.
8. Cao L. Data science: a comprehensive overview. *ACM Computing Surveys*. 2017. Vol. 50, N 3, Article 43, 42 p.
9. Gandomi A. and Haider M. Beyond the hype: Big data concepts, methods, and analytics. *Intern. Jour. of Information Management*. 2015. Vol. 35, N 2. P. 137–144.
10. Tsai C.-W., Lai C.-F., Chao H.-C. and Vasilakos A.V. Big data analytics: a survey. *Journal of Big Data*. 2015. Vol. 2, N 1. P. 1–32.
11. Watson H.J. Tutorial: Big Data analytics: Concepts, technologies, and applications. *Comm. of the Association for Information Systems*. 2014. Vol. 34, Article 65. P. 1247–1268.
12. Fan J., Han F. and Liu H. Challenges of Big Data analysis. *Nat. Scient. Rev*. 2014., Vol. 1, N 2. P. 293–314.
13. Franke B., Plante J.-F., Roscher R., Lee E.A., Smyth C., Hatefi A., Chen F., Gil E., Schwing A.G., Selvitella A., Hoffman M.M., Grosse R., Hendricks D. and Reid N. Statistical inference, learning and models in Big Data.

Intern. Statistical Review. 2016. Vol. 84, N. 3. P. 371–389.
14. Zafarani R., Abbasi M.A. and Liu H. Social media mining. An introduction. Cambridge University Press, 2019. 380 p.
15. Andon P.I. and Balabanov O.S. Vyjavlenie znanij i izyskanija v bazah dannyh. Podhody, modeli, metody i sistemy. *Problems in programming*. 2000. N 1–2. P. 513–526. (Kyjv, UA). [In Russian].
16. Balabanov O.S. Knowledge extraction from databases – advanced computer technologies for intellectual data analysis. *Mathematical Machines and Systems*. 2001. N 1–2. P. 40–54. [In Russian].
17. Azzalini A. and Scarpa B. Data analysis and Data Mining: An introduction. – N.Y.: Oxford University Press, 2012. 288 p.
18. Swanson N.R. and Xiong W. Big Data analytics in economics: What have we learned so far, and where should we go from here? *Canadian J. of Economics*. 2018, Vol. 51, Issue 3. P. 695–746.
19. Graham E. and Timmermann A. Forecasting in Economics and Finance. *Annual Review of Economics*. (2016). Vol. 8. P. 81–110.
20. Weihs C. and Ickstadt K. Data Science: the impact of statistics. *Intern. J. of Data Science and Analytics*. 2018. Vol. 6. P. 189–194.
21. The role of statistics in the era of big data. Special issue of the journal: *Statistics and Probability Letters*. May 2018. Vol. 136.
22. Secchi P. On the role of statistics in the era of big data: A call for a debate. *Statistics and Probability Letters*. 2018. Vol. 136. P. 10–14.
23. Witten I.H., Eibe F., Hall M.A. (3rd ed.).Data mining: practical machine learning tools and techniques. Morgan Kaufmann, 2011. 629 p.
24. Maimon O., Rokach L. (Eds.) Data Mining and Knowledge Discovery Handbook. 2nd ed., Springer-Verlag New-York Inc., 2010. 1285 p.
25. Murphy K.P. Machine learning: a probabilistic perspective. MIT Press, Cambridge, Massachusetts, 2012. 1055 p.
26. Hastie T., Tibshirani R. and Friedman J. The elements of statistical learning. (2nd ed.). Springer. 2009. 745 p.
27. Efron B. and Hastie T. Computer age statistical inference. Cambridge University Press, 2016. 475 p.
28. Efron B. Large-scale inference. Stanford University Press, 2010. 263 p.
29. James G., Witten D., Hastie T. and Tibshirani R. An introduction to statistical learning with applications in R. Springer, N.Y., 2013. 426 p.
30. Berkhin P. A survey of clustering data mining techniques. In: Kogan J., Nicholas C., Teboulle M. (eds.). *Grouping multidimensional data*. Springer-Verlag: Berlin-Heidelberg, 2006. P. 25–71.
31. Bouveyron C., Brunet-Saumard C. Model-based clustering of high-dimensional data: A review. *Computational Statistics and Data Analysis*. 2014. Vol. 71. P. 52–78.
32. Kurban H., Jenne M. and Dalkilic M.M. Using data to build a better EM: EM* for big data. *Intern. J. of Data Science and Analytics*. 2017. Vol. 4, Issue 2. P. 83–97.
33. LeCun Y., Bengio Y., Hinton G. Deep learning. *Nature*. 2015. Vol. 521, P.436–444.
34. Esling P. and Agón C. Time-series data mining. *ACM Computing Surveys*. 2012. Vol. 45, Issue 1. P. 12–34.
35. Chandola V., Banerjee A. and Kumar V. Anomaly detection for discrete sequences: a survey. *IEEE Trans. on Knowledge and Data Eng*. (TKDE). 2012. Vol. 24, N 5. P. 823–839.
36. Truong C., Oudre L. and Vayatis N. Selective review of offline change point detection methods. [Electronic resource] URL: https://arxiv.org/abs/1801.00718.
37. Aminikhanghahi S. and Cook D.J. A survey of methods for time series change point detection. *Knowledge and Information Systems*. 2017. Vol. 51, Issue 2. P. 339–367.
38. Frick K., Munk A. and Sieling H. Multiscale change point inference. *J. Roy. Statist. Soc., ser. B*. 2014. Vol. 76, Pt. 3. P. 495–580.
39. Wang T. and Samworth R.J. High dimensional change point estimation via sparse projection. *J. Roy. Statist. Soc., ser. B*. 2018. Vol. 80, Pt. 1. P. 57–83.
40. Liao T.W. Clustering of time series data – a survey. *Pattern Recognition*. 2005. Vol. 38. P. 1857–1874.
41. Atluri G., Karpatne A. and Kumar V. Spatio-temporal Data Mining: a survey of problems and methods. *ACM Computing Surveys*. 2018. Vol. 51, Issue 4, Article N 83.
42. Lee T.-W., Girolami M., Bell A.J., Sejnowski T.J. A unifying information-theoretic framework for Independent Component Analysis. *Intern. J. Computers and Mathematics with Applications*. 2000. Vol. 39. P. 1–21.
43. Neville J. and Jensen D. Relational Dependency Networks. *Jour. of Machine Learning Res*. 2007. Vol. 8. P. 653–692.
44. De Raedt L., Kersting K., Natarajan S. and Poole D. Statistical relational artificial intelligence: Logic, probability, and computation. *Synthesis Lectures on Artificial*

Intelligence and Machine Learning. 2016. Vol. 10, N 2. P.1–89.
45. Kazemi S.M., Buchman D., Kersting K., Natarajan S. and Poole D. Relational logistic regression: The directed analog of Markov logic networks. *Workshops at the Twenty-Eighth AAAI Conf. on Artificial Intelligence*. 2014. P. 41–43.
46. Pearl J. Causality: models, reasoning, and inference. Cambridge: Cambridge Univ. Press, 2000. 526 p.
47. Spirtes P., Glymour C. and Scheines R. Causation, prediction and search. New York: MIT Press, 2001. 543 p.
48. Peters J., Janzing D. and Schölkopf B. Elements of Causal Inference. Foundations and Learning Algorithms. MIT Press, Cambridge, MA, USA, 2017. 265 p.
49. Balabanov O.S. Knowledge discovery in data and causal models in analytical informatics. *Problems in programming*. 2017. N 3. P. 96–112. (ISSN 1727–4907). [in Ukrainian].)
50. Raghu V.K., Ramsey J.D., Morris A., Manatakis D.V., Sprites P., Chrysanthis P.K., Glymour C., Benos P.V. Comparison of strategies for scalable causal discovery of latent variable models from mixed data. *Intern. Jour. of Data Science and Analytics*. 2018. Vol. 6, Issue 1. P. 33–45.
51. Tsagris M., Borboudakis G., Lagani V., Tsamardinos I. Constraint-based causal discovery with mixed data. *Intern. Jour. of Data Science and Analytics*. 2018. Vol. 6, Issue 1. P. 19–30.
52. Pearl J. The seven tools of causal inference, with reflections on machine learning. *Communications of the ACM*. 2019. Vol. 62, Issue 3. P. 54–60.
53. Pearl J. and Bareinboim E. External validity: From do-calculus to transportability across populations. *Statistical Science*. 2014. Vol. 29, N 4. P. 579–595.
54. Malinsky D. and Spirtes P. Causal structure learning from multivariate time series in settings with unmeasured confounding. *Proc. of 2018 ACM SIGKDD Workshop on Causal Discovery*, August 2018, London, UK. PMLR, Vol. 92. P. 23–47.
55. Entner D. and Hoyer P.O. On causal discovery from time series data using FCI. *Proc. of the 5th European Workshop on Probabilistic graphical models*. 2010, Helsinki, Finland. P. 121–128.
56. Runge J. Causal network reconstruction from timeseries: From theoretical assumptions to practical estimation. *Chaos*. 2018. Vol. 28, paper 075310. 20 p.
57. Balabanov O.S. Upper bound on the sum of correlations of three indicators under the absence of a common factor. *Cybernetics and Systems Analysis*. 2019. Vol. 55, N 2. P. 174–185.
58. Balabanov O.S. From covariation to causation: Discovery of dependency structures in data. *System research and information technologies*. 2011. N 4, P. 104–118. [In Ukrainian]
59. Colombo D., Maathuis M.H., Kalisch M. and Richardson T.S. Learning high-dimensional directed acyclic graphs with latent and selection variables. *Annals of Statistics*. 2012. Vol. 40, Issue 1. P. 294–321.
60. Colombo D., Maathuis M.H. Order-independent constraint-based causal structure learning. *Jour. of Machine Learning Research.* 2014. Vol.15. P. 3921−3962.
61. Kernel-based conditional independence test and application in causal discovery / K.Zhang, J. Peters, D. Janzing, B. Schölkopf. / *Proc. of the 27th Conf. on Uncertainty in Artificial Intelligence*, (UAI-2011). Corvallis, Oregon: AUAI Press, 2011. P. 804–813.
62. Balabanov A.S. Minimal separators in dependency structures: Properties and identification. *Cybernetics and Systems Analysis*. 2008. Vol. 44, N 6. P. 803–815.
63. Balabanov O.S. Vidtvorennya kauzalnych merezh na osnovi analizu markovskich vlastyvostej [Reconstruction of causal networks via analysis of Markov properties]. *Mathematical Machines and Systems*. 2016. N 1. P. 16–26. [In Ukrainian]
64. Granger C.W.J. Investigating causal relations by econometric models and cross-spectral methods. *Econometrica*. 1969. Vol. 37. P. 424–459.
65. Swanson N.R. and Granger C.W.J. Impulse response functions based on a causal approach to residual orthogonalization in vector autoregressions. *J. of the American Statistical Association*. 1997. Vol. 92, N 437, P. 357–367.
66. Gong M., Zhang K., Schölkopf B., Tao D. and Geiger P. Discovering temporal causal relations from subsampled data. *Proc. of the 32nd Intern. Conf. on Machine Learning*, 2015. P. 1898–1906.
67. Malinsky D. and Spirtes P. Learning the structure of a nonstationary vector autoregression. The 22nd Intern. Conf. on Artificial Intelligence and Statistics. *Proc. of*

Machine Learning Research, *PMLR*, 2019, Vol. 89. P. 2986–2994.
68. Harford T. Big data: A big mistake? *Significance*. 2014. Vol. 11, N 5. P. 14–19.
69. Bühlmann P. and van de Geer S. Statistics for high-dimensional data: Methods, theory and applications. Springer, 2011. 556 p.
70. Donoho D.L. High-dimensional data analysis: the curses and blessings of dimensionality – In: American Mathematical Society Conf. "*Math Challenges of the 21st Century*", 2000, Los Angeles. P. 1–32.
71. Bareinboim E., Tian J., Pearl J. Recovering from selection bias in causal and statistical inference. *Proc. of the 28th AAAI Conf. on Artificial Intelligence*. 2014. P. 2419–2416. (July 27–31, 2014, Québec Convention Center, Québec City, Québec, Canada).

Одержано 17.07.2019

Про автора:

Балабанов Олександр Степанович,
доктор фізико-математичних наук,
провідний науковий співробітник.
Кількість наукових публікацій в
українських виданнях – 60.
Кількість наукових публікацій в
зарубіжних виданнях – 12.
Індекс Хірша – 6.
http://orcid.org/0000-0001-9141-9074.

***Місце роботи автора*:**

Інститут програмних систем
НАН України,
03187, м. Київ-187,
проспект Академіка Глушкова, 40.
Тел.: (044) 526 3420.
E-mail: bas@isofts.kiev.ua

УДК 004.62 https://doi.org/10.15407/pp2019.03.086

В.А. Резниченко

ЧТО ТАКОЕ BIG DATA

В статье делается попытка раскрытия сути понятия Big Data на основе анализа материалов из различных источников. Даются определяющие характеристики Big Data, приводится их классификация, кратко описывается история возникновения и развития, представлены основополагающие принципы работы, кратко излагаются методы и технологии анализа и визуализации, описывается жизненный цикл управления данными с использованием технологии Big Data.
Ключевые слова: большие данные, технология больших данных, база данных, жизненный цикл.

Введение

“Так что же такое Big Data? Это неожиданно обрушившаяся на человечество лавина данных, это принципиально новая информационная технология, это также можно считать техническая и технологическая революции в информатике. Показательно, что из более чем 153 миллиона страниц в Web, содержащих словосочетание Big Data, 122 миллиона содержит еще и слово definition – более двух третей пишущих о Big Data пытается дать свое определение. Такая массовая заинтересованность свидетельствует в пользу того, что, скорее всего, в Big Data есть что-то качественно иное, чем то, к чему подталкивает нас обыденное восприятие этого словосочетания. В этой статье делается попытка раскрытия сути понятия Big Data на основе анализа материалов из различных источников.

1. Материальные и информационные технологии

Приведем рассуждения относительно материальных и информационных технологий, почерпнутые из [1]. К информационной технологии надо относиться как к материальной технологии. Практически все известные материальные технологии сводятся к процессу переработки, обработки или сборки специфического для них исходного сырья или каких-то иных компонентов с целью получения качественно новых продуктов.

Логически информационные технологии мало чем отличаются от материальных технологий, на входе сырые данные, на выходе – структурированные, в форме, более удобной для восприятия человеком, данные, извлеченная из них информация, которая силой интеллекта (естественного или искусственного) превращается в полезное знание. Данные – это выраженные в разной форме сырые факты, которые сами по себе не несут полезного смысла до тех пор, пока не поставлены в контекст, должным образом не организованы и не упорядочены в процессе обработки. Информация появляется в результате анализа обработанных данных человеком (компьютером), этот анализ придает данным смысл и обеспечивает им потребительские качества.

На информационные технологии должны распространяться общие закономерности, согласно которых развиваются все остальные технологии, а это прежде всего увеличение количества перерабатываемого сырья способствует повышению качества переработки.

2. Проблемы больших данных

Мировой объем оцифрованной информации растет по экспоненте. Начиная с 1980-х годов цифровая информация удваивается каждые 40 месяцев. Поданным компании IBS, к 2003 году мир накопил 5 эксабайтов данных (1 ЭБ = 1 млрд гигабайтов), а теперь это количество порождается каждые два дня. К 2008 году этот объем вырос до 0,18 зеттабайта (1 ЗБ = = 1024 эксабайта), к 2011 году – до 1,76 зеттабайта, к 2013 году – до 4,4 зеттабайта. В мае 2015 года глобальное количество данных превысило 6,5 зеттабайта. К 2020 году, по прогнозам, человечество

сформирует 40–44 зеттабайтов информации, а к 2025 г. – 163 зеттабайт. В настоящее время площадь всех крупных датацентров в мире равна площади 6000 футбольных полей Как справиться с такими объемами?

Приведем следующую цитату из [1]: «Данных становится все больше и больше, но при всем этом упускается из виду то обстоятельство, что проблема отнюдь не внешняя, она вызвана не столько обрушившимися в невероятном количестве данными, сколько неспособностью старыми методами справиться с новыми объемами. Наблюдается дисбаланс – способность порождать данные оказалась сильнее, чем способность их перерабатывать.» Под именем Big Data скрывается намечающийся качественный переход в компьютерных технологиях, способный повлечь за собой серьезные изменения. Не случайно этот переход называют новой технической революцией.

3. Определяющие характеристики Big Data

Для Big Data были сформулированы определяющие характеристики. Впервые в 2001 г. признаки «Три V» выделил ведущий аналитик Gartner Дуг Лани [2], а именно, объем, скорость, разнообразие.

Volume (объем). Считается, что Big Data начинаются с объемов в петабайты (1015 байт). Чтобы представить, что это за объем, приведем пример. В Национальной библиотеке Украины им. В.И. Вернадского функционирует портал Научной периодики Украины. Редакции более 2700 научных периодических изданий Украины предоставляют все свои статьи на протяжении 10 лет. За это время объем портала составил около 1 миллиона статей. Если предположить, что размер статьи в среднем составляет 1 МБ, то объем ресурсов этого портала составляет 1 ТБ. Это на три порядка ниже минимального объема для Big Data, то есть через десять тысяч лет успешного функционирования этого портала он накопит объемы, характерные для Big Data.

Big Data появляются тогда, когда сотни миллионов людей объединяются в сообщества и выкладывают свои информационные ресурсы, либо объединенные центры научных исследований предоставляют данные результатов своих исследований, например в 2017 году дата-центр CERN превысил размер 200 петабайт и ежегодно этот объем увеличивается на 15 петабайт. Если поместить в DVD все порожденные в мире за день данные и положить эти диски друг на друга, то получится стопка, дважды превышающая расстояние до Луны.

Velocity (скорость). Является одной из наиболее важных характеристик Big Data с точки зрения их практического использования. Под скоростью подразумевается как скорость прироста (поступления, накопления) данных, так и скорость их обработки с целью получения конечных результатов. Кроме того, в эту категорию включаются характеристики интенсивности и объемов информационных потоков. Для этого технология обработки таких данных должна допускать возможность их анализа уже в момент их порождения (иногда называемой «оперативной аналитикой » - in- memory analytics), то есть до того, как они попадут в хранилище данных. Несколько цифр, характеризующих эту категорию, которые взяты из [3] и некоторых других источников.

YouTube: Имеет более 1 миллиарда зарегистрированных пользователей и ежемесячно сайт посещают 1,9 миллиарда пользователей. Ежеминутно закачивается новых фильмов на 100 часов и скачивается фильмов на 700 тысяч часов. Для просмотра фильмов, выгруженных в YouTube в течении дня, потребуется 15 лет.

Facebook: Имеет 1,4 миллиарда пользователей. Ежедневно на сайт выгружается 100 терабайт данных и ежеминутно ставятся более 34 тысячи лайков. Каждую минуту загружается 200 000 фотографий. Каждый месяц выкладывается в открытый доступ 30 млрд новых источников информации.

Twitter: Сайт имеет более 645 миллиона пользователей. Каждый день генерируется 175 миллион твитов.

Google: Каждую минуту отрабатывается 2,4 миллиона поисковых запросов (40 000 запросов в секунду). Каждый день обрабатывается 25 петабайт данных.

Каждую минуту в мире посылается 204 миллиона е-писем.

По словам специалистов, к категории Big Data относится большинство потоков данных свыше 100 Гб в день.

Variety (разноообразие). Возможность воспринимать, хранить и обрабатывать различные данные. Говоря о многообразии, подразумевается следующее.

Различные источники получения данных. Приведем примеры источников возникновения больших данных:

- непрерывно поступающие данные с измерительных устройств,
- события от радиочастотных идентификаторов,
- потоки сообщений из социальных сетей,
- метеорологические данные,
- данные дистанционного зондирования Земли,
- потоки данных о местонахождении абонентов сетей сотовой связи, устройств аудио- и видеорегистрации.

Различные способы представления данных, например, сигналы, поступающие от датчиков, отличаются от текстов научных статей.

Различные форматы хранения (поступления) данных. Это могут быть тексты, аудио- и видео данные, изображения. Более того, одни и те же данные могут быть представлены в различных форматах. Произносимая человеком речь может быть представлена в аудио-формате и в виде текстового файла.

Семантическое разнообразие. Семантика одних и тех же данных может быть представлена по-разному, например, возраст человека может быть указан количественно или в виде таких терминов, как ребенок, юноша, взрослый человек.

Различная степень структурированности данных. Традиционные базы данных позволяют хранить структурированные данные, но фактически в настоящее время порождаемые данные на 80 % являются слабо структурированными или даже неструктурированными.

Технология Big Data позволяет объединять и обрабатывать данные, обладающие приведенному выше многообразию.

Зикопоулоус [4] предложил добавить еще 2 признака – достоверность и ценность (значимость), таким образом получив «5V»:

Veracity (достоверность). Свойство, которое характеризует надежность данных. Технология создания и использования традиционных БД предполагает, что в БД поступают тщательно отобранные и проверенные данные. В Big Data дело обстоит иначе. Исходные данные могут быть «сырыми» (неполными, неточными, нечеткими, расплывчатыми, искаженными), то есть поступают без какой-либо предварительной обработки, они могут быть субъективными, случайными и содержать много «шума». Еще один критерий этой характеристики – степень доверия к поступающим данным. Хотя Big Data предоставляют прекрасные возможности для анализа и принятия решений, однако их ценность во многом зависит от качества исходных данных. Технология Big Data учитывает эту характеристику и позволяет надежно работать с такими данными.

Value (ценность). Когда мы говорим о ценности данных, то подразумеваем их значимость с точки зрения прикладных задач. По расчетам IBS, только 1,5 % накопленных массивов данных имеет информационную значимость. Большое количество данных – это хорошо, но если они не представляют никакого интереса, то они бесполезны.

Со временем стали предлагать дополнительные определяющие характеристики Big Data [5–9], которые получили название «7V» и “10V”. Приведем этот дополнительный список.

Variability (изменчивость). Под изменчивостью в Big Data подразумевается ситуация, когда постоянно изменяется смысл данных. Например, это имеет место, когда сбор и обработка данных происходит в процессе анализа естественноязыковых тестов и особенно при переводе с одного языка на другой.

Volatility (волатильность, актуальность). Характеристика, которая определяет, какой период времени устаревания данных, когда они становятся нерелевантными или бесполезными. Как долго их надо хранить? До эры Big Data данные могли храниться неопределенно долго, использование для этих целей несколько десятков терабайт не было обременительным. Более того, их можно было хранить в действующей базе данных, не вызывая при этом проблем с производительностью. Однако при наличии Big Data, учитывая характеристики объема и скорости, следует тщательно следить за волатильностью данных. Необходимо установить правила управления хранением данных с тем, чтобы обеспечить эффективное их использование.

Vulnerability (уязвимость). Большие данные порождают новые проблемы их безопасности. Взлом больших данных приводит к большому взлому. Примером может служить взлом базы данных социальной сети LinkedIn, в результате которого было выкрадено167 млн учетных записей и 360 миллионов сведений о e-mail.

Validity (пригодность, обоснованность). Эта характеристика тесно связана с достоверностью и характеризует, в какой мере располагаемые данные являются точными и правильными с точки зрения их предполагаемого использования. По оценке Forbes [10] ученые следующим образом тратят свое время для работы с данными:

– сбор данных 19 %,

– очистка и систематизация данных – 60 %,

– подбор тестовых данных – 3 %,

– анализ данных для построения модели – 9 %,

– уточнение алгоритмов 4 %,

– другие виды работ с данными 5 %.

Таким образом, ученый тратит 80 % своего времени на подбор и подготовку данных прежде, чем приступить к их анализу. Преимуществом использования больших данных для проведения аналитических исследований можно в полной мере воспользоваться только тогда, когда данные тщательно отобраны, являются релевантными и достоверными.

Visualization (визуализация). После получения и обработки данных их надо представить таким образом, чтобы они были читабельными и доступными. Именно это и подразумевает визуализация

Как было уже отмечено, в вебе имеет множество определений Big Data. В частности, по адресу [11] дается 43 определения Big Data. Обобщая эти материалы, дадим следующие определение.

Big Data (большие данные) – это огромные объемы неоднородной, неструктурированной или слабо структурированной, существенно распределенной и интенсивно растущей, изменяющейся и используемой цифровой информации, которую невозможно обработать традиционными средствами. А также методы, технологии и средства их сбора, хранения и обработки и анализа с целью получения воспринимаемых человеком результатов.

4. КлассификацияBig Data

Редактор журнала Web 2.0 Journal Дайон Хинчклифф (Dion Hinchcliffe) дал классификацию Big Data [12], позволяющую соотнести технологию с результатом, который ждут от обработки Big Data. Хинчклиф делит подходы к Big Data на три группы: Fast Data **(**быстрые данные), их объем измеряется терабайтами-петабайтными; Big Analytics **(**большая аналитика) – петабайтные-экзабайтные данные и Deep Insight (глубокое проникновение) – экзабайты-зеттабайты. Группы различаются между собой не только оперируемыми объемами данных, но и качеством решения задач по их обработке.

Обработка для **Fast Data** не предполагает получения новых знаний, ее ре-

зультаты соотносятся с априорными знаниями и позволяют судить о том, как протекают те или иные процессы, она позволяет лучше и детальнее увидеть происходящее, подтвердить или отвергнуть какие-то гипотезы. Только небольшая часть из существующих сейчас технологий подходит для решения задач Fast Data, в этот список попадают некоторые технологии работы с хранилищами. Скорость работы этих технологий должна возрастать синхронно с ростом объемов данных.

Задачи, решаемые средствами **Big Analytics,** заметно отличаются, причем не только количественно, но и качественно, а соответствующие технологии должны помогать в получении новых знаний — они служат для преобразования зафиксированной в данных информации в новое знание. Однако на этом среднем уровне не предполагается наличие искусственного интеллекта при выборе решений или каких-либо автономных действий аналитической системы — она строится по принципу «обучения с учителем». Иначе говоря, весь ее аналитический потенциал закладывается в нее в процессе обучения.

Высший уровень, **Deep Insight**, предполагает обучение без учителя (unsupervised learning) и использование современных методов аналитики, а также различные способы визуализации. На этом уровне возможно обнаружение знаний и закономерностей, априорно неизвестных.

Далее на рис. 1 показана схема Дайон Хинчклиффа взаимодействия трех составляющих Big Data.

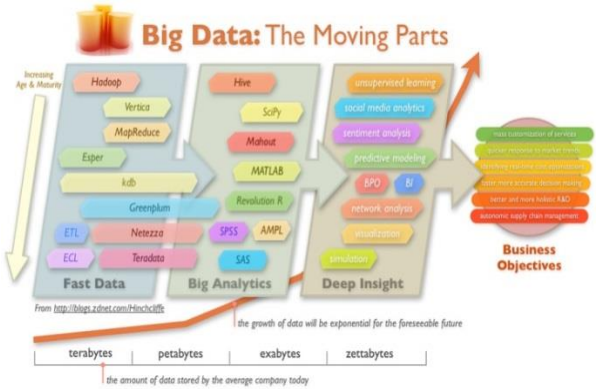


Рис. 1

5. Некоторые вехи в истории развития Big Data

Широкое использование термина «большие данные» связывают с Клиффордом Линчем (Clifford Lynch), редактором журнала Nature, подготовившим к 3 сентября 2008 года специальный выпуск номера старейшего британского научного журнала, посвященный поиску ответа на вопрос «Как могут повлиять на будущее науки технологии, открывающие возможности работы с большими объёмами данных?», в котором были собраны материалы о феномене взрывного роста объёмов и многообразия обрабатываемых данных и технологических перспективах в парадигме вероятного скачка «от количества к качеству»; термин был предложен по аналогии с расхожими в деловой англоязычной среде метафорами «большая нефть», «большая руда», отражающими не столько количество чего-то, сколько переход количества в качество. Этот специальный номер подытоживает предшествующие дискуссии о роли данных в науке вообще и в электронной науке (e-science) в частности.

Этот термин был сначала введен в академической среде и прежде всего обсуждалась проблема роста и многообразия научных данных, но начиная с 2009 года термин широко распространился в деловой среде.

В 2010 году появляются первые продукты и технологии, относящиеся исключительно и непосредственно к проблеме обработки больших данных.

К 2011 году большинство крупнейших поставщиков информационных технологий в своих деловых стратегиях начинают использовать понятие «большие данные», это, в частности, относится к IBM, Oracle, Microsoft, Hewlett-Packard, EMC, а основные аналитики рынка информационных технологий посвящают концепции специальные исследования.

Большой шум вокруг темы больших данных возник после того, как в июне 2011 года консалтинговая компания McKinsey выпустила доклад «Большие данные: следующий рубеж в инновациях, конкуренции и производительности», в котором

оценила потенциальный рынок больших данных в миллиарды долларов.

В этом же году аналитическая компания Gartner отметила большие данные как тренд номер два в информационно-технологической инфраструктуре (после виртуализации и как более существенный, чем энергосбережение и мониторинг). В это же время прогнозировалось, что технология больших данных окажет наибольшее влияние на информационные технологии, в производстве, здравоохранении, торговле, государственном управлении.

В 2012 году администрация президента США выделила 200 миллионов долларов для того, чтобы различные американские ведомства организовывали конкурсы по внедрению технологий больших данных в жизнь. Если в 2009 году американские венчурные фонды вложили в отрасль всего 1,1 миллиарда долларов, то в 2012 – уже 4,5 миллиарда долларов.

С 2013 года большие данные как академический предмет начинают изучать в появившихся вузовских программах по науке о данных и вычислительным наукам и инженерии.

В 2015 году Gartner исключил большие данные из цикла зрелости новых технологий и прекратил выпускать выходивший в 2011–2014 годы отдельный цикл зрелости технологий больших данных, мотивировав это переходом от этапа шумихи к практическому применению.

6. Принципы работы с Big Data

Исходя из определения Big Data, можно сформулировать следующие основные принципы работы с такими данными [13]:

распределенность. Хранить информацию в одном месте бессмысленно и практически невозможно. Поэтому технология работы с Big Data должна использовать распределенное хранение, управление, обработку и анализ данных, хранящихся в разнообразных хранилищах данных во всем мире;

горизонтальная масштабируемость. Поскольку данных может быть сколь угодно много – любая система, которая подразумевает обработку больших данных, должна быть расширяемой. В 2 раза вырос объём данных – в 2 раза увеличили кластер и всё продолжило работать с такой же производительностью;

отказоустойчивость. Принцип горизонтальной масштабируемости подразумевает, что машин в кластере может быть много. Например, Hadoop-кластер Yahoo имеет более 42000 машин. Это означает, что часть этих машин будет гарантированно выходить из строя. Методы работы с большими данными должны учитывать возможность таких сбоев и переживать их без каких-либо значимых последствий;

локальность данных. В больших распределённых системах данные распределены по большому количеству машин. Если данные физически находятся на одном сервере, а обрабатываются на другом – расходы на передачу данных могут превысить расходы на саму обработку. Поэтому одним из важнейших принципов проектирования BigData-решений является принцип локальности данных – по возможности обрабатываем данные на той же машине, на которой они хранятся;

интерпретация данных в процессе их обработки (schema-on-read). Данные поступают в хранилище такими, как есть, без какого-либо их предварительного описания, без указания их структуры и семантики. И только в процессе их выборки для обработки происходит их «осмысливание».

Все современные средства работы с большими данными так или иначе следуют этим пятерым принципам.

7. Методы и технологии анализа и визуализации, применимые к Big Data

К настоящему времени создано и адаптировано множество методов и технологий для сбора, агрегирования, манипулирования, анализа и визуализации больших данных. Эти методы и технологии заимствованы из различных областей, включая статистику, информатику, прикладную математику и экономику. Это означает, что для извлечения выгоды из

больших данных, следует использовать гибкий междисциплинарный подход. Некоторые методы и технологии были разработаны для оперирования значительно меньшими объемами и разнообразием данных, но были успешно адаптированы для Big Data. Другие были разработаны в последнее время, в частности, для сбора и анализа больших данных. Далее приводится перечень и краткое описание методов и технологий анализа и визуализации, применимые к Big Data , которые взяты из отчета McKinsey [14].

7.1. Методы анализа Big Data

Методы класса Data Mining:

– **обучение ассоциативным** правилам (association rule learning) – это метод, базирующийся на правилах, используется для обучения машин способам обнаружения зависимостей между данными в больших базах данных;

– **классификация** – методы категоризации новых данных на основе принципов, ранее применённых к уже наличествующим данным;

– **кластерный анализ** – статистический метод классификации объектов, который приводит к разделению разнообразных групп на более мелкие группы подобных (сходных) объектов, для которых критерий подобия заранее не известен;

– **регрессионный анализ**.

Краудсорсинг (crowdsourcing) – метод сбора, категоризация и обогащение данных силами широкого круга лиц, привлечённых на основании публичной оферты, без вступления в трудовые отношения, обычно посредством использования сетевых медиа.

Смешение и интеграция данных (data fusion and integration) – набор методов, позволяющих интегрировать и анализировать разнородные данные из разнообразных источников для глубинного анализа более точно и эффективно, чем из единственного источника данных. В качестве примеров методов этого класса является цифровая обработка сигналов и обработка естественного языка.

Обучение ассоциативным правилам (association rule learning). Совокупность методов для анализа необходимых взаимосвязей, то есть «ассоциативных правил»," среди переменных в больших базах данных.

Машинное обучение (machine learning). Класс методов искусственного интеллекта, характерной чертой которых является не прямое решение задачи, а обучение в процессе применения решений множества сходных задач. Включает обучение с учителем (supervised learning) и без учителя (unsupervised learning), а также Ensemble learning – использование моделей, построенных на базе статистического анализа или машинного обучения для получения комплексных прогнозов на основе базовых моделей (constituent models).

Обработка естественного языка (Natural language processing – NLP). Общее направление искусственного интеллекта и математической лингвистики. Оно изучает проблемы компьютерного анализа и синтеза естественных языков. Применительно к искусственному интеллекту анализ означает понимание языка, а синтез – генерацию грамотного текста. Многие NLP-методы являются методами машинного обучения.

Искусственные нейронные сети (artificial neural networks). Математическая модель, построенная по принципу организации и функционирования биологических нейронных сетей – сетей нервных клеток живого организма.

Сетевой анализ (network analysis). Набор методов, используемых для описания и анализа отношений между дискретными узлами в графе или сети. В анализе социальной сети анализируются связи между людьми в сообществе или организации, например, как перемещается информация или кто имеет наибольшее влияние на кого.

Распознавание образов (pattern recognition). Набор методов машинного обучения, развивающих основы и методы классификации и идентификации предметов, явлений, процессов, сигналов, ситуаций и т. п. объектов, которые характери-

зуются конечным набором некоторых свойств и признаков.

Прогнозная аналитика (predictive analytics). Класс методов анализа данных, концентрирующийся на прогнозировании будущего поведения объектов и субъектов с целью принятия оптимальных решений.

Анализ тональности текста (sentiment analysis). Класс методов контент-анализа в компьютерной лингвистике, предназначенный для автоматизированного выявления в текстах эмоционально окрашенной лексики и эмоциональной оценки авторов (мнений) по отношению к объектам, речь о которых идёт в тексте.

Имитационное моделирование. (simulation modeling) – метод исследования, при котором изучаемая система заменяется моделью, с достаточной точностью описывающей реальную систему (построенная модель описывает процессы так, как они проходили бы в действительности), с которой проводятся эксперименты, с целью получения информации об этой системе.

Пространственный анализ (Spatial analysis) – набор методов, которые анализируют топологические, геометрические или географические свойства, представленные в наборе данных. Часто данные для пространственного анализа поступают из географических информационных систем (ГИС).

Статистический анализ, примеры: A/B-тестирование (контрольная группа элементов сравнивается с набором тестовых групп, в которых один или несколько показателей были изменены, для того, чтобы выяснить, какие из изменений улучшают целевой показатель) и анализ временных рядов.

Анализ временных рядов (time series analysis) – совокупность математико-статистических методов анализа, предназначенных для выявления структуры временных рядов и для их прогнозирования. Сюда относятся, в частности, методы регрессионного анализа. Выявление структуры временного ряда необходимо для того, чтобы построить математическую модель того явления, которое является источником анализируемого временного ряда.

7.2. Технологии и средства работы с Big Data

Существует множество технологий для агрегации, манипулирования, управления и анализа больших данных. Далее приводится список наиболее известных и используемых технологий и средств. Они приводятся в алфавитном порядке.

Big Table. Запатентованная распределенная система баз данных, построенная на основе Google File System.

Business intelligence (BI) (бизнес-аналитика). Совокупность методологий, процессов, архитектур и технологий, которые преобразуют большие объемы «сырых» данных в осмысленную и полезную информацию, пригодную для бизнес-анализа и для поддержки принятия оптимальных тактических и стратегических решений.

Cassandra. Свободно распространяемая система управления базами данных, предназначенная для манипулирования данными огромного объема в распределенных системах.

Cloud computing (облачные вычисления). Вычислительная парадигма, в которой высокомасштабируемые вычислительные ресурсы, обычно сконфигурированные в виде распределенных систем, предоставляются в сетях качестве сервисов.

Data Warehouse (хранилище данных). Предметно-ориентированная информационная база данных, специально разработанная и предназначенная для подготовки отчётов и анализа данных с целью поддержки принятия решений в организации и является одной из основных компонент бизнес-анализа. Выступает центральным репозиторием данных, поступающих из различных источников. Хранит текущие и исторические данные. Строится на базе систем управления базами данных и систем поддержки принятия решений.

Distributed system (распределенная система). Множество компьютеров, взаи-

модействующих по сети и объединенных для решения общей вычислительной задачи.

Dynamo. Запатентованная распределенная система хранения данных, разработанная в Amazon.

Extract, transform, and load (ETL) (извлечь, преобразовать, загрузить). ПОР, используемое для извлечения данных из внешних источников, преобразования их для удовлетворения операционных потребностей, и загрузка их в базу данных или хранилище данных.

Google File System. Запатентованная распределенная файловая система. На ее основе построен Hadoop.

Hadoop. Проект фонда Apache Software Foundation, свободно распространяемый набор утилит, библиотек ифреймворк для разработки и выполнения распределённых программ, работающих на кластерах из сотен и тысяч узлов. Используется для реализации поисковых и контекстных механизмов многих высоконагруженных веб-сайтов, в том числе, для Yahoo! и Facebook. Базируется на MapReduce и Google File System.

HBase. Свободно распространяемая распределенная нереляционная база данных, созданныя на основе Big Table Google.

MapReduce. Модель распределённых вычислений, представленная компанией Google, используемая для параллельных вычислений над очень большими, вплоть до нескольких петабайт, наборами данных в компьютерных кластерах. Эта моделоь реализована в Hadoop.

Mashup. Веб-приложение, объединяющее данные из нескольких источников в один интегрированный, например, при объединении картографических данных Google Maps с данными о недвижимости с Craigslist получается новый уникальный веб-сервис, изначально не предлагаемый ни одним из источников данных.

R. Свободно распространяемый язык программирования среда программирования для статистических и графических вычислений.

Stream processing. Технология, предназначенная для обработки больших потоков данных в реальном масштабе времени.

7.3. Визуализация Big Data

Наглядное представление результатов анализа больших данных таким образом, чтобы ее можно было легко воспринимать, является ключевой проблемой анализа данных, имеет принципиальное значение для их интерпретации. Восприятие человека ограничено, и ученые продолжают вести исследования в области совершенствования современных методов представления данных в виде изображений, диаграмм или анимаций. В качестве иллюстрации приводим несколько прогрессивных методов визуализации, относительно недавно получивших распространение.

Облако тегов (Tag cloud) рис. 2. Каждому элементу в облаке тегов присваивается определенный весовой коэффициент, который коррелирует с размером шрифта. В случае анализа текста величина весового коэффициента напрямую зависит от частоты употребления (цитирования) определенного слова или словосочетания. Позволяет читателю в сжатые сроки получить представление о ключевых моментах сколько угодно большого текста или набора текстов.

Рис. 2

Clustergram (кластерграмма) рис. 3. Метод визуализации, использующийся при кластерном анализе. Показывает, как отдельные элементы множества данных

соотносятся с кластерами по мере изменения их количества. Выбор оптимального количества кластеров – важная составляющая кластерного анализа. Этот способ визуализации позволяет аналитику лучше понять, как результаты кластеризации изменяются по мере изменения количества кластеров.

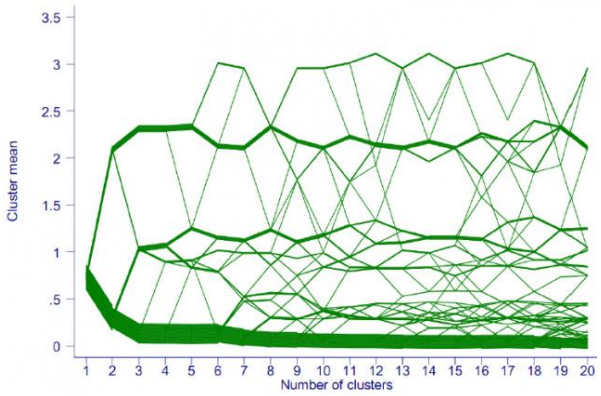


Рис. 3

History flow (исторический поток) рис. 4. Помогает следить за эволюцией документа, над созданием которого работает одновременно большое количество авторов. В History flow (исторический поток). Помогает следить за эволюцией документа, над созданием которого работает одновременно большое количество авторов. В частности, это типичная ситуация для сервисов wiki. По горизонтальной оси откладывается время, по вертикальной – вклад каждого из соавторов, т. е. объем введенного текста. Каждому уникальному автору присваивается определенный цвет на диаграмме. Приведенная диаграмма – результат анализа для слова «ислам» в Википедии. Хорошо видно, как возрастала активность авторов с течением времени.

Spatial information flow (пространственный поток) рис. 5. Эта диаграмма позволяет отслеживать пространственное распределение информации. Приведенная в качестве примера диаграмма построена с помощью сервиса New York Talk Exchange. Она визуализирует интенсивность обмена IP-трафиком между Нью-Йорком и другими городами мира. Чем ярче линия – тем больше данных передается за единицу времени. Таким образом, не составляет труда выделить регионы, наиболее близкие к Нью-Йорку в контексте информационного обмена.

Рис. 4

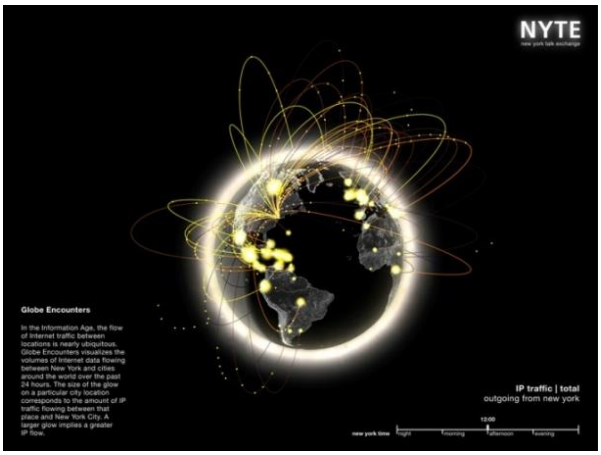


Рис. 5

8. Жизненный цикл управления данными с использованием технологии Big Data

Опишем в общих чертах жизненный цикл управления данных, который использует технологию Big Data. Идея этого цикла взята из работы [15] Предлагаемый жизненный цикл данных состоит из следующих этапов: сбор, фильтрация и классификация, анализ данных, хранение, обмен и публикация, а также поиск и обнаружение данных. Далее кратко описывается каждый этап согласно показанному на рис. 6. жизненному циклу.

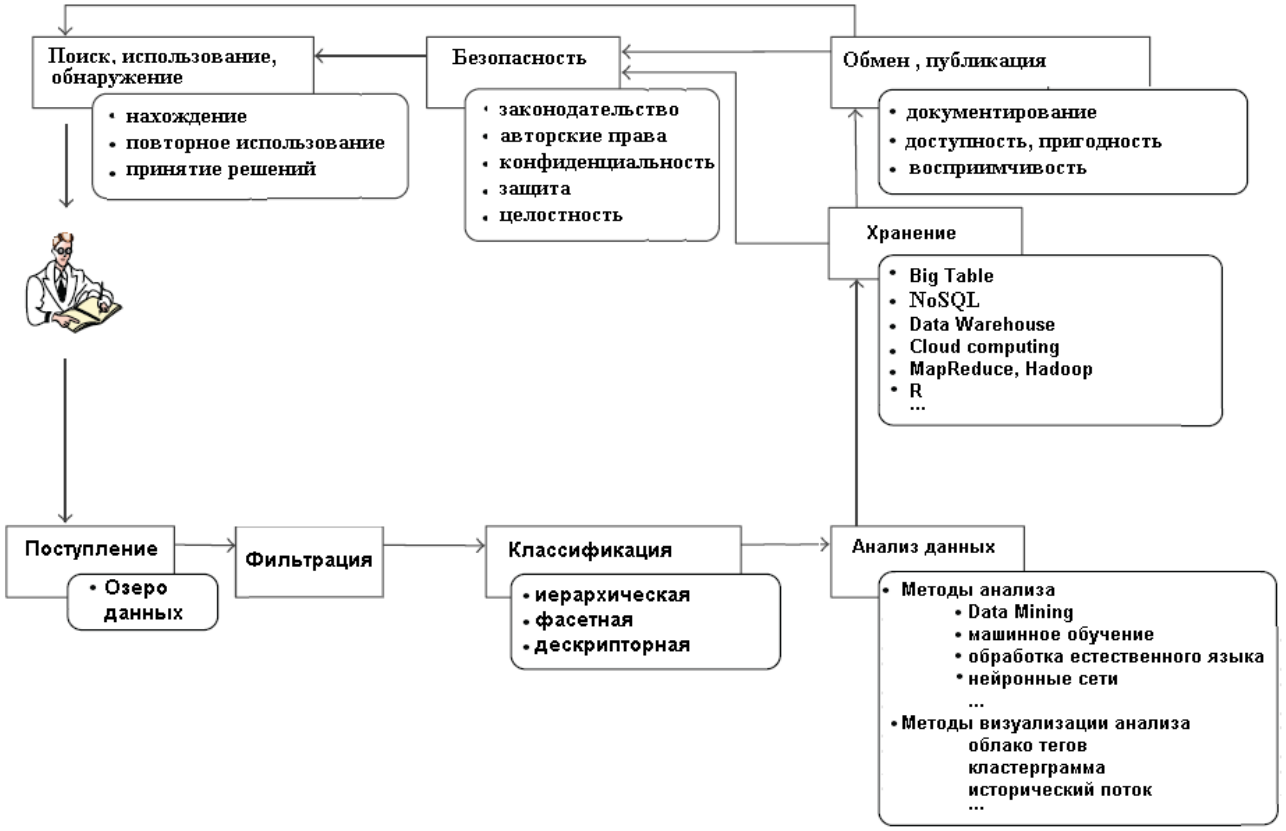


Рис. 6

8.1. Поступление данных. Поступление (сбор) данных – это первый этап жизненного цикла данных. Большое количество данных поступает из различных источников. Такими источниками могут быть: файлы журналов, которые ведутся на серверах, датчики различного вида, мобильные устройства, данные, поступающие со спутников, результаты научных исследований, данные вычислительных экспериментов, результаты выполнения поисковых запросов, данные, порождаемые в социальных сетях, и многие другие. При сборе данных используются разнообразные методы получения исходных сырых данных из различных источников. Рассмотрим несколько методов сбора данных и используемые ими технологии.

– Файлы журналов (log-файлы). Этот метод используется для автоматической регистрации данных, связанных с различными событиями, происходящими в автоматизированных системах. Log-файлы используются практически во всех компьютерных системах, например, веб-сервера фиксирует все транзакции, выполняемые сервером. При наличии очень больших файлов журнала их информация запоминается в базах данных, а не в виде тестовых файлов.

– Сенсорные данные (Sensor data.). Часто датчики используются для съема физических характеристик, которые затем преобразуются в воспринимаемые цифровые сигналы для их сохранения и обработки. К сенсорным данным можно отнести, например, данные, которые поступают в виде звуковых, вибрационных, голосовых волн, результатов физических, химических, биологических, метеорологических или других видов исследований, результатов съема характеристик (показателей) производственных процессов.

– Мобильные устройства. С помощью различные технологий, которые встраиваются в мобильные устройства, можно получать и передавать информацию географическом местоположении, воспринимать аудио- и видеоинформацию, делать фотографии, с помощью сенсорных экранов и гравитационных датчиков получать

информацию о состоянии здоровья человека.

В результате сбора таких данных образуется так называемое озеро данных (Data lake). Это централизованное хранилище больших данных в сыром, необработанном виде. В нем хранят данные из разных источников, разных форматов, структурированные, слабо структурированные, неструктурированные и бинарные данные (изображения, аудио видео-данные)). Они хранятся как правило, в несистематизированном виде такими, как есть, без какой либо предварительной обработки. Это обходится значительно дешевле традиционных хранилищ, в которые помещаются только структурированные данные. Data lake позволяют анализировать большие данные в исходном виде.

8.2. Фильтрация данных. В исходных данных может быть много шума. Так, например, при некачественной аудиозаписи фоновый шум может быть настолько сильным, что не позволяет выделить полезную аудио-информацию с использование современных средств распознавания, или камера видео-наблюдения произвела съемку в темное время и изображение абсолютно черным. Фильтрация позволяет ибавиться от такой информации.

8.3. Классификация данных. Любые поступающие данные всегда обладают какой-то минимальной информацией. Например, известно, где именно установлена видео-камера, куда она направлена и к какому времени суток привязаны те или иные кадры, или что собой представляют поступающие научные данные, результатами какого эксперимента они являются, при каких условиях эксперимент проводился, и так далее. Таким образом, любые поступающие данные обладают так называемыми метаданными, которые можно использовать для проведения первоначальной классификации, которая является первоначальным шагом выявления семантики данных. Эта семантика служит хорошей основой для проведения последующего анализа данных.

Методы классификации – это совокупность приемов разделения множества объектов на подмножества. В науке известны три метода классификации объектов: иерархический, фасетный, дескрипторный. Эти методы различаются разной стратегией применения классификационных признаков.

Иерархический метод. Это метод, при котором заданное множество последовательно делится на подчиненные подмножества, постепенно конкретизируя объект классификации. При этом основанием деления служит некоторый выбранный признак. Совокупность получившихся группировок при этом образует иерархическую древовидную структуру.

Фасетный метод. Подразумевает параллельное разделение множества объектов на независимые классификационные группы. При этом не предполагается жёсткой классификационной структуры и заранее построенных конечных групп. Классификационные группировки образуются путём комбинации значений, взятых из соответствующих фасетов.

Дескрипторный метод. Суть этого метода заключается в следующем: отбирается совокупность ключевых слов или словосочетаний, описывающих определенную предметную область или совокупность однородных объектов, они подвергаются нормализации, на основании этого создается словарь дескрипторов, который служит основой для проведения классификации.

8.4. Анализ данных. Анализ данных позволяет воспринять и обработать огромные объемы Big Data. Анализ данных является сложной задачей и во многом зависит от тех задач, которые надо решать с использованием этих данных, выдвигаемых требований к точности и скорости решения, наличия технических средств и, наконец, состояний исходных данных. Анализ данных включает решения следующих двух основных задач:

– на первом этапе должна быть решена задача раскрытия синтаксиса данных, то есть выявление структуры данных, например, какие объекты предоставляемые данные представляют, какими свойствами они обладают, что собой

представляют значения этих свойств, каким образом взаимосвязаны объекты, какова природа и каковы характеристики этих связей;

– второй этап связан с раскрытием семантики данных. Это так называемый этап интеллектуального анализа данных (data mining). В разделе «Методы анализа Big Data» приводится краткое описание используемых методов. Для гибкой организации анализа данных в работе [16] были предложены следующие три принципа: во-первых, для достижения поставленных целей следует использовать не единственный, а множество релевантных методов анализа. Во-вторых, для хранения данных следует использовать различные методы и устройства хранения, которые могут быть распределены по компьютерам сети. В-третьих, следует предоставлять высокоэффективные методы и средства доступа и обработки данных.

Анализ данных производится с учетом следующих факторов: гетерогенность, точность и сложность данных, возможность их масштабирования.

8.5. Хранение, совместное использование, публикация. После сбора, очистки и анализа полученные данные запоминаются в соответствующих хранилищах, к ним предоставляется доступ и/или они публикуются для ознакомления с ними широкого круга заинтересованных лиц. Большие по объему и интенсивно использующиеся наборы данных. Big Data должны храниться и управляться с большой степенью надежности, доступности и простоте использования. Инфраструктура хранения должна обладать достаточной степенью гибкости. Система хранения должна быть распределенной. Такая распределенная система хранения должно обеспечить поддержку целостности, обеспечение доступности, устойчивости к отказам различного вида.

8.6. Безопасность. Безопасность данных – это защита данных от несанкционированного (случайного или намеренного) доступа, изменения или разрушения.

Сфера применения Big Data в современном мире практически не имеет границ. Раскрытие, изменение или разрушение данных в Big Data может иметь катастрофические последствия. При этом следует отметить, что все среды для работы с большими данными подвержены рискам В связи с этим необходимо обеспечивать надежную защиту Big Data при их хранении, передаче и обработке за счет внедрения и использования процедур и технологических решений в области защиты информации.

8.7. Поиск, повторное использование, обнаружение. Поиск данных обеспечивает (гарантирует) качество данных, увеличение их значимости и сохранности посредством механизма повторного использования и сохранения с целью выявления новой более осмысленной информации. Сфера этой деятельности включает поиск, обнаружение, управление, аутентификацию, архивирование, сохранение и представление данных .После публикации данных другие исследователи должны иметь возможность аутентифицировать и регенерировать их в соответствии со своими интересами для проведения своих исследований. Возможность повторного использования опубликованных данных также должна быть гарантирована в научных сообществах. При многократном использовании определение семантики опубликованных данных является обычной ситуацией. Обычно эта процедура выполняется вручную. В Европейском Союзе активно поддерживается концепция открытой науки, например, инициированием Европейского облака открытой науки для обеспечения открытого доступа к результатам научных исследований из финансируемых государством проектов.

Литература

1. Chernyuk L. Big Data – new theory and practice. Otkrytye sistemy. SUBD 2011 № 10. URL: https://www.osp.ru/os/2011/10/13010990/

2. Laney Doug (2001) 3D Data Management: Controlling Data Volume, Velocity, and Variety. Technical Report 949, METAGroup (now Gartner). [Electronic resourse]: https://blogs.gartner.com/doug-laney/files/2012/01/ad949-3D-Data-Management-Controlling-Data-Volume-Velocity-and-Variety.pdf
3. Nawsher Khan, Ibrar Yaqoob, Ibrahim Abaker Targio Hashem, Zakira Inayat, Waleed KamaleldinMahmoud Ali, Muhammad Alam, Muhammad Shiraz, and Abdullah Gani1. Big Data: Survey, Technologies, Opportunities, and Challenges // Hindawi Publishing Corporation The Scientific World Journal Volume 2014, Article ID 712826, 18 pages, URL: http://dx.doi.org/10.1155/2014/712826
4. Zikopoulos P., Parasuraman K., Deutsch T., Giles J., Corrigan D. (2013) Harness the power of big data The IBM big data platform. McGraw Hill Professional, New York, NY. - [Электронный ресурс]: ftp://public.dhe.ibm.com/software/pdf/at/SWP10/Harness_the_Power_of_Big_Data.pdf
5. The Four V′s of Big Data (англ.). IBM (2011). Проверено 19 февраля 2017. http://www.ibmbigdatahub.com/sites/default/files/infographic_file/4-Vs-of-big-data.jpg
6. Neil Biehn. The Missing V's in Big Data: Viability and Value (англ.). Wired (1 May 2013). Проверено 19 февраля 2017. https://www.wired.com/insights/2013/05/the-missing-vs-in-big-data-viability-and-value/
7. Eileen McNulty. Understanding Big Data: The Seven V's (англ.). Dataconomy (22 May 2014). Проверено 19 февраля 2017. http://dataconomy.com/2014/05/seven-vs-big-data/
8. Tom McNeill. The Eight V's of Supercomputing and Big Data. https://www.nimbix.net/eight-vs-supercomputing-big-data/
9. George Firican. The 10 Vs of Big Data - https://tdwi.org/articles/2017/02/08/10-vs-of-big-data.aspx
10. Gil Press. Cleaning Big Data: Most Time-Consuming, Least Enjoyable Data Science Task, Survey Says. http://www.forbes.com/sites/gilpress/2016/03/23/data-preparation-most-time-consuming-least-enjoyable-data-science-task-survey-says/
11. Jennifer Dutcher. What Is Big Data? https://datascience.berkeley.edu/what-is-big-data/
12. Dion Hinchcliffe. Big Data, The Moving Parts: Fast Data, Big Analytics, and Deep Insight. - https://www.flickr.com/photos/dionh/7550578346/in/photostream/
13. Big Data от А до Я. Часть 1: Принципы работы с большими данными, парадигма MapReduce. https://habr.com/company/dca/blog/267361/
14. James Manyika, Michael Chui, Brad Brown, Jacques Bughin, Richard Dobbs, Charles Roxburgh, Angela Hung Byers. Big data: The next frontier for innovation, competition, and productivity. McKinsey Global Institute, 2011. https://bigdatawg.nist.gov/pdf/MGI_big_data_full_report.pdf
15. Nawsher Khan, Ibrar Yaqoob, Ibrahim Abaker Targio Hashem, Zakira Inayat, Waleed KamaleldinMahmoud Ali, Muhammad Alam, Muhammad Shiraz, and Abdullah Gani. Big Data: Survey, Technologies, Opportunities, and Challenges. Hindawi Publishing Corporation The Scientific World Journal, Volume 2014, Article ID 712826, 18 pages. URL: http://dx.doi.org/10.1155/2014/712826
16. E. Begoli and J. Horey, "Design principles for effective knowledge discovery frombig data," in Proceedings of the 10thWorking IEEE/IFIP Conference on Software Architecture (ECSA '12). P. 215–218, August 2012.

Referenses

1. Chernyuk L. Big Data – new theory and practice. Otkrytye sistemy. SUBD 2011 № 10. URL: https://www.osp.ru/os/2011/10/13010990/
2. Laney Doug (2001) 3D Data Management: Controlling Data Volume, Velocity, and Variety. Technical Report 949, METAGroup (now Gartner). [Electronic resourse]: https://blogs.gartner.com/doug-laney/files/2012/01/ad949-3D-Data-Management-Controlling-Data-Volume-Velocity-and-Variety.pdf
3. Nawsher Khan, Ibrar Yaqoob, Ibrahim Abaker Targio Hashem, Zakira Inayat, Waleed KamaleldinMahmoud Ali, Muhammad Alam, Muhammad Shiraz, and Abdullah Gani1. Big Data: Survey, Technologies, Opportunities, and Challenges // Hindawi Publishing Corporation The Scientific World Journal Volume 2014, Article ID 712826, 18 pages, URL: http://dx.doi.org/10.1155/2014/712826
4. Zikopoulos P., Parasuraman K., Deutsch T., Giles J., Corrigan D. (2013) Harness the power of big data The IBM big data platform.

McGraw Hill Professional, New York, NY. - [Электронный ресурс]: ftp://public.dhe.ibm.com/software/pdf/at/SWP10/Harness_the_Power_of_Big_Data.pdf
5. The Four V´s of Big Data (англ.). IBM (2011). Проверено 19 февраля 2017. http://www.ibmbigdatahub.com/sites/default/files/infographic_file/4-Vs-of-big-data.jpg
6. Neil Biehn. The Missing V's in Big Data: Viability and Value (англ.). Wired (1 May 2013). Проверено 19 февраля 2017. https://www.wired.com/insights/2013/05/the-missing-vs-in-big-data-viability-and-value/
7. Eileen McNulty. Understanding Big Data: The Seven V's (англ.). Dataconomy (22 May 2014). Проверено 19 февраля 2017. http://dataconomy.com/2014/05/seven-vs-big-data/
8. Tom McNeill. The Eight V's of Supercomputing and Big Data. https://www.nimbix.net/eight-vs-supercomputing-big-data/
9. George Firican. The 10 Vs of Big Data - https://tdwi.org/articles/2017/02/08/10-vs-of-big-data.aspx
10. Gil Press. Cleaning Big Data: Most Time-Consuming, Least Enjoyable Data Science Task, Survey Says. http://www.forbes.com/sites/gilpress/2016/03/23/data-preparation-most-time-consuming-least-enjoyable-data-science-task-survey-says/
11. Jennifer Dutcher. What Is Big Data? https://datascience.berkeley.edu/what-is-big-data/
12. Dion Hinchcliffe. Big Data, The Moving Parts: Fast Data, Big Analytics, and Deep Insight. - https://www.flickr.com/photos/dionh/7550578346/in/photostream/
13. Big Data from A to Z. Part 1: Principles of working with Big Data, paradigm MapReduce. https://habr.com/company/dca/blog/267361/
14. James Manyika, Michael Chui, Brad Brown, Jacques Bughin, Richard Dobbs, Charles Roxburgh, Angela Hung Byers. Big data: The next frontier for innovation, competition, and productivity. McKinsey Global Institute, 2011. https://bigdatawg.nist.gov/pdf/MGI_big_data_full_report.pdf
15. Nawsher Khan, Ibrar Yaqoob, Ibrahim Abaker Targio Hashem, Zakira Inayat, Waleed KamaleldinMahmoud Ali, Muhammad Alam, Muhammad Shiraz, and Abdullah Gani. Big Data: Survey, Technologies, Opportunities, and Challenges. Hindawi Publishing Corporation The Scientific World Journal, Volume 2014, Article ID 712826, 18 pages. URL: http://dx.doi.org/10.1155/2014/712826
16. Begoli E. and Horey J. "Design principles for effective knowledge discovery frombig data," in Proceedings of the 10thWorking IEEE/IFIP Conference on Software Architecture (ECSA '12). P. 215–218, August 2012.

Получено 05.07.2019

Об авторе:

Резниченко Валерий Анатольевич,
кандидат физико-математических наук,
старший научный сотрудник Института программных систем НАН Украины.
Количество научных публикаций в украинских изданиях – 61.
Количество научных публикаций в зарубежных изданиях – 4.
http://orcid.org/0000-0002-4451-8931

Место работы автора:

Институт программных систем
НАН Украины.
03187, Киев,
проспект Академика Глушкова, 40.
Тел.: +38 (044) 526 5139.
E-mail: vreznichenko_47@mail.ru

УДК 004.94 https://doi.org/10.15407/pp2019.03.101

О. Захарова

ПЛАТФОРМИ ВЕЛИКИХ ДАНИХ. ОСНОВНІ ЗАДАЧІ, ВЛАСТИВОСТІ ТА ПЕРЕВАГИ

Дана робота представляє огляд існуючих платформ великих даних. Мета полягає у визначені основних проблем та рішень, які існують в цій сфері, а також властивостей платформ великих даних, що визначають їх можливості, переваги чи недоліки у вирішенні цих проблем. Актуальність теми обумовлена стрімким розвитком мобільних пристроїв і прикладних систем, відповідним ростом обсягів інформації та нездатністю традиційних систем обробляти таки обсяги даних у придатні терміни. Тобто, це платформа інформаційних технологій класу підприємства, яка забезпечує властивості та функціональність прикладної системи в одному рішенні для розробки, розгортання, обробки та управління великими даними. Метою створення та використання таких платформ є покращення масштабованості, доступності, продуктивності та безпеки організацій, які працюють з великими даними. Платформи великих даних надають можливість обробляти об'ємні багатоструктурні дані в режимі реального часу та дозволяють різним користувачам застосовувати її для виконання різних задач, що пов'язані із використання великих даних. В роботі розглядаються фреймворки, що розроблені для вирішення задач великих даних, аналізуються їх характеристики, принципи роботи, переваги у контексті проблем, які вони здатні вирішувати, визначаються існуючі «прогалини» та напрямки подальшого розвитку. Вирішення проблем великих даних, а саме забезпечення ефективного зберігання, обробки та аналізу даних, дозволить зробити інформацію кориснішою, а підприємства, які працюють з великими даними, конкурентноздатнішими.

Ключові слова: платформа великих даних, машинне навчання, Apache Hadoop, документоорієнтоване сховище, сховище типу «ключ-значення», стовпчикове сховище, графове сховище, управління даними, розподілене зберігання, потокові обчислення, NoSQL бази даних, розподілена файлова система, MapReduce, TaskTracker, JobTracker, Apache Spark, обробка великих даних, аналітика великих даних.

Вступ

Сьогодні, в умовах зростаючої конкуренції, успіх будь-якої організації визначається можливістю миттєвого доступу до операційних даних та аналізу цих даних. Компанії намагаються своєчасно отримувати необхідну інформацію, щоб оперативно реагувати на змінення ринку. Але обсяг операційних даних може бути настільки великим, що традиційні системи на базі жорстких дисків не справляються з їх обробкою у необхідні терміни. В результаті керівники не отримують потрібну інформацію, а між збором даних та їх аналізом триває занадто багато часу.

Внаслідок технологічної революції, протягом останніх років постійно зростає кількість ядер процесорів та обсяг інтегрованої кеш-пам'яті в обчислювальних системах. Сьогодні обсяг основної пам'яті є практично не обмеженим, що дозволяє завантажувати дані будь-якого розміру. З іншого боку, стрімке розширення ринку мобільних пристроїв та прикладних систем призвело до того, що інформація для прийняття рішень стала доступною для бізнес-користувачів саме в той момент, коли вона необхідна. Індустрія обробки інформації усвідомила вибухове зростання даних, які складно укласти в реляційні сховища завдяки їх неструктурованій природі та не можливо швидко привести до ладу внаслідок великої швидкості зростання обсягів. Це дані від багатьох вимірювальних приладів, з потоків повідомлень у соціальних мережах, метеорологічна інформація та дані журналів подій. Ще одна причина, з якої існуючі раніше рішення виявилися не придатними, – дуже висока вартість зберігання дійсно великих обсягів даних в реляційних БД великих виробників [1].

Так виник термін "великі дані". Він застосовується до наборів даних, які настільки великі, що з ними не можна працювати, використовуючи традиційні системи керування даними. Це набори даних, розмір яких перевищує можливості загально використовуваних програмних засобів та систем зберігання для вирішення задач витягнення, зберігання, управління та обробки у допустимий час.

У таких умовах, організація, щоб лишатися конкурентоспроможною, повинна мати здатність виявляти змінення ситуації на ринку, нові ризики та можливості у режимі реального часу. Традиційні методи управління даними та сховища даних не дозволяють адекватно обробляти та аналізувати дані великих обсягів. Великі дані потрібно не лише генерувати та радіти їх обсягу й різноманітності, інформація, яку вони містять, повинна бути корисною. А для цього її необхідно правильно зберігати, обробляти, аналізувати та синтезувати нові знання з існуючих – тобто робити виводи. Це пояснює стрімкий розвиток платформи великих даних на протязі останніх років.

Платформа великих даних

Платформа великих даних – це інструмент, розроблений постачальниками програмного забезпечення (ПЗ) для управління даними з метою покращення масштабованості, доступності, продуктивності та безпеки організацій, які працюють з великими даними.

Платформа призначена для обробки в режимі реального часу об'ємних багатоструктурних даних. Різні користувачі можуть її використовувати для виконання різних задач. Так, наприклад, інженери даних – для очищення, агрегування та підготовки даних для аналізу, бізнес-користувачі – для запуску запитів, а вчені вважають її корисною при аналізі шаблонів з наборів великих даних за допомогою алгоритмів машинного навчання.

Це платформа інформаційних технологій (ІТ) класу підприємства, яка забезпечує властивості та функціональність прикладної системи в одному рішенні для розробки, розгортання, обробки та управління великими даними. Програмне забезпечення (ПЗ) аналітики великих даних допомагає розкрити приховані шаблони, невідомі кореляції, ринкові тенденції, вподобання клієнтів та іншу корисну інформацію з широкого різноманіття наборів даних.

Головне питання організації роботи з великими даними на корпоративному рівні: обрати реляційну (SQL) чи нереляційну (NoSQL) базу даних? Головною причиною відмови від SQL баз даних (БД) є не правильна робота з самою базою. Більшість компаній не можуть собі дозволити тримати спеціалістів для постійного налагодження баз даних, а для того, щоб розпочати використовувати NoSQL БД не потрібно додаткових розробок. При розробці NoSQL БД особлива увага приділяється забезпеченню високої масштабованості та гнучкості рішень. NoSQL [2] БД – це, перш за все, швидкий доступ до даних, що зберігаються в оперативній пам'яті, гнучкість використання та можливість швидкого розподілення даних між вузлами. Однак можливі такі сценарії, коли дані згодом виходять з-під контроля або вже просто не вміщуються в оперативній пам'яті.

Основні властивості та переваги платформ великих даних

До основних властивостей платформ великих даних можна віднести:

- *забезпечення ефективного зберігання та обробки даних, а також їх інтеграції, управління, витягнення, трансформації, та завантаження* (*ETL*);
- *використання системи Hadoop*: забезпечують функції для масового зберігання даних будь-якого типу, величезну потужність обробки та можливість обробляти практично необмежену кількість паралельних задач;
- *потокові обчислення*: забезпечують функції для *затягування* даних у потік, обробки даних та передачі їх назад єдиним потоком;
- *функції розвиненої аналітики та машинного навчання*;
- *функції управління життєвим циклом контенту та документів;*
- *функції інтеграції великих даних з будь-якого джерела*;
- *управління даними*: містять комплексну систему безпеки, рішення для управління даними та забезпечують дотримання вимог щодо захисту даних.

До головних переваг платформи великих даних та ПЗ аналітики великих даних можна віднести:

- *точні дані.* Платформа великих даних пропонує точні дані, що сприяє прийняттю правильних рішень. Її аналітичні засоби зменшують ризик отримання недостовірних даних, які виникають внаслідок використання сирих, не проаналізованих даних;
- *підвищення ефективності праці.* Платформа спрощує отримання джерела необхідної інформації. Пропонує також інформацію, що може стати у нагоді в майбутньому, таким чином, зберігаючи час та підвищуючи ефективність роботи користувачів;
- *швидкі відповіді на складні питання.* Ефективне управління бізнесом вимагає швидких адекватних відповідей на критичні питання, які впливають на успішність бізнес-операції. Платформа великих даних дозволяє робити це більш надійно. Деякі критичні питання, відповіді на які вимагають тижнів або місяців, за наявності правильного інструменту можуть вирішуватись лише за кілька годин або хвилин;
- *безпека даних.* Забезпечує безпечну інфраструктуру, яка гарантує безпеку даних.

Задачі та ПЗ великих даних

Програмні засоби великих даних можна класифікувати за задачами, які вони вирішують. У процес створення рішення повинні бути інтегровані різні засоби для зберігання, управління та аналізу великих даних. Інструменти великих даних відповідно до їх задач включають наступні групи:

- ПЗ зберігання великих даних;
- ПЗ управління великими даними;
- ПЗ обробки великих даних;
- методи та засоби візуалізації великих даних;
- методи та засоби аналітики великих даних.

Таким чином, відповідний фреймворк [3] повинен відображати інструменти для зберігання, управління та обробки великих даних, інструменти та методи аналітики, візуалізації та оцінювання у різні етапи процесу побудови рішення. Аналітика великих даних може застосовуватись до виявлення знань та обгрунтованого прийняття рішень.

Зберігання та управління великими даними

Традиційні методи структурованого зберігання та витягування даних, такі як реляційні БД, вітрини або SQL сховища даних, мають певні обмеження, які роблять їх не придатними для роботи з великими даними, а саме вони:

- не дозволяють включати нові джерела даних без їх попереднього очищення та інтеграції,
- не дозволяють швидко виробляти та адаптувати дані,
- не забезпечують можливості синхронізації логічного та фізичного вмісту БД з швидкою еволюцією даних,
- не забезпечують поточні потреби аналізу даних.

Необхідність порівняно недорогого зберігання та обробки гігантських обсягів неструктурованої інформації призвела до створення спеціалізованого ПЗ, яке дозволило розподіляти дані за кластерами з сотень та тисяч вузлів, а також обробляти їх у паралельному режимі. Засоби нового покоління для зберігання та управління не структурованими (не реляційними) даними, а саме NoSQL БД, дозволили використовувати репозиторій даних без додаткових розробок, підготовки або налагодження, забезпечили високу масштабованість, розподілення даних між вузлами та швидкий доступ до даних, що зберігаються в оперативній пам’яті. NoSQL [2] БД дозволяють записувати задачі управління даними на прикладному рівні. Кожна база, в даному випадку, є колекцією незалежних документів, де кожний документ підтримує власні дані та схеми та може мати метадані – оглядову інформацію про дані документа. Прикладна програма може мати доступ до багатьох БД, розташованих у різних місцях.

Нові вимоги до зберігання, управління та обробки даних обумовили виникнення Hadoop – фреймворка з відкритим кодом під крилом Apache Sotware Foundati-

on, що дозволяє створювати розподілені системи на базі відносно недорогого обладнання масового попиту. З часом Hadoop був розширений набором бібліотек та утиліт, та сформував навколо себе екосистему проектів з розподіленої обробки даних. Розглянемо його більш детально.

Apache Hadoop фреймворк

Apache Hadoop [4] забезпечує розподілене зберігання та обробку дуже великих наборів даних на комп'ютерних кластерах з промислового комп'ютерного обладнання. Тобто, замість того, щоб використовувати один великий комп'ютер, Hadoop дозволяє кластеризувати апаратне забезпечення для паралельного виконання аналізу масивних наборів даних. Сервіси Hadoop забезпечують виконання наступних функцій:

- зберігання даних;
- обробка даних;
- доступ до даних;
- управління даними;
- безпека та
- операції з даними.

Екосистема Hadoop [5, 10] складається з багатьох модулів (процедур, бібліотек та властивостей), які розглядаються як частини фреймворка. Кожний модуль виконує певну задачу, необхідну для виконання аналітики великих даних. Ядро Hadoop складається з двох основних модулів: розподіленої файлової системи Hadoop Distributed File System (HDFS) та базового інструменту для обробки даних MapReduce та підтримується майже всіма відомими постачальниками систем великих даних. Також до найбільш використовуваних відносять планувальник завдань та управління кластерами YARN і множину загальних утиліт Hadoop Common.

Розподілена файлова система. HDFS [4] дозволяє зберігати дані у простому доступному форматі. Це досягається завдяки використанню великої кількості пов'язаних пристроїв зберігання даних та механізму MapReduce для їх обробки.

"Файлова система" є методом, що застосовується комп'ютером для зберігання даних таким чином, щоб їх можна було знаходити та використовувати. Зазвичай, він визначається операційною системою (ОС) комп'ютера, але система Hadoop використовує власну файлову систему, яка надбудовується "над" файловою системою хост-комп'ютера. Це означає, що доступ до даних можна отримати з будь-якого комп'ютера, на якому встановлена будь-яка підтримувана ОС.

Hadoop розділяє файли на великі блоки та розподіляє їх між вузлами у кластері. Потім він передає пакетований код у вузли для обробки даних у паралельному режимі. В даному підході вузли маніпулюють даними, до яких вони мають доступ. Це дозволяє обробляти набір даних швидше та ефективніше, ніж в більш традиційній архітектурі суперкомп'ютера, яка спирається на паралельну файлову систему, де обчислення та дані розподіляються у високошвидкісній мережі.

HDFS є розподіленою, масштабованою та портативною файловою системою, що написана на Java для Hadoop фреймворк. Вона забезпечує виконання команд та Java інтерфейсів (API), подібних до інших файлових систем, для зв'язку використовує протокол TCP/IP. Клієнти для спілкування один з одним використовують виклики віддаленої процедури (RPC). Надійність зберігання даних досягається шляхом реплікації між декількома хостами. Щоб зменшити трафік у мережі, Hadoop необхідно знати, які сервери є найближчими до даних або інформації, яка може забезпечити встановлення мостів з HDFS.

Hadoop може працювати безпосередньо з будь-якою розподіленою файловою системою, яка може бути встановлена основною операційною системою.

Прикладами файлових систем, що підтримуються Hadoop (окрім HDFS), є:

- FTP (зберігає всі свої дані на віддалених FTP-серверах);
- сховище об'єктів Amazon S3 (Simple Storage Service), орієнтоване на кластери, які розміщені на інфраструктурі Amazon Elastic Compute Cloud типу сервер-на-запит;
- Windows Azure Storage Blobs (WASB), розширення HDFS, яке дозволяє

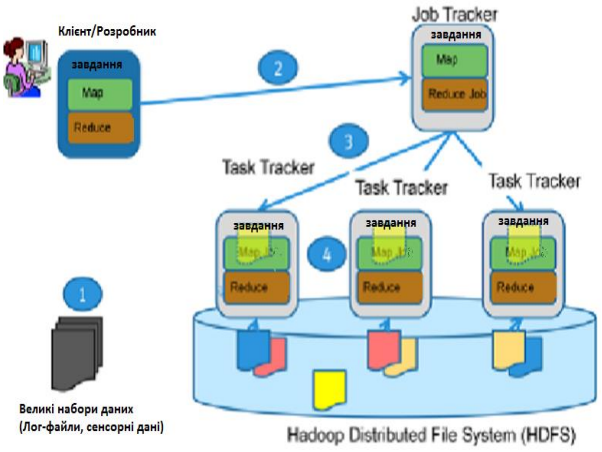
розподілення Hadoop для доступу до даних в Azure блог-сховищах без постійного переміщення даних у кластер.

Існують також файлові системи, які не розповсюджуються разом з Hadoop, але постачаються як альтернативні, що використовуються за замовченням, з де-якими його комерційними рішеннями. Наприклад: IBM General Parallel File System, Parascale, Appistry (драйвер файлової системи Hadoop для використання з CloudIQ Storage), драйвер файлової системи IBRIX Fusion, альтернативна файлова система MapR FS, що заміщує HDFS системою повністю випадкового доступу для читання/запису файлів.

Модуль MapReduce. MapReduce названий за двома головними операціями, які він виконує, а саме: читання даних з БД і переведення їх у формат, що підходить для аналізу (map), та виконання математичних операцій (reduce).

Функціонування MapReduce [11] забезпечується двома компонентами: JobTracker та TaskTracker. Клієнтські прикладні програми направляють завдання MapReduce до JobTracker, JobTracker працює з доступними у кластері вузлами TaskTracker, щоб наблизитись до потрібних для виконання цих завдань даних. JobTracker відомо, які вузли містять дані, та які інші комп'ютери є поруч (рис. 1).

Якщо робота не може бути виконана на тому вузлі, де розміщені дані, перевага надається вузлам, що розміщені на тій самій стойці. Таким чином, скорочується трафік на головній магістральній мережі. Якщо TaskTracker виходить з ладу або зазнає затримку, здійснюється перепланування частини завдань. TaskTracker в кожному вузлі породжує окремий процес віртуальної машини Java (JVM). Це дозволяє запобігти виходу з ладу TaskTracker, якщо запущене на виконання завдання зруйнує свою JVM. Кожні кілька хвилин з TaskTracker до JobTracker надсилається імпульс, щоб перевірити його стан. Стани JobTracker і TaskTracker та інформація про їх роботу відображаються у контейнері сервлетів Jetty та її можна також переглядати у веббраузері [11].



❶ Дуже великий набір даних. HDFS зберігає репліки даних у вузлах даних.

❷ Клієнт виконує Map та Reduce завдання на конкретному наборі даних та відсилає їх JobTracker.

❸ JobTracker розподіляє завдання серед TaskTracker. TaskTracker запускає механізм відображення (map), результат роботи якого зберігається у HDFS.

❹ Запускається завдання Reduce на даних, що вже оброблені завданням Map

Рис. 1. Робота Map Reduce модуля

Слід зазначити, що даний підхід має певні обмеження:

- Алгоритм розподілення роботи TaskTracker є дуже простим. Кожний TaskTracker має множину наявних слотів. Кожна активна задача займає один слот. JobTracker розподіляє роботу на TaskTracker з наявним слотом, найближчий до даних. При цьому не приймається до уваги поточна завантаженість системи призначеної машини – її реальна доступність.
- Якщо один TaskTracker дуже повільний, це може затримати роботу MapReduce в цілому. Однак, коли дозволене паралельне виконання, окрема задача може виконуватися на декількох підпорядкованих вузлах.

В первинному варіанті, для впорядкування завдань з робочої черги, Hadoop підтримує First-In-First-Out (FIFO) планування та опціональне планування пріоритетів, які використовуються за замовченням. Згодом до планувальника завдань бу-

ла додана можливість використовувати альтернативні планувальники, такі як Fair (Facebook AI Research) або Capacity. Планувальник Fair [16] є розробкою Facebook. Розробники мали за мету забезпечити швидку відповідь для невеликих завдань та якість сервісу для виробничих завдань. Завдання групуються у пули і ресурси розподіляються між цими пулами. За замовченням для кожного користувача є окремий пул, так що кожний користувач отримує рівну частку кластера. На відміну від планувальника Hadoop, що використовується за замовченням та формує чергу завдань, Fair дозволяє коротким завданням завершуватись у розумний час, не очікуючи довго своєї черги. Це також є простим способом спільного використання кластера між кількома користувачами, яке також може працювати з пріоритетами завдань. Ці пріоритети використовуються як ваги для визначення частки загального часу обчислення, яке отримує кожне завдання.

Планувальник Capacity, розроблений Yahoo, підтримує декілька властивостей, подібних властивостям Fair, а саме: кожній черзі виділяється частка загального ресурсу, вільні ресурси виділяються чергам за їх потужністю. У черзі завдання з більшим пріоритетом мають пріорітетніший доступ до її ресурсів.

HDFS не обмежується MapReduce завданнями. Вона може працювати з іншими прикладними програмами, багато з яких є розробками Apache, наприклад, база даних HBase, система машинного навчання Apache Mahout, система Apache Hive Data Warehouse. Теоретично, Hadoop може використовуватися для будь якого типу завдань, які є швидше пакетно-орієнтованими, ніж тими, що виконуються у реальному часі, а також для завдань з дуже інтенсивними даними і завдань, для яких корисна паралельна обробка даних. Hadoop також може використовуватися для доповнення системи реального часу, такої як, наприклад, лямбда архітектура, Apache Storm, Flink або Spark.

Hadoop Common та планувальник Yarn. Hadoop Common забезпечує інструменти, які дозволяють комп’ютерним системам користувача читати дані, що зберігаються у файловій системі Hadoop.

YARN керує ресурсами систем, які зберігають дані та виконують їх аналіз.

Використання Hadoop. Hadoop також містить безліч інших інструментів з відкритим кодом, призначених для створення додаткових функцій на компонентах ядра Hadoop.

Так, Apache Tez є фреймворком наступного покоління, який може використовуватися замість Hadoop MapReduce, в якості двигуна. Amazon EMR включає конектор EMRFS, який дозволяє Hadoop використовувати для зберігання даних сховище Amazon S3. Amazon EMR також може використовуватися для легкого встановлення та налаштування у кластері таких інструментів, як Hive, Pig, Hue, Ganglia, Oozie та HBase. Окрім Hadoop на Amazon EMR можна запускати інші фреймворки, такі як Apache Spark для обробки даних у пам'яті або Presto для виконання інтерактивних запитів.

Гнучка природа системи Hadoop дозволяє компаніям, коли вони потребують змін, додавати або змінювати власну систему даних, використовуючи дешеві та легко-доступні частини від будь-яких постачальників інформаційних систем. Сьогодні Hadoop є найбільш використовуваною системою для зберігання та обробки даних на виробничому апаратному забезпеченні. Hadoop використовують майже всі великі постачальники он-лайн продуктів, та кожний має можливість його вільно модифікувати відповідно до своїх цілей. Ці зміни, які вносять до ПЗ експерти, наприклад, Amazon чи Google, відсилаються до спільноти розробників, де вони часто використовуються в подальшому для вдосконалення "офіційного" продукту. Така форма колаборативної розробки є ключовою властивістю ПЗ з відкритим кодом.

Слід зазначити, що використання базових модулів Hadoop Apache є складним навіть для фахівця галузі інформаційних технологій, тому були розроблені комерційні версії продукту такі, як наприклад, Cloudera, що спрощують задачу інсталяції та запуску Hadoop, а також пропонують послуги навчання та підтримки. Завдя-

ки гнучкій природі Hadoop, компанії при розширенні бізнесу мають можливість корегувати та розширювати операції аналізу даних. Підтримка спільноти відкритого коду робить аналіз великих даних доступним для кожного.

Apache Spark

Apache Spark – фреймворк (містить більш ніж 80 операторів для роботи з даними) з відкрититм кодом, який був створений для розподіленої обробки великих даних. На відміну від класичного обробника з ядра Hadoop, що реалізує дворівневу концепцію MapReduce з дисковим сховищем, він використовує спеціалізовані примитиви для рекурентної обробки в оперативній пам'яті, завдяки чому дозволяє отримувати значне прискорення роботи для деяких класів задач. Зокрема, можливість багатократного доступу до даних користувача, що завантажені в оперативну пам'ять, робить бібліотеку дуже привабливою для алгоритмів машинного навчання. Фактично, Spark є переосмисленим MapReduce, але працює у 10-100 разів швидше, залежно від того, працює він в пам'яті або на диску. Spark підтримує мови програмування Scala, Python, Java, R.

Головним поняттям у Spark є Resilient Distributed Dataset (RDD) [14] – це розподілена структура даних, яка розміщується в оперативній пам'яті (рис. 2). Кожний RDD є фрагментом даних, що розподілені по вузлах кластера. RDD є незмінними структурами, тому після виконання перетворень створюються нові RDD. RDD обробляються паралельно за допомогою трансформацій/дій, які виконуються одночасно во всіх розділах (partition). RDD є відмовостійкими: якщо розділ втрачається в результаті відмови вузла, він може бути відновлений з вихідних джерел.

Рис. 2. Розподілення RDD

Фактично RDD являє собою набір даних, над яким можна виконувати перетворення двох типів: трансформації та дії. Відповідно, вся робота з цими структурами полягає у послідовності цих перетворень.

Трансформація. Як правило, перетворює якимось чином елементи даного набору даних. Результатом застосування її до RDD є новий RDD. Далі наведений неповний перелік найрозповсюдженіших трансформацій, кожна з яких повертає новий RDD (рис. 3):

- map(f) – застосовує функцію f до кожного елемента набора даних;
- filter(f) – повертає всі елементи набору даних, на яких функція f повернула істинне значення;
- distinct([numTasks]) – повертає набір даних, який містить унікальні елементи вихідного набору даних;

Також підтримуються наступні операції над множинами:

- union(Dataset) – об'єднання з набором даних Dataset,
- intersection(Dataset) – перетин з набором даних Dataset,
- cartesian(Dataset) – результатом операції є новий набір даних, який містить пари (A,B), де A належить вихідному набору даних, а B – набору даних Dataset.

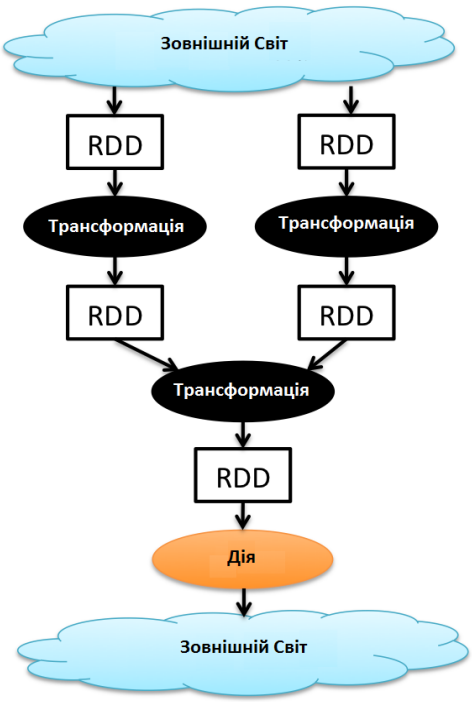


Рис. 3. Робота з RDD

Дії. Застосовуються, коли необхідно матеріалізувати результат – як правило, зберегти дані на диску, або вивести частину даних у консоль. Найбільш розповсюдженими діями, які можна застосувати до RDD, є:

- saveAsTextFile(path) – зберігає дані у текстовому файлі (hdfs) на локальну машину або у будь-яку іншу файлову систему, яка підтримується, path визначає шлях для збереження файлу;
- collect() – повертає елементи набора даних у вигляді масива. Як правило, використовується після застосування до набору даних фільтрів та перетворень для візуалізації або додаткового аналізу результату;
- take(n) – повертає у вигляді масива перші n елементів набору даних;
- count() – повертає кількість елементів у наборі даних;
- reduce(f). Функція f (приймає на вхід 2 аргументи, повертає одне значення) повинна обов'язково бути комутативною та асоціативною.

Spark не змушує думати в парадигмі MapReduce, а дозволяє створювати зрозумілий код, який спрямований саме на виконання поставленої бізнес-задачі. Фреймворк бере на себе розподілення та фрагментацію кода та даних, які автоматично передаються на кластер.

Spark має також колекцію бібліотек (набір готових алгоритмів, підходів та практик), що дозволяють комбінувати існуючі рішення в межах одного програмного кода для досягнення поставленої мети. На цей час Spark містить наступні бібліотеки:

- Spark SQL;
- Spark Streaming (аналіз у реальному часі);
- MLib (машинне навчання);
- GraphX (робота з графами).

Spark SQL. Це модуль Apache Spark, який є частиною ядра Spark та інтегрує реляційну обробку даних та процедурний API Spark. Він може працювати разом з Hive (HiveQL/ SQL) або його заміщувати. Окрім цього, модуль здатний взаємодіяти з інструментами бізнес-аналітики.

Spark SQL підтримує реляційну обробку даних як в межах програм Spark (через RDD), так і з зовнішніх джерел даних. Він може взаємодіяти з новими джерелами даних, включаючи слабоструктуровані дані та зовнішні бази даних, що підтримують федеративні запити.

Spark SQL реалізує та оптимізує реляційну обробку, підтримуючи наступні підходи:

- перетворення даних у більш ефективні формати (з точки зору сховища, мережі та операцій введення/ виведення), зокрема, в різні формати, що орієнтовані на стовбці (columnar format);
- розбиття даних на секції;
- зменшення кількості операцій читання на основі статистики;
- оптимізація операцій над даними;
- виконання оптимізації наскільки можливо пізніше, коли доступна вся інформація по конвейєрах даних.

Spark SQL використовує оптимізатор запитів Catalyst для інтелектуального планування запитів.

Spark SQL може підтримувати пакетний та потоковий SQL. Ядро Spark забезпечує обробку пакетних навантажень через RDD. RDD можуть посилатися на статичні набори даних, а за допомогою розвиненого API Spark можна маніпулювати RDD в оперативній пам'яті із застосуванням «ледачих» обчислень.

Spark Streaming. Реалізує абстракцію DStream (discretized stream, дискретизований потік), що являє собою безперервний потік даних. DStream може бути створений з потоку вихідних даних; на основі таких джерел, як Kafka або Flume, або за допомогою виконання операцій з іншими DStream. По суті, DStream є послідовністю RDD (рис. 4).

Рис. 4. Структура DStream

RDD, що створений за допомогою DStream, можна перетворювати у DataFrame та виконувати SQL запити до нього. Доступ до потоку може надаватися для будь-якої зовнішньої прикладної програми, що підтримує SQL, за допомогою JDBC-драйвера. Пакети потокових даних зберігаються у пам'яті вузла. До цих даних можна будувати інтерактивні запити, використовуючи SQL або API Spark. Для виконання SQL-запитів до Dstream використовується StreamSQL, що поєднує Spark Streaming з Catalyst. StreamSQL є розширенням SQL, яке додатково забезпечує підтримку наступних потокових операцій:

- виборка (SELECT) з потоку для обчислення функцій або фільтрації даних (за допомогою умови WHERE);
- з'єднання (JOIN) потоку з одним або декількома наборами даних для створення нового потоку;
- застосування віконних функцій та виконання агрегацій. Потік можна налаштувати таким чином, щоб він створював набори даних обмеженого розміру. За допомогою віконних функцій можна виконувати складний відбір повідомлень на основі значень полей. Після створення обмеженого пакета можна виконувати аналітику на ньому.

В основу підходу для реалізації аналітики реального часу покладено лямбда-архітектуру, що застосовується для створення аналітичних систем реального часу в контексті великих потокових даних (рис. 5).

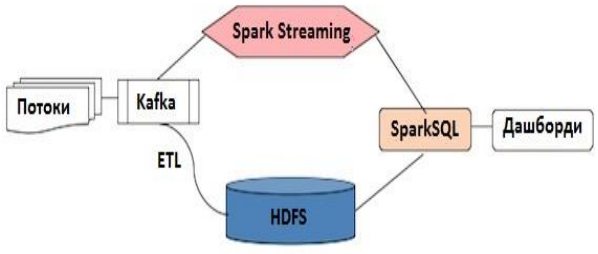


Рис. 5. Логічна схема реалізації аналітики великих даних в реальному часі за допомогою Spark SQL

MLib. Бібліотека для машинного навчання. Її метою є зробити машинне навчання масштабованим та простим. Вона містить розповсюджені алгоритми і утиліти машинного навчання, та дозволяє розпаралелювати на кластері алгоритми машинного навчання (класифікація, регресія, кластеризація і т. і.) лише за пару строк коду. Окрім цього, SparkMLib якісно працює з локальними даними, використовуючи пакет лінійної алгебри Breeze. MLib має добре продуманий API, працює з даними у будь-якому форматі на базі Hadoop та не потребує попереднього встановлення.

GraphX. Розподілений фреймворк обробки графів на основі Apache Spark. Графи є наявною та простою для розуміння моделлю даних. Розподілені обчислення кардинально спростили зберігання та обробку графів.

Головним механізмом ітерації графа в GraphX є розроблений Google алгоритм Pregel. Головною ідеєю цього алгоритму є передача повідомлень між вузлами у графі, які називають супер кроками завдяки послідовності ітерацій. Ітерація часто формується як "think like a vertex", тобто стан поточного вузла залежить лише від стану його сусідів. Використання Pregel є особливо доцільним, коли задачу складно вирішити за допомогою звичайного MapReduce.

Головним примитивом для обходу графа у GraphX є триплет: поточний вузол, вузол, до якого здійснюється перехід, та ребро між ними. Pregel вимагає визначення відстані між вузлами за замовченням, як правило, це PositiveInfinity – UDF (user defined function) функція для кожного вузла, що дозволяє обробити вхідне повідомлення та порахувати наступний вузол, а також UDF функції для злиття двох вхідних повідомлень. Ці функції повинні бути комутативними та асоціативними [15].

GraphX містить статичну та динамічну версії реалізації алгоритму PageRank, який для кожного вузла графа призначає вагу серед решти вузлів. Наприклад, якщо користувач Твітера має велику кількість підписок від інших користувачів, то він буде мати високий рейтинг, тобто, його можна буде легко знайти у пошуковій системі. Статична версія має фіксовану кількість ітерацій, тоді як динаміч-

на версія буде працювати доки рейтинг не почне зходитися до заданого значення.

Через те що GraphX побудований на основі незмінних RDD, графи теж незмінні, тому GraphX непридатний для роботи з графами, які оновлюються, тим більше транзакціями, як у графових БД.

GraphX надає два окремі API для реалізації масово паралельних алгоритмів (таких як PageRank): Pregel-подібний та більш загальний — MapReduce API.

Основні типи NoSQL сховищ

На сьогодні виділяють чотири основних типи NoSQL сховищ [12]:

- *сховище «ключ-значення».* В ньому є велика хеш-таблиця, що містить ключі та значення. (Приклади: Riak, Amazon DynamoDB);
- *документоорієнтоване сховище.* Зберігає документи, які складаються з тегованих елементів. (Приклад: CouchDB);
- *стовпчикове сховище.* У кожному блоці зберігаються дані лише з однієї колонки. (Приклади: HBase, Cassandra);
- *сховище на основі графів.* Мережеве сховище, яке використовує вузли та ребра для відображення та зберігання даних. (Приклад: Neo4J).

Сховище типу «ключ-значення». Відсутність схеми у сховищах типу «ключ-значення» є важливою перевагою для зберігання великих даних. Ключ може бути синтетичним або автосгенерованим, а значення може бути представлено строкою, JSON, блобом (BLOB, Binary Large Object) тощо. Такі сховища, як правило, використовують хеш-таблицю, яка містить унікальний ключ та посилання на певний об'єкт даних. Існує поняття блока – логічної групи ключів, що фізично не поєднують дані у групи. У різних блоках можуть бути ідентичні ключи.

Продуктивність обробки даних значно збільшується за рахунок механизмів хешування, що працюють на основі маппінгів. Щоб прочитати значення, необхідно знати ключ та блок, оскільки насправді ключ є хешем (блок + ключ).

Модель «ключ-значення» проста в реалізації. Такі сховища є доступними та толерантними до розділення, але явно програють у питаннях погодженості даних. В якості недоліків сховищ типу «ключ-значення» можна зазначити:

- модель не надає стандартних можливостей баз даних таких, як атомарність транзакцій або погодженість даних при одночасному виконанні декількох транзакцій. Такі можливості повинні надаватися самою прикладною програмою.
- при збільшенні об'ємів даних, підтримка унікальних ключей може стати проблемою. Для її вирішення необхідно якось ускладнити процес генерації строк, щоб вони залишалися унікальними серед дуже великої множини ключей.

Документоорієнтоване сховище. Дані, які представлені парами ключ-значення, стискаються як сховище документів, що є схожим зі сховищем «ключ-значення». Але на відміну від сховища «ключ-значення», документи, які зберігаються, мають визначену структуру та кодування даних. Деякі зі стандартних розповсюджених кодировок, що використовуються – це XML, JSON та BSON.

Однією з ключових відмінностей між сховищами «ключ-значення» та документоорієнтованим є те, що останнє включає метадані, які пов'язані зі вмістом, що зберігається. Це надає можливість робити запити на основі цього вмісту. Найпопулярнішими прикладами документоорієнтованих сховищ є CouchDB та MongoDB. CouchDB використовує JSON для зберігання даних, JavaScript в якості мови запитів з використанням MapReduce та HTTP для API. Дані та відношення не зберігаються в таблицях так, як це відбувається у традиційних реляційних БД, а за сутністю є набором незалежних документів. Той факт, що такі сховища працюють без схеми, спрощує задачу додавання полів до JSON-документа без необхідності попереднього заявлення про зміни.

Стовпчикове сховище. У стовпчикових NoSQL сховищах дані зберігаються у комірках, що згруповані у стовпчики, а не строки даних. Стовпчики логічно групуються у стовпчикові сімейства. Стовп-

чикові сімейства можуть складатися з практично необмеженої кількості стовпчиків, які можуть створюватися під час роботи програми або під час визначення схеми. Читання та запис відбувається із використанням стовпчиків, а не строк.

Порівняно зі зберіганням даних у строках, як у більшості реляційних БД, переваги зберігання у стовпчиках полягає у швидкому пошуці/доступі та агрегації даних. Реляційні БД зберігають кожну строку як безперервний запис на диску. Різні строки зберігаються у різних місцях на диску, в той час як стовпчикові сховища зберігають всі комірки, що відносяться до стовпчика, як безперервний запис, що прискорює пошук/доступ.

Стовпчикові сховища використовують наступну модель даних:

- стовпчикове сімейство – структура, яка може легко групувати колонки та суперколонки;
- ключ – постійне ім'я запису. У ключів може бути різна кількість стовпчиків, тому сховище може розширюватися нерівномірно;
- простір ключів – визначає зовнішній рівень організації, як правило, ім'я прикладної програми/БД.
- стовпчик – має впорядкований список елементів – кортежів з іменами та значеннями.

Найвідомішими представниками стовпчикових сховищ є Google BigTable та HBase з Cassandra.

BigTable є високопродуктивним, стислим та пропрієтарним сховищем даних від Google. Воно має наступні атрибути:

- розрідженість – деякі комірки можуть бути порожніми;
- розподіленість – дані розділені між багатьма вузлами;
- постійність – дані зберігаються на диску;
- багатомірність – більш ніж одне вимірювання;
- співставлення – ключ та значення;
- відсортованість.

На стовпчики можна посилатися за допомогою стовпчикового сімейства.

Графове сховище. У графовому сховищі немає строгого формату SQL або представлення таблиць та стовпчиків, замість цього використовується гнучке графічне представлення, яке ідеально підходить для вирішення проблем масштабованості. Графові структури використовуються разом із ребрами, вузлами та властивостями, що забезпечує безіндексну суміжність. При використанні графового сховища дані можуть бути легко перетворені з однієї моделі в іншу (рис. 6).

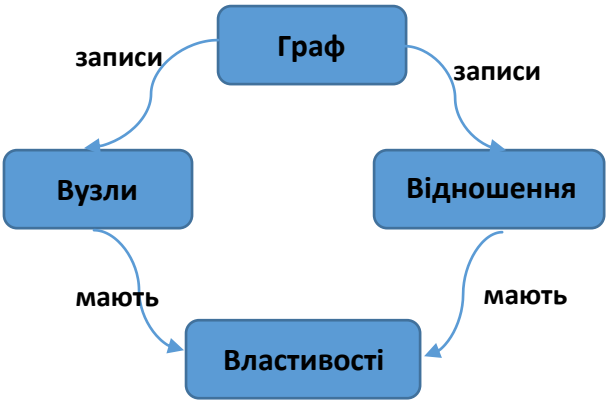


Рис. 6. Принципи використання графової моделі

- Такі сховища використовують ребра та вузли для представлення даних.
- Вузли пов'язані між собою певними відношеннями, які представлені ребрами між ними.
- Вузли та відношення мають деякі властивості.

Розмічений, спрямований, атрибутований мультиграф (рис. 7) – це граф, який містить вузли, які помічені певними властивостями та які мають зв'язки один з одним, що представлені спрямованими ребрами. Наприклад, зв'язок «Аліса знає Боба» виражена ребром з відповідними властивостями. Будь-який рейтинг «вам рекомендовано», представлений на різних сайтах, часто вираховується виходячи з того, як інші користувачі оцінили продукт. Графові БД відмінно підходять для вирішення такого типу задач.

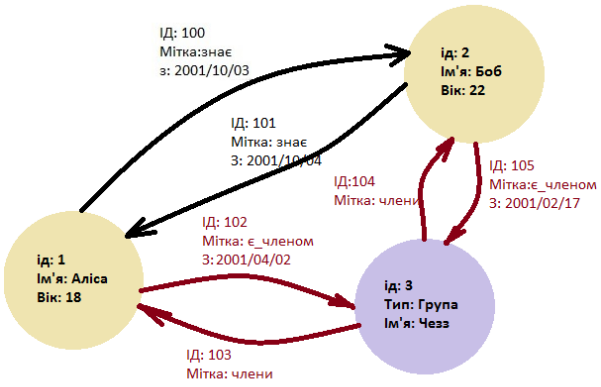


Рис. 7. Приклад атрибутованого мультиграфу

Прикладами найпопулярніших графових сховищ є InfoGrid та Infinite Graph. InfoGrid дозволяє з'єднувати множини ребер та вузлів, що спрощує представлення набора інформації зі складними взаємними посиланнями. InfoGrid пропонує два типи сховищ:

- MeshBase — підходить для автономного розгортання;
- NetMeshBase — підійде для великих розподілених графів та має додаткові можливості для взаємодії з іншими подібними сховищами.

Постачальники та ПЗ великих даних

В області розробки підходів до роботи з пам'яттю найбільшим чемпіоном є SAP зі своєю Hana платформою, але, слід зазначити, що зараз Microsoft та Oracle також вводять спеціальні опції для роботи з пам'яттю для своїх провідних баз даних. Постачальники ПЗ, що фокусуються на використанні аналітичних БД, включаючи Actian, HP Vertica та Teradata ввели спеціальні опції для співвідношення високих-ОЗУ-дисків, а також пропонують інструменти для розміщення конкретних даних у пам'яті для виконання ультра-швидкого аналізу. Прогрес, що має місце у зростанні пропускної здатності та обчислювальної потужності, вдосконалив й можливості потокової обробки та проведення аналізу в реальному часі.

До великих постачальників сервісів обробки даних [13] можна віднести IBM, Microsoft, Oracle, SAP, які пропонують все від ПЗ інтеграції даних та систем керування базами даних (DBMS) до ПЗ для бізнес-аналізу та аналітичної обробки, а також Hadoop опцій для роботи з пам'яттю та потокової обробки. Teradata більш вузько зосереджений на керуванні даними, та подібно Pivotal, він має тісні зв'язки з лідером аналітичного ринку SAS.

Багато постачальників пропонують реалізовані окремі опції хмарних технологій, але такі розробники, як 1010data та Amazon Web Services (AWS) використовують хмарну модель у повному обсязі в своєму ПЗ. З них двох Amazon має найширшу вибірку продуктів і є очевидним вибором для тих, хто працює з великими навантаженнями та зберігає велику кількість даних на AWS платформі. 1010data має високо масштабований сервіс БД та підтримує можливості управління інформацією, бізнес-аналіз та аналітику, що обслуговуються в стилі приватної хмари.

Hadoop довів свою користь та переваги у вартості там, де є екстремальними об'єм та різноманітність даних. На сьогодні це найбільш відомий та поширений програмно-апаратний комплекс для роботи з великими даними. Він виявився настільки гарним, що став фундаментом декількох комерційних реалізацій на його основі, а саме: Cloudera, MapR та Hortonworks, кожна з яких пропонує власний дистрибутив. На сьогодні всі постачальники традиційних BI-систем, як MicroStrategy або SAS, забезпечують інтерфейс з Hadoop. Виробники MPP-систем (масово-паралельних архітектур) у свою чергу забезпечують суттєво більш міцну інтеграцію з Hadoop, коли дані, що зберігаються і в Hadoop, і в реляційній СКБД, можуть оброблятися в одному SQL-запиті. Oracle, IBM, Teradata. Cloudera, Hortonworks та MapR, що також включили Hadoop до своїх продуктових лінійок, роблять все можливе, щоб перемістити Hadoop з високо-масштабованого зберігання даних та Map Reduce обробки у світ аналітики.

Менші постачальники такі як Actian, InfiniDB/Calpont, HP Vertica, Info-

bright та Kognitio, навпаки фокусуються загалом скоріше на аналітиці, ніж на обробці транзакцій.

Такі постачальники аналітики, як Alpine Data Labs, Revolution Analytics та SAS, працюють з платформами, які забезпечуються сторонніми постачальниками СКБД та дистриб'ютерами Hadoop, хоча, зокрема, SAS розмиває це розмежування зі зростаючою підтримкою для середовищ SAS-керованих рядів даних «у-пам'яті» та Hadoop. NoSQL та NewSQL СКБД фокусуються на високо-масштабованій обробці транзакцій, а не на аналітиці.

Взагалі програмні засоби для роботи з великими даними не заміщують решту інструментів обробки, бізнес-аналітики, візуалізації та прогнозування, а лише допомагають підтримати терабайти нових даних та спрямовують їх у потрібне русло.

Так, відповідно до аналітичних платформ для великих даних, деякі експерти вважають найбільш універсальною платформу Pentaho, а для вирішення задач машинного самонавчання, таких як, наприклад, кластеризація, класифікація, регресія та інші, краще підходять Mahout та Spark. Серед найбільш технологічних MPP – платформ спеціалісти виділяють Vertica та Teradata Aster. Останнім часом з'явилася множина платформ, які підтримують швидку аналітику для великих даних, наприклад, MemSQL або Splice Machine.

Окремої уваги заслуговує Intel платформа з відкритим кодом для Hadoop. Привабливість рішення Intel для Hadoop обумовлює також й фактор "апаратного забезпечення", а саме – оптимізація, що виконана Intel з урахуванням архітектури процесорів Xeon та специфіки роботи твердотільних накопичувачів з контролерами Intel, дозволяє досягти значного приросту продуктивності. Процесори Xeon прискорюють операції шифрування або дешифрування за алгоритмом AES (Advanced Encryption Standard), що реалізується за допомогою додаткового набору команд AES-NI (New Instruction). Окрім цього, платформа Intel для Hadoop також пропонує розширені можливості у галузі обробки потокових даних.

Різноманіття платформ для роботи з великими даних доповнюється величезною кількістю прикладних програмних продуктів, комерційних чи безкоштовних, для аналітичної обробки таких даних. Нижче наведений невеликий перелік найпоширеніших прикладів такого ПЗ:

- *Cluvio* – сучасна платформа аналітики даних, що дозволяє виконувати SQL запити, обробляти дані, візуалізувати результати та створювати гарні, інтерактивні дашборди за лічені хвилини. Підтримує потужне вбудовування, що дозволяє додавати аналітичні властивості до будь-якого веб-сайту або веб-застосунку.
- *IBM SPSS Statistics* дозволяє виявляти нові зв'язки між даними та будувати прогнози. Він дозволяє отримувати легкий доступ до даних, управляти та аналізувати набори даних, не маючи попереднього статистичного досвіду. Це дозволяє практично виключити довготривалу підготовку даних та швидко створювати, маніпулювати та розповсюджувати інформацію для прийняття рішень.
- *Qlik Sense Desktop* – безкоштовний продукт, що надає можливість інтерактивного створення звітів та дашбордів з діаграмами та графіками. Програма візуалізації спрощує аналіз даних та допомагає створювати інформовані бізнес-рішення швидше, ніж будь-коли раніше. Перетворення електронних таблиць у більш чіткі візуалізації робить процес аналізу простішим та швидшим для перегляду всіх користувачів.
- *Elasticsearch* – розповсюджений пошуковий та аналітичний движок на базі Apache Logstash, Kibana та Beats складають "Elastic Stack", розроблений фірмою Elastic. Також Elasticsearch забезпечується хостінг Elastic Cloud.
- *Cyfe* – бізнес-панель для управління даними компанії за допомогою звітів, попередньо побудованих віджитів тощо.
- *Forestpin Analytics* – платформа аналізу даних для знаходження нерівностей, кореляцій та дублювань за допомогою простого дашборда.

Висновки

Кількість підприємств, що використовують великі дані, безперервно зростає. Практика останніх років продемонструвала, що застосування результатів аналізу великих масивів даних може принести реальний ефект. Але, окрім переваг існує велика кількість проблем, вирішення яких вимагає застосування досить значних ресурсів.

Для систем, що отримують аналітичні дані в масштабі, близькому до реального часу, ключовими є вимоги не лише до продуктивності, але й до часу відгуку (наприклад, IBM каже про час відгуку, менший за мілісекунди). Це дуже обмежує вибір аналітичних платформ. Неможливо використовувати колосальні обчислювальні можливості Hadoop, якщо накладні витрати на ініціювання та завершення тривіальної MapReduce-програми складають десятки секунд. Забезпечити прийнятний час відгуку можуть або досить недешеві MPP-платформи (такі як Netezza, Teradata, Greenplum), або розподілені системи з розвиненою індексацією або високим рівнем резидентності даних в оперативній пам'яті.

Багато аналітичних систем все ще використовують реляційну модель даних, внаслідок чого вибір платформ обмежується такими рішеннями, як GridGain або Gigaspaces XAP. Для роботи з потоковими даними в режимі он-лайн були створені технології Storm, Spark Streaming та Akka. Але аналіз даних за допомогою SQL на Hadoop не дозволяє досягти того максимуму, який пропонує платформа.

Компанії обирають Hadoop, щоб збирати складні та різноманітні дані: історія відвідувань веб-сайтів, логи, дані про використання мобільних пристроїв й інформації з соцмереж та багато іншого. Цими даними складно оперувати у СКБД. Можна витягувати структуровані дані з Hadoop для SQL-аналізу, але більш перспективними є такі підходи як машинне самонавчання та інші, що дозволяють співвіднести нові дані зі вже накопиченою, проаналізованою та структурованою інформацією. BI та SQL системи досить добре себе проявили, але постійно виникають нові потреби та нові питання, що виходять за межі поточних можливостей. Уже не достатньо просто управляти даними. Окрім цього, компанії не можуть покладатися лише на аналітику, вони потребують також рішень зі сфери BI, системи збору та передачі оперативної інформації та інше. Межа між цими поняттями почала стиратися, а SAS, Alpine Data Labs та інші стали підтримувати кластеризовані серверні середовища, вимогливі до пам'яті та Hadoop.

Література

1. Великі дані: великі проблеми. Є.Є. Чехарин. Міжнародний електронний науковий журнал.
2. https://proglib.io/p/nosql-db-part-1/
3. Big Data: Survey, Technologies, Opportunities, and Challenges. Nawsher Khan, Ibrar Yaqoob, Ibrahim Abaker Targio Hashem, ZakiraInayat, Waleed Kamaleldin Mahmoud Ali, Muhammad Alam, Muhammad Shiraz and Abdullah Gani. *The Scientific World Journal*. January 2014.
4. Sagiroglu S. and Sinanc D. Big data: are view. Proceedings of the International Conference on Collaboration Technologies and Systems (CTS'13), P. 42–47, IEEE, SanDiego, Calif, USA, May 2013.
5. Wang D. An efficient cloud storage model for heterogeneous cloud infrastructures. Procedia Engineering. 2011. Vol. 23. P. 510–515.
6. Bakshi K. Considerations for big data: architecture and approach. Proceedings of the IEEE Aerospace Conference. P. 1–7, BigSky, Mont, USA, March 2012.
7. Aho A.V. Computation and computational thinking. The Computer Journal. 2012. Vol. 55, N 7. P. 833–835.
8. Bhatnagar S.S.V. and Srinivasa S. Big Data Analytics, 2012.
9. Pastorelli M., Barbuzzi A., Carra D., Dell'Amico M. and Michiardi P. HFSP: size-based scheduling for Hadoop. Proceedings of the IEEE International Congress on Big Data (BigData '13), 2013. P. 51–59.
10. Katal A., Wazid M., and Goudar R.H. Big data: issues, challenges, tools and good practices. Proceedings of the 6th International Conference on Contemporary Computing (IC3'13). 2013. P. 404–409.
11. Big Data Analytics: A Literature Review Paper. Nada Elgendy and Ahmed Elragal. Con-

ference Paper in Lecture Notes in Computer Science · August 2014.
12. https://tproger.ru/translations/types-of-nosql-db/
13. https://www.forbes.com/sites/bernardmarr/2016/02/09/how-to-find-the-best-big-data-product-or-service-vendors/
14. http://datareview.info/article/analitika-v-rezhime-realnogo-vremeni-s-pomoshhyu-spark-sql/
15. https://habr.com/post/415939/
16. https://hadoop.apache.org/docs/r1.2.1/fair_scheduler.html

References

1. Big data: big problems. E. E. Cheharin. International Electronic Scientific Journal, ISSN 2307-2334.
2. https://proglib.io/p/nosql-db-part-1/
3. Big Data: Survey, Technologies, Opportunities, and Challenges. Nawsher Khan, Ibrar Yaqoob, Ibrahim Abaker Targio Hashem, ZakiraInayat, Waleed Kamaleldin Mahmoud Ali, Muhammad Alam, Muhammad Shiraz and Abdullah Gani. The Scientific World Journal · January 2014.
4. Sagiroglu S. and Sinanc D. "Big data: are view," in Proceedings of the International Conference on Collaboration Technologies and Systems (CTS'13), P. 42–47, IEEE, SanDiego, Calif, USA, May 2013.
5. Wang D. "An efficient cloud storage model for heterogeneous cloud infrastructures," Procedia Engineering. 2011. Vol. 23. P. 510–515.
6. K. Bakshi, "Considerations for big data: architecture and approach, "in Proceedings of the IEEE Aerospace Conference. P. 1–7. BigSky, Mont, USA, March 2012.
7. Aho A.V. "Computation and computational thinking," The Computer Journal. 2012. Vol. 55, N 7. P. 833–835.
8. Bhatnagar S.S.V. and Srinivasa S. Big Data Analytics, 2012.
9. Pastorelli M., Barbuzzi A., Carra D., Dell'Amico M. and Michiardi P. "HFSP: size-based scheduling for Hadoop," in Proceedings of the IEEE International Congress on Big Data (BigData '13), 2013. P. 51–59.
10. Katal A., Wazid M., and Goudar R.H. "Big data: issues, challenges, tools and good practices," in Proceedings of the 6th International Conference on Contemporary Computing (IC3'13). 2013. P. 404–409.
11. Big Data Analytics: A Literature Review Paper. Nada Elgendy and Ahmed Elragal. Conference Paper in Lecture Notes in Computer Science · August 2014.
12. https://tproger.ru/translations/types-of-nosql-db/
13. https://www.forbes.com/sites/bernardmarr/2016/02/09/how-to-find-the-best-big-data-product-or-service-vendors/
14. http://datareview.info/article/analitika-v-rezhime-realnogo-vremeni-s-pomoshhyu-spark-sql/
15. https://habr.com/post/415939/
16. https://hadoop.apache.org/docs/r1.2.1/fair_scheduler.html

Одержано 28.02.2019

Про автора:

Захарова Ольга Вікторівна,
кандидат технічних наук,
старший науковий співробітник.
Кількість наукових публікацій в
українських виданнях – 28.
http://orcid.org/0000-0002-9579-2973.

Місце роботи автора:

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

УДК 004.932.2:[551.501:551.576.2] https://doi.org/10.15407/pp2019.03.116

А.Ю. Дорошенко, В.М. Шпиг, О.М. Овдій

ПРОГРАМНА СИСТЕМА АНАЛІЗУ ХМАРНОСТІ ЗА ДАНИМИ СУПУТНИКОВИХ СПОСТЕРЕЖЕНЬ

У даній роботі за допомогою онлайнового діалогового конструктору для автоматизації проектування робочих процесів ОКРП було спроектовано робочий процес для обробки архіву супутникових знімків з метою аналізу показників хмарності на території України. Робочий процес для реалізації поставленої задачі було розроблено для системи Apache Oozie, призначеної для управління роботами розподіленої платформи Apache Hadoop. З метою підтримки масштабованості та оптимізації обробки для реалізації аналізу знімків було використано фреймворк Apache Spark. Проведено експерименти, що довели ефективність обраного підходу та відповідність отриманих результатів даним фактичних спостережень.

Ключові слова: розподілені обчислення, робочі процеси, Apache Hadoop, Apache Spark, аналіз зображень, моніторинг, хмарність.

Вступ

В області метеорології задача обробки та аналізу великих обсягів даних є надзвичайно актуальною. Це зумовлено наявністю величезних об’ємів історичних та поточних даних метеорологічних величин, великим різноманіттям технічних засобів вимірювань і спостережень та складністю моделей. Однією із основних таких величин у моделях та синоптичних методах прогнозу погоди, кліматології, агрометеорології тощо є хмарність, тобто кількість хмар, яка характеризує ступінь покриття небосхилу хмарами і визначається спостерігачем у десятих частках одиниці (балах). Вона є одним із найважливіших факторів для перебігу багатьох фізичних процесів в атмосфері та біля земної поверхні. Змінюючись у часі та просторі, хмарність суттєво впливає на радіаційний та тепловий режими Землі, а отже, має безпосередній вплив на кліматичну систему планети, її глобальні та регіональні зміни. З іншого боку інформація про загальну хмарність, купчасто-дощові хмари та пов’язані із ними небезпечні та стихійні явища (наприклад, гроза, град, смерч, шквал і т. п.) є важливою як для цивільної, так і військової авіації, багатьох галузей економіки держави.

Протягом десятиліть в Україні кліматологічні дослідження хмарності (географічні та сезонні особливості, добовий хід тощо), розробка синоптичних та інших методів прогнозування, верифікація моделей прогнозу погоди проводяться із використанням даних лише наземних спостережень [1–5]. Подібного характеру дослідження виконуються і в інших країнах у такий же спосіб [6, 7]. Водночас наявність супутникових даних відкриває широкий спектр можливостей покращення безпосереднього моніторингу хмарного покриву, пов’язаних із ним атмосферних опадів та інших явищ, отримання нових даних (шляхом обробки та аналізу різнорідної інформації) та для започаткування цілого вектору досліджень в Україні. Використання супутникової інформації дає можливість більш тонко здійснювати аналіз поля хмарності у регіонах зі складним рельєфом, оскільки орографічні особливості можуть змінювати хмарні системи на відносно коротких просторово-часових масштабах, а частота отримання даних дозволяє виокремити не тільки великомасштабне перенесення, але і особливості місцевої циркуляції у гірських районах, кластерізацію опадів тощо [8, 9].

На даний момент Україна не має своїх власних метеорологічних супутників, для потреб оперативного прогнозування та наукових досліджень використовуються дані EUMETSAT (Європейська організація з експлуатації метеорологічних супутників) та NOAA (Національна адміністрація з океанічних та

атмосферних досліджень, США), частина яких зберігається у вигляді знімків. Зокрема, дані щодо хмарного покриву та яскравісної температури в каналі 10,8 мкм радіометра SEVIRI геостаціонарного супутника MSG (EUMETSAT), які і було використано з метою створення системи аналізу хмарності для території України.

У даній роботі для вирішення задачі обробки супутникових знімків було вирішено використовувати розподілену обчислювальну платформу Apache Hadoop [10]. За допомогою системи для автоматизації проектування робочих процесів на основі алгеброалгоритмічного та онтологічного інструментарію ОКРП [11] було спроектовано робочий процес для системи Apache Oozie, призначеної для управління роботами Apache Hadoop. Система ОКРП є подальшим розвитком методологій та інструментів, що ґрунтуються на засобах високорівневої алгеброалгоритмічної формалізації і автоматизації перетворень програм [12–16], зокрема експериментальної інструментальної системи ІПС для конструювання та оптимізації паралельних програм, а також її онлайн версії системи ОДСП. З метою підтримки масштабованості та оптимізації обробки великого обсягу супутникових знімків для реалізації аналізу зображень було використано фреймворк Apache Spark. Проведено експерименти, які довели ефективність обраного підходу.

1. Програмна реалізація обробки супутникових знімків

У рамках даної роботи була поставлена задача обробки архіву супутникових знімків для обчислення та аналізу показників хмарності на території України. Знімки зберігаються у форматі збереження графічної інформації JPEG. На супутникових знімках в залежності від значень яскравісної температури хмарність зображено різними кольорами спектру. Конвективна хмарність ідентифікувалася згідно [17] з урахуванням географічного розташування України.

Для вирішення задачі необхідно обчислити загальну хмарність, конвективну та неконвективну. При аналізі знімків виникло декілька проблем. Перша проблема пов’язана з тим, що JPEG є форматом, що використовує стиснення з втратами і в зв’язку з великим ступенем стиснення вихідних зображень втрата та спотворення даних є дуже великими. Цю проблему було вирішено за рахунок введення похибок та попередньої обробки зображення з метою вилучення шумів. Наступною проблемою були нанесені на зображення обриси кордонів областей та назв населених пунктів. Цю проблему було вирішено шляхом використання маски для їх вилучення з розрахунків з подальшим обчисленням значення кольорів точок зображення, що потрапили під маску, за найближчими точками поза маскою. В зв’язку з тим, що всі ці заходи не в змозі повністю відновити зображення, в обчисленнях присутні похибки.

Для реалізації аналізу супутникових знімків було вирішено використати систему для автоматизації проектування робочих процесів ОКРП [11]. В даній системі поєднання інструментів онтологій та алгеброалгоритмічних інструментів забезпечує значний потенціал для адаптації, оптимізації, інтеграції та модифікації розроблюваних робочих процесів. Робочі процеси [18] забезпечують систематичний спосіб опису необхідних методів та інтерфейс між фахівцями та обчислювальним середовищем, спрощуючи задачу обробки та аналізу даних з різних джерел на широкому спектрі обчислювальних платформ.

Робочий процес для реалізації поставленої задачі було розроблено для системи Apache Oozie [19]. Apache Oozie є системою керування робочими процесами для управління роботами розподіленої платформи Apache Hadoop. Apache Hadoop призначена для розподіленого зберігання й обробки великих об’ємів даних на великих комп’ютерних кластерах.

З метою підтримки масштабованості та оптимізації обробки великого обсягу супутникових знімків для реалізації кроку аналізу знімків по отриманню даних про хмарність використовується

фреймворк Apache Spark [20, 21]. Apache Spark – це кластерний обчислювальна фреймворк з відкритим вихідним кодом для розподіленої обробки великих об'ємів даних. Spark не використовує MapReduce [22] як механізм виконання, замість цього він використовує власне розподілене середовище. Тим не менш, Spark має багато спільного з MapReduce з точки зору API та середовища виконання. Spark тісно інтегрований з Apache Hadoop і він може запускатися як на Hadoop YARN так і у відокремленому режимі. Spark може працювати з різноманітними форматами файлів та сховищами даних Hadoop такими як розподілена файлова система HDFS, розподілена база даних HBase і т. д.

На відміну від MapReduce, де дані завжди завантажуються з диска, Spark здатен між робочими задачами зберігати їх у оперативній пам'яті, що дозволяє значно збільшити швидкість виконання для деяких класів задач. Ця можливість дозволяє Spark перевершити за продуктивністю еквівалентний робочий процес MapReduce (на порядок або більше в деяких випадках).

Spark підтримує такі мови програмування як Scala, Java і Python.

Для більш детального статистичного аналізу отриманих з супутникових знімків даних про хмарність застосовується програмне середовище R [23], що призначене для статистичних обчислень та аналізу даних.

На рис. 1 показано схему розробленого робочого процесу для аналізу хмарності за супутниковими знімками.

Робочий процес складається з наступних кроків:

– **start** – запуск робочого процесу;

– **spark-job** – крок робочого процесу, який запускає програму Apache Spark, що виконує обробку масиву супутникових знімків для отримання показників хмарності. Деталі реалізації даного кроку описані нижче;

– **fork** – розбиває шлях виконання робочого процесу на кілька паралельних. Це дозволяє паралельно провести статистичний аналіз отриманих на попередньому кроці даних про хмарність;

– **r-mean-by-month** – shell-скрипт, який запускає на виконання програму R для статистичного аналізу даних та виводу результатів. Програма обчислює середні показники хмарності за місяцями та графічно представляє результат зі зберіганням графіків у форматі PDF;

– **r-mean-by-day** – shell-срипт, який запускає на виконання програму R для статистичного аналізу даних та виводу

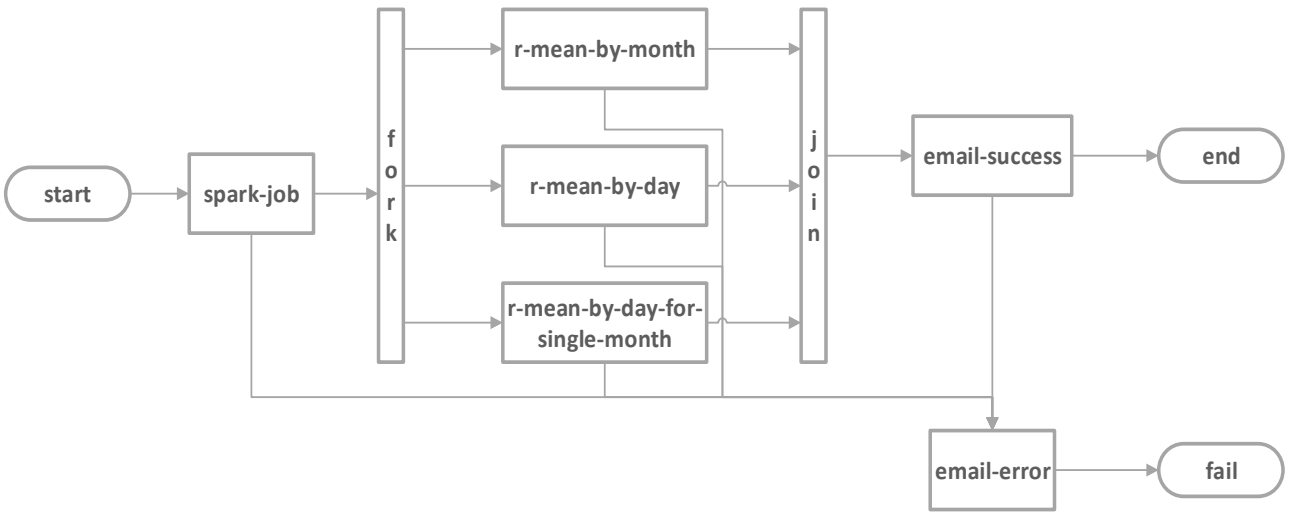


Рис. 1. Схема робочого процесу аналізу супутникових знімків

результатів. Програма обчислює середні показники хмарності за днями та графічно представляє результат зі зберіганням графіків у форматі PDF;

– **r-mean-by-day-for-single-month** – shell-скрипт, який запускає на виконання програму R для статистичного аналізу даних та виводу результатів. Програма обчислює середні показники хмарності за днями для обраного місяця та графічно представляє результат зі зберіганням графіків у форматі PDF;

– **join** – очікує, поки закінчяться всі паралельні шляхи виконання робочого процесу;

– **email-success** – сповіщає про вдале виконання робочого процесу та надсилає результати;

– **email-error** – сповіщення у разі виникнення помилок та збою робочого процесу.

– **end** – успішне завершення робочого процесу;

– **fail** – примусове завершення робочого процесу, наприклад, у разі виникнення помилок.

Програму для обробки та аналізу супутникових знімків на платформі Apache Spark, що виконується у рамках спроектованого робочого процесу, було реалізовано мовою Java. Вона складається з таких основних кроків:

– завантаження зображень та налаштувань;

– перевірки потрапляння точок зображення, що аналізується, у вибрану область (вся територія України, Київська область і т. д.) з використанням масиву координат досліджуваної області за допомогою методу трасування променем;

– фільтрація точок зображення за маскою, для вилучення нанесених на карту кордонів регіонів та назв населених пунктів. Обчислення значення кольорів точок, що потрапили під маску виконується за допомогою обходу масиву точок за спіраллю на предмет пошуку найближчої точки, що не потрапила під маску;

– обхід масиву точок та визначення наявності в них хмарності (див. рис. 2) та її типу (конвективна, неконвективна) на підставі попадання кольору точки у відповідний діапазон кольорового спектру;

– обробка проміжних результатів для вилучення шумів;

– обрахунок площі хмарності різних типів та збереження результатів.

Далі наведено фрагмент згенерованого коду специфікації робочого процесу для цільової системи управління робочими процесами, у даному випадку Apache Oozie.

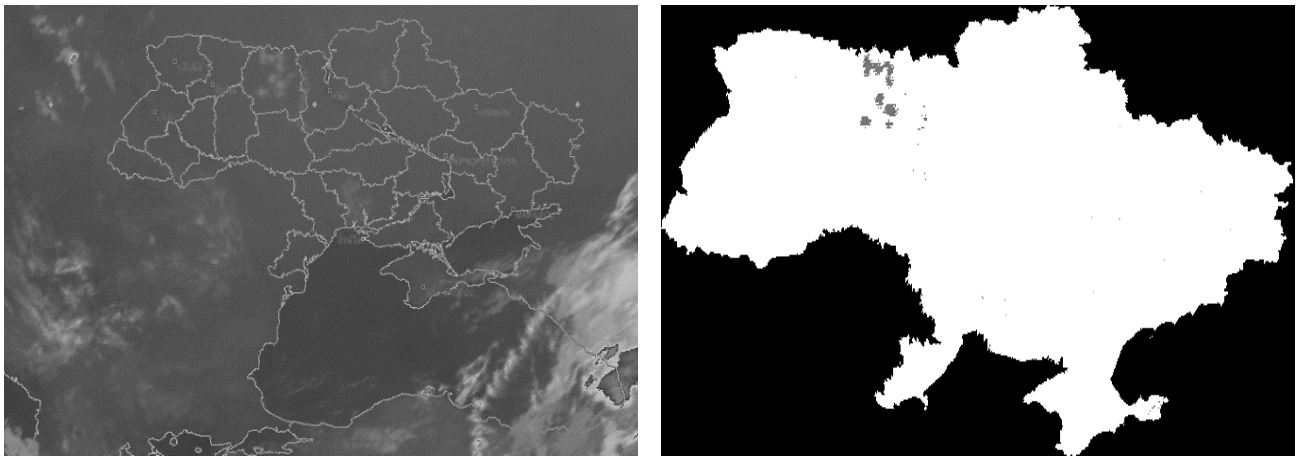


Рис. 2. Фрагмент супутникового знімку та результат фільтрації точок за маскою з ідентифікованою хмарністю

```
<workflow-app xmlns="uri:oozie:workflow:1.0" name="forecast-wf">
<startto="spark-job"/>
<actionname="spark-job">
<sparkxmlns="uri:oozie:spark-action:1.0">
...
</spark>
<okto="r-analyse"/>
<errorto="email-error"/>
</action>
<forkname="r-analyse">
<pathstart="r-mean-by-month" />
<pathstart="r-mean-by-day" />
<pathstart="r-mean-by-day-for-single-month" />
</fork>
<actionname="r-mean-by-month">
<shellxmlns="uri:oozie:shell-action:1.0">
...
</shell>
<okto="email-error"/>
<errorto="email-error"/>
</action>
    ...
<joinname="join-r-analyse" to="email-success"/>
<actionname="email-success">
    ...
</action>
<actionname="email-error">
    ...
</action>
<killname="fail">
    ...
</kill>
<endname="end"/>
</workflow-app>
```

В рамках роботи було обчислено показники хмарності для усієї території України та окремо для Київської області.

2. Результати експерименту

Розроблений в рамках даної роботи робочий процес було застосовано для обробки архіву супутникових знімків, збережених у форматі JPEG. Архів містить знімки за кожну добу з інтервалом 15 хвилин. Для прикладу було обрано вибірку за один рік з 01.03.2018 по 28.02.2019 об’ємом 8 Гб.

Для виконання робочого процесу використано фізичну обчислювальну машину з такими характеристиками:

– 4-ядерний процесор IntelCore i7-6700HQ з частотою 2.6 ГГц,

– об’єм оперативної пам’яті 32 Гб.

Для проведення експерименту було розгорнуто інфраструктуру Apache Hadoop у псевдорозподіленому режимі. Архів знімків було перенесено у розподілену файлову систему HDFS.

При випробуваннях було досліджено швидкість виконання робочого процесу при однаковому об’ємі вихідних даних для різної кількості потоків виконання програми Apache Spark.

Графік на рис. 3 показує залежність часу виконання робочого процесу для обрахунку показників хмарності по Київській області для різної кількості потоків виконання Apache Spark. Значення прискорення при подвоєнні кількості потоків у середньому складає 1,8.

У результаті проведених обчислень було отримано часові розподіли площі, вкритої хмарним покривом, у межах України різного ступеня деталізації. На рис. 4 та рис. 5 показано річний хід на території України та для березня 2018 р. середні добові значення у межах Київської області площ загальної (позначено суцільною лінією), конвективної (позна-

чено штриховою лінією) та неконвективної хмарності (позначено штрихпунктирною лінією). Отриманий річний хід (див. рис. 4) на якісному рівні добре узгоджується із даними багаторічних спостережень, приведеними у [2]. Так, найбільше хмар спостерігається у зимові місяці, весною хмарність зменшується, влітку тенденція зменшення продовжується (проте просторовий розподіл хмар більш неоднорідний, ніж зимою та весною; з'являються сприятливі умови для розвитку потужних конвективних хмар), восени хмарність починає збільшуватися. Ясно, що у межах окремо взятого місяця (див. рис. 5), протягом року та від одного року до іншого значення загальної хмарності та окремих її типів будуть суттєво відрізнятися. Проте відповідно до отриманих результатів, можна висловити припущення, що використання супутникових даних дає вищі значення хмарності, ніж значення отримані на основі наземних спостережень. Згідно [2] у зимові місяці майже на всій території країни близько 70 – 75 % небосхилу закрито хмарами, у нашому випадку за даними супутникових спостережень для періоду в один рік це – близько 90 %. Навесні за даними багаторічних наземних спостережень це – 60 %, у березні – травні 2018 р. за даними супутникових спостережень мало місце зменшення від 83 % до 28 % відповідно. Аналогічна схожість має місце і для інших сезонів. На рис. 6 приведено дані спостережень за загальною хмарністю метеостанції Київ для березня 2018 р. Розриви у кривій (лакуни) пояснюються відсутністю даних в архіві Українського гідрометеорологічного інституту ДСНС України та НАН України. Порівнюючи рис. 5 та 6 і беручи до уваги, що площа Київської області становить 28131 км^2, можемо бачити добре узгодження ходу загальної хмарності як за величиною площі, так і за ступенем покриття небосхилу. Проте є відмінність щодо настання мінімумів. У Києві вони спостерігались 11, 19 та 24 березня, а по області 13, 20 та 23 березня 2018 р.

Розбіжності, які мають місце, можна пояснити одночасним і рівномірним охопленням великої площі, відсутністю місцевих перешкод (закритості частини горизонту) для спостереження, а також проходженням атмосферних фронтів (коли над однією частиною точок наземних спостережень ще не спостерігається, а над іншою вже спостерігається суцільний хмарний покрив).

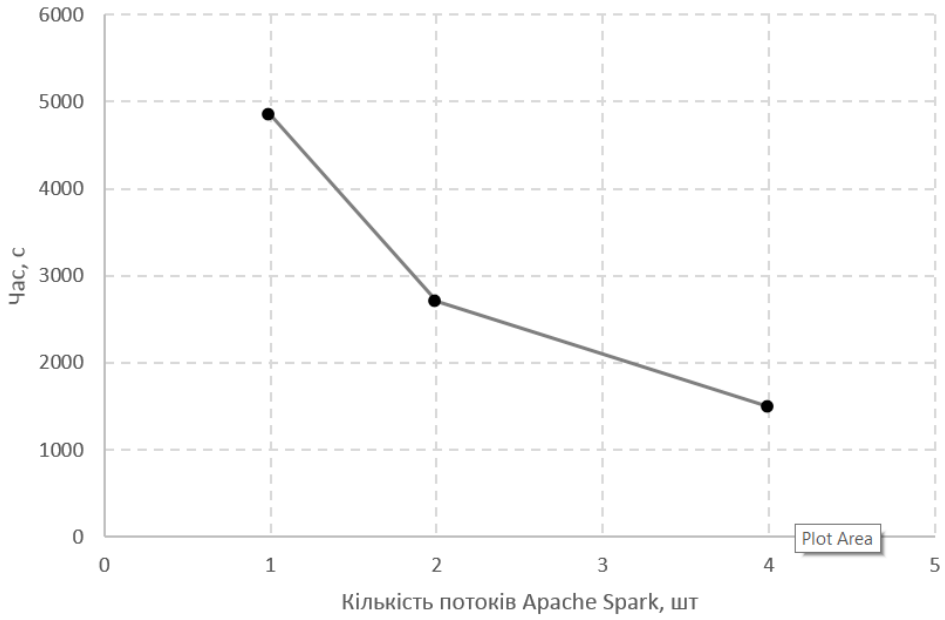


Рис. 3. Залежність часу виконання робочого процесу від кількості потоків виконання Apache Spark

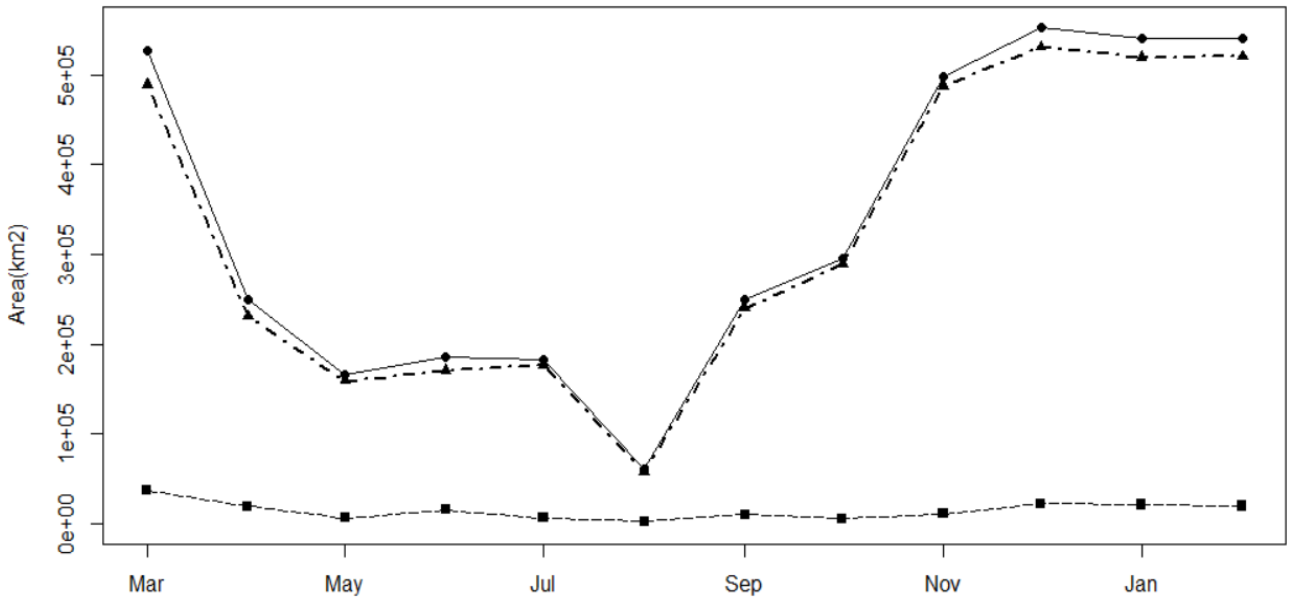


Рис. 4. Річний хід площ загальної, неконвективної та конвективної хмарності [10^5 км^2] на території України

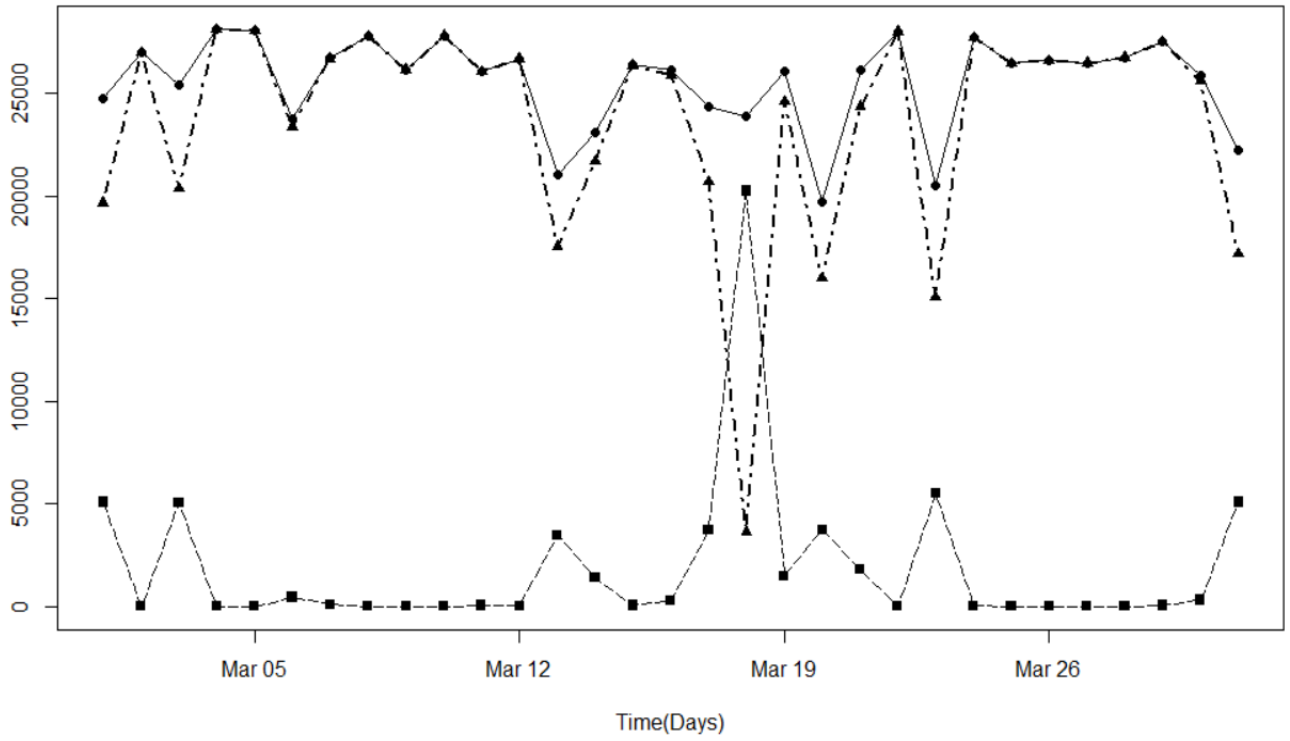


Рис. 5. Середні добові значення площ загальної, неконвективної та конвективної хмарності [км^2] на території Київської області протягом березня 2018 р.

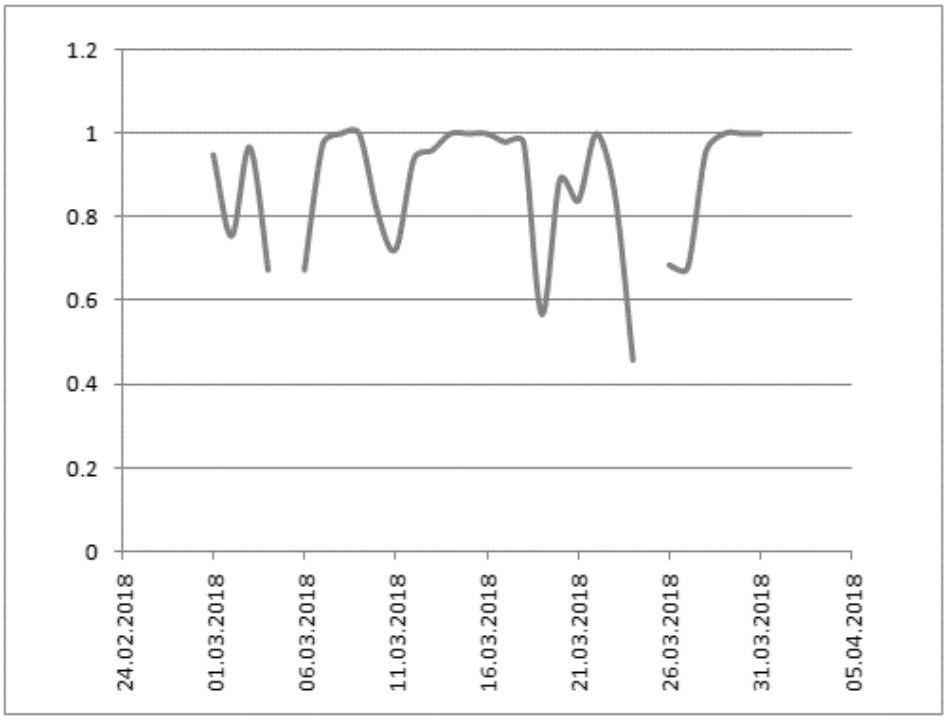


Рис. 6. Середня за добу загальна хмарність за даними спостережень у м. Київ у березні 2018 р.

Висновки

В даній роботі за допомогою онлайнового діалогового конструктору для автоматизації проектування робочих процесів ОКРП було спроектовано робочий процес для обробки архіву супутникових знімків з метою аналізу показників хмарності на території України. Робочий процес для реалізації поставленої задачі було розроблено для системи Apache Oozie, призначеної для управління роботами розподіленої платформи Apache Hadoop. З метою підтримки масштабованості та оптимізації обробки великого обсягу супутникових знімків для реалізації аналізу зображень було використано фреймворк Apache Spark. Проведено експеримент щодо виконання розробленого робочого процесу при різній кількості потоків Apache Spark, результати якого продемонстрували хороший показник ефективності розпаралелювання обчислень. Показано, що отримані у ході експерименту результати на якісному рівні добре узгоджуються із даними багаторічних наземних спостережень та фактичними даними спостережень. Висловлено припущення, що використання супутникових даних дає вищі значення хмарності, що пояснюється одночасним і рівномірним охопленням великої території, відсутністю місцевих перешкод (закритості частини горизонту) та проходженням атмосферних фронтів.

Література

1. Климат Украины [Под редакцией Г.Ф. Приходько, А.В. Ткаченко, В.Н. Бабиченко]. Ленинград: Гидрометеоиздат, 1967. 227 с.
2. Клімат України [За редакцією В.М. Ліпінського, В.А. Дячука, В.М. Бабіченко]. Київ: видавництво Раєвського, 2003. 344 с.
3. Заболоцька Т.М., Шпиг В.М. Трансформація баричного поля та хмарності у випадку тривалих і сильних опадів. *Наукові праці Українського науково-дослідного гідрометеорологічного інституту*. 2014. № 266. С. 12–19.
4. Заболоцька Т.М., Шпиг В.М. Кількісні зміни хмарності як показник тривалості періоду глобального потепління. *Наукові праці Українського науково-дослідного гід-*

рометеорологічного інституту. 2015. № 267. С. 23–27.
5. Shpyg V. et al. The application of regional NWP models to operational weather forecasting in Ukraine. CAS Technical Conference (TECO) on “Responding to the Environmental Stressors of the 21st Century”: 18–19 November 2013: Conf. Materials. 2013. URL: http://www.wmo.int/pages/prog/arep/cas/documents/Ukraine-NWPModels.pdf (дата звернення: 27.06.2019).
6. Хлебникова Е.И., Салль И.А. Особенности климатических изменений облачного покрова над территорией России. Метеорология и гидрология. 2009. № 7. С. 5–13.
7. Warren S.G., Eastman R.M., Hahn C.J. A survey of Changes in Cloud Cover and Cloud Types over Land from Surface Observations, 1971-96. Climate. 2007. N 20. P. 717–738.
8. Giovannettone J.P. and Barros A.P. Probing Regional Orographic Controls of Precipitation and Cloudiness in the Central Andes Using Satellite Data. *Journal of Hydrometeorology February*. 2009. Vol. 10, N 1. P. 167–182.
9. Sumargo E. and Cayan D. R. Variability of Cloudiness over Mountain Terrain in the Western United States. *Journal of Hydrometeorology*. 2017. Vol. 18, N 5. P. 1227–1245.
10. Apache Hadoop: сайт. URL: http://hadoop.apache.org/ (дата звернення: 1.06.2019).
11. Овдій О.М. До питання автоматизації проектування робочих процесів на основі алгебро-алгоритмічного та онтологічного інструментарію. *Проблеми програмування*. 2019. № 1. С. 37–47.
12. Дорошенко Е.А., Яценко Е.А. О синтезе программ на языке Java по алгеброалгоритмическим спецификациям. *Проблеми програмування.* 2006. № 4. С. 58–70.
13. Андон Ф.И., Дорошенко А.Е., Цейтлин Г.Е., Яценко Е.А. Алгеброалгоритмические модели и методы параллельного программирования. Киев: Академпериодика, 2007. 631 с.
14. Андон Ф.И., Дорошенко А.Е., Бекетов А.Г., Иовчев В.А., Яценко Е.А. Инструментальные средства автоматизации параллельного программирования на основе алгебры алгоритмов. *Кибернетика и системный анализ*. 2015. № 1. С. 162–170.
15. Дорошенко А.Ю., Іваненко П.А., Овдій О.М., Яценко О.А. Автоматизоване проектування програм для розв’язання задачі метеорологічного прогнозування. *Проблеми програмування.* 2016. № 1. С. 102–115.
16. Дорошенко Е.А., Овдей О.М., Яценко Е.А. Онтологические и алгеброалгоритмические средства автоматизации проектирования параллельных программ для "облачных" платформ. *Кибернетика и системный анализ.* 2017. Т. 53, № 2. С. 181–192.
17. Setvák M. et al. Blended "Sandwich" Image Products in Nowcasting. 2012 EUMETSAT Meteorological Satellite Conference. URL: https://www.eumetsat.int/website/wcm/idc/idcplg?IdcService=GET_FILE&dDocName=PDF_CONF_P61_S7_11_SETVAK_V&RevisionSelectionMethod=LatestReleased&Rendition=Web (дата звернення: 27.06.2019).
18. Workflow Management Coalition: сайт. URL: https://www.wfmc.org/ (дата звернення: 1.06.2019).
19. Apache Oozie Workflow Scheduler for Hadoop: сайт. URL: http://oozie.apache.org/ (дата звернення: 1.06.2019).
20. Apache Spark: сайт. URL: https://spark.apache.org/ (дата звернення: 1.06.2019).
21. White T. Hadoop: The Definitive Guide, 4th Edition. O'Reilly Media, Inc. 2015. 756p.
22. Dean J., Ghemawat S. Mapreduce: Simplified data processing on large clusters. In Proc. of the 6th USENIX OSDI. 2004. P. 137-150.
23. The R Project for Statistical Computing: сайт. URL: https://www.r-project.org/ (дата звернення: 1.06.2019).

References

1. Climate of Ukraine [Ed. by G.F. Prikhodko, A.V. Tkachenko, V.N. Babichenko] (1967). Leningrad: Hydrometeoizdat. 227 p. (in Russian).
2. Climate of Ukraine [Ed. by V.M. Lipinsky, V.A. Diachuk, V.M. Babichenko] (2003). Kyiv: Publishing House of Raevsky. 344 p. (in Ukrainian).
3. Zabolotska, T.M., Shpyg V.M. (2014). Transformation of baric field and cloudiness in the case of long-term and heavy precipitation. Scientific Proceedings of the UHMI. 266. P. 12–19. (in Ukrainian).
4. Zabolotska, T.M., Shpyg V.M. (2015). Quantitative changes of cloud cover as indicator of global warming period. Scientific Proceedings of the UHMI. 267. P. 23–27. (in Ukrainian).
5. Shpyg V. et al. (2013). The application of regional NWP models to operational weather forecasting in Ukraine. CAS Technical Con-

ference (TECO) on “Responding to the Environmental Stressors of the 21st Century”. [online] Available from: http://www.wmo.int/pages/prog/arep/cas/documents/Ukraine-NWPModels.pdf [Accessed 27 Jun. 2019].
6. Khlebnikova E.I., Sall I.A. (2009). Peculiarities of climatic changes in cloud cover over the Russian Federation. Russian Meteorology and Hydrology. 7. P. 5–13. (in Russian).
7. Warren S.G., Eastman R.M., Hahn C.J. (2007). A survey of Changes in Cloud Cover and Cloud Types over Land from Surface Observations, 1971-96. Climate. N 20. P. 717–738.
8. Giovannettone J.P. and Barros A.P. (2009). Probing Regional Orographic Controls of Precipitation and Cloudiness in the Central Andes Using Satellite Data. Journal of Hydrometeorology February. Vol. 10, N 1. P. 167–182.
9. Sumargo E. and Cayan D.R. (2017). Variability of Cloudiness over Mountain Terrain in the Western United States. Journal of Hydrometeorology. Vol. 18, N 5. P. 1227–1245.
10. Hadoop.apache.org. Apache Hadoop Official Website. [online] Available from: http://hadoop.apache.org/ [Accessed 1 Jun. 2019].
11. Ovdii, O.M. (2018) On the issue of automating the workflow design based on algebra-algorithmic and ontological tools. Problems in programming. (1). P. 37–47. (in Ukrainian).
12. Doroshenko, A.Yu. & Yatsenko O.A. (2006) About the synthesis of Java programs by algebra-algorithmic specifications. Problems in programming. (4). P. 58–70. (in Russian).
13. Andon, P.I. et al. (2007) Algebra-algorithmic models and methods of parallel programming. Kiev: Academperiodika. (in Russian).
14. Andon, P.I., Doroshenko, A.Yu., Beketov, O.G., Iovchev, V.O. & Yatsenko O.A. (2015) Software tools for automation of parallel programming on the basis of algebra of algorithms. Cybernetics and systems analysis. (1). P. 162–170. (in Russian).
15. Doroshenko, A.Yu., Ivanenko, P.A., Ovdii, O.M., & Yatsenko, O.A. (2016) Automated design of programs for solving the task of meteorological forecasting. Problems in programming. (1). P. 102–115. (in Ukrainian).
16. Doroshenko, A.Yu., Ovdii, O.M. & Yatsenko O.A. (2017) Ontological and algebra-algorithmic tools for automated design of parallel programs for cloud platforms. Cybernetics and Systems Analysis. 53(2). P. 181–192. (in Russian).
17. Setvák M. et al. (2012). Blended "Sandwich" Image Products in Nowcasting. EUMETSAT Meteorological Satellite Conference. [online] Available from: https://www.eumetsat.int/website/wcm/idc/idcplg?IdcService=GET_FILE&dDocName=PDF_CONF_P61_S7_11_SETVAK_V&RevisionSelectionMethod=LatestReleased&Rendition=Web [Accessed 27 Jun. 2019].
18. Wfmc.org. Workflow Management Coalition. [online] Available from: https://www.wfmc.org/ [Accessed 1 Jun. 2019].
19. Oozie.apache.org. Apache Oozie Workflow Scheduler for Hadoop Official Website. [online] Available from: http://oozie.apache.org/ [Accessed 1 Jun. 2019].
20. Spark.apache.org Unified analytics engine for BigData.[online] Available from: https://spark.apache.org/ [Accessed 1 Jun. 2019].
21. White T. (2015) Hadoop: The Definitive Guide, 4th Edition. O'Reilly Media, Inc.
22. Dean J. & Ghemawat S. (2004) Mapreduce: Simplified data processing on large clusters. In: 6th USENIX OSDI, pp. 137-150.
23. R-project.org. The R Project for Statistical Computing Official Website. [online] Available from: https://www.r-project.org/ [Accessed 1 Jun. 2019].

Одержано 01.07.2019

Про авторів:

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

Шпиг Віталій Михайлович,
кандидат географічних наук,
завідувач відділу фізики атмосфери
Українського гідрометеорологічного
інституту ДСНС України та НАН України,
експерт Комісії з атмосферних наук
Всесвітньої метеорологічної організації.
Кількість наукових публікацій в
українських виданнях – понад 40.
Кількість наукових публікацій в
зарубіжних виданнях – понад 50.
Індекс Хірша – 2.
https://orcid.org/0000-0003-1055-7120,

Овдій Ольга Михайлівна,
молодший науковий співробітник
Інституту програмних систем
НАН України.
Кількість наукових публікацій в
українських виданнях – 22.
Кількість наукових публікацій в
зарубіжних виданнях – 5.
http://orcid.org/0000-0002-8891-7002.

Місце роботи авторів:

Інститут програмних систем
НАН України,
03187, м. Київ-187,
проспект Академіка Глушкова, 40.
Тел.: (38)(044) 526 6033.
E-mail: doroshenkoanatoliy2@gmail.com,
olga.ovdiy@gmail.com;

Український гідрометеорологічний
Інститут ДСНС України та НАН України,
03028, м. Київ,
проспект Науки, 37.
Тел.: (38)(044) 525 8630.
E-mail: vitold82@i.ua.

УДК 004.8

Модифікація метамови нормальних форм знань / О.П. Кургаєв. – С. 3 – 10.

Набір базових відношень метамови нормальних форм знань (альтернатива, конкатенація, заперечення й ітерація) розширено двома новими відношеннями: обов'язковості (не нульовим числом повторень), необов'язковості деякої структури й структурними дужками. Уведення нових відношень виконано описом їхніх структур у базових відношеннях метамови нормальних форм знань. Дано текстову й графічну форми самоопису модифікованої метамови нормальних форм знань, розширеної стилістичними відношеннями опису інформаційних структур. Наведено з мінімально необхідними коментарями приклади графічних варіантів подання структур нових відношень термінів, використаних у самоопису модифікованої метамови нормальних форм знань.

Ключові слова: модифікована метамова, самоопис метамови, граф самоопису метамови, відношення обов'язковості, відношення необов'язковості, структурні дужки.

UDC 004.8

The modification of the meta-language of normal forms of knowledge / A.F. Kurgaev. – P. 3 – 10.

The set of the relations of normal forms of knowledge (alternative, concatenation, negation and iteration) is expanded by the two new relations: commitment relation – non-zero number of repetitions of some structure, non-commitment relation of some structure and structure brackets. The introduction of new relations is implemented by describing their structures with the basic relations of the meta-language of normal forms of knowledge. The text and graphical descriptions of the meta-language of normal forms of knowledge are presented, extended by the stylistic relations of the information structure descriptions. The examples of graphical variants of representing the structures of new relations between terms, which are used to self-describe the modified meta-language of normal forms of knowledge are presented with the minimally required comments.

Key words: modified meta-language, self-description of the meta-language, self-description graph of the meta-language, commitment relation, non-commitment relation, structure brackets.

УДК 004.42:510.69

Отношения логического следствия в логиках частичных предикатов с композицией предикатного дополнения / О.С. Шкильняк. – С. 11 – 27.

В работе исследованы программно-ориентированные логики частичных предикатов, названные LC. Характерной их особенностью является нали-

UDC 004.42:510.69

Relations of logical consequence in logics of partial predicates with composition of predicate complement / O.S. Shkilniak. – P. 11 –27.

In this paper we study software-oriented logics of partial predicates with new special non-monotonic operation (composition) of the predicate comple-

чие новой операции (композиции) предикатного дополнения $\tilde{}$. Операции такого типа используются в различных вариантах программных логик Флойда-Хоара с частичными перед- и после-условиями. Описаны композиционные алгебры и языки LC. Предложен ряд отношений логического следствия в LC (типов $|=_T$, $|=_F$, $|=_{TF}$, а также отношение ${}^{Pc}|=_{IR}$) и отношений логического следствия в условиях неопределенности (типов $|=_T^{\perp}$, $|=_F^{\perp}$, $|=_{TF}^{\perp}$). Исследованы свойства этих отношений, установлены соотношения между ними. Для предложенных отношений описаны условия их гарантированного наличия, приведены свойства декомпозиции формул и свойства элиминации кванторов. Для отношений типов $|=_T$ и $|=_F$ доказана теорема элиминации условий неопределенности. Это позволяет свести отношения ${}^{P}|=_T^{\perp}$, ${}^{P}|=_F^{\perp}$, ${}^{R}|=_T^{\perp}$, ${}^{R}|=_F^{\perp}$ к отношениям ${}^{Pc}|=_T$, ${}^{Pc}|=_F$, ${}^{Rc}|=_T$, ${}^{Rc}|=_F$. В то же время отношение $|=_{IR}^{\perp}$ невозможно свести к отношению ${}^{Pc}|=_{IR}$. Более того, для ${}^{Pc}|=_{IR}$ невозможно корректно определить условия декомпозиции формул $\tilde{}\Phi$. Установленные свойства LC свидетельствуют о ее существенном отличии от традиционной логики квазиарных предикатов. Для предложенных отношений логического следствия в LC и отношений логического следствия в условиях неопределенности в последующих работах планируется построение первопорядковых исчислений секвенциального типа.

Ключевые слова: логика, частичный предикат, композиционная алгебра, логическое следствие.

УДК 004.42:510.69

Первопорядковые композиционно-номинативные логики с сущностями слабой и строгой равности / С.С. Шкильняк. – С. 28 – 44.

В работе исследованы новые программно-ориентированные логические

ment. We denote these logics by LC and composition of the predicate complement by $\tilde{}$. Such operations are used in various versions of the Floyd-Hoare program logic with partial pre- and post-conditions. We describe first order composition algebras and LC languages. For LC, a number of logical consequence relations (${}^{Pc}|=_T$, ${}^{Pc}|=_F$, ${}^{Rc}|=_T$, ${}^{Rc}|=_F$, ${}^{Pc}|=_{TF}$, ${}^{Rc}|=_{TF}$, ${}^{Pc}|=_{IR}$) and logical consequence relations under the conditions of undefinedness (${}^{P}|=_T^{\perp}$, ${}^{P}|=_F^{\perp}$, ${}^{R}|=_T^{\perp}$, ${}^{R}|=_F^{\perp}$, ${}^{P}|=_{TF}^{\perp}$, ${}^{R}|=_{TF}^{\perp}$) are specified. Properties of the defined relations are investigated, differences and the relationship between them are given. For the introduced relations, we describe the conditions for their guaranteed presence, the decomposition conditions for formulas and the properties of quantifier elimination. The theorem of elimination of the conditions of undefinedness for the relations $|=_T^{\perp}$ and $|=_F$ is proved. Thus, the relations ${}^{P}|=_T^{\perp}$, ${}^{P}|=_F^{\perp}$, ${}^{R}|=_T^{\perp}$ and ${}^{R}|=_F^{\perp}$ can be expressed by ${}^{Pc}|=_T$, ${}^{Pc}|=_F$, ${}^{Rc}|=_T$ and ${}^{Rc}|=_F$ respectively. However, it is shown that $|=_{IR}^{\perp}$ cannot be expressed by ${}^{Pc}|=_{IR}$. Moreover, it is impossible to define correctly the decomposition conditions for $\tilde{}\Phi$ formulas for ${}^{Pc}|=_{IR}$. Properties of decomposition conditions for $\tilde{}\Phi$ formulas are different for the relations $|=_T$ and $|=_F$, therefore properties of decomposition and equivalent transformations must be specified indirectly through the corresponding properties of $|=_T$ and $|=_F$. First order sequent calculi for the introduced logical consequence relations for LC and logical consequence relations under the conditions of undefinedness will be constructed in in the forthcoming articles.

Key words: logic, partial predicate, composition algebra, logical consequence.

UDC 004.42:510.69

First-order composition-nominative logics with predicates of weak equality and of strong equality / S.S. Shkilniak. – P. 28 – 44.

Development of the new software-oriented logical formalisms is a topical

формализмы – композиционно-номинативные логики с предикатами равенства и композицией предикатного дополнения, такие логики названы LCE. Можно выделить предикаты слабого равенства и строгого равенства. Отсюда получаем LCE с предикатами слабого равенства, их назовем LCEw, и LCE с предикатами строгого равенства, их назовем LCEs. LCE можно рассматривать на первопорядковом уровне и реноминативном уровне. Рассмотрены композиционные алгебры LCE, исследованы свойства их композиций, описаны первопорядковые языки этих логик. Основное внимание сосредоточено на исследовании свойств, связанных с предикатами равенства и композицией предикатного дополнения. Введен и исследован ряд отношений логического следствия в первопорядковых LCE, рассмотрены их особенности. В частности, установлена определенная вырожденность LCEw, для которой корректным оказывается только отношение неопровержимого логического следствия в условиях неопределенности. Подробное исследование в LCE отношений логического следствия в условиях неопределенности будет проведено в последующих роботах. Свойства отношений логического следствия в LCE является семантической основой построения соответствующих исчислений секвенциального типа. Такое построения тоже планируется осуществить в последующих роботах.

Ключевые слова: логика, композиция, предикат, равенство, логическое следствие.

problem. The paper introduces logics of partial predicates with predicate complement and equality predicates, we denote them LCE. They extend logics of quasiary predicates with equality and logics with predicate complement. The composition of the predicate complement is used in Floyd-Hoare program logics' extensions on the class of partial predicates. We define predicates of weak equality and of strong equality. Thus, LCE with predicates of weak equality (denoted by LCEw) and LCE with predicates of strong equality (denoted by LCEs) can be specified. LCE can be studied on the first order and renominative levels. We consider composition algebras of LCE, investigate properties of their compositions and describe first order languages of such logics. We concentrate on the properties related to the equality predicates and the composition of the predicate complement. Various variants of logical consequence relations for the first order LCE are introduced and studied: ${}^{P}|{=}_{T}$, ${}^{P}|{=}_{F}$, ${}^{R}|{=}_{T}$, ${}^{R}|{=}_{F}$, ${}^{P}|{=}_{TF}$, ${}^{R}|{=}_{TF}$, ${}^{P}|{=}_{IR}$. In particular, we obtained that LCEw are somewhat degenerate, as for them all the relations are incorrect except for the irrefutability logical consequence relation under the conditions of undefinedness $|{=}_{IR}^{\perp}$. At the same time, all of the listed relations are correct for LCEs. Properties of the logical consequence relations are the semantic basis for construction of the respective calculi of sequential type. Further investigation of logical consequence relations for LCE includes adding the conditions of undefinedness and constructing the corresponding sequent calculi; it is planned to be displayed in the forthcoming articles.

Key words: logic, predicate, composition, equality, logical consequence.

УДК 502:004.45 (075.8)

UDC 502:004.45 (075.8)

Основи програмування в контексті інженерії програмного забезпечення / М.О. Сидоров. – С. 45 – 57.

Basics of programming in the context of software engineering / M.O. Sydorov. – P. 45 – 57.

У статті розглядається застосування конструкційного підходу до побудови

The article discusses the use of a constructive approach to building a program

програми, який систематично культивується в інженерії програмного забезпечення і стало можливим завдяки низки фундаментальних результатів, отриманих в теорії програмування. По-перше, на основі відомої структурної теореми, аргументовано відмовилися від використання оператора go to і запропонували метод структурного програмування, що забезпечило реальний шлях до створення зрозумілих програм. По-друге, поняття підпрограми, хоча і використовувалося тільки для зменшення рутинної роботи в процесі програмування, стало першим засобом модульного представлення програм. Пізніше блок і підпрограма склали основу блок-орієнтованих (процедурних, підпрограмних) мов і методу процедурного (підпрограмного) програмування. По-третє, для відповіді на питання, що відносяться до визначення меж, розмірів і устрою модуля ввели поняття зв'язування частин, що складають модуль і з'єднання для вказівки з'єднання між модулями; конкретні критерії модулярізації; запропонували устрій модуля на основі поняття приховування інформації. Модуль реалізований в мові програмування Modula, а пізніше Modula-2, в яких використовувалося поняття модуля на основі методу модульного (композиційного) програмування. При розробці мови Simula 67, були закладені основи об'єктно-орієнтованих мов, які отримали розвиток завдяки роботам по концепціям успадкування, пізнього зв'язування і посилань і були завершені розробкою об'єктно-орієнтованих мов і методом об'єктно-орієнтованого (класифікаційного) програмування. Таким чином, була створена основа для повторного, багаторазового використання і компонентної розробки програмного забезпечення. Зараз ці роботи розвиваються в напрямку дослідження і створення програмного забезпечення як системи систем (system of systems), використовуючи зв'язок системного аналізу та інженерії програмного забезпечення, і розвиваючи системну інженерію програмного забезпечення. У статті, для навчання основам програмування, як засіб, що дозволяє уточнити поняття програмної конструкції, використовується класифікація, а як класифікаційна ознака – рівень інкапсуляції, який будується на основі принципів інженерії програмного забезпечення – інкапсуляції і багаторівневого подання, Застосовуючи принцип інкапсуляції

that is systematically cultivated in software engineering and made possible by a number of fundamental results obtained in programming theory. On the first, based on the structural theorem, reasonably was refused to use the go to operator and proposed a structured programming method, which provided a real way to create understandable programs. On the second, the concept of a subroutine, although it was used only to reduce routine work in the programming process, was the first means of modular presentation of programs. Later, the block and the subroutine formed the basis of block-oriented (procedural, subroutine) languages and the procedural (subroutine) programming method. In the third, to answer questions related to the definition of boundaries, sizes, and building module, the concept of connecting parts that make up a module and connections to indicate the connection between modules; specific modularization criteria; proposed a device module based on the concept of information hiding were introduced. The module is implemented in the programming language Modula, and later Modula-2. The concept of a module based on the modular (compositional) programming method. In the Simula 67 language, the foundations of object-oriented languages were introduced. They developed through the work on the concepts of inheritance, late binding and links, and were completed by the development of object-oriented languages and object-oriented (classification) programming. Thus, a framework for reuse, systematic reuse and component software development was created. Now these works are developing in the direction of research and creation of software as system of systems, using the link between system analysis and software engineering, and developing software system engineering. In the article, for learning the basics of programming, as a tool that allows clarifying the concept of program design, classification is used. As a classification feature - the level of encapsulation, which is based on the principles of software engineering – encapsulation and multi-level presentation -, is used. Applying the principle of encapsulation at different levels of representation program structure corresponding to different degrees of software abstraction, the concept of encapsulation level was obtained. Using this concept, you can find out the types of software constructions and the corresponding methods of programming (construction) programs. Using intro-

на різних рівнях представлення структури програми, які відповідають різним ступеням абстракції програмного забезпечення, отримано поняття рівня інкапсуляції. Скориставшись цим поняттям, можна з'ясувати типи програмних конструкцій і відповідні методи програмування (конструювання) програм. На основі введених понять і конструкційного підходу до побудови програми автором створена дидактика основ програмування, яка впроваджена шляхом використання в лекціях для студентів спеціальності «Інженерія програмного забезпечення» (121) і при написанні автором навчального посібника для студентів і аспірантів, зазначеної спеціальності по основам програмування.

Ключові слова: навчання, основи програмування, мови програмування, інженерія програмного забезпечення.

duced the concepts and the constructive approach to building the program, the author created the didactics of the basics of programming, which was deployment through lectures for students of the specialty “Software Engineering” (121) and a textbook of the author for students and post-graduate students of the specialty.

Key words: learning, basics of programming, programming languages, software engineering.

УДК 004.855:519.216

UDC 004.855:519.216

Задачи и методы анализа больших данных (обзор) / А.С. Балабанов. – С. 58 – 85.

Tasks and methods of Big Data analysis (a survey) / O.S. Balabanov. – P. 58 – 85.

Рассмотрены основные задачи и методы глубокого анализа больших данных. В изложении сделан акцент на «физическом» смысле задач и методов, без математических деталей. Спектр анализа и использования больших данных охватывает четыре концептуальных класса заданий: «интеллектуальный» поиск информации; массированную (конвейерную) переработку данных; экстракцию знаний из данных (открытие закономерностей) и индукцию модели объекта (среды). Освещено суть типичных классов задач большой аналитики: группирование случаев (кластеризация данных); вывод целе-определенных моделей (классификация, регрессия); вывод генеративных моделей; открытие структур и закономерностей. Рассмотрены ключевые методы кластеризации, регрессии и классификации (включая глубокое обучение), а также вывод генеративных

We review tasks and methods most relevant to Big Data analysis. Emphasis is made on the conceptual and pragmatic issues of the tasks and methods (avoiding unnecessary mathematical details). We suggest that all scope of jobs with Big Data fall into four conceptual modes (types): four modes of large-scale usage of Big Data: 1) intelligent information retrieval; 2) massive (large-scale) conveyed data processing (mining); 3) model inference from data; 4) knowledge extraction from data (regularities detection and structures discovery). The essence of various tasks (clustering, regression, generative model inference, structures discovery etc.) are elucidated. We compare key methods of clustering, regression, classification, deep learning, generative model inference and causal discovery. Cluster analysis may be divided into methods based on mean distance, methods based on local distance and methods based on a model. The targeted (predic-

моделей. Методы решения целеопределенных задач делятся на те, что выводят модель в явном виде (модель «отделяется» от данных) и та методы, «привязанные к данным». Охарактеризовано особенности анализа темпоральных данных (сегментация, выявление точек изменения и т. д.). Детальнее изложено индуктивный вывод каузальных сетей методами, основанными на независимости. Указаны особенности вывода динамических каузальных сетей. Отдельно подытожены общие особенности применения статистических методов в анализе больших данных.

Ключевые слова: большие данные, анализ данных, вывод генеративной модели, статистические методы, кластеризация, регрессия, прогноз, открытие закономерностей, темпоральные данные, каузальные сети.

УДК 004.62

Що таке Big Data / В.А. Резніченко. – С. 86 – 100.

У статті робиться спроба розкриття суті поняття Big Data на основі аналізу матеріалів з різних джерел. Даються визначальні характеристики Big Data, наводиться їх класифікація, коротко описується історія виникнення та розвитку, представлені основоположні принципи роботи, коротко викладаються методи і технології аналізу та візуалізації, описується життєвий цикл управління даними з використанням технології Big Data.

Ключові слова: великі дані, технологія великих даних, база даних, життєвий цикл.

tive) methods fall into two categories: methods which infer a model; “tied to data” methods which compute prediction directly from data. Common tasks of temporal data analysis are briefly overviewed. Among diverse methods of generative model inference we make focus on causal network learning because models of this class are very expressive, flexible and are able to predict effects of interventions under varying conditions. Independence-based approach to causal network inference from data is characterized. We give a few comments on specificity of task of dynamical causal network inference from timeseries. Challenges of Big Data analysis raised by data multidimensionality, heterogeneity and huge volume are presented. Some statistical issues related to the challenges are summarized.

Key words: Big Data, data analysis, generative model inference, statistical methods, clustering, regression, prediction, pattern discovery, temporal data, causal networks.

UDC 004.62

What is Big Data / V.A. Reznichenko. – P. 86 – 100.

The article attempts to uncover the essence of the concept of Big Data based on the analysis of materials from various sources. Defining characteristics of Big Data are given, their classification is given, a brief description of the origin and development is presented, the basic principles of operation are presented, methods and technologies for analysis and visualization are outlined, the life cycle of data management using Big Data technology is described.

Key words: Big Data, Big Data technology, database, life cycle.

УДК 004.94

Платформы больших данных. Основные задачи, свойства и достоинства / О. Захарова. – С. 101 – 115.

Данная работа представляет обзор существующих платформ больших данных. Цель состоит в определении основных проблем и решений, существующих в этой области, а также свойств платформ больших данных, которые определяют их возможности, достоинства или недостатки в решении этих проблем. Актуальность темы обусловлена стремительным развитием мобильных устройств и прикладных систем, соответствующим ростом объемов информации и неспособностью традиционных систем обрабатывать такие объемы данных в приемлемые сроки. То есть, это платформа информационных технологий класса предприятия, которая обеспечивает свойства и функциональность прикладной системы в одном решении для разработки, разворачивания, обработки и управления большими данными. Цель создания и использования таких платформ заключается в улучшении масштабируемости, доступности, производительности и безопасности организаций, работающих з большими данными. Платформы больших данных позволяют обрабатывать объемные многоструктурные данные в режиме реального времени, а также разным пользователям использовать их для выполнения разных задач с использованием больших данных. Рассматриваются фреймворки, разработанные для решения задач больших данных, анализируются их характеристики, принципы работы, возможности в контексте проблем, которые они способны решать, определяются существующие «пробелы» и направления развития. Решение проблем больших данных, а именно обеспечение эффективного хранения, обработки и анализа данных, позволит сделать информацию более полезной, а предприятия, работающие с большими данными, более конкурентноспособными.

Ключевые слова: платформа больших данных, машинное обучение, Hadoop,

UDC 004.94

Big data platforms. Main objectives, features and advantages / O. Zakharova. – P. 101 – 115.

This paper presents an overview of existing big data platforms. The goal is to identify the main problems and solutions that exist in this area, as well as the properties of the big data platforms that determine their capabilities, advantages or weaknesses in solving these problems. The relevance of the topic is due to the rapid evaluation of mobile devices and application systems, the corresponding increase in the volume of information and the inability of traditional systems to process such amounts of data in a reasonable time. That is, it is an information technology platform enterprise class that provides the properties and functionality of an application in one solution for developing, deploying, processing and managing big data. The goal of creating and using such platforms is to improve the scalability, availability, performance, and security of organizations working with big data. Big data platforms enable to process multi-structured data in real time and allow different users to use them for various tasks related to using big data. The paper discusses frameworks developed for solving big data problems, analyzes their characteristics, operating principles and capabilities in the context of the problems they are able to solve, it also identifies existing “gaps” and directions for further development. Solving the problems of big data, namely ensuring the effective storage, processing and analysis of data, will make information more useful, and companies that work with big data more competitive.

Key words: big data platform, machine learning, Apache Hadoop, document-oriented storage, «key-value» storage, column storage, graph storage, data management, distributed storage, stream computing, NoSQL data bases, distributed file system, MapReduce, TaskTracker, JobTracker, Apache Spark, big data processing, big data analytics.

документоориентированное хранилище, хранилище типа «ключ-значение», колоночное хранилище, графовое хранилище, управление данными, распределенное хранение, потоковые вычисления, NoSQL базы данных, распределенная файловая система, MapReduce, TaskTracker, JobTracker, Apache Spark, обработка больших данных, аналитика больших данных.

УДК 004.932.2:[551.501:551.576.2]

Программная система анализа облачности по данным спутниковых наблюдений / А.Е. Дорошенко, В.М. Шпиг, О.М. Овдей. – С. 116–126.

В данной работе с помощью онлайнового диалогового конструктора для автоматизации проектирования рабочих процессов ОКРП был спроектирован рабочий процесс для обработки архива спутниковых снимков с целью анализа показателей облачности на территории Украины. Рабочий процесс для реализации поставленной задачи был разработан для системы Apache Oozie, предназначенной для управления работами распределенной платформы Apache Hadoop. С целью поддержки масштабируемости и оптимизации обработки большого объема спутниковых снимков для реализации анализа изображений был использован фреймворк Apache Spark. Проведен эксперимент по выполнению разработанного рабочего процесса при различном количестве потоков Apache Spark, результаты которого продемонстрировали хороший показатель эффективности распараллеливания вычислений. Показано, что полученные в ходе эксперимента результаты на качественном уровне хорошо согласуются с данными многолетних наземных наблюдений и фактическими данными наблюдений.

Ключевые слова: распределенные вычисления, рабочие процессы, Apache Hadoop, Apache Spark, анализ изображений, мониторинг, облачность.

UDC 004.932.2:[551.501:551.576.2]

Software system for analyzing cloudiness based on satellite observations / A.Yu. Doroshenko, V.M. Shpyg, O.M. Ovdii. – P. 116–126.

The article presents software system for analyzing cloudiness indicators in Ukraine based on satellite observations. This system was developed using an online dialog designer for automated workflow design based on algebra-algorithmic and ontological tools ODWF. The workflow for the implementation of the task was designed for Apache Oozie and distributed platform Apache Hadoop. In order to support scalability and optimize the processing of large volumes of satellite images, the Apache Spark framework was used to implement image analysis. An experiment was conducted to run the developed workflow on a different number of Apache Spark threads, the results of which demonstrated a good efficiency of calculation parallelization. It is shown that the results obtained during the experiment at a qualitative level are in good agreement with the data of long-term ground-based observations and actual observational data.

Key words: distributed computing, workflow, Apache Hadoop, Apache Spark, image analysis, monitoring, cloudiness.

ДО УВАГИ АВТОРІВ!

У журналі "Проблеми програмування" публікуються наукові матеріали, які раніше не публікувалися в інших виданнях.

Мова статті: українська, російська, англійська. Обсяг статті — від 6 до 16 сторінок формату А4.

Документ зберігається у форматі doc або docx. Ім'я подається транслітерацією, як прізвище автора (авторів), наприклад, «Petrenko.doc».

Автори можуть користуватися електронною поштою і також телефаксом для ділової переписки та передачі до редакції тексту статті та правки при коректурі. E-mail редакції: tsok@isofts.kiev.ua. FAX: +380 (44) 526 6263, Телефон: 526 5065.

1. Оформлення файлу з текстом статті.

При підготовці файлу використовуються: стиль нормальний (звичайний) або normal; шрифт Times New Roman, розмір шрифта 12 пт.; міжрядковий інтервал – 1,0; абзацний відступ – 1,25 см; вирівнювання – по ширині. У тексті не допускається вирівнювання пропусками; розстановка переносів – автоматична. Формат паперу А4, розміри полів документа – 20 мм. Текст статті після анотації має бути оформлений у 2 колонки, ширина яких – 7,86 см, а пробіл між ними – 1,27 см.

2. Послідовність розміщення та оформлення матеріалу статті.

УДК: індекс за універсальною десятковою класифікацією.

Автори: ініціали та прізвища авторів, курсив (світлий).

Заголовок 1 **(назва статті):** не містить абревіатур та строго відповідає змісту статті. Шрифт 15 пт, напівжирний, регістр верхній.

Анотація **(мовою статті):** 50–100 слів, не містить абревіатур, зрозумілих із змісту статті. Шрифт 10 пт, звичайний.

Ключові слова **(мовою статті):** не більше10 слів, не містить абревіатур, зрозумілих із змісту статті, подаються в називному відмінку, розділені комами. Шрифт 10 пт, звичайний.

Заголовок 2 **(назва розділу):** шрифт 14 пт, напівжирний; абзац із центральним вирівнюванням, без переносів. Заголовки нижчого рівня (пункти і т. п.) у самостійний абзац не виділяються і проходять першим реченням текстового абзацу, шрифт 12 пт, напівжирний.

Основний текст статті, має такі необхідні елементи:

постановка проблеми в загальному вигляді і її зв'язок з важливими науковими або практичними завданнями;

аналіз останніх досліджень і публікацій, у яких розпочато рішення даної проблеми і на які спирається автор, виділення невирішених раніше частин загальної проблеми, яким присвячується дана стаття;

формулювання цілей статті (постановка задачі);

виклад основного матеріалу дослідження з повним обґрунтуванням отриманих наукових результатів;

висновки з даного дослідження і перспективи подальших розробок у даному напрямку;

подяка (за наявності такої).

Формули створюються в редакторі Microsoft Equation 3.0 або MathType. Формули, на які є посилання в тексті, повинні мати наскрізну нумерацію. Номер формули друкується в круглих дужках біля краю правого поля. Розмір основного шрифту редактора формул – 12 пт. Розміри символів у формулах: звичайний – 12 пт, великий індекс – 9 пт, дрібний індекс – 7 пт, великий символ – 18 пт, дрібний символ – 11 пт. Не допускається масштабування формульних об'єктів.

Рисунки мають бути створені вбудованим редактором Word Picture або експортовані з прикладних програм Windows у графічних форматах (bmp, pcx, gif, jpg або tif). Рисунки розташовуються по центру. Нумерація рисунків здійснюється відповідно до порядку

згадування у тексті. Нумеровані підписи розміщуються під рисунком з позначенням «Рис. », далі вказується номер рисунка і текст підпису.

Таблиці мають бути підготовлені стандартним вбудованим в Word інструментарієм "Таблиця". Таблиці нумеруються за порядком згадування. На номер таблиці повинно бути посилання в тексті. Номер таблиці вказується в окремому рядку з вирівнюванням по правій стороні (наприклад, «Таблиця 1»). Назви таблиць розміщуються над таблицею з вирівнюванням по центру. Мінімальний розмір шрифту в таблицях – 11 пт.

Література**:** нумерований список джерел згідно ДСТУ 8302:2015 від 01.07.2016 р., шрифт 11 пт, відступ: спеціальний, навислий, 0,63 см.

Література англійською мовою (References): список використовуваних джерел згідно ***Harvard Style***. Джерела з заголовками на латиниці наводяться без перекладу. Для літератури джерел на мовах, що не використовують латинський алфавіт, необхідно забезпечити переведення назв джерел і вказати після них у дужках мову оригіналу. Прізвища та ініціали авторів, слід транслітерувати за правилами як для закордонного паспорта. Приклади оформлення бібліографічних посилань згідно з вимогами ***Harvard Style*** наведені в багатьох публікаціях, наприклад, за електронною адресою http://www.staffs.ac.uk/assets/harvard_referencing_examples_tcm44-39847.pdf

Дані про авторів**:** мають починатися рядком "Про авторів:", напівжирний курсив. Далі вказуються для кожного з авторів ПІБ повністю, наукове звання, посада, адреса, кількість публікацій в українських виданнях (приблизна), кількість публікацій в зарубіжних індексованих виданнях (приблизна), індекс Хірша (за наявності), обов'язково номер ORCID (сайт ORCID http://orcid.org/).

Дані про місце роботи авторів**:** починаються рядком "Місце роботи авторів:", напівжирний курсив. Далі вказуються місце роботи, адреса, телефон, факс, електронна пошта, контактний телефон.

3. Оформлення файлу з анотаціями.

Файл з анотаціями містить інформацію двома мовами (наприклад, якщо стаття написана на українській мові, то анотації та ключові слова – на російській та англійській мовах) та має бути оформлений у дві колонки: УДК (шрифт – 8 пт); назва статті (шрифт – 12 пт, напівжирний); прізвища та ініціали авторів (шрифт – 12 пт); текст анотації, ключові слова (шрифт – 10 пт).

Вимоги до анотації англійською мовою: обсяг від 100 до 250 слів, інформативність, оригінальність (не є калькою української або російськомовної анотації), змістовність (відображає основний зміст статті і результати досліджень), структурованість (дотримується логіки опису результатів у статті).

Документ зберігається у форматі doc або docx. Ім'я подається транслітерацією, як прізвище автора (авторів), наприклад, «Petrenko_Annot.doc».

Примітка: Підписний індекс журналу "Проблеми програмування" – **90853**.