КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ТАРАСА ШЕВЧЕНКА
Факультет комп'ютерних наук та кібернетики
Катедра теорії та технології програмування

Кваліфікаційна робота
на здобуття ступеня бакалавра

за спеціальністю 122 Комп'ютерні науки

на тему:

НЕМОНОТОННІ ЛОГІКИ ТА ЇХ ЗАСТОСУВАННЯ В ПРОГРАМУВАННІ

Виконав студент 4-го курсу
Олександр ГАЛАВАЙ _____
(підпис)

Науковий керівник:
доктор фіз.-мат. наук, професор
Степан ШКІЛЬНЯК _____
(підпис)

Засвідчую, що в цій роботі немає запозичень з праць інших авторів без відповідних посилань.

Студент _____
(підпис)

Роботу розглянуто й допущено до захисту на засіданні катедри теорії та технології програмування
«_01_» _червня__ 2022 р.,
протокол № 10

Завідувач катедри
Микола НІКІТЧЕНКО _____
(підпис)

Київ – 2022

РЕФЕРАТ

Обсяг роботи 55 сторінки, 8 ілюстрацій, 21 джерело та 3 додатки.

АВТОЕПІСТЕМІЧНА ЛОГІКА, БАЗА ЗНАНЬ, ЕПІСТЕМІЧНА ЛОГІКА, ЗАПЕРЕЧЕННЯ ЯК ВІДМОВА, ЛОГІКА ПЕРШОГО ПОРЯДКУ, МОДАЛЬНА ЛОГІКА, НЕЙРОННА МЕРЕЖА, НЕМОНОТОННА ЛОГІКА, ПРЕДИКАТ, ШТУЧНИЙ ІНТЕЛЕКТ, PROLOG.

Об’єктом роботи є немонотонні логіки та прикладні програми, в яких вони використовуються.

Метою роботи є аналіз властивостей немонотонних логік та визначення їх семантичних особливостей, порівняння характерних властивостей традиційних логік та немонотонних логік, демонстрація використання апарату немонотонних логік при розробці рішень для прикладних задач.

Методи розроблення: застосування засобів математичної логіки, моделювання системи, абстракція задачі. Інструменти розроблення: вільно поширюване середовище SWI-Prolog (його онлайн-аналог), формальна мова A-Prolog, мови програмування Prolog, Python.

Результат роботи: проведено порівняльний аналіз традиційних та немонотонних логік; проведено семантичний аналіз програм, в яких застосовуються немонотонні логіки; розроблено прикладні рішення, які використовують апарат немонотонних логік. Було продемонстровано, яким чином немонотонні логіки допомагають у розробці деяких типів систем штучного інтелекту.

Результати роботи можуть бути використані для продовження дослідницької діяльности, пов’язаної з теоретичним та практичним застосуванням математичної логіки у програмуванні та в штучному інтелекті, що дає змогу розширити сферу можливостей використання штучного інтелекту на практиці.

ЗМІСТ

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАЧЕННЯ

CWA – Closed-world assumption, припущення про закритий світ;

БЗ – база знань;

ЛЗЗ – логіка за замовчуванням;

ПЛ – пропозиційна логіка;

ПС – предикатні символи;

ФМ – формальна мова;

ФунСим – функціональні символи;

ФС – формальна система;

ШНМ – штучна нейронна мережа.

ВСТУП

Оцінка сучасного стану об'єкта розробки. Розробка комп'ютерів зі штучним інтелектом, на відмінну від звичайних персональних, потребує створення нового покоління інформаційних систем, які орієнтовані перш за все не на збільшення потужности та швидкодії, а на асинхронність і підтримку прийняття рішень. Характерною властивістю штучного інтелекту є його можливість до пошуку рішень в умовах НЕ-факторів на основі немонотонних логік та м'яких обчислень. Тому таким логікам приділяється особлива увага у зв'язку зі створенням та розвитком сучасних інформаційних і програмних систем.

Актуальність роботи та підстави для її виконання. Хоча немонотонні логіки є достатньо молодими, останнім часом вони набувають популярности саме із стрімким розвитком інформатики, в особливості її важливої підгалузі – штучного інтелекту. Тому апарат немонотонних логік вдало використовується для моделювання систем, які намагаються імітувати діяльність людського мозку. Адже люди постійно мають приймати рішення і робити висновки у складному неоднозначному світі, так як отримані знання за своєю суттю є неповними і можуть містити суперечливу інформацію.

Впродовж останніх десятиліть з'являється багато свідчень на користь практичного застосування ідеї немонотонних логік у вигляді певних програмних продуктів та систем у рамках окремих проєктів. За допомогою немонотонних логік, наприклад, можна вдало побудувати базу знань або розробити певну нейронну мережу. Тому в наш час гостро стоїть питання їх якісної розробки апарату немонотонних логік задля вирішення низки прикладних задач програмування та штучного інтелекту. Саме цим зумовлена актуальність даної роботи.

Мета й завдання роботи. Метою кваліфікаційної роботи є аналіз властивостей немонотонних логік та визначення їх семантичних особливостей, порівняння характерних властивостей традиційних логік та немонотонних логік, а

також використання апарату немонотонних логік при розробці рішень для прикладних задач.

Для досягнення мети роботи визначено наступні завдання:

1. Проаналізувати теоретичні відомості, які стосуються традиційних класичних (монотонних) логік та немонотонних логік;
2. Порівняти основні властивості традиційних логік та немонотонних логік;
3. Встановити семантичні особливості програм, розроблених з використанням немонотонних логік;
4. Розробити прикладну базу знань та промоделювати роботу нейронної мережі із застосуванням апарату немонотонних логік.

Об’єкт, методи й засоби розроблення. Об’єктом даної роботи є немонотонні логіки та прикладні програми, в яких вони використовуються. В процесі виконання роботи було опрацьовано й проаналізовано властивості немонотонних логік та безпосередньо розроблено декілька прикладних програм із використанням немонотонних логік.

Розробці прикладних програм із використанням немонотонних логік передувала самостійна постановка конкретної задачі в кожному випадку, а також створення своєрідних логічних алгоритмів їх вирішення. Аналіз особливостей даного типу логік та виділення їх відмінностей з класичними (монотонними) логіками стали загальними для реалізації прикладної частини роботи.

Для розробки програмної частини роботи використано різні підходи реалізації в силу відмінностей умов задач. Обов’язковим є акцентування на використання апарату немонотонних логік. У реалізації першої задачі використано мови програмування A-Prolog (як мова проєктування), Prolog (як мова розробки). Для вирішення другої задачі використовується раніше сформульована модель, а на основі неї моделюється рішення з використанням однієї з мов програмування.

Можливі сфери застосування. Теоретичні та практичні результати роботи можуть бути використані для подальшої дослідницької роботи у галузі прикладного застосування математичної логіки у комп’ютерних науках. Також важливим є можливість застосування таких програмно-орієнтованих логік у інформатиці, зокрема у штучному інтелекті.

РОЗДІЛ 1 ОСНОВНІ ВІДОМОСТІ ПРО СИСТЕМИ МАТЕМАТИЧНОЇ ЛОГІКИ

В наш час розроблено досить багато різноманітних систем з використанням апарату математичної логіки, які вдало використовуються у програмуванні та штучному інтелекті. Зазвичай вони базуються на класичній логіці предикатів. Нічого дивного тут немає, адже класична логіка добре досліджена та має великий досвід застосування (згадаємо хоча б схемотехніку комп'ютерів). В основі багатьох спеціальних логік, які орієнтуються на розв'язання задач тієї чи іншої прикладної галузі, теж лежить апарат класичної логіки. Водночас, як зазначено, зокрема, в [1, 2], класична логіка має принципові обмеження, які роблять її практичне використання більш складним. Так набуває актуальности проблема створення і дослідження нових логік, більш орієнтованих на вирішення проблем програмування та штучного інтелекту.

1.1 Пропозиційна логіка

Спочатку опишемо ПЛ, так як для це базова, найабстрактніша логіка, виходячи з якої можна вже будувати більш складні логічні системи.

Семантичними моделями ПЛ є пропозиційні композиційні системи вигляду (A, Pr, C), де Pr – множина абстрактних предикатів вигляду $A \to \{T, F\}$ (в підрозділі 1.1 предикатом вважаємо висловлення про певний(-і) суб'єкт(и), яке виражає їхні властивості та відношення між ними та може бути розглянуто з погляду істинности або хибности), C – множина композицій над Pr, яка визначається множиною деяких базових пропозиційних композицій.

Пропозиційний рівень логіки характерний тим, що на ньому не досліджується внутрішня або суб'єктно-предикатна структура висловлень чи предикатів. На пропозиційному рівні предикати розглядаються як функції виду $P: A \to \{T, F\}$, де A – множина абстрактних даних, елементи якої невідмінні один

від одного. Для створення більш складних висловлень чи предикатів із простіших застосовуються логічні операції (композиції), які в свою чергу не враховують особливости даних, а беруть до уваги лише вироблені предикатами істиннісні значення T та F. Тому пропозиційні композиції називають логічними зв'язками.

Такими основними традиційними логічними зв'язками є: унарна композиція заперечення (¬) та бінарні композиції : диз'юнкція (∨), кон'юнкція (&), імплікація (→) та еквіваленція (↔).

У випадку логіки часткових предикатів визначення пропозиційних комбінацій подібні до визначень класичних логічних зв'язок для висловлювань і класичних тотальних предикатів. Такі зв'язки називаються клінієвими, оскільки вони відповідають логічним зв'язкам сильної тризначної логіки Кліні [1, 2].

Нехай маємо деякі предикати P та Q, тоді означимо нові предикати з використанням логічних композицій ¬, ∨, &, →, ↔:

$$(\neg P)(d) = \begin{cases} F, & \text{якщо } P(d)\downarrow = T, \\ T, & \text{якщо } P(d)\downarrow = F, \\ \bot & \text{у інших випадках.} \end{cases}$$

$$(P \vee Q)(d) = \begin{cases} T, & \text{якщо } P(d)\downarrow = T \text{ або } Q(d)\downarrow = T, \\ F, & \text{якщо } P(d)\downarrow = F \text{ та } Q(d)\downarrow = F, \\ \bot & \text{у інших випадках.} \end{cases}$$

$$(P\&Q)(d) = \begin{cases} T, & \text{якщо } P(d)\downarrow = T \text{ та } Q(d)\downarrow = T, \\ F, & \text{якщо } P(d)\downarrow = F \text{ або } Q(d)\downarrow = F, \\ \bot & \text{у інших випадках.} \end{cases}$$

$$(P \to Q)(d) = \begin{cases} T, & \text{якщо } P(d)\downarrow = F \text{ або } Q(d)\downarrow = T, \\ F, & \text{якщо } P(d)\downarrow = T \text{ та } Q(d)\downarrow = F, \\ \bot & \text{у інших випадках.} \end{cases}$$

$$(P \leftrightarrow Q)(d) = \begin{cases} T, & \text{якщо } P(d)\downarrow \text{ та } Q(d)\downarrow, P(d) = Q(d)\downarrow, \\ F, & \text{якщо } P(d)\downarrow \text{ та } Q(d)\downarrow, P(d) \neq Q(d)\downarrow, \\ & \bot, \text{якщо } P(d)\uparrow \text{ або } Q(d)\uparrow. \end{cases}$$

де $P(d)\downarrow = T$ означає, що $P(d)$ визначене і рівне істині, $P(d)\uparrow$ означає, що $P(d)$ не визначене.

Візьмемо за базові композиції ¬ та ∨. Тоді інші композиції &, →, ↔ виражаються через базові пропозиційні композиції та є похідними.

Означення 1.1.1. Нехай маємо нескінченну множину пропозиційних символів (імен) Ps. Множиною $\mathcal{L}_{Ps}$ назвемо алфавіт ПЛ, який складається з множини $Ps \cup \{\neg,\vee\}$ ($Ps \cup \{\neg,\vee,\&,\rightarrow\}$). [1]

Теорема 1.1.1. Нехай $\tau: Ps \rightarrow \{T,F\}$ – істиннісна оцінка мови ПЛ. Тоді існує [2] єдина функція $\bar{\tau}: \mathcal{L}_{Ps} \rightarrow \{T,F\}$ така що:

1. $\forall d \in Ps\ \ \bar{\tau}(d) = \tau(d)$;
2. $\bar{\tau}(\neg\Phi) = \neg\bar{\tau}(\Phi)$;
3. $\bar{\tau}(\Phi\&\Psi) = \bar{\tau}(\Phi)\ \&\ \bar{\tau}(\Psi)$;
4. $\bar{\tau}(\Phi \vee \Psi) = \bar{\tau}(\Phi)\ \vee\ \bar{\tau}(\Psi)$;
5. $\bar{\tau}(\Phi \rightarrow \Psi) = \bar{\tau}(\Phi)\ \rightarrow\ \bar{\tau}(\Psi)$.

Правильно побудовані вирази мови ПЛ називаються пропозиційними формулами (ПФ). Індуктивно означимо ПФ [1]:

1. $\forall\ A\ \in Ps$ є ПФ, такі ПФ – атомарні;
2. Якщо Φ і Ψ є ПФ, то $\neg\Phi, \Phi\ \vee\ \Psi, \Phi\ \&\ \Psi, \Phi\ \rightarrow\ \Psi$ є ПФ.

Множину всіх ПФ позначимо Fp.

Означення 1.1.2. Нехай $\tau: Ps \rightarrow \{T,F\}$ – істиннісна оцінка мови ПЛ. Кажуть, що ПФ $\Phi\ \in \mathcal{L}_{Ps}$ задовольняє τ, якщо $\tau(\Phi) = T$. Це записують як $\vDash_\tau \Phi$ або просто $\vDash \Phi$.

Означення 1.1.3. ПФ $\Phi\ \in \mathcal{L}_{Ps}$ називається тавтологією, якщо для кожної істиннісної оцінки мови ПЛ, $\vDash \Phi$; якщо ПФ істинна на кожному наборі значень її пропозиційних імен [3].

Введемо поняття формальної системи.

Формальна система (ФС)– це трійка *(L,A,P)*, яка складається з:

- L – мова системи (її слова – формули);
- A – множина аксіом (кожна аксіома є формулою);
- P – множина правил виведення, які мають вигляд $P_1, P_2, \dots, P_n \vdash P$, де $P_1, P_2, \dots, P_n$ – засновки, а P – висновок.

Звідси отримуємо поняття пропозиційного числення.

Пропозиційне числення (ПЧ) – це наступна ФС (L, Ax, P):

- L – мова пропозиційної логіки (ПЛ)
- Ax – множина аксіом ПЧ
- P – множина правил виведення (ПВ) ПЧ,

де Ax задається єдиною схемою пропозиційних аксіом $\neg\Phi \vee \Phi$, а множина ПВ має наступний вигляд:

П1) $\Phi \vdash \Psi \vee \Phi$ – правило розширення;

П2) $\Phi \vee \Phi \vdash \Phi$ – правило скорочення;

П3) $\Phi \vee (\Psi \vee \Xi) \vdash (\Phi \vee \Psi) \vee \Xi$ – правило асоціативності;

П4) $\Phi \vee \Psi, \neg\Phi \vee \Xi \vdash \Psi \vee \Xi$ – правило перетину.

Теорема ПЧ ПФ, яка виводиться із пропозиційних аксіом з використанням скінченної кількости застосувань правил виведення П1-П4.

Позначимо $\vdash \Phi$ як те, що ПФ Φ є теоремою.

Наведемо декілька прикладів виведення у ПЧ.

Теорема 1.1.2. Якщо $\vdash A \vee B$, то $\vdash B \vee A$ (правило комутативности (ПК)).

Нехай $\vdash A \vee B$ за припущенням, а як аксіому візьмемо $\vdash \neg A \vee A$, тому за правилом виведення П4 маємо $\vdash B \vee A$.

Теорема 1.1.3. Якщо $\vdash A$ та $\vdash A \to B$, то $\vdash B$.

Нехай $\vdash A$ з умови, за П1 маємо $\vdash B \vee A$, звідки $\vdash A \vee B$. Нехай $\vdash A \to B$ за умовою, тобто $\vdash \neg A \vee B$, тому за П4 маємо $\vdash B \vee B$, звідки $\vdash B$ за П2.

Теорема 1.1.4. Кожна теорема ПЧ є тавтологією [1].

Приклад 1.1.1. Доведемо, що $\vdash A \to (\neg A \to B)$.

За аксіомою $\vdash \neg A \vee A$, за П1 $\vdash B \vee (\neg A \vee A)$, за ПК $\vdash (\neg A \vee A) \vee B$, за правилом асоціативности $\vdash \neg A \vee (A \vee B)$, що еквівалентно $\vdash A \to (\neg A \to B)$.

1.2 Предикати та їх різновиди

У підрозділі 1.1 дотично розглядалося поняття предикату, однак в повній мірі воно не було розкрито. Тому означимо його, так як поняття предикату є центральним у досліджені і побудові немонотонних логік.

Предикатом на множині D назвемо часткову неоднозначну, взагалі кажучи, функцію вигляду: $P: D \to \{T, F\}$, де D – деяка множина даних.

Таким чином, частковим неоднозначним предикатом вважаємо відношення між певною множиною даних D та множиною істинісних значень $\{T, F\}$, тому такі предикати називають [2, 5] предикатами реляційного типу.

Нехай $P(d)$ – множина значень, які предикат може набувати на $d \in D$. $P(d) \subseteq \{T, F\}$, з чого отримується, що $P(d)$ приймає одне із $\{\emptyset\}$, $\{T\}$, $\{F\}$, $\{T, F\}$.

Таким чином предикат $P: D \to \{T, F\}$ задається наступними множинами – областями істинности та хибности відповідно:

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

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

Предикат P – *однозначний*, якщо $T(P) \cap F(P) = \emptyset$.

Предикат P – *тотальний* за умови $T(P) \cup F(P) = D$.

Дамо визначення наступних різновидів предикатів вигляду $P: D \to \{T, F\}$:

- P – неспростовний (частково істинний), якщо $F(P) = \emptyset$;
- P – виконуваний, якщо $T(P) = \emptyset$;
- P – всюди невизначений, якщо $F(P) = T(P) = \emptyset$;
- P – тотально істинний, якщо $T(P) = D$;
- P – тотально хибний, якщо $F(P) = D$;
- P – тотожно істинний, якщо $T(P) = D$ і $F(P) = \emptyset$;
- P – тотожно хибний, якщо $T(P) = \emptyset$ і $F(P) = D$;
- P – тотально амбівалентний, якщо $F(P) = T(P) = D$.

Області істинности та хибности для однозначних предикатів наступні:

$T(P) = \{d \in D | P(d) \downarrow = T\}$,

$F(P) = \{d \in D | P(d) \downarrow = F\}$.

Так як предикат за визначенням є частковою неоднозначною функцією, то на нього можна ввести поняття монотоности та немонотоности. Для цього нехай на множинні D введемо відношення порядку за включенням даних.

Предикат P – *монотонний*, якщо $d \subseteq d' \Rightarrow P(d) \subseteq P(d')$.

Тоді, відповідно предикат P – *антитонний*, якщо $d \subseteq d' \Rightarrow P(d') \subseteq P(d)$.

Якщо однозначний предикат P монотонний, тоді такий предикат стає [2, 5] *еквітонним*: $d \subseteq d'$ та $P(d) \downarrow \Rightarrow P(d') \downarrow = P(d)$.

Для *монотонних однозначних часткових предикатів* при розширенні вхідних даних їх інформативність не зменшуватиметься.

Для *монотонних тотальних предикатів* при збільшенні інформативности вхідних даних сама інформативність може лише зменшуватися. В цьому сенсі для таких предикатів поняття монотоности малозмістовне, але змістовним є двоїсте поняття антитонности.

Для *тотальних предикатів* антитоність означає те, що інформативність предиката не зменшуватиметься при збільшенні інформативности вхідних даних.

Примітним є те: якщо P – антитонний, то $P(\emptyset)$ містить у собі з усіх значень, які предикат P може приймати на D, тому $P(\emptyset) = E_P$.

В той самий час поняття антитоности для однозначних предикатів малозмістовне, так як для них антитонними можуть бути [2, 5] тільки майже константні предикати:

$P(d) \cong T, \forall d \in D$ або $P(d) \cong F, \forall d \in D$.

1.3 Класичні логіки першого порядку

В пропозиційній логіці легко можна представити ті чи інші висловлення. Однак, на жаль, пропозиційної логіки не достатньо для опису структури висловлень. Вона має обмежену виразність. Тому висловлення, наприклад, *«Тарас любить футбол»* або *«Деякі рибалки недоброчесні»*, не можуть бути адекватно представлені у пропозиційній логіці. Тому для повнішого опису висловлень використовується потужніша логіка першого порядку.

Стисло опишемо в підрозділі класичні логіки першого порядку. Їх визначальною особливістю є те, що предикати – однозначні скінченно-арні тотальні відображення, функції – однозначні скінченно-арні, причому базові функції та предикати – однозначні n-арні тотальні.

Синтаксис класичної логіки першого порядку визначає, яка колекція символів описує логічні вирази у логіці першого порядку. Такими описами є формули мови першого порядку. Основним її синтаксичними елементами, цеглинками побудови, є символи алфавіту.

Алфавіт класичної мови першого порядку складається [1, 4] з таких символів:

- константні символи (1, 2, 3, непарне, сонце, кіт,…);
- предметні імена (змінні) $x, y, z, a, b,$ ... заданої арности;
- предикатні символи (ПС) $p_0, p_1, p_2,$ ... заданої арности;

- функціональні символи (ФунСим) $f_0, f_1, f_2, \ldots$ заданої арности;
- символи логічних зв'язок $\neg, \vee, \&, \rightarrow$;
- символи кванторів $\exists -$ існування, $\forall -$ загальности.

Позначимо деякі ці множини. Множину ПС позначимо Ps, множина ФунСим нехай буде Fs, множина константних символів – Cn, при цьому $Cn \subseteq Fs$. Спеціальний ПС $=$ за умови, що $= \in Ps$, завжди інтерпретується як предикат рівности, при цьому рівність зазвичай трактується як тотожність.

Множину $\sigma = Fs \cup Ps$ ФунСим і ПС назвемо *сигнатурою* мови першого порядку.

Базовими конструкціями мови логіки першого порядку є формули та терми. Останні використовуються для позначення, назви суб'єктів, а формули – для запису тверджень про суб'єкти.

Індуктивне означення *терма*:

- довільне предметне ім'я та довільний константний символ є термом; такі терми – атомарні;
- якщо $t_1, \ldots, t_n$ – терми, а f – n-арний функціональний символ, тоді $f(t_1, \ldots, t_n)$ – терм.

Атомарною формулою називається вираз вигляду $p(t_1, \ldots, t_n)$, де p – n-арний ПС, а $t_1, \ldots, t_n$ – терми.

Дамо індуктивне означення *формули* мови логіки першого порядку:

- кожна атомарна формула є формулою;
- якщо Φ і Ψ є формулами, то $\neg\Phi, \Phi \vee \Psi, \Phi \,\&\, \Psi, \Phi \rightarrow \Psi$ є формулами;
- якщо Φ – формула, x – предметне ім'я, то $\exists x\Phi$ і $\forall x\Phi$ є формулами. [1]

Приклад 1.3.1. Запишемо висловлення *«Ромео кохає Джульєту»* в термінах логіки першого порядку.

«Ромео кохає Джульєту» $\Rightarrow$ *кохання (Ромео, Джульєта)*

Видно, що в логіці першого порядку це описує атомарна формула.

Розглянемо декілька прикладів на запис висловлень та предикатів формулами мови першого порядку.

Приклад 1.3.2. «Всі птахи літають»

У висловленні предикатом є *«літати(птахи)» / «fly(bird)».* Це представлено наступним чином: $\forall x\ bird(x) \rightarrow fly(x)$.

Приклад 1.3.3. «Деякі хлопці грають у хокей»

У висловленні предикатом є $play(x, y)$, де $x = boys$ та $y = game$. Тому висловлення представляється наступним чином: $\exists x\ boys(x) \rightarrow play(x, hockey)$.

Приклад 1.3.4. «Лише один студент не склав іспит з Програмування»

У висловленні предикатом є $failed(x, y)$, де $x = student$ та $y = subject$. Тому висловлення представляється наступним чином:

$\exists x\ (student(x) \rightarrow failed(x, Programming)\ \&\ \forall y(\neg(x = y)\ \&\ student(y) \rightarrow \neg failed(y, Programming)))$.

1.4 Першопорядкові логіки часткових квазіарних предикатів

Ми розглянули класичну логіку першого порядку, однак її можливостей недостатньо для дослідження немонотонних логік. Наступним кроком буде стислий опис першопорядкових логік часткових квазіарних предикатів [1, 2, 5].

Нехай V і A – довільні множини, які тлумачимо як множину предметних імен і множину предметних значень відповідно.

V-A-іменна множина (V-A-ІМ) – це часткова однозначна функція $d: V \rightarrow A$.

Таку множину можемо подати у вигляді $[v_1 \longmapsto a_1, \dots, v_n \longmapsto a_n, \dots]$, де $v_i \in V, a_i \in A,\ v_i \neq v_j$ при $i \neq j$.

Клас усіх V-A-ІМ будемо позначати ${}^{V}A$.

Введемо функцію asn: ${}^{V}A \to 2^{V}$ наступним чином:

$asn(d) = \{v \in V \mid v \longmapsto a \text{ для деякого } a \in A\}$

Для V-A-ІМ введемо операцію $\|_{-x}$, яка видаляє компоненти з іменем x, та операцію накладки ∇:

$\|_{-x} = [v \longmapsto a \in d \mid v \neq x]$, $\delta\nabla\eta = [v \longmapsto a \in \delta \mid v \notin asn(\eta)]$.

Задамо параметричну операцію *реномінації*

$$R^{v_1,\dots,v_n}_{x_1,\dots,x_n}(d) = d\nabla[v_1 \longmapsto d(x_1), \dots, v_n \longmapsto d(x_n)].$$

Для скорочення запису вважатимемо скорочення $\bar{v}$ рівним $v_1, \dots, v_n$.

Тоді операцію реномінації $R^{v_1,\dots,v_n}_{x_1,\dots,x_n}$ можна записати скорочено $R^{\bar{v}}_{\bar{x}}$.

При послідовному використанні двох операцій $R^{\bar{v}}_{\bar{x}}$ і $R^{\bar{u}}_{\bar{y}}$ можна застосувати операцію згортки, яку позначатимемо як $R^{\bar{v}}_{\bar{x}} \bullet R^{\bar{u}}_{\bar{y}}$.

Тепер дамо означення квазіарного предикату:

Під V-A-квазіарним предикатом розуміють [1, 2, 5] часткову, взагалі кажучи, неоднозначну функцію вигляду P: ${}^{V}A \to \{T, F\}$.

Квазіарні предикати мають основні властивості, в цілому аналогічні властивостям класичних предикатів, які описані у підрозділі 1.2.

Часткові неоднозначні V-A-квазіарні предикати називатимемо реляційного типу або *R*-предикати.

Часткові однозначні V-A-квазіарні предикати назвемо *P*-предикатами, тотальні – *T*-предикати, тотальні однозначні – *TS*-предикати.

Класи таких предикатів позначимо відповідно: PrR^{V}_{A}, PrP^{V}_{A}, PrT^{V}_{A}, $PrTS^{V}_{A}$.

Для квазіарних предикатів, як для предикатів реляційного типу, вводимо поняття монотонности та антитонности. Монотонні *R*-предикати, антитонні *R*-предикати, еквітонні *P*-предикати, антитонні *T*-предикати будемо відповідно називати *RM*-предикатами, *RA*-предикатами, *PE*-предикатами, *TA*-предикатами. Відповідні класи таких V-A-квазіарних предикатів позначимо так:

$PrRM_A^V,\ PrRA_A^V,\ PrPE_A^V,\ PrTA_A^V.$

Дуальним до предиката P називають [2, 5] предикат $\tilde{P}$, такий:

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

Якщо розглядати V-A-квазіарний предикат P як реляцію $P \subseteq {}^V A \times Bool$, то можна вважати $\bar{P}$ доповненням до P як до реляції. Тоді маємо $T(\bar{P}) = \overline{T(P)}$ та $F(\bar{P}) = \overline{F(P)}$. Звідси отримуємо, що $T(\bar{P}) = F(\tilde{P})$, $F(\bar{P}) = T(\tilde{P})$.

Твердження 1.4.1. Якщо P – монотонний, то $\bar{P}$ та $\tilde{P}$ є антитонними; якщо P – антитонний, то $\bar{P}$ та $\tilde{P}$ є монотонними.

Твердження 1.4.2. $P \in PrP_A^V \Leftrightarrow \bar{P} \in PrT_A^V \Leftrightarrow \tilde{P} \in PrT_A^V$; $P \in PrT_A^V \Leftrightarrow \bar{P} \in PrP_A^V \Leftrightarrow \tilde{P} \in PrP_A^V$;

Введемо відображення дуалізації $\delta: PrR_A^V \to PrR_A^V$ таким чином:

$\delta(P) = \tilde{P}$ для $\forall P \in PrR_A^V$.

Відображення дуалізації інволютивне: $\delta(\delta(P)) = P$ для $\forall P \in PrR_A^V$.

Твердження 1.4.3 (див. [5]). $(T) = T$, $\delta(F) = F$, $\delta(PrP_A^V) = PrT_A^V$, $\delta(PrT_A^V) = PrP_A^V$, $\delta(PrTS_A^V) = PrTS_A^V$, $\delta(PrPE_A^V) = PrTA_A^V$, $\delta(PrTA_A^V) = PrTA_A^V$, $\delta(PrRM_A^V) = PrRA_A^V$.

Семантичними моделями чистої першопорядкової логіки квазіарних предикатів є [2, 5] композиційні системи вигляду $({}^V A, Pr, CQ)$. Така композиційна система $({}^V A, Pr, CQ)$ задає дві алгебри: композиційну алгебру предикатів (Pr, CQ) та алгебру даних (A, Pr).

Алфавітом мови чистої першопорядкової логіки квазіарних предикатів є множина V предметних імен (змінних), в якій виділена множина $U \subseteq V$ тотально неістотних імен; множина Ps ПС; множина $Cs = \{\neg, \vee, R_{\bar{x}}^{\bar{v}}, \exists x\}$ символів базових композицій.

Сигнатурою мови називають множину Ps, натомість четвірку $\Sigma = (V, U, Cs, Ps)$ назвемо розширеною сигнатурою мови.

Індуктивно означимо множину формул Fr:

1. $Ps \subseteq Fr$; формули $p \in Ps$ назвемо атомарними;
2. Якщо Φ і $\Psi \in Fr$, то $\neg\Phi, \Phi \vee \Psi, R_{\bar{x}}^{\bar{v}}\Phi, \exists x\Phi \in Fr$.

Інтерпретуємо мову на композиційних системах $({}^{V}A, Pr, CQ)$. Імена $x \in V$ позначають елементи множини базових даних A, символи композицій – композиції із CQ. Символи множини Ps позначають базові предикати в множині Pr, для опису цього позначення задаймо тотальне однозначне відображення $I: Ps \to Pr$. Із базових предикатів за допомогою композицій можемо побудувати складніші, які позначаються формулами.

Відображення інтерпретації формул $I: Fr \to Pr$ задаємо як розширення відображення $I: Ps \to Pr$ згідно побудови формул із простіших за допомогою символів Cs:

$I(\neg\Phi) = \neg(I(\Phi)),\ \ I(\Phi \vee \Psi) = I(\Phi) \vee I(\Psi),\ \ I\left(R_{\bar{x}}^{\bar{v}}(\Phi)\right) = R_{\bar{x}}^{\bar{v}}(I(\Phi)),\ \ I(\exists x\Phi) = \exists x(I(\Phi))$.

Назвемо трійку $\boldsymbol{J} = (CS, \Sigma, I)$ *інтерпретацією* мови чистої першопорядкової логіки часткових квазіарних предикатів сигнатури Σ.

Інтерпретації скорочено позначаються (A, I).

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

Дуальною [5] *до інтерпретації* $\boldsymbol{J} = (A, I)$ називається інтерпретація $\delta(\boldsymbol{J}) = (A, I_\delta)$, якщо $\forall\, p \in Ps$ маємо: $T\left(p_{\delta(J)}\right) = \overline{F(p_J)}$ та $F\left(p_{\delta(J)}\right) = \overline{T(p_J)}$. Аналогічно, $\boldsymbol{J}$ дуальна до $\delta(\boldsymbol{J})$: $T\left(p_J\right) = \overline{F(p_{\delta(J)})}$ та $F\left(p_J\right) = \overline{T(p_{\delta(J)})}$.

1.5 Модальні логіки

До цього розглядалися лише традиційні логіки, які орієнтувалися на опис одного стану світу. Однак, ще з античности філософи розглядали висловлення, які вимагали певної оцінки істиности або хибности. До прикладу, «всі А можливо В», «кожні Х кращі за Y» і т.д. В основу таких висловлень покладаються спеціальні

слова, які певним чином характеризують міру їх істиности або ж надають наше відношення до них. Такі властивості тверджень називаються *модальностями*. Вони виділяють властивості висловлень, які певним чином характеризують міру їх істиности чи наше ставлення до них. Існує стільки ж модальностей, скільки способів описати речі. Серед модальних логік виокремлюють алетичні, епістимічні, темпоральні, деонтичні та багато інших, відштовхуючись від сенсу та назви самих модальностей.

Основними модальностями, які трапляються майже на кожному кроці, є «необхідно» і «можливо». Їх називають алетичними модальностями, вони використовуються в однойменній модальній логіці. Ці модальності показують, як та чи інша ситуація, про яку йде мова, детермінована рядом законів та фактів. Тому алетичні модальності характеризують істинність тверджень з позиції або законів логіки, або ж загалом відомих фактів та законів природи. Таким чином можна говорити [6] про два погляди на алетичні модальності – логічний та онтологічний.

Розглянемо для прикладу висловлення «Можливо, що зірка впаде з неба». З логічної точки зору, таке висловлення істинне, адже воно виражає лише те, що існує деяке А, яке має певну властивість. Інтерпретуючи модальність «можливо» з точки зору онтологічного підходу, висловлення є фальшем, так як суперечить здоровому глузду та всім законам природи загалом.

На основі алетичних модальностей здійснимо загальну характеристику модальних логік. Варто зазначити, що для дослідження модальних логік суттєвим є поняття суперпозиції модальностей, наприклад, «необхідно, що можливо», «необхідно, що необхідно», «можливо, що необхідно, що можливо» і тому подібні. Звичайно, якщо продовжувати генерувати ще більшу суперпозицію, то отримана модальність буде завеликою для сприйняття.

Модальності «необхідно» та «можливо» традиційно позначаються символами □ та ◊ відповідно.

Таким чином, основними модальними операторами загальної модальної логіки є □ – «необхідно» та ◊ – «можливо». Вони двоїсті одне до одного і пов'язані наступними рівностями:

$\Diamond P = \neg \Box \neg P$;

$\Box P = \neg \Diamond \neg P$.

Розглянемо детальніше відомі аксіоматичні системи алетичної логіки на пропозиційному рівні. Мовою таких систем є розширення мови пропозиційної логіки. Алфавіт мови містить у собі множину ПС *Ps*, символи основних логічних зв'язок ∨ і ¬ та символ □ модального оператора «необхідно».

Тоді множина формул *Fm* визначається індуктивно:

1) Кожний $P \in Ps$ є формулою. Такі формули є атомарними.
2) Нехай Φ і Ψ – формули. Тоді ¬ Φ, Φ ∨ Ψ, □ Φ – формули.

Далі розглянемо аксіоматичні системи алетичної логіки. У традиційній модальній логіці існує безліч різних таких систем. Тому обмежимось розглядом систем S1-S5 (див [4]), аксіоми для яких були введені Карлом Льюїсом.

Первісною системою є *S1°*. В її основу покладені наступні аксіоми:

AX1) ⊢ Φ ∧ Ψ → Φ

AX2) ⊢ Φ ∧ Ψ → Ψ ∧ Φ

AX3) ⊢ Φ ∧ (Ψ ∧ Σ) → (Ψ ∧ Φ) ∧ Σ

AX4) ⊢ Φ → Φ ∧ Φ

AX5) ⊢ ((Φ → Ψ) ∧ (Ψ → Σ)) → (Φ → Σ)

Правила виведення:

П1) Якщо Φ та Ψ є формулами, то ¬ Φ, ◊ Φ, Φ ∧ Ψ є також формулами. (правило перетворення)

П2) Якщо ⊢Φ та ⊢ Ψ, тоді ⊢ Φ ∧ Ψ. (правило злиття)

П3) Якщо ⊢ Φ та ⊢ Φ → Ψ, тоді ⊢ Ψ. (правило розділення для →)

Система S1 – це система [6], отримана шляхом додання до аксіом системи *S1°* аксіоми:

AXS1) ⊢ Φ →◊ Φ

Система S2 – це система [6], отримана шляхом додання до аксіом системи S1 аксіоми:

AXS2) ⊢ ◊ (Φ ∧ Ψ) → ◊Φ

Правила виведення для *S2* такі самі як і для *S1*.

Система S3. Множина аксіом *S3* складається з аксіом системи *S1* та наступної:

AXS3) ⊢ (Φ → Ψ) → (◊Φ → ◊ Ψ)

Правила виведення залишаються [6] такими, як і для системи *S1*.

Тепер розглянемо дві системи, на яких ґрунтуються системи *S4*, *S5*.

Система K. Множина аксіом системи містить у собі аксіоми пропозиційної логіки та наступною аксіомою:

AXK) ⊢ □ (Φ → Ψ) → (□Φ → □ Ψ)

Правила виведення складаються з правил виведення пропозиційної логіки (modus ponens), до яких долучається правило модалізації:

ПМ) Φ ⊢ □Φ

Вводиться [6] поняття нормальної системи модальної логіки, яка містить в собі схему аксіом AXK та правило ПМ.

Система T. Множина аксіом складається з аксіом системи *K* та наступної аксіоми:

AX□) ⊢ □Φ → Φ *(«Мої знання істинні»)*

Правила виведення такі ж, як і для системи *K*.

Система S4. Для системи *S4* [6] множини аксіом та правил виведення такі самі як і для системи *T*, додаючи ще одну аксіому за схемою:

AXS4) ⊢ □Φ → □□Φ *(«Мені відомо, що саме я знаю»)*

Система S5. Множина аксіом системи *S5* складається [6] з аксіом системи *T* та наступної аксіоми:

AXS5) $\vdash \Diamond\Phi \rightarrow \Box\Diamond\Phi$ *(«Я знаю, що саме я не знаю»)*

Правила виведення такі самі, як і для системи *K*.

Після означення аксіоматичних систем модальної логіки необхідно описати семантику для них. Найбільш простим могло би бути використання алгебраїчних семантик, проте вони не зовсім прийнятні з інтуїтивної точки зору. Тому для кращої змістовности інтерпретації модальних систем вдало підходять семантики можливих світів або реляційні семантики.

Реляційна модель (модель можливих світів) – це трійка $M = (S, \rhd, I)$, де S – множина світів, $\rhd$ – бінарне відношення на S, I – відображення $I: Ps \times S \rightarrow \{T, F\}$ інтерпретації атомарних формул на світах.

Нехай $\alpha, \beta \in$ S. $\alpha \rhd \beta$ трактується так: світ β можливий відносно світу α, або світ β досяжний зі світу α. Тобто довільне твердження, істинне у світі β, можливе у світі α. Тому відношення $\rhd$ називається відношенням досяжности. За такого розуміння будь-яке твердження, істинне в α, можливе в α, звідки $\alpha \rhd \alpha$. Це означає, що відношення $\rhd$ рефлексивне.

Відображення інтерпретації атомарних формул $I: Ps \times S \rightarrow \{T, F\}$ можна індуктивно розширити до відображення $J: Fm \times S \rightarrow \{T, F\}$:

1) $J(A, \alpha) = I(A, \alpha)$ для всіх A $\in Ps$;

2) $J(\neg\Phi, \alpha) = \neg J(\Phi, \alpha)$;

3) $J(\Phi \vee \Psi, \alpha) = J(\Phi, \alpha) \vee J(\Psi, \alpha)$;

4) $J(\Box\Phi, \alpha) = T \Leftrightarrow J(\Phi, \beta) = T$ для всіх $\beta \in$ S таких, що $\alpha \rhd \beta$, інакше

J($\Box\Phi$, α) = F.

З останнього видно, що $\Box\Phi$ істинна у світі α, якщо Φ істинна в усіх світах, які досяжні з α. [1]

1.5.1 Загальна характеристика епістемічної логіки

Епістемічна логіка досить нова, адже була заснована (Я.Хінтікка) у другій половині XX століття. Її поява була спричинена застосовуванням апарату модальної логіки для порівняльної логічної характеристики того, що є знанням, а що є вірою. Тому епістемічна логіка використовується для того, щоб формалізувати міркування про знання певних експертів (агентів знання), які володіють неповною інформацією. Таким чином ця логіка стає важливою для використання у інформатиці, особливо у штучному інтелекті.

Перш за все опишемо [7] постулати епістемічної логіки:

1. Знання пов'язане з реальністю.
2. Вірування пов'язані з внутрішнім світом опіній.
3. Опінія – це уявлення суб'єкта про свою розумову діяльність.
4. Сумнів – невпевненість у знанні.
5. Незнання – це особливий стан.
6. Знання – це обґрунтована віра.

На основі цього розглянемо просту формальну систему для опису знань експерта. Нехай є множина ПС Ps, які отримані з атомарних предикатів, та множина символів логічних композицій ($\neg, \vee$ як основні та $\&, \rightarrow$ як похідні). Тоді введемо у формальну систему дві модальності епістемічної логіки – «відомо» і «вірую». Відповідно позначатимемо їх операторами знань $\boldsymbol{K}$ і $\boldsymbol{B}$. В цьому контексті їх можна трактувати наступним чином: $\boldsymbol{K}P$ – як «відомо, що P», а $\boldsymbol{B}P$ – як «вірую, що P».

Розглянемо основний алфавіт мови пропозиційної епістемічної логіки з одним експертом. Він складається з множини ПС Ps, символів пропозиційних композицій $\neg, \vee$ та символу $\boldsymbol{K}$, як модального оператора знання. Множину формул такої логіки означимо індуктивно:

- $\forall\, p \in Ps$ є формулою; такі формули атомарні;
- якщо Φ і Ψ є формулами, то $\neg\Phi, \Phi \vee \Psi, \boldsymbol{K}\Phi$ є формулами.

Множина аксіом пропозиційної логіки складається з аксіом пропозиційної логіки та наступних заданих аксіом:

AxEN) $\boldsymbol{K}(\Phi \to \Psi) \to (\boldsymbol{K}\Phi \to \boldsymbol{K}\Psi)$;

AxRe) $\boldsymbol{K}\Phi \to \Phi$ (аксіома реальности знання);

AxPR) $\boldsymbol{K}\Phi \to \boldsymbol{K}\boldsymbol{K}\Phi$ (аксіома позитивної рефлексії – *«якщо я знаю, то я знаю, що знаю»*);

AxNR)$\neg\boldsymbol{K}\Phi \to \boldsymbol{K}\neg\boldsymbol{K}\Phi$ (аксіома негативної рефлексії – *«якщо я не знаю, то я знаю, що я не знаю»*).

Правила виведення складаються з правил виведення пропозиційної логіки, а також правила знання:

ПК) $\Phi \vdash \boldsymbol{K}\Phi$

Система епістемічної модальної логіки, яка містить у собі аксіоми AxEN і AxRe, є аналогом системи *T* алетичної модальної логіки (див. підрозділ 1.5). Таку систему називають $T_{(1)}$.

Система епістемічної модальної логіки, яка включає у собі аксіоми AxEN, AxRe і AxPR, є аналогом системи *S4* алетичної модальної логіки (див. підрозділ 1.5). Таку систему називають $S4_{(1)}$.

Система епістемічної модальної логіки, яка містить у собі аксіоми AxEN, AxRe і AxNR, є аналогом системи *S5* алетичної модальної логіки (див. підрозділ 1.5). Таку систему називають $S5_{(1)}$.

У більш загальних випадках розглядається скінченна множина операторів знання та віри, що відповідає скінченній множині експертів оператори $\boldsymbol{K}x$ та $\boldsymbol{B}x$. У такому випадку вони трактуються так:

$\boldsymbol{K}xP$ – «експерт x знає, що P»;

$\boldsymbol{B}xP$ – «експерт x вірить, що P».

Між знанням і вірою маємо наступні співвідношення:

$\boldsymbol{K}xP \to \boldsymbol{B}xP$;

$\boldsymbol{B}xP \rightarrow \boldsymbol{K}x\boldsymbol{B}xP$;

$\boldsymbol{K}xP \rightarrow \boldsymbol{B}x\boldsymbol{K}xP$.

А для логіки знання і віри отримуємо таке:

$\boldsymbol{B}x\neg P \rightarrow \neg\boldsymbol{B}xP$;

$\boldsymbol{K}x\neg P \rightarrow \neg\boldsymbol{K}xP$;

$\boldsymbol{B}y\boldsymbol{K}xP \leftrightarrow \boldsymbol{B}y(\boldsymbol{B}xP \,\&\, P)$. [1]

На відмінну від деяких класів модальних логік, таких як темпоральна, у епістемічній оперують поняттям «сценарії», а не «світи». Тому введемо множину W як множину можливих сценаріїв.

Аби оцінювати істинність висловлень у епістемічній логіці, використовують *модель сценаріїв* $\mathcal{M} = (W, R_K, V)$. В цій моделі W – непорожня множина певних сценаріїв, R_K – бінарне відношення (епістемічної доступности) на W таке, що для $\forall w, v \in W \;\; wR_Kv$ означає, що сценарій v епістемічно досяжний зі сценарію w, а V – відображення $V\colon Ps \times W \rightarrow \{T, F\}$ інтерпретації атомарних формул на сценаріях.

З огляду на те, що кожен експерт вважає за правду лиши ті факти, які йому відомі, запропонована вище модель має бути доповнена тим, що відношення R_K – рефлексивне, тобто $\forall w \in W \; wR_Kw$. Тоді таку *модель* називатимемо *епістемічною*.

Тип моделі сценаріїв, який був запропонований, представляє не лише зміст своїх знань, а також зміст вірувань або ж опіній, які в свою чергу доповнюють перших. Розширимо мову моделі раніше згаданим модальним оператором $\boldsymbol{B}P$, а також додаймо до моделі відношення доксастичної доступности R_B. Відношення wR_Bv означає, що все, що експерт вірить в w, є істиною у v.

На відмінну від відношення R_K відношення доксастичної доступности не обов'язково є рефлексивним [8]: не все, в що вірить експерт у w, може бути істиною у w. Натомість, відношення R_B може бути серійним: $\forall w \in W \, \exists v \in W\colon wR_Bv$, тобто існує сценарій, в якому все, у що вірує експерт, є істиною.

Розглянемо сформульований нами актуальний на сьогодні приклад, у якому використовується апарат епістемічної логіки.

Приклад 1.5.1.1. Двоє шпигунів виконують завдання із розвідки місцевості. Перший шпигун втратив зв'язок зі своїм колегою. В одному епістемічному випадку перший шпигун може подумати, що другий дезертирує (d). В іншому – що ні $(\neg d)$. При цьому в обох епістемічних випадках він отримував останнє повідомлення від другого два тижні тому (tw). Отже, перший шпигун знає, що востаннє він отримував повідомлення від другого два тижні тому, але при цьому він не знає чи дезертирував він, чи ні. Дану ситуацію можна записати у наступному вигляді: $\boldsymbol{K}tw \,\&\, \neg(\boldsymbol{K}d \vee \boldsymbol{K}(\neg d))$.

На рис. 1.5.1.1 представлена проста епістемічна модель для поданого прикладу.

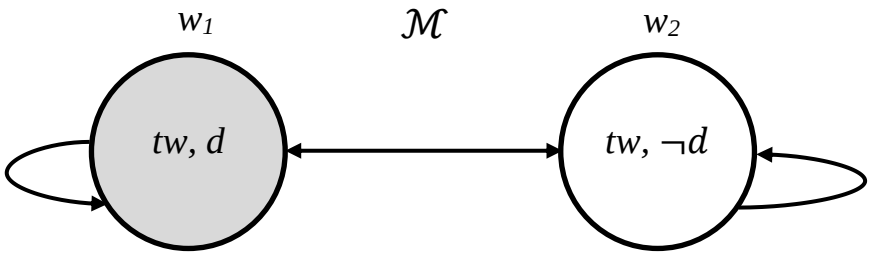

Рис. 1.5.1.1 – проста епістемічна модель

На ньому зображено кола для кожного сценарію, які з'єднані стрілкою відношенням $w_1 R_B w_2$. Припустимо, що другий шпигун дезертирував, а першому про це не відомо. Тому на основі раніше отриманої моделі створимо нову: $\boldsymbol{K}tw \,\&\, \neg\big(\boldsymbol{K}d \vee \boldsymbol{K}(\neg d)\big) \,\&\, d$, де d – істина в $w1$ за умовою, натомість ні $\boldsymbol{K}d$, ні $\boldsymbol{K}(\neg d)$ не є істинами, так як ні d, ні $\neg d$ по окремості не є істинами в усіх сценаріях епістемічно досяжних з w_1, тобто з w_1 та w_2. Натомість $\boldsymbol{K}tw$ є істинним в w_1, так як tw є істиною в усіх епістемічно досяжних з w_1 сценаріях. Тоді цього достатньо, аби показати, що модель $\boldsymbol{K}tw \,\&\, \neg\big(\boldsymbol{K}d \vee \boldsymbol{K}(\neg d)\big) \,\&\, d$ для випадку дезертирства є істинною.

1.5.2 Загальна характеристика автоепістемічної логіки

Автоепістемічна логіка знайшла своє застосування в багатьох областях досліджень, які мають справу з формальним представленням знань, наприклад, у

теорії знань, автоматизованих міркуваннях або логічному програмуванні. Вона базується на звичайній епістемічній логіці, однак в автоепістемічній покладена ідея інтроспекції, тобто усвідомлення власних знань в тій мірі, яка б дозволяла самооцінити, що експерт знає, а що не знає.

Розглянемо алфавіт мови пропозиційної автоепістемічної логіки. Він складається з множини константних символів Cs, ПС Ps, символів пропозиційних композицій $\neg$, $\vee$ та символу $\boldsymbol{K}$ – модального оператора самознання, що виражає факти, які відомі тому, хто про них говорить. Назвемо множину ПС Sp стабільним розширенням множини Ps, яка представляється у вигляді:

$$Sp = Ps \cup \{\boldsymbol{K}\Phi \mid \Phi \in Ps\} \cup \{\neg\boldsymbol{K}\Phi \mid \Phi \notin Ps\}$$

Семантика автоепістемічної логіки базується [3, 9] на розширеннях семантичних моделей традиційної пропозиційної логіки. У пропозиційній моделі визначається, які формули істинні чи хибні, натомість розширення теорії Sp визначає ще, які формули вигляду $\boldsymbol{K}\Phi$ є істинними, а які хибними. В термінах можливих сценаріїв, розширення Sp складається з модальної моделі *S5*, в якій можливі світи (сценарії) складаються лише зі світів (сценаріїв), де Sp є істинною. Тобто можливі сценарії не обов'язково містять усі такі послідовні світи, так як у звичайній модальній логіці, де у формул є вже значення істинности перед перевіркою вивідности.

Таким чином, автоепістемічна логіка є розширенням модальної системи *S5*, оскільки формули вигляду $\neg\boldsymbol{K}\Phi$ та $\neg\boldsymbol{K}(\neg\Phi)$ тавтології у термінах автоепістемічної логіки, але не в системі *S5*.

Приклад 1.5.2.1 Нехай маємо множину предикатів:

$$Ps = \{mobile \vee notebook\}$$

Тоді множина стабільних розширень буде мати вигляд:

$$Sp = \{mobile \vee notebook, \boldsymbol{K}(mobile \vee notebook), \boldsymbol{K}\boldsymbol{K}(mobile \vee notebook), \ldots,$$
$$\neg\boldsymbol{K}(mobile), \boldsymbol{K}\neg\boldsymbol{K}(mobile), \ldots, \neg\boldsymbol{K}(notebook), \boldsymbol{K}\neg\boldsymbol{K}(notebook), \ldots\}$$

Приклад 1.5.2.2 Нехай маємо множину предикатів:

$$Ps = \{\boldsymbol{K}\ (mobile)\ \vee \boldsymbol{K}(notebook), \boldsymbol{K}(mobile) \rightarrow\ mobile, \boldsymbol{K}(notebook) \rightarrow notebook\}$$

Тоді множина стабільних розширень буде мати вигляд:

$$Sp = \{\boldsymbol{K}\ (mobile)\ \vee \boldsymbol{K}(notebook), \quad \boldsymbol{K}(mobile) \rightarrow\ mobile,$$
$$\boldsymbol{K}(notebook) \rightarrow notebook, \boldsymbol{K}(\boldsymbol{K}\ (mobile) \vee \boldsymbol{K}(notebook)),$$
$$\boldsymbol{K}(\boldsymbol{K}(mobile) \rightarrow\ mobile), \boldsymbol{K}(\boldsymbol{K}(notebook) \rightarrow notebook), \ldots,$$
$$\neg\boldsymbol{K}(mobile), \boldsymbol{K}\neg\boldsymbol{K}(mobile), \ldots, \neg\boldsymbol{K}(notebook), \boldsymbol{K}\neg\boldsymbol{K}(notebook), \ldots\}$$

1.6 Висновки стосовно проаналізованих теоретичних відомостей

На основі проаналізованих відомостей про системи математичної логіки були виділені основні моменти, які стосуються основних різновидів логік, аби підготувати фундамент для дослідження немонотонних логік та їх практичного застосування. Перш за все, ми описали пропозиційну логіку, розглянули основні логічні терміни. Далі розглянули дуже важливе поняття предиката, від якого будемо далі відштовхуватися у немонотонних логіках. Опісля, виокремили логіку першого порядку, так як за допомогою її апарату вдало формулюються та розв'язуються задачі, пов'язані з використанням штучного інтелекту. Додатково розглянули першопорядкові логіки часткових квазіарних предикатів як такі, що безпосередньо використовуються в апараті немонотонних логік.

Також було стисло описано модальні логіки, охарактеризовано їх загальні риси на прикладі алетичної логіки. Окремої уваги заслуговує епістемічна логіка, яка, як було зазначено, з успіхом застосовується у розв'язанні практичних задач. Вона, зокрема, надає добре підґрунтя для побудови автоепістемічної логіки, в основу якої покладено практичний принцип немонотоности, про який будемо вести мову в наступному розділі даної роботи.

РОЗДІЛ 2 НЕМОНОТОННІ ЛОГІКИ ТА ЇХ ПРАКТИЧНЕ ЗАСТОСУВАННЯ

Як ми побачили, існує дуже багато логік, які мають свої особливості, та які можна успішно застосовувати у програмуванні. Однак особливе місце посідають немонотонні логіки, про які мало що було сказано в попередньому розділі, водночас це цілий пласт сучасної математичної логіки. І на те є причина, так як немонотонні логіки вирішують фундаментальну проблему застосування класичного логічного апарату в комп'ютерних науках при моделюванні реальних проблем: нова інформація може не лише розширювати отримані раніше знання (у сенсі монотонного розширення), а й може кардинально змінити їх, при цьому попередні висновки можуть виявитися хибними та мають бути вилучені.

Явища немонотонного характеру присутні в усіх сферах нашого повсякденного життя, що спричинене невизначеністю та неповнотою отримуваної інформації, а також через, взагалі кажучи, обмеженість міркування людей. З іншого боку, люди достатньо добре визначають відповідні контексти міркувань. Тому міркування лише на основі неповної інформації може бути проведене цілком усвідомлено та зроблене задля ефективности. У наш час комп'ютерні системи стають невід'ємною частиною нашого життя, тому зростає потреба у використанні штучного інтелекту, для якого характерним є немонотонний спосіб міркувань.

2.1 Відомості про немонотонні логіки

Спочатку з'ясуємо, якими мають бути немонотонні логіки, та які властивості вони мусять мати. Немонотонна логіка є логічною системою із немонотонним відношенням логічного наслідку. Інакше кажучи, вона призначена для фіксації і представлення висновків, які далі перевіряються [9]. Експерти роблять попередні висновки та можуть відкликати їх на основі нових додаткових доказів. Традиційні логічні системи мають монотонне відношення наслідку. Це означає, що додання нової формули до множини засновків ніколи не призводить до скорочення

множини наслідків. На інтуїтивному рівні це означає, що додання нових знань не може скоротити набір тих, які вже відомі.

Традиційна (монотонна) логіка не може впоратися з багатьма задачами, такими як міркування за замовчуванням (наслідки можуть бути отримані лише через відсутність доведення зворотнього), абдуктивне мислення (наслідки виводяться як найбільш ймовірні пояснення), мислення про знання (незнання наслідків має бути вилучено, коли наслідки стають відомі) та перегляд переконань/віри (нові знання можуть суперечити попередньому переконанню/вірі). Розглянемо [10] деякі із цих задач на теоретичному рівні:

1) *Абдуктивне мислення* – це процес отримання найбільш ймовірних пояснень відомих фактів. Абдуктивна логіка не має бути монотонною, так як найімовірніші пояснення фактів не обов'язково вірні. Наприклад, найбільш ймовірним поясненням появи світла вночі на вулицях міста є наявність вуличного освітлення; однак від цього пояснення слід відмовитися, якщо знати, що справжньою появою світла на вулиці міста є сяйво Місяця або поява блискавки під час грози. Так як старе пояснення (*«наявність вуличного освітлення»*) вилучено через появу нової частини знань («*поява Місяця або блискавки*»), будь-яка логіка, яка таким чином моделює висновки, немонотонна.
2) Якщо логіка містить твердження, які означають, що про щось невідомо, то така логіка не може бути монотонною. Справді, вивчення чогось, що раніше не було відомо, призводить до видалення твердження, яка вказувало, що дана частина знань невідома. Таким чином видалення твердження, яке зумовлене доданням нових знань, порушує умову монотоности. Дане питання вивчає *автоепістемічна логіка*, загальна характеристика якої була подана у попередньому розділі.
3) *Перегляд переконань* – це процес зміни переконань, аби пристосуватися до нових переконань, які можуть не відповідати старим. За умови, що нове переконання вірне, деякі зі старих повинні бути вилучені, щоб була збережена послідовність міркувань. Тому при доданні нового переконання

можливість подальшого його спростування робить будь-яку логіку перегляду переконань немонотонною.

4) *Перегляд віри* – це процес зміни віри, аби прийняти нову віру, яка може бути несумісна зі старою. За умови, що переконання у новій вірі є вірними, старі мають бути вилучені, щоб була збережена послідовність міркувань. Ця відмова у відповідь на додання нової віри робить будь-яку логіку перегляду віри немонотонною.

2.1.1 Формалізація пропозиційної немонотонної логіки

Існує щонайменше два підходи до формалізації немонотонних логік, кожен з яких має свої особливості. Перший спосіб: *теоретично-доказова формалізація* немонотонної логіки починається з прийняття деяких немонотонних правил виведення, потім описуються контексти, в яких ці немонотонні правила можуть застосовуватися у виводах. Натомість інша *теоретично-модельна формалізація* немонотонної логіки в свою чергу починається з обмеження семантики обраної монотонної логіки певними спеціальними моделями до мінімальної моделі, на якій вводиться набір немонотонних правил виведення, можливо, з деякими обмеженнями, в контексті яких ці правила можуть застосовуватися, так щоб результуюча система була надійною і повною відносно обмеженої семантики.

Як було зазначено раніше, міркування людей, як правило, немонотонні, тому вони зазвичай роблять висновки з попередніх висновків, яких би не могло бути за наявности деякої додаткової інформації. Тому для формалізації потрібно ввести припущення про закритий світ (Closed-world assumption – CWA). Воно може бути подано наступним чином:

CWA*: Якщо A не може бути доведено, тоді робимо висновок, що* $\neg A$.

Зрозуміло, що люди часто користуються даним міркуванням: не приймають інформацію/знання, поки вона не буде формально доведена. Тому додавання CWA-правила робить логіку немонотонною [11]. Тому, виходячи з CWA, ми можемо

прийняти$\neg A$, якщо A не може бути доведено. Натомість, якщо ми розширимо логіку, в якій A може бути доведено, то ми не можемо використовувати CWA-правило для прийняття $\neg A$.

Виходячи з цього, спробуємо самостійно формалізуємо пропозиційну немонотонну логіку, використовуючи знання, які були отримані при аналізі літератури у розділі 1.

Означення 2.1.1.1 *Нормальною формальною системою* назвемо ФС вигляду *(L,A,P)*, для якої виконуються умови:

- якщо $\forall \Phi \in L$, тоді $\exists \neg \Phi \in L$;
- $\perp \in L$.

Означення 2.1.1.2 Нехай *(L,A,P)* – нормальна ФС. Тоді припустимо, що *(L,A,P,B)* – немонотонна ФС, якщо $B \subseteq L$.

Нехай *(L,A,P,B)* – немонотонна система, в якій *(L,A,P)* – нормальна ФС.

Означення 2.1.1.3 Нехай $\Delta_1, \Delta_2 \subseteq B$. Тоді за CWA Δ_1 виключає Δ_2 для теорії T, якщо:

1) $T \cup \Delta_1 \nvdash \perp$;
2) $T \cup \Delta_1 \cup \Delta_2 \vdash \perp$.

Приклад 2.1.1.1 $T = \{\neg p \to q, \neg q \to p, \neg r \to s\}$. Звідси маємо, що $\{\neg p\}$ виключає $\{\neg q\}$, а $\{\neg q\}$ виключає $\{\neg p\}$.

Означення 2.1.1.4 Δ прийнятна для теорії T, якщо $T \cup \Delta \nvdash \perp$.

Означення 2.1.1.5 Δ строго прийнятна для теорії T, якщо $\forall \Delta'$, прийнятної для T, Δ – прийнятна для $T \cup \Delta'$: $T \cup \Delta \cup \Delta' \nvdash \perp$.

Приклад 2.1.1.2 $T = \{\neg p \to q, \neg q \to p, \neg r \to s\}$. Маємо, що $\{\neg p\}$ та $\{\neg q\}$ – прийнятні для T, а $\{\neg r\}$ – строго прийнятна для T.

Означення 2.1.1.6 $T\nmid \Phi$ – *немонотонний наслідок* в *(L,A,P,B)*, якщо $\exists \Phi_1, \dots, \Phi_n \in L : \forall\, i = \overline{1,n}\ \Phi_i \in T$ та $\exists (S, \Phi_i) \in (L, A, P, B) : S \subseteq T \cup \{\Phi_1, \dots, \Phi_{i-1}\}$ та $\exists \Phi_i \in B$: $\{\Phi_i\}$ прийнятна для $T \cup \{\Phi_1, \dots, \Phi_{i-1}\}$.

Означення 2.1.1.7 $T \Vdash \sim \Phi$ – *слабкий немонотонний наслідок* в *(L,A,P,B)*, якщо $\exists \Phi_1, \dots, \Phi_n \in L : \forall\, i = \overline{1,n}\ \Phi_i \in T$ та $\exists (S, \Phi_i) \in (L, A, P, B) :$ $S \subseteq T \cup \{\Phi_1, \dots, \Phi_{i-1}\}$ та $\exists \Phi_i \in B$: $\{\Phi_i\}$ строго прийнятна для $T \cup \{\Phi_1, \dots, \Phi_{i-1}\}$.

Теорема 2.1.1.1:

1) $T\nmid \Phi$, якщо існує скінчена множина $\Delta \subseteq A$, так що $T \cup \Delta \vdash \Phi$ і Δ – прийнятна для T.
2) $T \Vdash \sim \Phi$, якщо існує скінчена множина $\Delta \subseteq A$, так що $T \cup \Delta \vdash \Phi$ і Δ – строго прийнятна для T.

Приклад 2.1.1.3 Нехай *(L,A,P,B)* – немонотонна ФС. $B = \{\neg guilty(x)\}$, $T = \{make(x, crime) \rightarrow guilty(x),\ make(Tom, crime)\}$. Тоді маємо:

$$T \vdash guilty(Tom), T \nmid guilty(Tom),$$
$$T \widetilde{\nmid} \neg guilty(Tom), T \Vdash\!\!\!\!\!\!\widetilde{\ \ \sim} \neg guilty(Tom),$$
$$T \nmid \neg guilty(Sam), T \Vdash \sim \neg guilty(Sam).$$

Таким чином з відомих даних ми робимо висновок, що Tom винний, а стосовно Sam ми припускаємо, що він не винний, так як ми не знаємо, чи вчиняв він злочин згідно з CWA.

У традиційних монотонних логіках якщо формула Ф є логічним наслідком теорії (множини формул) T, то для довільної теорії (множини формул) T' формула Ф також буде логічним $T \cup T'$.

Як приклад, наведемо твердження «всі птахи вміють літати», і ми одразу робимо висновок, що будь-який птах може літати. Однак, якщо нам скажуть, що птахами є страус та пінгвін, тоді ми маємо повернутися до попереднього твердження, про яке вже не можемо сказати, що всі птахи вміють літати.

У [11] Раймонд Рейтер пропонує використовувати *логіку за замовчуванням (ЛЗЗ)*. Вона базується на традиційних логіках. В ній використовуються такий самий інструментарій, як у звичайній логіці першого порядку, множина ПС Ps та множина правил виведення, з використанням додаткової множини правил виведення D у формі правил:

$$\frac{\alpha : M\beta_1, \dots, M\beta_n}{\gamma},$$

де α, β_i, γ – формули логіки першого порядку; α – передумова, β_i – припущення, γ – висновок. Правила такого вигляду слід читати як: якщо передумова α виконується та кожна з вимог β_i істинна, то можна зробити висновок γ. Теорія T описує, що є істиною, тоді як D дозволяє отримати додаткову інформацію, використовуючи логіку міркування за замовчуванням. Семантика такої теорії за замовчуванням (T, D) отримана розширенням. Розширенням Σ теорії (T, D) є множина формул логіки першого порядку, яка задовольняє наступним вимогам:

1) $\Sigma \supseteq T$;
2) Множина Σ є замкненою над логічними операціями;
3) Якщо R – правило в D, так що формула $\alpha \in \Sigma$ та $\forall i \ \neg\beta_i \notin \Sigma$, то $\gamma \in \Sigma$;
4) Множина Σ є розширенням (D, Σ). [11]

Теорія *ЛЗЗ* може не мати розширення, вона може мати одне або декілька розширень. Кожне розширення є множиною вірувань, якими володіє експерт. Таким чином самостійно формалізуємо попередній приклад про птахів у логіці за замовчуванням. Нехай множина D складається з правила:

$$\frac{bird(x) : M\neg anomal(x)}{flies(x)}, \frac{: \boldsymbol{not}(anomal(x))}{\neg anomal(x)}, \frac{bird(x)\ \&\ (ostrich(x)\ \vee penguin(x)):}{anomal(x)}$$

та наступних отриманих теорій:

$$T_1 = \{bird(x)\}$$

$$T_2 = \{bird(x), \neg anomal(x)\}$$

Так (D, T_1) має одне розширення, в якому предикат $bird(x)$ істинний, в іншому випадку, у розширенні (D, T_2) предикату $flies(x)$ не має місця, так як у теорії T_2 присутній предикат $\neg anomal(x)$.

2.2 Логічне програмування та немонотонна логіка

Як ми побачили, немонотонна логіка дуже вдало може описуватися в термінах логіки першого порядку із застосуванням ЛЗЗ. Тому її можна успішно використовувати у логічному програмуванні. Водночас досить довго між ними не було міцних зв'язків, незважаючи на наявність багатьох спільних цілей та прийомів. Найвідомішою мовою логічного програмування є Prolog, в яку не так давно були включені немонотоності. Серед іншого це призвело до розвитку представлень знань та створення схожої на Prolog мови – ФМ A-Prolog [12]. Розглянемо її та покажемо її застосування як на теоретичному, так і на практичному рівнях.

Синтаксис ФМ A-Prolog визначений сигнатурою σ, яка складається з типів $types(\sigma) = \{\tau_0, \dots, \tau_m\}$, константних об'єктів $obj(\tau, \sigma) = \{c_0, \dots, c_m\}$ для кожного типу τ, ФС та ПС відповідно: $func(\sigma) = \{f_0, \dots, f_k\}$, $pred(\sigma) = \{p_0, \dots, p_n\}$. Вважаємо, що сигнатура містить символи для цілих чисел і для стандартних математичних функцій та відношень. Терми будуються за аналогією першопорядкових логік: звичайні літерали будуються у вигляді $p(t_1, \dots, t_n)$, де t_i – терми власних типів, p – n-арний ПС; натомість заперечувальний літерал має вигляд $\neg p(t_1, \dots, t_n)$. Символ $\neg$ у ФМ A-Prolog називається класичним (сильним) запереченням. Літерали $p(t_1, \dots, t_n)$ та $\neg p(t_1, \dots, t_n)$ називається протилежними. Літерали та терми без змінних називатимемо базовими. Множини всіх базових термів, атомів та літералів відповідно позначатимемо $terms(\sigma)$, $atoms(\sigma)$, $lit(\sigma)$ відповідно. Для множини ПС Ps з σ $atoms(Ps, \sigma)$ ($lit(Ps, \sigma)$) позначають множини базових атомів (літералів) сигнатури σ, які створені з ПС з Ps. Узгоджені множини базових літералів сигнатури σ, що містять всі арифметичні літерали, які істинні за

стандартної інтерпретації, називаються станами сигнатури σ і позначаються $states(\sigma)$.

Правила ФМ A-Prolog мають наступний вигляд (*формула 2.2.1*):

$$l_0 \; or \; \dots or \; l_k \; \leftarrow l_{k+1}, \dots, l_m, \textbf{not} \; l_{m+1}, \dots, \textbf{not} \; l_n,$$

де l_i – літерали, $\textbf{not}$ – логічна композиція, яка називається заперечення як відмова (*NAF*) або заперечення за замовчуванням, or – епістемічна диз'юнкція [13, 14].

Множину $\{\textbf{not} \; l_i, \dots, \textbf{not} \; l_{i+k}\}$ можна скорочено позначати $\textbf{not}\{ \; l_i, \dots, l_{i+k}\}$.

Якщо q – деяке правило мови A-Prolog, тоді $head(q) = \{l_0 or \dots or \; l_k\}, pos(q) = \{l_{k+1}, \dots, l_m\}, neg(q) = \{l_{m+1}, \dots, l_n\}$, тому $body(q) = pos(q), \textbf{not} \; neg(q)$.

Правило, в якому $head(q) = \emptyset$, називається обмеженням та записується:

$$\leftarrow l_{k+1}, \dots, l_m, \textbf{not} \; l_{m+1}, \dots, \textbf{not} \; l_n$$

При $k = 0$ маємо:

$$l_0 \leftarrow l_1, \dots, l_m, \textbf{not} \; l_{m+1}, \dots, \textbf{not} \; l_n.$$

Заперечення за замовчуванням ($\textbf{not}$) трактується як нова логічна композиція. Інтуїтивно $\textbf{not} \; l$ означає, що «не має причини вірити в l». Також замість класичної диз'юнкції використовується композиція or, зміст якої відрізняється від $\vee$. Формула $P \vee Q$ означає, що «P є істинною або Q є істинною», в той самий час композиція $P \; or \; Q \; \leftarrow$ інтерпретується епістемічно та означає «вважається, що P є істинною або вважається, що Q є істинною» (див. [15, 16]).

Означення 2.2.1 Програмою у мові A-Prolog є пара $\{\sigma, П\}$, де σ – сигнатура, а П – множина правил, побудованих згідно формули 2.2.1. Позначатимемо сигнатуру як $\sigma(П)$.

Семантика логічної програми П допомагає отримати П множину відповідей – базових літералів над сигнатурою $\sigma(П)$, що відповідають множині правил, задані самою програмою П. Вважаємо, що для отримання такої множини необхідно, щоб виконувалися наступні вимоги:

- Множина відповідей має задовольняти правилам П, які трактуються як обмеження: Якщо «вірять» у $body(q)$, то мають «вірити» в хоча б один з літералів $head(q)$.
- Дотримання принципу раціональности, який означає, що «не можна вірити тому, в що примушують нас вірити».

Правило форми: $\neg l \leftarrow \boldsymbol{not}\ l$ будемо називати *запереченням як відмова*. Це означає, що якщо вичерпний пошук доведення l завершився невдачою, тоді стверджуємо, що $\neg l$. Більш широко це немонотонне правило логічного наслідку розглядається в [17].

Дамо точне визначення множини відповідей для програми, яка не містить заперечення як відмова, для інших же випадків використаємо [14]. Нехай П – програма, а S – довільний стан із $\sigma(\text{П})$. S є замкненою над П, якщо для кожного правила вигляду:

$$l_0\ or\ \dots or\ l_k\ \leftarrow l_{k+1}, \dots, l_m$$

програми П, виконується $\{l_{k+1}, \dots, l_m\} \subseteq S$, $\{l_0, \dots, l_k\} \cap S \neq \emptyset$.

Означення 2.2.1 Множина S є множиною відповідей програми П, якщо S – мінімальна серед множин, замкнена над П.

2.2.1 Практичне застосування немонотонної логіки у A-Prolog та Prolog

Після аналізу можливостей використання мови немонотонних логік розв'яжемо декілька власноруч сформульованих задач, спочатку використовуючи ФМ A-Prolog, а потім мову Prolog для перевірки відповідей.

Приклад 2.2.1.1 $\text{П} = \begin{bmatrix} p(a)\ \leftarrow\ p(c), \boldsymbol{not}\ q(a). \\ p(b)\ \leftarrow \neg q(b), \boldsymbol{not}\ (\neg q(b)). \\ p(c)\ \leftarrow q(c),\ \boldsymbol{not}\ p(b). \\ q(a). \\ q(b). \\ q(c). \end{bmatrix}$

Використовуючи означення множини відповідей [14] такої задачі, одразу легко бачити, що $S_0' = \{q(a), q(b), q(c)\}$ є множиною розв'язків. Розглядаючи перші три предикати, отримуємо, що в контексті цієї задачі вірними є множина $\{p(c)\}$.Тоді остаточною множиною відповідей буде $S_0 = \{q(a), q(b), q(c), p(c)\}$. Таким чином $\Pi \models q(a), \Pi \models q(b), \Pi \models q(c), \Pi \not\models p(a), \Pi \not\models p(b), \Pi \models p(c)$.

Тепер спробуємо записати дану задачу на мові програмування Prolog та порівняємо отримані відповіді при викликах відповідних предикатів. У Додатку А наведено рис. 2.2.1.1, на якому представлено запис даної задачі на мові Prolog та множина відповідей. Бачимо, що вони співпадають, отже, наші міркування стосовно розв'язку цієї задачі аналогічні міркуванням логічної мови програмування.

Приклад 2.2.1.2 $\Pi = \begin{bmatrix} p \leftarrow \textbf{not}\ q. \\ q \leftarrow s, \textbf{not}\ (\neg p). \\ s \leftarrow p. \end{bmatrix}$

В даному випадку бачимо, що програма зациклюється, тому однозначної множини розв'язків не має, так як не існує першого однозначного предиката, то кожен з літералів є невизначеним і тому множина розв'язків у цій задачі порожня.

Записана задача у мові програмування Prolog та отриманий результат представлені в Додатку А на рис. 2.2.1.2.

Приклад 2.2.1.3 $\Pi = \begin{bmatrix} l(0). \\ l(q(q(X))) \leftarrow \textbf{not}\ (l(X)). \\ p(q(X)) \leftarrow l(X), \textbf{not}\ p(X). \\ p(X) \leftarrow l(X), \textbf{not}\ p(q(X)). \end{bmatrix}$

В цій задачі одразу видно, що розв'язком точно буде $l(0)$. Наступне правило також буде завжди істинним, так як предикат викликає сам себе, а в силу оператора заперечення як відмова для предикату $l(X)$ буде існувати безліч розв'язків. Натомість на двох наступних рядках бачимо, що правила викликають предикат $q(X)$, який існує лише всередині правил і не є окремим. Також видно, що третє та четверте правила викликають одне одного, з чого можна зробити висновок, що для

будь-якого X $p(X)$ буде вірним, при цьому $p(X)$ потрапляє в рекурсію. Тому така програма має безліч розв'язків.

Записана на мові програмування Prolog задача представлена на рис. 2.2.1.2-а), а отриманий результат – на рис. 2.2.1.2-б), які представлені в Додатку А.

Тепер самостійно поставимо задачу на побудову бази знань, в якій використовуються елементи немонотонних логік.

Приклад 2.2.1.4 Необхідно побудувати БЗ структури підрозділів університету. Він складається з факультетів, які в свою чергу складаються з катедр. На кожній катедрі є співробітники та перелік дисциплін, що викладаються на ній, а також які дисципліни яким співробітником викладаються.

Простенька БЗ була реалізована на мові Prolog, код та скріншоти роботи якої знаходяться в Додатку Б.

Спочатку було описано атомарні предикати, які є істинними. У лістингу ці факти мають вигляд $member(name, cathedra), course(title, cathedra)$. Перший з них позначає співробітника тої чи іншої катедри, а другий – курс, який читається на катедрі (див. Рис. 2.2.1.4 (б, в)). Далі застосовуємо для цих фактів заперечення як відмову. Цей інструмент вдало використовується в Prolog для опису винятків з певних правил. Також воно дозволяє збільшити шанси появи програмних помилок, які часто супроводжуються використанням т.з. «червоних скорочень» (red cut). У синтаксисі мови Prolog заперечення як відмову позначають \+. Також опишемо виняткові ситуації для таких фактів на мові A-Prolog:

$$\neg member(P, Y) \leftarrow \textbf{not}\ member(P, Y).$$
$$\neg course(C, Y) \leftarrow \textbf{not}\ course(C, Y).$$

Окремо опишемо факти, які будуть позначати хто з викладачів викладає який курс, у вигляді: $teaches(name\_member, title\_course)$ (див. Рис. 2.2.1.4 (а)). Особливої уваги у цьому випадку потрібно описати винятковим ситуаціям з викладачами. Виняткова ситуація наступна – дисципліну «Логіка» має викладатися викладачами з катедри ТТП. Тому опишемо це правило на мові A-Prolog:

$$\neg teaches(P, C) \leftarrow \big(\neg member(Ps, tc), course(C, tc)\big); (\neg member(Ps, mi), course(C, mi)), \textbf{not}\ memeber\_ttp(P, C), \textbf{not}\ teaches(P, C).$$

$$memeber\_ttp(P, logic) \leftarrow member(P, ttp).$$

Означимо ієрархію катедра-факультет-університет, в якій також буде описана виняткова ситуація із застосуванням заперечення як відмови:

$$\neg part(E1, E2) \leftarrow \textbf{not}\ part(E1, E2).$$

Із використанням ФМ A-Prolog ми задали модель простенької БЗ, яка була реалізована із використанням елементів немонотонних логік, які вбудовані у мові Prolog. Викличемо предикат $teaches(name\_member, title\_course)$, і побачимо, що викладача *oleksandr* , який мав би викладати предмет *logic*, нема у результуючому виводі предиката, тобто використання заперечення як відмову дало нам скорочення попередніх знань. Результат виклику предиката представлено на Рис. 2.2.1.5 у Додатку Б.

2.3 Немонотонна логіка та штучні нейронні мережі

У цьому підрозділі на прикладах ми розглянемо практичне застосування апарату немонотонних логік на штучних нейронних мережах (ШНМ). Міркування, які представляються в немонотонній логіці, відіграють важливу роль у розвитку сучасних ШНМ, намагаючись імітувати людську розумову діяльність. Немонотонні ШНМ отримують знання, які представляються множинною схемою з певними винятками, тобто немонотонною множинною схемою спадкування.

Спочатку наведемо загальне означення класу задач ШНМ. Воно буде використовуватися як підґрунтя до визначення немонотонних мереж.

Нейронною мережею N назвемо четвірку (S, F, C, G), де S – множина можливих станів ШНМ, розмірність якої відповідає кількості параметрів необхідних для опису стану системи, тобто один стан представляється у вигляді $\bar{x} = (x_1, \dots, x_n)$, де $\forall i = \overline{1, n}\ \ 0 \le x_i \le 1$; C – множина можливих конфігурацій

(вагових коефіцієнтів) мережі, тобто $c \in C$ описує для кожної пари нейронів i та j зв'язок c_{ij}, позитивне значення якого називається подразником, а негативне – інгібітором; F – множина функцій активацій, кожна така функція $f_c \in F$ для даної конфігурації $c \in C$ описує, як активність нейрона впливає на стан мережі; G – множина навчальних функцій, які описують, як змінюються результати конфігурацій.

Множини S, F та C, G взаємо пов'язані за допомогою наступних рівнянь:

$$x(t+1) = f_{c(t)}\big(x(t)\big);$$

$$c(t+1) = g_{x(t)}\big(c(t)\big),$$

де $s \in S, f \in F, c \in C$ та $g \in G$.

Багато теорій будуються на схемах. Звичайно, термін *схема* досить обширний і потребує певного уточнення його використання. В нашому випадку ми будемо використовувати означення схеми як загальну назву структур, що використовуються у [18]. Задаймо загальні риси такої схеми:

- Схеми можуть використовуватися для представлення об'єктів, подій або ситуацій;
- Одна схема або її частина може бути в складі іншої схеми;
- Схеми підтримують CWA, тобто вони мають здатність до заповнення пропущеної інформації.

Виходячи з цього, *схемами* є множини «мікрофункцій», які виникають з колективної поведінки нейронів як найменшої одиниці. Кожна така «мікрофункція» представлена у вигляді нейроноподібної одиниці, яка взаємодіє з іншими, подразнюючи або пригнічуючи їх. Детальніше про це описано в [19].

У теорії нейронних мереж існує спосіб визначення поняття схем, які володіють схожими властивостями. Виходячи з [19], деяка схема α відповідає вектору $(\alpha_1, \dots, \alpha_n)$ в множині станів S. Таким чином така схема представляється в ШНМ у вигляді вектора активности $\bar{x} = (x_1, \dots, x_n)$, де $\forall i = \overline{1,n}\ x_i \geq \alpha_i$, тобто $\alpha =$

$\{\bar{x} \in S \mid \forall i = \overline{1,n}\ x_i \geq \alpha_i\}$. Також означимо відношення часткового порядку «величини інформаційного змісту» серед схем α та β: $\alpha \geq \beta$ – відношення часткового порядку, якщо $\forall i = \overline{1,n}\ \alpha_i \geq \beta_i$. Мінімальною схемою в нейронній мережі вважатимемо $\overline{\mathbf{0}} = (0, \dots, 0)$, а максимальною – $\overline{\mathbf{1}} = (1, \dots, 1)$.

Якщо $\alpha \geq \beta$, то схему β вважаємо більш загальною схемою за α, а α вважаємо частиною β, в той самий час частина схеми α, яке не є β, є варіативною частиною схеми β.

Розглянемо деякі елементарні операції над схемами, які безпосередньо використовуються у немонотонних ШНМ. Нехай є дві схеми $\alpha = (\alpha_1, \dots, \alpha_n)$ та $\beta = (\beta_1, \dots, \beta_n)$, тоді визначені [19] наступні операції:

- Доповнення $\tilde{\alpha}$ до схеми α: $\tilde{\alpha} = (1 - \alpha_1, \dots, 1 - \alpha_n)$;
- Кон'юнкція схем $\alpha \bullet \beta = \gamma$, де $\gamma = (\gamma_1, \dots, \gamma_n)$ та $\forall i = \overline{1,n}\ \gamma_i = \max(\alpha_i, \beta_i)$;
- Диз'юнкція схем $\alpha \oplus \beta = \gamma$, де $\gamma = (\gamma_1, \dots, \gamma_n)$ та $\forall i = \overline{1,n}\ \gamma_i = \min(\alpha_i, \beta_i)$.

В нейронних мережах при подачі на вхід інформації до певних нейронів може відбуватися стабілізація отриманої інформації в них. Такі стани назвемо резонансними.

Розглянемо нейронну мережу $N = (S, F, C, G)$, множина конфігурацій якої буде незмінною. Нехай для фіксованого $c \in C$ $f_c^0(x) = f_c(x)$ та $f_c^{n+1}(x) = f_c(x) \circ f_c^n(x)$. Тоді стан $s \in S$ називається *резонуючим*, якщо виконуються наступні умови:

1) $f_c(s) = s$ (рівновага);
2) $\forall p \in S\ \forall \varepsilon > 0\ \exists \delta > 0$: $|p - s| < \delta \implies \forall n \geq 0 :\ |f_c^n(p) - s| < \varepsilon$ (стійкість);
3) $\forall p \in S\ \exists \delta > 0$: $|p - s| < \delta \Rightarrow \lim\limits_{n \to \infty} f_c^n(p) = s$ (асимптотична стійкість).

Нейронна мережа називається *резонуючою*, якщо $\forall c \in C\ \forall x \in S\ \exists n > 0 :$ $f_c^n(p)$ є резонуючою функцією активації.

Якщо існує скінченна $\lim\limits_{n \to \infty} f_c^n(p)$, запишемо її у вигляді $[p]_c$, де $[\,.\,]_c$ – резонансна функція для c. Функція $[\,.\,]_c$ може бути інтерпретована для заповнення

пропущеної інформації у схемі, так як схема представлена у вигляді $[\alpha]_c$ містить інформацію, яку очікують отримати з α як при вхідному значенні.

Представимо функцію $f_\alpha(x)$ як $f_\alpha(x) = f(x) \bullet \alpha, \forall x \in S$. Тоді функцію $[x]^\alpha$ для конфігурацій представимо: $[x]^\alpha = \lim\limits_{n\to\infty} f_\alpha^n(x)$.

Головною ідею немонотонних ШНМ є [20] поняття немонотонного наслідку $\nmid$ між схемами, яке визначається наступним чином: якщо $[\alpha]^\alpha \geq \beta$, тоді $\alpha \nmid \beta$.

Приклад 2.3.1 Розглянемо приклад ШНМ (Рис. 2.3.1), яка складається з 6 однакових нейронів з відповідними активностями $x_1, x_2, x_3, x_4, x_5, x_6$.

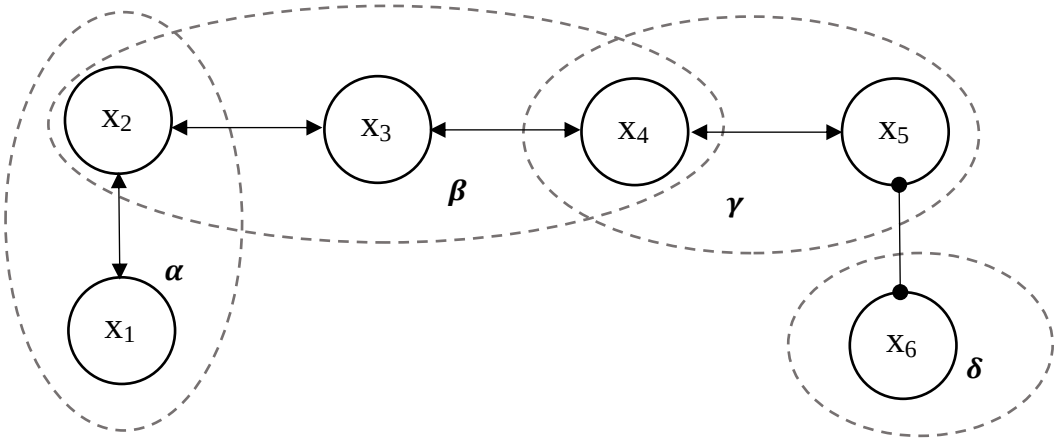

Рис. 2.3.1 – Представлення нейронної мережі у схемах

Нейрони, які взаємодіють, з'єднані лініями. Лініями зі стрілками на кінці позначені нейрони, які подразнюють одне одного, а з великими крапками – пригнічують.

Розділимо дану нейромережу на схеми та розглянемо її з точки зору схем.

Нехай ми поділили мережу на чотири схеми $\alpha, \beta, \gamma, \delta$, яким відповідають вектори активностей схем $\alpha = (1,1,0,0,0,0), \beta = (0,1,1,1,0,0), \gamma = (0,0,0,1,1,0), \delta = (0,0,0,0,1,1)$. Припустимо, що x_6 приглушує x_5 значно більше, ніж x_4 подразнює x_5. Активувавши схему α на вході, мережа активує і β, це позначимо $\alpha \nmid \beta$.

В свою чергу при подачі ще інформації на β як початкової схеми у мережі може активуватися γ. Тоді це представимо у вигляді $\alpha \bullet \beta \nmid \gamma$.

Розширивши множину входів до $(\alpha \bullet \beta) \bullet \delta$ мережа не зможе збудити схему γ, так як, ми припустили раніше, x_6 приглушує x_5 значно більше, ніж x_4 подразнює x_5, і це запишемо у вигляді $(\alpha \bullet \beta) \bullet \delta \widehat{\dashv} \gamma$.

Тепер самостійно означимо властивості оператора немонотонного наслідку, виходячи з означень схем. Представимо їх:

- *Рефлексивність.* Посилаючись з означення немонотонного наслідку, $\alpha \dashv \beta$, якщо $[\alpha]^{\alpha} \geq \beta$, маємо, що якщо $\alpha \dashv \alpha$, то $[\alpha]^{\alpha} \geq \alpha$, а це означає, що $[\alpha]^{\alpha} = \lim\limits_{n \to \infty} f_{\alpha}^{n}(\alpha) = \alpha$. Тобто оператор немонотонного наслідку рефлексивний: $\alpha \dashv \alpha$.
- *Послаблення наслідку.* Якщо $\alpha \vdash \beta$, то $\alpha \dashv \beta$. Тобто якщо схема β логічно отримується з α, то при частковому порядку оператора $\dashv$ схема β логічно немонотонно отримується з α.
- *Об'єднання немонотонних наслідків.* Якщо $\alpha \dashv \beta$ та $\alpha \dashv \gamma$, тоді $\alpha \dashv \beta \bullet \gamma$. З формальної точки зору маємо: $[\alpha]^{\alpha} \geq \beta \;\&\; [\alpha]^{\alpha} \geq \gamma \Rightarrow [\alpha]^{\alpha} \geq \beta \bullet \gamma$.

Приклад 2.3.2 Нехай ми маємо просту ШНМ типу Гопфілда (рис. 2.3.2) [21] із заданими числами величинами конфігурацій.

Необхідно показати немонотонну взаємодію виділених схем у даній нейронній мережі. Розглянемо її спочатку без формального дослідження. Між схемами α та β існує подразнюючий зв'язок, тобто $\alpha \dashv \beta$ та відповідно $\beta \dashv \alpha$.

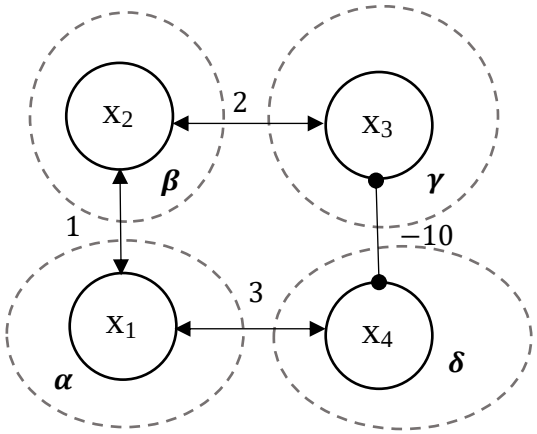

Рис. 2.3.2 – Штучна нейронна мережа типу Гопфілда

Припустимо, що вхідні дані подаються лише в схему α і $\bar{x} = (1,0,0,0)$, тоді $\alpha \dashv \delta$. Бачимо, що інгібіторський зв'язок між δ і γ значно більший, ніж подразнюючий між α і δ, то схема γ може не бути активною повністю, тоді при $\alpha \dashv \delta$ маємо $\alpha \widehat{\dashv} \gamma$.

Тепер змоделюємо роботу такої нейронної мережі у числовому вигляді шляхом активації обраних нейронів. Для мереж типу Гопфілда згідно з [21] існує наступне рівняння взаємодії нейронів у мережі:

$$x_i(t+1) = x_i(t) + \delta * d\big(x_i(t)\big) * \left(\big(1 - x_i(t)\big) \sum_{j=1}^{n} c_{ij}^{+} + x_i(t) \sum_{j=1}^{n} c_{ij}^{-} \right),$$

де $0 < \delta < 1$, c_{ij}^{+} та c_{ij}^{-} елементи матриць C^{+} та C^{-}відповідно, в яких $\forall i,j = \overline{1,n}$ $c_{ij}^{+} = c_{ji}^{+} \geq 0$ та $c_{ij}^{-} = c_{ji}^{-} \leq 0$, функція $d(x) \geq 0$ та $d'(x) > 0$.

Змоделюємо роботу ШНМ при всіх можливих початково активних нейронах, таких 16 випадків, та запишемо результати роботи у таблицю. Вкажемо вхідні дані необхідні для моделювання:

$$C = \begin{pmatrix} 0 & 1 & 0 & 3 \\ 1 & 0 & 2 & 0 \\ 0 & 2 & 0 & -10 \\ 3 & 0 & -10 & 0 \end{pmatrix}$$

$$d(x) = \frac{(\max\big((x+0.1), 0.5\big))^2}{((\max\big((x+0.1), 0.5\big))^2 + 0.2)}, \delta = 0.05$$

З результатів моделювання, представлених у додатку В у таблиці 2.3.1, видно, як при кожному можливому вхідному векторі активностей змінюється вектор активностей нейронів при стабілізації. Наприклад, у другому випадку подається вхідне значення лише на нейрон x_4, однак у ході її роботи таку саму інформацію отримує нейрон x_1, при цьому нейрони x_2 та x_3 містять у собі неповну інформацію, лише певну частину. Тоді тут наочно можна визначити, що зі схеми $\delta \dashv \alpha$. А в тринадцятому варіанті при вхідному векторі (1,1,0,0) маємо

немонотонний наслідок: $\alpha \bullet \beta \mid\!\sim \delta$, що і є немонотонним міркуванням у контексті штучних нейронних мереж.

2.4 Висновки стосовно застосування немонотонних логік

Було з'ясовано на теоретичному та практичному рівнях, як немонотонні логіки можуть застосовуватися у програмуванні, а саме при семантичному аналізі програм, при розробці бази знань та при роботі немонотонних штучних мереж. Досить вдалим можна вважати використання ФМ A-Prolog, за допомогою якої було показано аналітичні розв'язки програм у ній, які співпадають при реалізації даних програм у мові Prolog.

Задача розробки БЗ, яка розглядалася у даному розділі, напевно, не є дуже складною, однак на ній добре видно, як апарат немонотонних логік може використовуватися на практиці. Перед цим ми з'ясували, які виняткові ситуації можуть траплятися при її розробці, та описали їх спочатку на ФМ A-Prolog, застосовуючи правило заперечення як відмову, а потім реалізували дану базу знань на мові програмування Prolog.

Також були розглянуті штучні нейронні мережі, введено для них поняття схем та немонотонного наслідку. Був обраний найбільш зрозумілий приклад немонотонної штучної нейронної мережі. Разом з тим змоделювали роботу примітивної штучної нейронної мережі Гопфілда та на ній показали, яким чином нейрони немонотонно взаємодіють між собою, надаючи перевагу передачі інформації деяким з них.

Таким чином нам вдалося показати, як немонотонні логіки використовуються у задачах штучного інтелекту.

ВИСНОВКИ

У даній роботі було проведено аналіз властивостей деяких немонотонних логік та семантичних особливостей програм, в яких вони використовуються. Безпосередньо була розроблена база знань для системи базу знань структури підрозділів університету та змодельована робота штучної нейронної мережі, в яких на практиці використовується апарат немонотонних логік. Внаслідок цього було виконано мету роботи та її завдання:

1. У першому розділі наведено основні відомості про системи математичної логіки. Спочатку описано властивості та структура пропозиційної логіки, далі виокремлено центральне для логіки поняття предикату, акцентовано увагу на немонотонних предикатах. Описано класичну логіку першого порядку, показано її будову, а далі розглянуто першопорядкову логіку часткових квазіарних предикатів. Велику увагу приділено модальним логікам, виділено основні модальності. В класі модальних логік ми виокремили епістемічну та автоепістемічну логіки знання, показавши їх особливості як з теоретично-формального, так і з практичного погляду.
2. Далі ми визначили, що таке немонотонна логіка, охарактеризували її з філософського боку та вказали її різновиди. Засновуючись на знаннях про системи математичної логіки, вдалось самостійно формалізувати основні властивості пропозиційної немонотонної логіки. Додатково ввели основне припущення про закритий світ, яке дозволяє логіці стати немонотонною. Також ввели базовану на попередньому припущенні семантику логіки за замовчуванням.
3. Для подальшого огляду логіки за замовчуванням використали ФМ A-Prolog, яка є розвитком відомої мови програмування Prolog. Визначили її синтаксис та семантику, у цих термінах означили логічну зв’язку «заперечення як відмова». Ця зв’язка є тим самим провідником немонотоности у світ логічного програмування.

4. На основі вищезазначеного було вирішено декілька логічних задач на мові A-Prolog з подальшою їх реалізацією на мові логічного програмування Prolog та перевіркою розв’язків на істинність. Також було сформульовано задачу про побудову бази знань про викладання дисциплін викладачами катедр. Спочатку дана задача розглядалася з формального погляду із застосуванням апарату мови A-Prolog, а потім була запропонована реалізація у вигляді бази знань мовою Prolog. Характерною особливістю даної бази знань є те, що в ній використовується так зване CWA, характерне для немонотонних логік, яке було подане за допомогою оператора «заперечення як відмова». Це допоможе уникнути небажаних винятків у самій структурі бази знань. Також були розглянуті штучні нейронні мережі, означено їх властивості з точки зору логіки. Було розглянуто мережу типу Гопфілда та змодельовано її поведінку із застосуванням апарату немонотонної логіки.

Таким чином, ми показали, як властивості, подані за допомогою немонотонних логік, можуть реально допомогти при реалізації певних задач.

Підсумовуючи, можна сказати, що немонотонні логіки дійсно відіграють важливу роль у сучасному програмуванні. Безумовно, найперше це стосується штучного інтелекту, для якого побудова коректних баз знань та моделювання штучних нейронних мереж є дуже необхідними. Важливою особливістю такого підходу є те, що для формального опису програм існує спеціальна формальна мова, заснована на апараті логіки першого порядку. В подальшому дану тему можна більш детально дослідити у більш складних та цікавих типах штучних нейронних мережах, додатково ознайомившись із тензорним численням. Продемонстровані у даній роботі приклади із застосуванням немонотонної логіки дають змогу зробити висновок, що здавалося б така суто теоретична наука може успішно застосовуватися не тільки на теоретичному, а й на практичному рівні, зокрема при вирішенні задач штучного інтелекту.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. М. С. Нікітченко, С. С. Шкільняк. Математична логіка та теорія алгоритмів.– К.: «Київський університет», 2008. – 528с.
2. М. С. Нікітченко, С. С. Шкільняк. Прикладна логіка. – К.: «Київський університет», 2013. – 278с.
3. V. W. Marek, M. Truszczynski. Nonmonotonic Logic: Context-Dependent Reasoning. – K.: «Springer-Verlag», 1993. – 419c.
4. P. Smith. Beginning Mathematical Logic: A Study Guide. – K.: «Logic Matters, Cambridge», 2022. – 194c.
5. Чисті першопорядкові логіки квазіарних предикатів / М. С. Нікітченко, О. С. Шкільняк, С. С. Шкільняк // // Проблеми програмування. – 2016. – №2-3 – С.73-86.
6. R. Feys. Modal Logics. – K.: «Paris», 1965. – 520c.
7. А. Т. Ішмуратов. Вступ до філософської логіки. – К., 1997
8. W. H. Holliday. Epistemic Logic and Epistemology. – K.: «Springer», 2016. – 18c.
9. Logics in artificial intelligence: European Workshop, JELIA'98 Dagstuhl, Germany, October 12-15, 1998: proceedings. – Л., 1998. – 405с. // Режим доступу – Logics in artificial intelligence : European Workshop, JELIA'98 Dagstuhl, Germany, October 12-15, 1998 : proceedings : European Workshop JELIA'98 (1998 : Dagstuhl, Germany) : Free Download, Borrow, and Streaming : Internet Archive
10. Abduction in Nonmonotonic Theories [Електронний ресурс] / C. Delrieux. – «Universidad Nacional del Sur, Bahia Blanca – ARGENTINA» // Режим доступу – http://www.lip.uns.edu.ar/cmsra/delrieux0.pdf
11. Reiter R. Nonmonotonic Reasoning [Електронний ресурс] / R. Reiter. – K.: «Annual Reviews Inc.», 1987. – 40c. // Режим доступу – reiter-1.pdf (cornell.edu)
12. Teusink F. Logic Programming and Non-Monotonic Reasoning / Frank J.M. Teusink. – K.: «CWI», 1996. – 170c.

13. A-Prolog as a tool for declarative programming [Електронний ресурс] / M.Balduccini, M. Gelfond, M. Nogueira. – «Department of Computer Science The University of Texas at El Paso», 2000. – 10с. // Режим доступу – (PDF) A-Prolog as a tool for declarative programming (researchgate.net)
14. Gelfond M. Logic programming and knowledge representation – The A-Prolog perspective / M. Gelfond, N. Leone. – «Elsevier», 2002. – 36с.
15. Minker J. On indefinite data bases and the closed world assumption / J. Minker. – «CADE-82, New York», 1982 // Режим доступу – On indefinite databases and the closed world assumption | SpringerLink
16. Ben-Eliyahu R. Propositional semantics for dusjunctive logic programs [Електронний ресурс] / R. Ben-Eliyahu. Rachel, R. Dechter. – К.: «Annals of Mathematics and Artificial Intelligence», 1994 – 53-87с. // Режим доступу – (PDF) Propositional Semantics for Disjunctive Logic Programs | Rina Dechter - Academia.edu
17. L. C. Keith. Negation As Failure [Електронний ресурс] / Keith L. C. – «London, England» // Режим доступу – NegAsFailure.pdf (ic.ac.uk)
18. J. Piaget, B. Inhelder. Memory and Intelligence. – К.: «London: Routledge & Kegan Paul», 1973.
19. D. E. Rumelhart, P. Smolensky, J. L. McClelland, G. E. Hinton. Schemata and sequential thoughtprocesses in PDP models. – К.: «Cambridge, MA: MIT Press», 1986 – 7-57с.
20. M. A. Cohen, S. Grossberg. Absolutestability of global pattern formation and parallelmemory storage by competitive neural networks. – К.: «IEEE Transactions on Systems», 1983 – 815-826с.
21. J. J. Hopfield. Neurons with graded responsehave collective computational properties like those of two-state neurons. – К.: «Proceedings of the National Academy», 1984.

ДОДАТОК А

Практичне застосування апарату немонотонних логік при знаходженні множини істиннісних розв'язків логічних програм

```
p(a) :- p(c), \+ q(a).
p(b) :- not(q(b)), \+(not(q(b))).
p(c) :- q(c), \+(p(b)).
q(a).
q(b).
q(c).
```

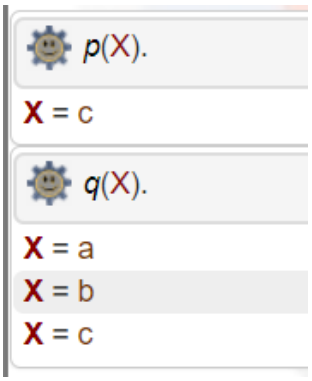

Рис. 2.2.1.1 Задача 2.2.1.1: а) – реалізація задачі 2.2.1.1; б) – множина її відповідей

```
p :- \+q.
q :- s, \+(not(p)).
s :- p.
```

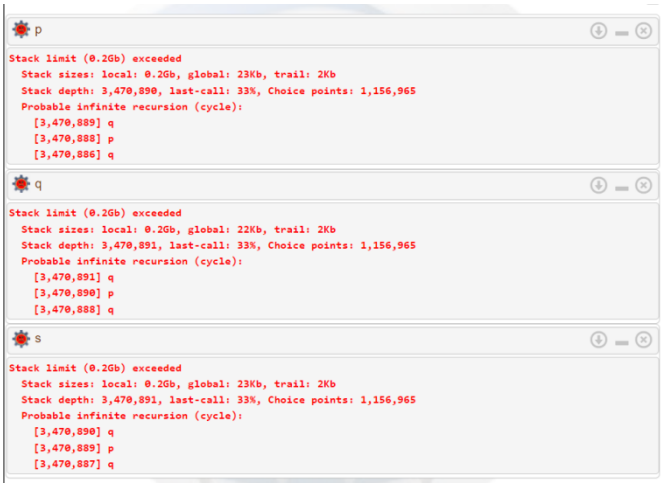

Рис. 2.2.1.2 Задача 2.2.1.2: а) – реалізація задачі 2.2.1.2; б) – множина її відповідей

```
l(0).
l(q(q(X))) :- \+ (l(X)).
p(q(X)) :- l(X), \+ (p(X)).
p(X) :- l(X), \+ (p(q(X))).
```

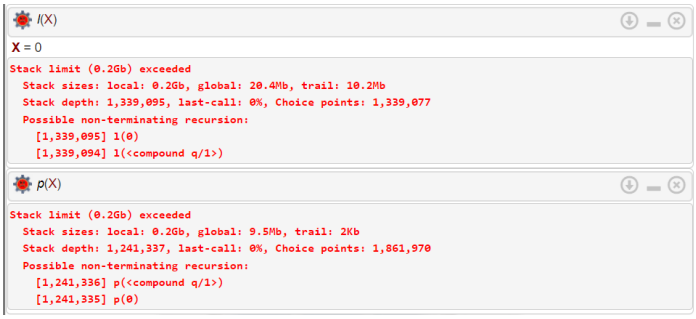

Рис. 2.2.1.3 Задача 2.2.1.3: а) – реалізація задачі 2.2.1.3; б) – множина її відповідей

ДОДАТОК Б

Лістинг програми до прикладу 2.2.1.4

```
member(vitalii, ttp).
member(anton, ttp).
member(petro, ttp).
member(anna, ttp).
member(danylo, ttp).
member(olha, tc).
member(oleksandr, tc).
member(tom, tc).
member(anatolii, mi).
member(serhii, mi).
member(nick, mi).
member(P,E1) :- part(E2, E1), member(P, E2).

course(java, ttp).
course(c, tc).
course(ai, mi).
course(logic, ttp).
course(algorithm, ttp).
course(programming, tc).
course(logic, ttp).
member_ttp (P, logic) :- member(P, ttp).
not(member(P, Y)) :- \+ (member(P, Y)).
not(course(P, cs)) :- \+ (course(P, cs)).
not(part(E1, E2)) :- part(E1,E2).
not(teaches(P, C)) :- (not(member(P, tc)), course(C,tc); not(member(P, mi)), course(C,mi)), \+ member_ttp(P,C), \+ teaches(P, C).

teaches(olha, programming).
teaches(nick, ai).
teaches(danylo, java).
teaches(oleksandr, c).
teaches(oleksandr, logic).
cathedra(ttp, cs).
```

```
cathedra(tc, cs).
cathedra(mi, cs).
part(cs, u).
part(E1, E2) :- cathedra(E1, E), part(E,E2).
```

```
member(vitalii, ttp).
member(anton, ttp).
member(petro, ttp).
member(anna, ttp).
member(danylo, ttp).
member(olha, tc).
member(oleksandr, tc).
member(tom, tc).
member(anatolii, mi).
member(serhii, mi).
member(nick, mi).
```

```
teaches(olha, programming).
teaches(nick, ai).
teaches(danylo, java).
teaches(oleksandr, c).
teaches(oleksandr, logic).
```

```
course(java, ttp).
course(c, tc).
course(ai, mi).
course(logic, ttp).
course(algorithm, ttp).
course(programming, tc).
course(logic, ttp).
```

Рис. 2.2.1.4 – Набір початкових фактів: а) – teaches (name_member,title_course); б) – member(name, cathedra); в) – course(title, cathedra).

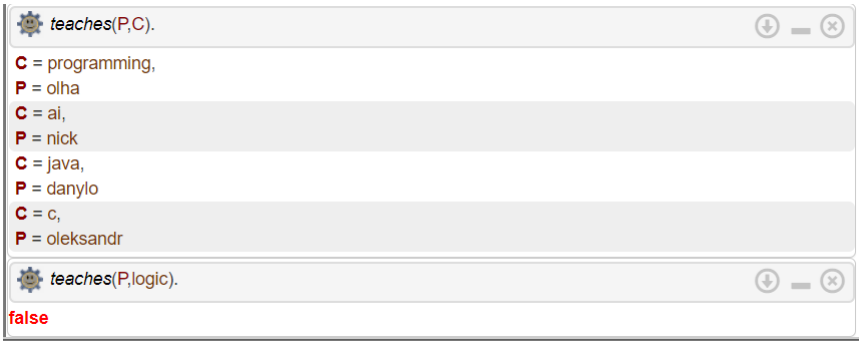

Рис. 2.2.1.5 – Результат пошуку викладачів у БЗ

ДОДАТОК В

Табл. 2.3.1 – Результати моделювання нейромережі типу Гопфілда

	Вектор вхідних значень				Результуючий вектор активностей			
	x_1	x_2	x_3	x_4	x_1^*	x_2^*	x_3^*	x_4^*
1	0	0	0	0	0	0	0	0
2	0	0	0	1	1	0.23	0.17	**1**
3	0	0	1	0	1	0.12	**1**	0
4	0	0	1	1	1	0.12	**1**	**1**
5	0	1	0	0	1	**1**	0.07	1
6	0	1	0	1	1	**1**	0.07	**1**
7	0	1	1	0	1	**1**	**1**	1
8	0	1	1	1	1	**1**	**1**	**1**
9	1	0	0	0	**1**	0.23	0.17	1
10	1	0	0	1	**1**	0.23	0.17	**1**
11	1	0	1	0	**1**	0.12	**1**	1
12	1	0	1	1	**1**	0.12	**1**	**1**
13	1	1	0	0	**1**	**1**	0.07	1
14	1	1	0	1	**1**	**1**	0.07	**1**
15	1	1	1	0	**1**	**1**	**1**	1
16	1	1	1	1	**1**	**1**	**1**	**1**