МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ТАРАСА ШЕВЧЕНКА

Кваліфікаційна наукова праця на правах рукопису

МОХАМАД АББАД

УДК 162:159.955(043.3)

РОЛЬ ЛОГІКИ В ІСТОРІЇ ТА ФІЛОСОФІЇ КОМП'ЮТЕРНИХ НАУК

Спеціальність 033 - філософія

Галузь знань - Логіка

Подається на здобуття наукового ступеня доктора філософії в галузі філософії

Дисертація містить результати власних досліджень.
Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело

_____ / М. Аввад /

Науковий керівник
Хоменко Ірина Вікторівна
Доктор філософських наук, професор

Київ - 2021

АНОТАЦІЯ

Аввад М. Роль логіки в історії та філософії комп'ютерних наук. - Кваліфікаційна наукова праця на правах рукопису.

Дисертація на здобуття наукового ступеня кандидата філософських наук за спеціальністю 033 - Філософія - (Гуманітарні науки). Київський національний університет імені Тараса Шевченка, Київ, 2021.

Результати цієї дисертації відповідають двом її основним частинам. Перша присвячена дослідженню місця логіки у філософії комп'ютерних наук та штучного інтелекту. Друга - це дослідження впливу логіки на розвиток комп'ютерних наук та штучного інтелекту.

У першій частині ми визначаємо комп'ютерні науки як багатогранну природознавчу дисципліну про інтелектуальні процеси, які можливо обчислити когнітивно та у цифровий спосіб. У цьому визначенні логіка розглядається як фундаментальна частина комп'ютерних наук наряду із інженерними рішеннями і в цілому постає набором концептів і практик, що відображають когнітивні процеси людського розуму. У контексті філософії комп'ютерних наук ми стверджуємо, що машина Тюрінга (МТ) представляє найбільш придатну модель обчислень завдяки своєму формальному визначенню та універсальним можливостям запозиченим від комп'ютерів. В наведених нами аргументах також ідеться про те, як модель МТ, в якості обчислювальної сукупності, може бути придатною і точною в характеристиці обчислюваних і, заснованих на логіці, частин пізнання. Ці формальні точність і ємність, на думку автора, дозволяють фіксувати

когнітивні логічні обчислення, які, наприклад, використовуються при маніпулюванні знанням.

У цій частині ми наводимо критичні аргументи проти точки зору радикального комп'ютаціоналізма, підкреслюючи різницю між пізнанням як динамічною, але неточно визначеною людською складовою та дуже чітко визначеною моделлю обчислень, представленою в МТ. Цим припускається певна *суттєва різниця* між пізнанням, яке можливо повністю виміряти / кваліфікувати, і машиною Тюрінга, де це не можливо. Крім того, порівнюючи пізнання з обчисленнями, ми торкнулися проблем сприйняття і свідомості як понять з невизначеною обчислюваністю. Також пропонується скептичний погляд щодо радикального гіперкомп'ютаціоналізма, який ґрунтується на складностях відшукати переконливу цілісну відповідність між людським пізнанням та гіперкомп'ютаційними моделями.

Що стосується питання морального статусу та моральної відповідальності роботів, то в даній дисертації ми стверджуємо, що моральна відповідальність має бути наслідком існуючого морального статусу. Крім того, ми вважаємо, що володіння свідомістю є необхідною умовою для надання морального статусу. Тому, на думку автора, інтелектуальні машини не мають жодного морального статусу за відсутності свідомості. Отже, ми дійшли висновку, що роботи не несуть ніякої етичної відповідальності, і ми показуємо небажані наслідки для людей, якщо припустити протилежну ситуацію. Нарешті, ми піднімаємо в цій частині проблему прозорості обчислень і показуємо, як обчислення

на основі логіки часто виявляється прозорішим, ніж за використання інших методів, таких як складні нейронні мережі.

У другій головній частині цієї дисертації ми аналізуємо теоретичні та практичні впливи логіки на комп'ютерні науки. Ми технічно пояснюємо, як Булева алгебра логіки еволюціонувала у Булеву логіку шляхом модифікації та адаптації своїх початкових законів. Потім ми показуємо, як ця нова логіка мала вирішальне значення 1) при проектуванні та аналізі сучасних схем, 2) при розробці криптографії, 3) при використанні коректуючого коду.

Друга складова цієї частини присвячена впливам з боку логіки предикатів Фреге. Описавши її високу виразність, ми детально показуємо, як відповідність між інтерпретаціями Ковальського та логікою предикатів призвела до фундаментальної семантичної еквівалентності між диз'юнктами Горна та логічним програмуванням. Потім ми надаємо докази ефективності правила резолюцій та диз'юнктів Горна в логічних розрахунках, зокрема в автоматизованому доведенні теорем. Одним із таких доказів є метод уніфікації, розгорнутий як техніка створення реального часу, поки будується доказ теореми; ця операція часто дозволяє уникнути експоненціального часу виконання.

У цій дисертації представлено багато доказів, щодо впливу теорії типів Рассела (як етапу розвитку логіки) на розробку мов програмування. Окрема увага приділяється відповідностям Каррі-Говарда між логікою та теорією типів, особливо між теоремою, що підлягає доведенню, та проблемою, яку потрібно вирішити. Серед інших моментів ми наголошуємо на правильності та ефективності програм, на

ролі типу даних при низькорівневому представленні пам'яті та на цілісності коду. Ми виокремлюємо вплив теорії типів на абстрагування даних, поліморфізм та успадкування. Крім того, ми демонструємо очевидність зв’язку між лямбда-численням та обмеженою квантифікацією з субтипуванням.

Друга частина цього дослідження, серед іншого, досліджує вплив логіки на штучний інтелект. Зокрема, ми показуємо роль логіки предикатів в системах автоматизованого доведення теорем (АДТ) за допомогою мов логічного програмування, таких як Пролог. У цьому контексті ми намічаємо значну подібність між функціонуванням логіки предикатів Фреге та логіки Пролог. Ми помічаємо, що Пролог створює список істин та уніфікацій цілей, які досягаються, автоматично використовуючи при цьому техніку пошуку з вертанням, щоб збільшити кількість вирішень. Потім ми проводимо паралель з логікою Фреге, яка спрямована на створення процесу генерування нових зростаючих істин на основі засновків та набору правил. Після цього, ми підкреслюємо різницю між машинно-орієнтованим і орієнтованим на людину обчисленням правил у контексті процедури співставлення/створення.

Ми досліджуємо використання індуктивного логічного програмування (ІЛП) у штучному інтелекті та представляємо вибірку успішних застосунків. Зокрема, ми надаємо докази впливу логіки першого порядку на методи пошукових систем, ігрові стратегії та поведінку користувачів у мобільних царинах.

Крім того, ми простежуємо плідну взаємодію між штучним інтелектом та теорією аргументації. Ми показуємо, які принципи

міркування теорії аргументації можливо застосувати у штучному інтелекті. Немонотонна логіка тут представлена як типовий приклад логіки, що з'явилася внаслідок розвитку комп'ютерних наук. Її роль особливо підкреслюється в застосуваннях ШІ і відображена в переконливих міркуваннях теорії аргументації.

Отримані нами результати в галузі історії обчислювальної логіки в основному полягають у виокремленні та аналізі основних ідей *обчислення за допомогою логіки*, що розвивалися в період від епохи Ляйбніца і до XX століття. Ми окремо досліджуємо універсальну обчислювальну машину Ляйбніца, яка мала бути практичною реалізацією його універсальної мовної характеристики та універсальної системи логічного числення (calculus ratiocinator) як символічної логіки.

Після відзначення провідної історичної ролі алгебри логіки Буля, ми приділяємо особливу увагу прориву логіки предикатів, який Фреге здійснив під час розробки програми логіцизму. Разом із цим, як уже зазначалося, показано очевидність визначального впливу цих подій на логіку та комп'ютерні науки. В історичному розгляді ми виявляємо також і інші великі досягнення логіки, такі як теорія типів Рассела, теорія доказів Гільберта, теореми Геделя про неповноту, а також зупиняємося на принципах логіки, які узгоджуються з останніми досягненнями комп'ютерних наук в двадцятому столітті.

Ключові слова: сучасна логіка, комп'ютерні науки (I), історія логіки, філософія комп'ютерних наук, філософія свідомості, штучний інтелект (ШІ), обчислення, інформація, Булева логіка, логіка предикатів, теорія типів, логічне програмування, інтелектуальні агенти, індуктивне

навчання, машинне навчання, робототехніка, теорія аргументації, нейронні мережі, пізнання, комп'ютаціоналізм, етичне обчислення.

СПИСОК ПУБЛІКАЦІЙ ЗДОБУВАЧА

1. **Awwad M**. *From Boole's Logic to Boolean Applications in Computer Science,* Educational Discourse: Collection of Scientific Papers. 2021. Vol. 32, № 4. P. 18–25.
2. **Awwad, M**., 2019. *From Philosophy to Computation: How Russell's Type Theory Impacts Programming Languages,* EESA Journal. 2019. Vol. 1, № 47 (part 7). P. 70–74.
3. **Awwad, M**., *The Role of Inductive Logic Programming in Some Machine Learning Applications,* Humanity Studies. 2018. Vol. 16, № Issue 31. P. 1–10.
4. **Awwad, M**., *Influences of Frege's Predicate Logic on Some Computational Models*, Future Human Image. 2018. Vol. 1, № 9. P. 5–19.

Опубліковані праці апробаційного характеру за матеріалами конференцій:

1. **Awwad, M**., *Frege's Logic and the Origins of Computing*, VIII Міжнародна науково-практична конференція "Викладання логіки та перспективи її розвитку", 17-18 травня 2018 р. : [матеріали доповідей та виступів]. – К.: Видавничо-поліграфічний центр "Київський університет", 2018.

2. **Awwad, M**., *The Role of Boole's Logic in Circuits Design*, International Scientific Conference - The Days of Science - Taras Shevchenko National University of Kyiv, Kyiv, Ukraine, April 23-24, 2019.

3. **Awwad, M**., *How a Logical Model strengthens digital Security*, International Scientific Conference- The War as a Problem of the World: Theory, Practice and Cultural Contexts- University of Zielona Góra, Zielona Góra, Poland, April 8-12, 2019.

ЗМІСТ

СПИСОК СКОРОЧЕНЬ

ШІ	Штучний інтелект
АДТ	Автоматизоване доведення теорем
КНФ	Кон'юнктивна нормальна форма
І	комп’ютерні науки
ЛПП	Логіка першого порядку
ІЛП	Індуктивне логічне програмування
ФЗ	Фон знань
ЗЯВ	Заперечення як відмова
ООП	Об'єктно-орієнтоване програмування
ЛП	Логіка предикатів
МП	Математичні принципи
ПР	Принцип резолюцій
МТ	Машина Тюрінга
СЗД	Система забезпечення достовірності

ВСТУП

Актуальність дослідження. Дане дослідження взаємозв'язків між логікою та комп’ютерними науками в значній мірі є актуальним і для філософії. До прикладу, розуміння природи обчислень як логічного механізму, математичної основи та наукового чи інженерного процесу істотно впливає на наші погляди щодо процесу пізнання та природи людського розуму. Більше того, стрімкий розвиток машинного навчання, автономних інтелектуальних агентів та робототехніки породив велику кількість етичних питань навколо штучного інтелекту. Зокрема, ці питання пов'язані з типами відносин, що можуть виникати між людьми та машинами. В основу таких відносин покладено багато фундаментальних понять, таких як свідомість, моральний статус, етичні обов'язки та етичне обчислення. Також філософськи актуальним вбачається намагання зрозуміти, як розвиток логіцизму, чисто філософської програми, по суті, посприяв утворенню багатьох розділів комп’ютерних наук.

Не викликає сумнівів також і актуальність цього дослідження в царинах логіки та комп’ютерних наук. Наприклад, з розвитком комп’ютерних наук, обчислювальна логіка та її прикладні аспекти стали захоплюючим та перспективним напрямком як в самій логіці, так і в теоретичних комп’ютерних науках. Крім того, вони є наріжним каменем як математичної логіки так і сучасних застосунків штучного інтелекту.

Типовими прикладами тут постають немонотонна логіка та індуктивне логічне програмування.

Іншим важливим аспектом актуальності даного дослідження є брак послідовного представлення в наявній дослідницькій літературі ґрунтовних теоретичних узагальнень щодо взаємодії логіки та комп'ютерних наук. Дійсно, рівень узагальнень, що зустрічається щодо техніки та технологій в комп'ютерних науках, часто або дуже високий і не дає бажаної точності, або дуже низький, що робить тему незрозумілою та, подекуди, недосяжною для непрофесіонала. Це один з проблемних моментів, що виявляється при дослідженні даної теми.

Більше уваги має бути приділено також дослідженню логічної природи та логічних засад комп'ютерних наук. Крім того, простежується нестача спеціальних досліджень обчислювальних структур логіки та їх історичного розвитку. Дане дослідження прагне по можливості вирішити ці проблеми шляхом більш деталізованого представлення, та здійснення необхідних теоретичних узагальнень щодо логічних принципів комп'ютерних наук. Крім того, ми здійснюємо таку деталізацію, що цілком здатна прояснити і поглибити розуміння відношення між, на приклад, алгеброю логіки та побудовою Булевих схем; між логікою предикатів та автоматизованим доведенням теорем; між теорією типів та основами мов програмування; між теорією аргументації та штучним інтелектом.

Зв’язок із науковими програмами, планами, темами. Наукове дослідження, представлене в цій дисертації, виконано в рамках дослідницької програми 17КФ041-01 "Сучасна логіка: теорія, історія та прикладні аспекти" кафедри логіки філософського факультету Київського національного університету імені Тараса Шевченка.

Мета і завдання дисертаційного дослідження. Головною метою даного дослідження є розкриття та прояснення суттєвих елементів, що пов'язують логіку та обчислення.

Для досягнення цієї мети було визначено **п’ять основних завдань**.

Перше завдання полягає в тому, щоб зрозуміти природу комп’ютерних наук та вивчити застосовність машини Тюрінга в якості обчислювальної моделі за замовчуванням в комп'ютаціоналізмі та в контексті філософії комп’ютерних наук.

Друге завдання - проаналізувати зв'язки між обчисленням, пізнанням та логікою, а також між обчисленням та етикою, особливо в частині морального статусу та відповідальності, прозорості обчислень та ролей людини і машини.

Третє завдання полягає в тому, щоб віднайти найвпливовіші логічні теорії, які успішно покладено в основу досягнень комп’ютерних наук, і з’ясувати теоретичні та технічні *передумови* цих впливів.

Четвертим завданням є вивченні ролі логіки у розробці штучного інтелекту (ШІ) та визначенні центральних принципів ШІ, які найбільше виграли від впливів логіки.

П'яте завдання полягає у виокремленні головних *обчислювальних* ідеї в історії логіки від Ляйбніца до XX ст.

Об'єктом даного дослідження виступає сучасна логіка.

Предметом дослідження є теоретичні та прикладні взаємозв'язки логіки та комп'ютерних наук.

Теоретико-методологічна база дослідження. Дослідницька методологія даної дисертації сформульована в рамках поліметодологічного підходу, який за своєю суттю є аналітико-синтетичним, порівняльним та орієнтованим на опитування. Аналітико-синтетичний метод був використаний у проясненні абстрактних понять, що досліджувалися у цій дисертації. Наприклад, за допомогою цього методу стало можливим: 1) цілісно проінтерпретувати природу комп'ютерних наук; 2) здійснювати аналітичну аргументацію та застосовувати модель машини Тюрінга в комп'ютаціоналізмі; 3) проаналізувати комп'ютаціоналізм та етичні елементи дослідження. Порівняльний метод був застосований при порівнянні обчислювальних теоретичних принципів логіки з їх практичними та технічними реалізаціями в комп'ютерних науках та ШІ. Без цього методу не можливо було б зрозуміти впливи логіки на її конкретні застосування в комп'ютерних науках. Метод послідовного огляду в основному

використовувався для вивчення історичного розвитку логіки майже хронологічним способом, починаючи з XVII століття. Цей метод дозволив виділити основні впливи в обчислювальній логіці та виявити їх безперервність у накопиченні знань. У підсумку, це дозволяє візуалізувати прогрес теоретичних принципів, що призвів до можливості реалізації їх в комп'ютерних науках. В основу цієї дисертації покладено роботи таких видатних логіків, як Г.В. Ляйбніц, Д. Буль, Г. Фреге, Б. Рассел, А.Н. Вайтгед, Д. Гільберт, К. Гедель, А. Тюрінг, А. Черч, Л. Кутюра та багатьох інших. Крім того, дослідження також базувалося на працях сучасних вчених, таких як М. Девіс, Д.М. Габбей, Л. Хаапаранта, Ф.Х. Емерен, І. Хоменко, В. Кніл, М. Кніл, В. Дж. Рапапорт, Дж. А. Робінсон, Дж. Х. Соракер, Р. Ковальський, А. Колмерауер, С. Магглтон, А. Горн, Д. Гілліз, Г. Піччініні, Дж. Й. Гальперн, Р. Л. Констебль.

Наукова новизна отриманих результатів. В процесі реалізації завдань дисертації, ми дійшли наступних *відповідних* висновків:

1- Ми визначаємо комп'ютерні науки як багатопрофільну *науку когнітивного опрацювання*, що має в своїй основі логічні теорії та інженерні реалізації. На користь цього визначення свідчать такі характерні риси комп'ютерних наук, як опрацювання інформації, процедурне мислення, використання абстракцій та синтетичної інженерії.

Ми переконливо обґрунтовуємо правильності розуміння машини Тюрінга як стандартної моделі обчислення в комп'ютаціоналізмі.

2- Ми пропонуємо аргументи проти радикального комп’ютаціоналізма та на користь часткового комп’ютаціоналізма особливо щодо представлення людських когнітивних феноменів як заснованих на логіці процесів. Ми представляємо скептичну позицію щодо гіперкомп’ютаціоналізма, обґрунтовану низкою концептуальних перешкод, лишаючи при цьому осторонь оцінку можливості його реалізації.

Крім того, ми представляємо умови, за яких роботи можуть виконувати людські завдання і ситуації, в яких вони не мають цього робити. Ми також доводимо, чому у роботів не має бути морального статусу і визначаємо наслідки впровадження принципів прозорості обчислень.

3- Ми вважаємо алгебраїчну логіку Буля, логіку предикатів Фреге та теорію типів Рассела найважливішими серед логічних теорій, що вплинули на розвиток комп’ютерних наук. Ми показуємо, як логіка Буля перетворилася з алгебраїчних часткових законів мислення на Булеву логіку і призвела до: 1) появи аналізу та проектування схем; 2) сильного впливу на криптографію та коректуючий код.

Ми наводимо велику кількість технічних і теоретичних причин значного впливу логіки предикатів на комп’ютерні науки. Ці причини в основному пов'язані з: 1) високим рівнем виразності логіки предикатів 2) відповідністю між логікою предикатів і інтерпретацією Ковальського 3) еквівалентністю між диз'юнктами Горна і семантикою логічного

програмування 4) роллю принципу резолюцій в проектуванні мов логічного програмування.

Крім того виявлено, що головні впливи теорії типів на комп'ютерні науки пов'язані із: 1) перевагами, що їх теорія типів надає щодо коректності, ефективності та цілісності програм; 2) роллю теорії типів в абстрагуванні даних, поліморфізмі та наслідуванні в мовах програмування; 3) роллю відповідностей Каррі-Говарда між теорією типів та логікою, особливо в частині, еквівалентності між доведенням та програмою, що пройшла перевірку на відповідність типів.

4- Ми прийшли до висновку, що логіка в основному вплинула на штучний інтелект через принципи, що застосовуються в мовах логічного програмування. Ми показуємо, що ці принципи пов'язані з інтерпретацією Ковальського та використанням вирішення/уніфікації за допомогою семантичної процедури при доведенні формул. Ми вважаємо індуктивне логічне програмування, автоматизоване доведення теорем (АДТ) та мови логічного програмування (такі як, наприклад, Пролог) такими, що серед інших базових досягнень ШІ, найбільше виграли від впливів логіки. Ми обґрунтовуємо таку вибірку репрезентативними успішними прикладами АДТ; ми також наводимо додаткові приклади використання індуктивного логічного програмування (ІЛП) у машинному навчанні, а саме технології пошукових систем, комбінаторну стратегію ігор та отримання даних у мобільних застосунках, що відстежують поведінку користувачів. Крім того, ми наголошуємо на важливості використання немонотонної логіки в якості

вірогідних міркувань у контексті теорії аргументації, та застосування цієї теорії у штучному інтелекті.

5- Ми виявили, що, починаючи з часів Ляйбніца, в історії обчислювальної логіки можна виокремити наступні основні ідеї: двійкова система Ляйбніца при проектуванні синтаксичної універсальної машини; універсальна характеристика Ляйбніца як мова міркування та його універсальна система логічного числення, як механізм міркування; Булева алгебра як символічна мова думок; силогістична логіка Венна та логіка відношень з використанням алгебри; вдосконалення де Морганом алгебри Буля; здійснене Пірсом та поглиблене Шредером, об'єднання силогістики та алгебри логіки; логіка предикатів Фреге як формальна високовиразна мова; теорія типів Рассела та сучасна символічна логіка з використанням логіки Пеано; теорія доведення Гільберта та проблеми повноти; теореми Геделя про повноту та неповноту; обчислюваність Тюрінга.

Практичне значення отриманих результатів. Ця дисертація може бути використана при пошуку репрезентативних філософських ідей, подібних до програми логіцизму, які істотно вплинули на розвиток комп’ютерних наук та появу вагомих здобутків в логіці. Дослідники, які цікавляться міждисциплінарними дослідженнями на перетині логіки, комп’ютерних наук, філософії, математики та когнітивних наук, можуть знайти в дисертації кілька елементів, що теоретично і практично пов'язують ці дисципліни. До прикладу, обговорювані в цьому

досліджені поняття комп’ютаціоналізма та гіперкомп’ютаціоналізма демонструють спільність сфери інтересів. Дослідники та студенти можуть включити порушені питання та пропозиції щодо етичних обчислень у свої дослідження. Це може бути корисним для пошуку кращих рішень відносно етичних наслідків практик штучного інтелекту. Ці наслідки можуть бути пов'язані з прозорістю обчислень, статусом машини та моральними обов'язками. Нарешті, зазначені етичні проблеми можуть бути використані в навчальних програмах з комп’ютерних наук, щоб ознайомити студентів із тим як штучний інтелект впиває на суспільство.

Особистий внесок здобувача. Ця дисертація виконана автором самостійно. Результати роботи та висновки представлені на основі особистих досліджень автора. Основна частини цих результатів опублікована автором у наукових журналах без співавторства.

Апробація результатів дисертації. Основні результати дисертації були представлені та обговорені на багатьох засіданнях кафедри логіки філософського факультету Київського національного університету імені Тараса Шевченка. Ці результати також були представлені на таких міжнародних наукових конференціях: VIII Міжнародна науково-практична конференція - Викладання логіки та перспективи її розвитку - 2018 (Київський національний університет імені Тараса Шевченка, Київ, 17 - 18 травня 2018 р.); Міжнародна наукова конференція - Дні науки філософського факультету - 2019 (Київський національний

університет імені Тараса Шевченка, Київ, 23 - 24 квітня 2019 р.); Міжнародна конференція - The War as a Problem of the World: Theory, Practice and Cultural Contexts - 2019 (Зеленогурський університет, Зелена Гура, Польща, 8 - 12 травня 2019 р.).

Публікації. Основні результати дослідження були представлені в семи публікаціях. Три з них були опубліковані в українських наукових журналах (один із яких міжнародний), одна стаття була опублікована в міжнародному зарубіжному журналі, дві інші були представлені в матеріалах українських міжнародних наукових конференцій, а одна в матеріалах міжнародної зарубіжної конференції.

Структура та обсяг дисертаційної роботи. Дисертаційна робота містить анотацію, вступ, чотири розділи, висновки до кожного із розділів, загальні висновки до дисертації та список літератури, що налічує 89 джерел. Основний текст дослідження (включаючи анотацію) викладено на 207 сторінках. Загальний обсяг рукопису налічує 211 сторінки.

Розділ 1. Вибірковий огляд історії логіки

У цьому розділі представлені окремі ідеї з історії логіки, принципово пов'язані з розвитком сучасної логіки. У ньому містяться визначення двійкової моделі обчислень Ляйбніца і пояснення двох його фундаментальних логічних концептів: універсальна система логічного числення (calculus ratiocinator) і універсальна характеристики, що доповнює його концепцію використання універсальної машини.

Підкресливши важливість робіт Де Моргана, Пірса та Гамільтона в часи математичного повороту в логіці, ми демонструємо, як Булева алгебра логіки була націлена на створення формальної синтаксичної моделі з використанням набору логічних законів. Це робилося задля можливості механічним чином виконувати логічний аналіз та обчислення. Ми також виокремлюємо логіку предикатів Фреге, як мову високої виразності, що використовує функції, зв'язані змінні та квантифікації та вважаємо її появу важливим проривом в історії логіки.

Більше того, ми показуємо, як внесок Рассела в логіку зіграв важливу роль у створенні логіки вищого порядку, і появі теорії типів, яка, в свою чергу, мала величезні наслідки для історії обчислювальної логіки.

В останніх частинах цього розділу розглядається теорія доведення Гільберта та його праці про послідовність аксіоматики. Вслід за ними

розглянуто теорему Геделя про повноту стосовно логіки першого порядку та його теореми про неповноту, що постали важливою віхою історію логіки, філософії та комп'ютерних наук. Усі попередньо обговорювані поняття логіки врешті перегруповано та представлено як основні складові досягнень комп'ютерних наук.

1.1. Логіка Ляйбніца і обчислювальна мрія

Метою цього підрозділу є показати, як Ляйбніц задумав створити універсальну машину з використанням двійкової системи більш ніж за двісті років до того, як були спроектовані цифрові комп'ютери. Ця машина мала використовувати два основні концепти, розроблені Ляйбніцом: "універсальна характеристика" та "універсальна система логічного числення" (calculus ratiocinator). Перший - це алфавіт людського мислення або концептуальна синтаксична *мова символів*. Другим концептом, який повинен був підтримувати перший, постала теоретична обчислювальна символічна логіка, що мала на меті дедукцію істини.. Спираючись на ці новаторські концепції, ми, в певному сенсі, стверджуємо в цьому розділі, що Ляйбніцева мрія(і) була повністю реалізована у XX столітті.

До кінця XVII століття, логіка в основному була еквівалентна арістотелівській теорії силогізму. Перші праці Готфріда Лейбніца (1646-1716) в царині логіки також в основному спиралися на силогістичну модель. Однак Ляйбніц ставив собі на меті створити потужнішу теорію,

під назвою *"універсальна система логічного числення" (calculus ratiocinator)* яка була призначена для формальної ідентифікації правильних висловлювань. Логічне числення може бути також застосоване і до довільних висловлювань, щоб систематичним чином визначати їх істинність або хибність. Ця ідея розглядалася як *універсальна характеристика*, що означає, що обчислення має бути автоматизованим для за будь-якого різновиду *характеризованих* висловлювань.

Наступний відомий уривок Ляйбніца в «Про мистецтво комбінаторики» (1685) чітко виражає його мету:

> … Якщо це буде здійснено, щоразу, коли виникатимуть розбіжності, не буде більше потреби сперечатися двом філософам, як нібито вони були математиками. Бо достатньо буде взяти ручки в руки і сісти біля рахівниці, кажучи один одному (або, якщо вони забажають і другові, покликаному на допомогу): Давайте обчислимо. (Ляйбніц) [1, p. 130]

Цій меті передувала захоплююча та новаторська ідея обчислень, яку Ляйбніц презентував у своїх працях: двійкова система. У наступному підрозділі описується коліщаткова машина Ляйбніца, заснована на концепції бінарності.

1.1.1. Колесо Ляйбніца на двійковій системі

У своїх трактатах *Прогрес дидактики* (1679) [2] та *Експлікація двійкової арифметики* (1703) [3]Ляйбніц з'ясував важливість використання двійкової системи, яка дозволяє виконувати арифметичні операції простіше, ніж за використання інших числових систем. Він дослідив використання двійкової системи, надавши детальний опис її застосовування до чотирьох основних арифметичних операцій (додавання, віднімання, множення та ділення). Машина Лейбніца, що інтегрувала новий механізм, отримала назву *Колесо Лейбніца* і була в змозі виконувати арифметичне множення та ділення. У відповідності зі своєю двійковою системою Ляйбніц описав бінарну машину, що працювала за допомогою отворів, паличок та канавок для кульок. В узагальненому вигляді все це описано в праці *Прогрес дидактики* наступним чином:

> Таке [двійкове] числення може бути реалізоване за допомогою машини. Наступний метод, безсумнівно, буде дуже легко втілити. [Машина з] отворами, які можна відкривати та закривати. Вони мають бути відкритими в тих місцях, які відповідають значенню "1", і залишатися закритими в місцях, що відповідають значенню "0". Крізь відчинені дверцята, дрібні кубики або скляні кульки повинні

> потрапляти в канавки; крізь інші нічого не буде падати. Він [набір дверцят] має пересуватися від стовпця до стовпця, як це потрібно для множення [...] Тому що можна встановити так, що дві завжди випадають разом, по інакшому вони не мають випадати (Ляйбніц)[2].

Цей оригінальний спосіб двійкового обчислення, ймовірно, став першим кроком до більш універсальної машини, здатної виконувати набагато складніші операції, особливо пов'язані з логічними міркуваннями. Про таку *універсальну* машину, що про неї мріяв Ляйбніц, йтиметься у наступному підрозділі.

1.1.2. Мрія Ляйбніца про "універсальну машину"

Однією з мрій Ляйбніца було знайти спосіб перетворення міркувань з поняттями в міркування з числами, використовуючи доступні сучасні йому інструменти. Його мрія в основному полягає у створенні універсальної символічної мови, яка може виражати всі аспекти знань і здатна надати правила для того, щоб поєднати засновки та виконати необхідні обчислення для перевірки існуючих чи отримання нових знань. Іншими словами, він хотів створити алфавіт, символи якого відповідають поняттям, що дозволило б перевіряти істинність речень за допомогою символьних обчислень.

Водночас, його мрія не зупиняється тільки на створенні зазначеної мови, він мріє надалі збудувати машину, здатну фізично виконувати такі обчислення і цим вивільнити людський розум від механічної роботи, дозволивши йому зосередитись на творчості.

Цю ідею часто вбачають передтечею того, що сучасні комп'ютери здатні символізувати, вирішуючи різні задачі з використанням двійкової системи.

Ляйбніц так описав типи обчислень універсальної машини своєї мрії:

> ... "Але якщо обмежитись науковим застосуванням, то старі геометричні та астрономічні таблиці можна було б виправити та побудувати нові, за допомогою яких можливо було б вимірювати всілякі криві та фігури... буде корисним розширити, наскільки це можливо, основні таблиці Піфагора: таблицю квадратів, кубів та інших степенів; таблиці комбінацій, варіацій та прогресій усіх видів... (Ляйбніц) [4, p. 8]

Задумана ним машина здатна розв’язати алгебраїчне рівняння та майже будь-які міркування та обчислення. Знову ж таки, універсальність

його проекту, втіленого цією машиною, чітко висловлене наступним фрагментом:

> Мій винахід передбачає застосування розуму, судження в кожній суперечності, аналіз усіх понять, оцінку ймовірності; він - це компас для навігації океаном нашого досвіду, опис усіх речей, таблиця всіх думок, [...] Мій винахід - це невинна магія, нехимерна Кабала, письмо, яке кожен може прочитати і якому дуже легко навчитися...(Ляйбніц) [5].

В основу ідеї створення універсальної машини покладено два згадані раніше фундаментальні концепти: універсальна (або загальна) характеристика та універсальна система логічного числення (calculus ratiocinator). У наступних підрозділах обговорюються, відповідно, ці дві, плекані Ляйбніцем ідеї.

1.1.3. Синтаксичний погляд на універсальну характеристику

Ляйбніц намагався об'єднати всю систему комбінацій у набір правил для того, щоб виконувати обчислення на кожному кроці та в результаті досягнути опрацювання потрібної глобальної системи. Здійснюючи це, Ляйбніц створив відповідний процес, що дозволяє вкласти зміст поняття у величину або число.

> Я ще раз згадав свій створений раніше план нової мови чи системи письма розуму, який міг би служити інструментом комунікації для всіх різних народів... Якби ми мали такий універсальний інструмент, ми могли б обговорювати проблеми метафізики або питання етики так само, як проблеми та питання математики або геометрії. […] Таким чином, в раз виникнення суперечки, два філософи могли сісти за стіл і просто обчислюючи, як два математики, вони могли б сказати: "Давайте це перевіримо...(Ляйбніц) [5].

Таке механічне обчислення, що розглядається як універсальна мова, йде пліч-о-пліч з ідеєю використання двійкового числення.

Працюючи над універсальною характеристикою, Ляйбніц досліджував взаємозв'язок між знаками та процесом міркування. Зв’язок між логікою та інтуїцією не існує a priori. Однак, науковий підхід здатний "встановити безпосередній контакт композицій знаків із змістом понять"[6, p. 3]. У цій роботі Тренделенбург вважає, що характеристика намагається створити відповідне значення сутності, особливо при аналізі матеріалу понять і, отже отримує можливість здійснювати над ними обчислення [6, p. 6]. Крім того, він вважав, що основні ідеї Булевої алгебри логіки такі, як $a^2 = a$, вже були відомими Ляйбницю, і тому

логічне обчислення Ляйбниця було доступним з 1840 року завдяки виданню Ердмана [7, p. 130].

Крім того, Ляйбніц мав на меті створення алфавіту, згідного з людським розумом, що підкріплений обчислювальним механізмом для роботи з ним. Символи або алфавіт, що використовуються в цій системі, часто вважаються такими, що дали початок слову характеристика (символізація). Кожен символ позначає конкретну ідею. Ляйбніц вважав, що можна створити загальну характеристику, що представлятиме справжній і цілісний людські розум і свідомість, і, врешті, вона могла бути універсальною. У цьому контексті універсальна характеристика розглядається як точна символічна мова, яка враховує складні зв’язки людських думок [4, p. 15].

Більше того, характеристика використовується не тільки для пошуку помилок, а й для з'ясування істини. Вона використовується не лише для доведення істин, але й для їх винаходження[8]. Ляйбніц вбачає в характеристиці модель, здатну охопити всі відомі та раціональні факти. Її межа повинна співпадати з межами розуму, що розширює здатність розуміння, і відображати відкриття, які розум може робити і доводити засобами логічних операцій. Отже, сила розуму збільшується, а сфера інтелектуальної інтуїції розширюється. Ідея Лейбніца щодо винаходження істин, спонукає нас провести паралель з ідеєю нарощування знань, реалізованою сучасними комп’ютеризованими механізмами, що використовують окремі концепти штучного інтелекту,

такі як індуктивне логічне програмування, яке можна віднести до індуктивного навчання.

1.1.4. Універсальна система логічного числення (calculus ratiocinator) як програмне забезпечення для міркування

Розробляючи проект універсальної характеристики, Ляйбніц мав намір створити алгебру понять, використовуючи мову першого порядку. Ця алгебра інтегрує символічні функції заперечення, кон'юнкції та тотожності понять. Ляйбніц був переконаний, що розвиток символічної мови дозволить механічно вирішити велику групу проблем.

Варто відзначити, що концепція машини Ляйбніца вимагає комбінування логіки і арифметики, що є непростим завданням. Насправді, при такому комбінуванні часто застосовується квантифікація змінних, які використовуються в арифметичних формулах; маніпуляції з ними, як правило, складніші, ніж звичайна оцінка [9].

Більше того, маніпулювання цими символами визначеної мови є способом застосування логічних правил дедукції. Ляйбніц називає обчислювальне ядро цієї маніпуляції *універсальна система логічного числення (Calculus Ratiocinator)* і її можна вважати еквівалентом сучасної символічної логіки [4, p. 17].

З використанням універсальної системи логічного числення (calculus ratiocinator) та узагальненої символічної мови, доведення

теорем стає послідовністю механічних кроків, починаючи від засновків і закінчуючи висновком, який необхідно довести. Ця система повинна дозволяти дедукувати характеристики понять із синтаксису символів. Первинні результати арифметики можуть бути, наприклад, представлені двійковою системою.

Таким чином, "характеристика" сприяє розуму за допомогою обчислювальних правил як "суддя суперечностей" [8]. Отже, коли потрібно обґрунтувати філософську дискусію чи суперечку, для Ляйбніца єдиним способом зробити це є *обчислення*.

1.1.5. Система L1 та інші розширення

Лейбніц розробив ієрархію логік, кожна з яких містить набір аксіом і правил, що описують її силу. Згідно з Ленценом, мабуть найважливішим численням, що його Ляйбніц розробив у 1686 році і позначив як L1, є алгебра концептів [10, p. 3], еквівалентна алгебрі множин; вона містить повний набір аксіом, і її можна вважати еквівалентом Булевої алгебри. Це числення базується на пропозиційному численні, що пізніше постане фундаментальним у пропозиційній логіці.

Вводячи та доводячи закони логіки концептів, Ляйбніц безпосередньо використовував закони пропозиційної логіки. В цьому процесі приймається аналогія між поняттями та висловлюваннями а, отже, концептуальні сполучники можна трактувати як пропозиційні

сполучники. Таке тлумачення перетворює алгебру, що має справу з концептами, в алгебру, що має справу з висловлюваннями[10, p. 4].

Таким чином, розширення цієї логіки Ляйбніца, яка допускає невизначені поняття, дозволяє отримати передчечу теорії квантифікації з поняттями. Ця ідея щодо індивідів-концептів виявилася досить точною, що дозволило згодом перебудувати її принципи, що призвело до визначення логічної системи другого порядку, під назвою L2. Ця система дозволяє по-різному, але точно представити категоричні аспекти теорії силогізму.

Насправді, традиційно починають з квантифікації елементів, яка продовжується квантифікацією предикатів. У Ляйбніцевій логіці — навпаки, спочатку відбувається квантифікація концептів. У цьому сенсі логіка другого порядку бере свій початок в пропозиційній логіці із строгою імплікацією.

1.1.6. Висновок

В дослідницькій літературі щодо логіки Ляйбніца можна зустріти багато негативних оцінок або суджень, що демонструють недооцінку її вартості. Наприклад, у важливій книзі *Розвиток логіки*, автори так описали досягнення Ляйбніца в логіці:

> Коли він починав, він, без сумніву, мав намір створити щось ширше, ніж традиційна логіка. [...] Але хоча він працював

над цією темою у 1679 р. та у 1690 р., йому так і не вдалося створити числення, яке б навіть охоплювало загалом теорію силогізму [...] (Kneale & Kneale) [11, p. 323].

На щастя, в середині 80-х років нарешті були надані різні докази, що демонструють фундаментальні результати, отримані цією логікою, всупереч оцінкам подружжя Ніл та Кутюра.

Ленцен у [10, p. 9] наступним чином описав деякі з найважливіших досягнень Ляйбніца:

1) "Ляйбніцева *інтенсіональна* інтерпретація концептів є еквівалентною сучасній ексистенсіональній інтерпретації" [10, p. 9].

2) “Ляйбніцева *алгебра концептів* еквівалентна Булевій алгебрі множин” [10, p. 9].

3) "Ляйбніцева теорія *невизначених концептів* передбачає появу сучасної теорії кванторів" [10, p. 9].

4) "Ляйбніцеве *універсальне числення* дозволяє по-різному вивести закони теорії силогізму" [10, p. 9].

Нарешті, погляди Ляйбніца в галузі логіки безпосередньо пов'язані з його поглядами в галузі математики і метафізики. Ця думка чітко сформульована у вислові: "*Моя метафізика - це суцільна*

математика... Я переконався, що справжня метафізика нічим не відрізняється від справжньої логіки"[10, p. 2].

1.2. Трансцендентальна логіка Канта та Діалектична логіка Гегеля

Не зважаючи на те, що Іммануїл Кант (1724-1804) безпосередньо не займався дослідженнями в галузі *формальної* логіки, він просував три центральні ідеї сучасної логіки: "розрізнення концепту та об'єкту, першість висловлювання як одиниці логічного аналізу та розуміння логіки як дисципліни, що займається вивченням структури логічних систем, а не лише правильності висновків" [12].

Внесок Канта в логіку широко пов'язують із трансцендентальною логікою, описаною ним самим наступним чином:

> […] Така наука, що визначала б походження, обсяг і об'єктивну значущість подібних знань, повинна називатися трансцендентальною логікою, бо вона має до діла виключно закони розсудку і розуму, проте лише оскільки вони відносяться до предметів a priori, а не до емпіричних, і до чистих розумових знань без розрізнення, як у загальній логіці (Кант) [13, p. 66].

Такий опис свідчить про те, що Кант вважає теорії Локка, та Ляйбніца теоретично обмеженими рамками загальної логіки. Переходячи до трансцендентальної логіки, Кант відкинув як емпіричні, так і раціоналістичні пояснення щодо відношення між ідеями. В рамках цього підходу Кант чітко розмежовував логіку, філософію та математику. Отже, математика не є релевантною для філософії, і філософія не в змозі запропонувати a priori гарантії, застосовні в математиці. Говорячи це, Кант намагався показати пізнавальну обмеженість загальної логіки і необхідність переходу до запропонованої ним трансцендентальної логіки.

У цьому контексті загальна логіка розуміється як інструмент, що стосується правил, що представляють форму думки, пристосовану до потреб загального розуміння. Це можна зробити абстрагуванням щодо заданих знань. Фреге з цього приводу сказав, що загальна логіка може дістатися лише до "законів мислення, але не до законів істини" [12]. Іншими словами, вона не в змозі отримати жодного формального принципу, що веде до визначення істини, оскільки ці принципи в основному будуються з використанням зв'язку знання з його об'єктом [12].

Для Канта сила розсудку полягає в можливості надати "єдність явищам за допомогою правил" [12]. Водночас, сила розуму полягає в "об'єднанні правил розсудку за принципами" [12]. Ця єдність, єдність розуму, сприймається як абсолютно відмінна від тієї єдності, яку може

запропонувати розсудок. Розсудок видається підрівнем розуму, або, що те ж саме, розум - це свого роду поняття, що виражає мета-рівень розсудку.

Розвиваючи свою трансцендентальну діалектику, Кант робить наголос на логічному застосуванні розуму, прийнятому в загальній логіці, та генерує силу розуму, яку описує як трансцендентальну. Він пов'язує "безпосередні умовиводи з розсудком, а опосередковані з розумом" [12]. Важливість його загальної характеризації умовиводу можливо пов'язати з тим, що умовивід базується не на порівнянні взятих ідей, а на застосуванні відповідних правил.

Крім того, Кант охарактеризував силу судження як здібність, що працює із правилами, а також дослідив концепти як правила. Це передбачає міцний взаємозв'язок між логікою, логічними формами судження та функціями єдності в судженні [12].

Що стосується логіки Гегеля, то Георг В. Гегель (1770-1831) був зосереджений більше на метафізичних термінах, таких як кількість, якість, буття та сутність, ніж на традиційних логічних поняттях. Тим не менш, він досліджував процеси сприйняття логіки як *науки про міркування*. Він намагався надати розуміння безпосередніх умовиводів як рефлексивного мислення, які, як правило, визнавалися само собою зрозумілими без необхідності особливого розгляду. Таким чином, його метою було вивчити характеризацію раціональних процесів мислення.

На Гегелеві погляди спочатку істотно впливала Кантова "Критика чистого розуму", в якій він досліджував чисті розсудкові поняття, використовуючи традиційні форми логічних суджень. Це вважається поворотом від чуттєвого споглядання до процесу (розсудкового) осягнення. Він вважав, що для розуміння процесів мислення ми повинні використовувати мову, здатну працювати з концептуальним значенням. Слід зосередитись на "інтелектуальних функціях та змінах значення" [12]. Зосереджуючись на дискурсивній динаміці міркувань, він показує, що поняття можуть змінювати значення. Таким чином, він дав аналітичний опис того, як відбувається зміна значень та їх злиття незалежно від будь-якого традиційного силогістичного пояснення. Гегель вважає, що будь-яка рефлексія щодо чистого мислення, з використанням його власних методів, не тільки не позбавлена вмісту окремих деталей, а і знаходиться під безпосереднім впливом людського досвіду. Завдяки цьому мисленню можливо виправити всі добре перевірені форми свідомості. Отже, людський досвід є необхідним елементом Гегелевої логіки для того щоб зрозуміти ідентифікацію мислення, яке схоплює "закономірності і процеси дійсності" [12].

Тут можна віднайти паралелі з традиційною метафізикою, яка використовувала розум, щоб схопити природу буття; Гегель показав, що людський розум завжди навчався, використовуючи власний досвід середовища, щоб охопити форми реальності. Отже, філософ може

обґрунтовано припустити, що чиста думка ґрунтується не лише на логічних принципах, а й на численних метафізичних концептах.

Резюмуючи, можна сказати, що немає нічого дивного в тому, щоб розуміти логіку Гегеля як нову версію традиційної метафізики [10, p. 137].

1.3. Математичний поворот у логіці, застосований в комп'ютингу

В кінці XIX століття зацікавленість логіків та математиків символічною логікою зменшується. Вайтгед влучно описав цю ситуацію наступними словами: "Символічна логіка [...] була відкинута багатьма логіками на тій підставі, що її інтерес - математичний, і багатьма математиками на тій підставі, що її інтерес - логічний" (Вайтгед) [10, p. 545].

Британський логік Джон Венн (1834-1923) вперше використав термін *символічна логіка*, що підкреслює важливість використання символів, пов'язавши їх із відповідними математичними теоріями, які приймають ці символи. Ці математичні теорії часто вдавалися до міркувань щодо того, як моделювати силогістичну логіку та логіку відношень в алгебраїчному ключі. Логіка відношень, розроблена Августом Де Морганом (1806-1971), включає використання квантифікації предикатів. Більш того, ці математичні теорії включають значне вдосконалення алгебри логіки, створеної Джорджем Булем (1815-1864), яка забезпечує кращу виразність і більш потужні

дедуктивні міркування, ніж силогістичної логіка. Цьому прогресу сприяли різні поняття алгебри, такі як диференціальні оператори і функціональні рівняння, вивчені відповідно Булем і де Морганом. Нові представлення силогізмів та нова алгебра логіки були об'єднані Ч.С. Пірсом (1839-1914) і ретельно досліджені Ернстом Шредером (1841-1902). Інший та інноваційний варіант символічної логіки був запропонований Готтлобом Фреге (1848-1932) у 1879 році, а пізніше і Джузеппе Пеано (1858-1932). Пеано сприяв символізації понять логіки, а також понять у математичних теоріях, таких як аналіз та геометрії. Деякі з цих символізованих понять включають, серед іншого, логічні сполучники, логіку предикатів та квантори. Слід зауважити, що Буль вже надав символи для логічних сполучників, а Фреге запропонував цілу теорію числення предикатів. Пеано досягнув більших успіхів у популяризації цієї нової символічної теорію, яку він, вслід за Де Морганом, називав математичною логікою [12]. Фактично, нова математична логіка Пеано містить логіку відношень, що надає дедуктивні методи і інші математичні об'єкти і структури. Цей термін успішно прижився у Великобританії і був ефективно використаний Бертраном Расселом (1872-1970) та А. Н. Вайтгедом (1861-1947) у їх монументальній праці *Principia Mathematica* для обстоювання позиції *логіцизму* (термін, введений Рудольфом Карнапом (1891-1970)), що означає зведення математики (або її частини) до логіки [11].

Важливо згадати в цьому контексті величезну роль Георга Кантора (1845-1918), який розробив теорію множин в 70-х роках XIX ст. Використовуючи цю теорію, яка має справу лише з множинами, можливо представити всі математичні поняття та докази формально, і, отже, як фундаментальну основу математики, в якій математичні об'єкти представлені у вигляді множин. Фактично в 1880-х Кантор передбачив можливість представлення цілих чисел з множин. Більше того, Кантор стверджував, що математичні теореми можна дедукувати використовуючи логіку предикатів та основні аксіоми теорії множин. Взявши цю теорію за основу, Рассел модифікував її, в основному розглядаючи множини еквіполентних множин, визначених предикатами [12]. Наступний підрозділ присвячений Булевій логіці або алгебрі логіки, метою якої було створення логіки, натхненної правилами традиційної алгебри.

1.4. Булева алгебра логіки та логічний аналіз

Головною метою Буля в логіці було віднайти таке логічне числення, яке виражатиме логіку у математичний спосіб. Таким чином, він розробив некількісний процес логічного маніпулювання, який розглядає алгебру логіки як абстрактний механізм з багатьма інтерпретаціями. В результаті, логіка може постати формальною системою, що перегрупує свої символи та позначення з метою здійснення логічного аналізу з використанням законів логіки. Інтерпретації операцій здійснюються a posteriori і без будь-якого

логічного опрацювання. Висловлюючись точніше, Буль наполягав на постійності еквівалентних форм, задля утвердження концепту незалежності від інтерпретації в алгебрі, ґрунтованій на символах. Застосовуючи це символічне числення в логіці, він уникав використання величин, типове для математики. Іншими словами, ця алгебра базується на використанні символів, законів та комбінацій, що мають загальне, послідовне та однозначне тлумачення [14].

Буль вважає, що логічні положення слід виражати алгебраїчними рівняннями із використанням символічної алгебраїчної мови, визначаючи тим самим логіку як математичну дисципліну. Ця програма ставить механічний аспект логічних дедукцій в основу алгебри логіки. Він висловив цю ідею наступним чином:

> Саме на основі цього загального принципу я маю на меті розробити логічне числення, що посяде своє місце серед визнаних форм математичного аналізу, незважаючи на те, що зараз воно знаходиться осторонь через характер своїх об’єктів та інструментів (Буль) [15].

Пов’язаність логіки та математики підкреслюється великою подібністю між символічною мовою алгебри та необхідними правилами логіки. Ця ідея допомагає отримати уявлення про логічні структури перш ніж застосовувати до них потрібні обчислення. Цей процес

продемонстрував, що математику можна використовувати і без квантифікації її структур. Іншими словами, це означає, що математика може мати справу з концептуальної обробкою, такою як логіка, без обов'язкового врахування традиційних кількісних аспектів [11]. Більше того, Буль намагався у своїх розробках провести поділ між об'єктами, що належать до різних класів, використовуючи символічну модель та застосовуючи закони таким самим способом, як це робиться в алгебрі. В підсумку, логічні висновки або докази можуть бути отримані алгебраїчно.

Буль визначив два центральні класи своєї алгебраїчної логіки. Перший представляє універсум, який містить усе (позначений як 1), а другий порожній клас (позначений як 0), який нічого не містить. На ці представлення з використанням двійкових значень, ймовірно, вплинули позначення ймовірності. Вони також враховують ще один внесок у розвиток символічної мови логіки.

Використання ідемпотентного закону, також є нововведенням у логіці Буля. Тут враховується факт, що перетин класу з самим собою, повторений багато разів, дає завжди один і той же клас. Ця ідея має особливе значення в логічних розрахунках.

Багато десятиліть потому оригінальна Булева логіка буде покладена в основу більш обмеженої алгебри, названої Булевою алгеброю (або, що одне і те ж саме, Булевою логікою). Основним

обмеженням цієї логіки є той факт, що будь-яка логічна змінна може мати виключно значення 0 або 1.

Підсумовуючи, Буль створив алгебру логіки, яка використовує найбільш подібний набір аксіом та засновків, прийнятих у традиційній алгебрі, що дозволяє виконувати обчислення в алгебраїчний спосіб. Однак найважливіша різниця між цими двома алгебрами полягає в інтерпретаціях, які можна отримати з системи незалежно від синтаксичної форми мови.

Наступний підрозділ присвячений одній з найбільш фундаментальних логік, логіці Фреге, або, так званій, логіці предикатів.

1.5. Логіка предикатів Фреге і основи математики

У своїй книзі "Числення понять..." (Begriffsschrift) Готтлоб Фреге (1948-1925) представив власну логіку предикатів, особливістю якої було використання квантифікації; це цінний інструмент в аналізі, щоб виразити, наприклад, концепти неперервності, рівномірну неперервність, конвергенцію тощо, особливо коли в цих визначеннях потрібні багаточисленні квантифікації. Він замінив традиційне неоднозначне представлення висловлювань суб'єктами і предикатами на більш загальне представлення за допомогою функцій і їх параметрів. До цього додається використання зв'язаних змінних і кванторів. У цьому контексті Фреге вважав, що необхідно уникати домінування слів над розумовими можливостями, щоб звільнити логіку від будь-якої

плутанини в інтерпретації і залежності від неоднозначності людської мови. Крім того, Фреге надав повну аксіоматичну систему, засновану на квантифікації загальності, запереченні та умовному дозволі заміни правил. Однією з найважливіших нових концепцій є введення ним операторної змінної в області дії квантора. Це дозволяє визначити відношення між логічними попередниками і наступниками, відношення один-до-одного та відношення логічної спадковості. Ці основні визначення уможливили формулювання математичної індукції з використанням виключно логічних позначень [10,11,14].

Ґрунтуючись на моделюванні логічної абстракції, Фреге включив визначення множин в свою аксіоматичну систему, зробивши теорію множин невід'ємною частиною логіки, відповідно зі своєю основною програмою логіцизму.

Кілька разів в подальшому, Рассел намагався *виправити* антиномію, що постала наслідком системи Фреге, запропонувавши теорію типів; водночас Фреге запропонував модифікацію свого проблемного п'ятого закону; всі ці пропозиції, як виявилося, породили більш складні антиномії, і, як наслідок, провал програми логіцизму.

Г. Фреге, якого часто вважають одним з найбільш значущих логіків в історії, запропонував інтерпретацію чисел, використовуючи свою систему логіки; згідно з цією ідеєю, арифметика (і вся математика) вважається складовою логіки. Важливим у цьому ключі, на думку Фреге,

є те, що доведення теорем повинно виконуватися простими кроками в рамках дедуктивного (логічного) механізму.

1.6. "Принципи..." Рассела та Вайтгеда та наслідки парадоксу

Внесок Рассела в логіку розпочався з розширення теорії числення Пеано, що пов'язало її з теорією відношень, розробленою Пірсом і Шредером, а також з використання здобутків Кантора в цьому питанні. Це було здійснено згідно із проектом Пеано з формалізації математики. Він намагався визначити відношення примітивно інтенсіональним чином і надати в цій теорії набір аксіом, включаючи властивості основних операцій (доповнення, змикання, продукт, об'єднання та перетин). Отже, функції були визначені як випадки відношень.

Вивчаючи роботу Фреге над проектом логіцизму, Рассел виявив його знаменитий парадокс і описав його у листі до Фреге 1902 року наступним чином:

> Нехай w є предикатом: бути предикатом, який не може бути предикатом самого себе. Чи може w бути присудженим самому собі? З кожної відповіді випливає її протилежність. Тому ми повинні зробити висновок, що w не є предикатом. Так само не існує класу (як сукупності) тих класів, котрі, взяті як сукупність, не належать собі. З цього я роблю

висновок, що за певних обставин визначена сукупність не формує тотальності (Рассел) [16, p. 125].

Цей парадокс мав драматичні наслідки для програми Фреге. Однак Рассел намагався вирішити цю проблему, запропонувавши теорію типів або "розгалужену теорію типів" [14].

Основна ідея цієї теорії полягає в тому, що кожна пропозиційна функція належить до області значущості, це означає, що кожен параметр функції повинен знаходитися в певному діапазоні значень, що дозволяє функції мати чітко істинне значення. Крім того, ці діапазони (до яких належать параметри) складають однорідні множини або класи, що називаються типами. Цим пропонується очевидне вирішення парадоксу [14]. Іншими словами, той факт, що елементи множини є однотипними, є головною умовою, за якої можливо розглядати елементи цієї множини. За цієї умови забезпечується більш глибоке розмежування між *частинами* які є однорідними, і *повнотою* що має точний діапазон значень; це відповідає різниці між "БУДЬ-ЯКИЙ" І "ВСІ". Подальші розробки в рамках цієї теорії вилилися у появу розгалуженої теорії типів, яку автори вважали визначеним вирішенням парадоксів [14].

Як вже згадувалося раніше, Рассел і Вайтгед намагалися довести, що математику можливо виразити за допомогою формальної логіки. Таким чином, вони вважали, що розділи математики можливо побудувати на основі логічних аксіом, що, в основному, походять від

правил висновування. Основними частинами цієї роботи є пропозиційне числення та експлікації дедукції, заснована на квантифікації теорія типів а також, теорії відношень та класів. Ця праця об'єднала в собі головні дослідження щодо аксіоматизації пропозиційної логіки.

Праця Рассела є різнобічно важливою навіть при тому, що в ній не вдалося надати повну аксіоматизацію математики. Спроба в логічний спосіб, хоча і не повністю, побудувати основні структури математики за допомогою декількох базових понять сама по собі є важливим внеском в логіку. Детальне вивчення (аксіоматизованої) пропозиційної логіки також мало значний вплив, особливо враховуючи подальші досягнення щодо її поширення на числення з використанням квантифікації та комбінаторної логіки. Можливо, поява логіки вищого порядку з творів Рассела-Вайтгеда є ще одним вагомим внеском; особливо через розповсюджене серед ключових логіків переконання, що цей тип логіки є релевантним у проекті аксіоматизації математики [14].

1.7. Питання Гільберта та теорія доведення

Давид Гільберт (1862-1943), німецький математик і логік, був дуже зацікавлений у вивченні аксіоматичних методів, особливо в умовах взаємозв'язку між логікою та основами математики. На основі своїх досліджень засад геометрії він спробував розширити це дослідження на природу аксіом, намагаючись досягнути повноти. На знаменитій конференції в Парижі 1900 року, він висловив глибоку переконаність

щодо можливості послідовного доведення арифметики та щодо існування вирішення для кожної математичної проблеми.

Більш точно, узгодженість аксіоматичної системи залежить від того, чи можливо знайти докази для результату і для його заперечення. Іншими словами, це вивчення можливостей довести (або спростувати) істинність чогось, якщо спочатку воно явно вважалося хибним.

Одне із найвідоміших запропонованих питань на згаданій конференції було пов'язане з проблемою розв'язності, в якій йдеться про те, чи може математичний вираз бути виведеним з його формальної аксіоматичної системи за допомогою скінченого числа операцій. Ця ідея може бути поширена, як вважав Гільберт, на питання щодо вирішення будь-якої математичної задачі, і тому він вважав проблему розв'язності пріоритетною в математичній логіці.

Більше того, пізніше Гілберт вважав, що проблема розв'язності може бути вирішена, якщо можливо віднайти процедуру для вирішення, чи є логічний вираз даного (першого порядку) правильним чи ні [14].

У своїй програмі узгодження Гільберт намагався відкинути інтуїціоністський погляд на засади математики, замінивши його формалістською точку зору, яка полягає в формалізації традиційної математики та доведенні її послідовним фінітним чином. Фінітизм Гільберта приблизно полягає в тому, що доведення будь-якого результату щодо кінцевих математичних об'єктів за допомогою

нескінченних (або ідеальних) об'єктів можливо і без використання цих останніх. Зокрема, будь-який доказ узгодженості теорії множин щодо скінченних об’єктів також може бути здійснений навіть за використання нескінченних ідеальних математичних об’єктів [14].

Якщо коротко, Гільберт висловив основні завдання, що стосуються основ математики, включаючи можливість фінітного доказу узгодженості. З іншого боку, його переконаність у тому, що математичні аксіоми є повними, а питання про достовірність будь-якого твердження першого порядку є вирішуваним, привела його до припущення, що всі сформульовані математичні проблеми мають вирішення.

Через декілька років з'ясувалося, що реалізація цих двох задач не можлива а сподівання Гільберта виявилися ілюзорними. Теорія квантифікації першого порядку виявилася невирішуваною, і згідно із Геделем, не існує аксіоматичної теорії, яка могла б бути повною в синтаксичному сенсі; що означає, що проект аксіоматизації теорії чисел нереалізовний.

Незважаючи на провал деяких завдань, що їх ставив Гільберт, його внески все ще мають вагу в математичній логіці; вони включають, серед іншого, визначення та задачі теорії доведення, методи теорії моделей та ініціюючі ідеї теорії рекурсії, включаючи принцип повноти в логіці (Кертіс Франк). Всі ці ідеї йшли поряд із своєрідною стійкістю Гільберта в доведенні несуперечливості аксіоматики з використанням лише

конкретних математичних конструкцій, таких як доведень, заснованих на примітивній рекурсії або індукції які є частинами фінітного підходу.

Деякі пов’язані внески Курта Геделя обговорюються в наступному підрозділі.

1.8. Відповіді Геделя та його теореми про неповноту

Курт Гедель (1906-1978) - один з найвидатніших логіків в історії. Його внески були визначальними для становлення і розвитку логіки та філософії у XX столітті. Найбільш продуктивним періодом для логіки Геделя були 30-ті роки XX го століття. У 1930 році, через рік після публікації теореми про повноту, він опублікував свої дуже відомі теореми про неповноту [17], які в цьому підрозділі будуть розглянуті більш детально.

Протягом того ж десятиліття йому вдалося довести несуперечливість "аксіоми вибору" та несуперечливість Канторової гіпотези про континуум, використовуючи ZF-аксіоми для теорії множин [18]. У той же час він опублікував фундаментальні праці з інтуїтивної логіки та теорії чисел, зокрема щодо деяких властивостей арифметики першого порядку, проблеми розв'язності для логіки предикатів та властивостей щодо доказів.

На початку 1940-х та 1950-х років, Гедель більше зацікавився філософією. Його праці дуже різнопланові і присвячені, серед іншого,

"взаємозв'язку між конструктивізмом та інтуїтивною логікою, обчислюваній функції, діалектичним інтерпретаціям, пов'язаним з арифметикою Гейтінга, і ґрунтовним дослідженням праць Ляйбніца" [18]. Цей інтерес продовжився вивченням філософських наслідків логічних теорем, засад та природи математики [18].

У 1929 році Гедель також запропонував вирішення проблеми повноти, яка Гільбертом та Аккерманом у 1928 році була сформульована у вигляді запитання: чи є точно задана система аксіом для логіки предикатів першого порядку повною? Це означає, що можна вивести кожне дійсне всередині системи.

Часто вважається, в тому числі й самим Геделем, що його теорема повноти є прямим наслідком результату, отриманого Скулемом та Ловенгаймом у 1923 році; основною причиною того, що Скулем не отримав в решті повноти, зазвичай вказують "ставлення до семантики та інфінітарних підходів" [18] як пояснив Гедель. Очевидно, сам Гедель виявив, що причиною цього відкриття була "відсутність епістемологічного підґрунтя щодо метаматематики та інфінітарних міркувань" [18].

Теореми неповноти Геделя представляють кульмінацію його відомих внесків у математичну логіку і значним чином вплинули на багато областей, серед яких, філософія та комп’ютерні науки.

Перша теорема стосується ідеї, що *“В рамках формальної арифметичної системи аксіом існують твердження, які не можливо довести (вивести) або спростувати* [14].

В більш точному формулюванні, система, описана в *Принципах математики* містить невивідні положення, а їх заперечення є також невивідними. Це означає, що такі положення, які репрезентують проблеми, є такими, що не мають розв'язання.

Гедель намагався звести несуперечливість аналізу до несуперечливості арифметики, використовуючи натуральну числову нумерацію, щоб представити деякі об'єкти аналізу, такі як доведення. Його підхід до кодування цих об'єктів, виконаний виключно з використанням концептів теорії чисел і за допомогою китайської теореми про остачі, привів його до парадоксальної ситуації, подібної до парадоксу брехуна [14, p. 401].

Однак його друга теорема про неповноту показує, що не можливо довести несуперечливість будь-якої теорії чисел [14]. Таким чином, для елементарної теорії чисел, не існує несуперечливості, що означає, що висловлювання, яке описує несуперечливість арифметики, є невирішуваним.

Фон Нейман щодо цього зауважив, що суперечливість (або нефінітна несуперечливість) доказу може бути застосована до будь-якої подібної математичної моделі зі описаних в "Принципах математики",

особливо при використанні методів або прийомів доведення в межах тієї самої моделі [14, p. 401].

Ці результати вказують на нездійсненність програми Гільберта (або принаймні її частини). Фактично фон Нейман негативно висловився щодо засад математики, сказавши Геделю, що "немає строгого обґрунтування класичної математики" [18].

У цьому демарші було показано, що поняття доказовості і істинності кардинально відрізняються; зокрема, при спробі Геделя звести несуперечливість аналізу до несуперечності арифметики, а доказовість - до істинності, були отримані парадокси.

Одним із багатьох філософських наслідків другої теореми є те, що класична арифметика включається в інтуїціоністську арифметику, коли беруться до уваги окремі інтерпретації. Отже, стосовно праць Гільберта, фінітна арифметика значно слабша за інтуїціоністську арифметику [14, p. 403]. Ці результати, безумовно, спонукали багатьох логіків (наприклад, Генцена) до пошуку кращого розуміння фінітних міркувань та природи несуперечливих доведень; в результаті, були розроблені нові зв'язані поняття про несуперечливість, такі як "відносно несуперечливе доведення".

Незважаючи на те, що в цій роботі перерахувати всі досягнення Геделя просто неможливо, інші його висновки, пов'язані з інтуїтивною логікою, наступні: 1) "багатозначної логіки з скінченим числом різних

істиннісних значень недостатньо для визначення інтуїтивної пропозиційної логіки" [14, p. 415] і 2) "між пропозиційною та інтуїтивною логіками можна отримати нескінченну ієрархію логік з фінітним числом значень" [14, p. 415].

1.9. Логіка та комп’ютери: зустріч у двадцятому столітті

Двадцяте століття стало кульмінацією цього захоплюючого історичного розвитку логіки, орієнтованої на обчислення, коли більшість із цих теорій було покладено в основи комп’ютерних наук. Майже кожна теорія щодо формальної чи математичної логіки, була прямо чи опосередковано задіяна у побудові принципів чи безпосередньо застосована у комп’ютерних науках. Ці реалізації, однак, не пов'язані виключно з формальною логікою і комп'ютерною обробкою; сюди входить велика кількість сучасних застосувань, що пов'язують неформальну логіку і теорію аргументації з комп'ютерами в рамках моделей штучного інтелекту. [19–22].

Як вже було сказано, Булева алгебраїчна логіка, логіка предикатів Фреге і теорія типів Рассела - разом постали основними складовими комп’ютерних наук. Доробок Тюрінга, Кліні та Черча в галузі теорії обчислень (і не тільки) безпосередньо ґрунтується на математичній логіці. Логіка предикатів, розвинута Фреге, має ключовий вплив на створення мов програмування в цілому, і мов логічного програмування зокрема. Перші використовуються для розробки комп’ютерного

програмного забезпечення майже в усіх сферах знань, другі є вихідним пунктом програм штучного інтелекту. У наш час штучний інтелект представляє суть останніх досягнень практичної комп'ютерних наук. Він виражає собою прогрес технологій, потужність машин, щоденне розширення меж людських можливостей, але також виражає страх перед машиною, сумніви щодо неоднозначних відносин між людьми та машинами та місця, яке для кожної із сторін буде прийнятним або бажаним. Завдяки природі та сфері досліджень штучного інтелекту дуже швидко виникає дотична філософія. Філософія штучного інтелекту продовжує займатися багатьма сучасними, складними питаннями. Етичні проблеми штучного інтелекту стимулюють зростаюче зацікавлення та сучасні дискусії. Статус машини як морального суб'єкта, влада, якою люди мають її наділяти та рівень довіри, який люди можуть собі дозволити щодо машин, це - тільки частина цікавих і захоплюючих тем. Філософію штучного інтелекту неможливо повністю зрозуміти без взяття до уваги філософії комп'ютерних наук в цілому, яка розглядає суть таких понять, як обчислення, інформація, алгоритм, обробка та питання, про те, що ми здатні обчислити і що нам слід обчисляти. В цьому контексті постає, наприклад, проблема прозорості обчислень, яка полягає в тому, що для розуміння рішення, що приймається комп'ютером потрібно відстежити програми, що призвели до такого рішення. Ці проблеми показують доцільність розгляду філософії комп'ютерних наук в цілому коли йдеться про філософію штучного інтелекту.

Були порушені також і інші питання філософії штучного інтелекту, зокрема щодо потенційної відповідності між пізнанням та обчисленням. Одне з них, питання комп'ютерної теорії розуму, яку полягає в тому, чи можна реалізувати чи пояснити пізнання засобами обчислень. Інше подібне питання стосується гіперкомп’ютаціоналізма. Представники даної позиції стверджують, що пізнання може бути еквівалентним деяким обчислювальним моделям, сильнішим, ніж моделі машин Тюрінга, у тому сенсі, що вони здатні обчислювати функції, які не піддаються обчислюванню Тюрінга. Ці моделі включають, але не обмежуються наступними елементами: класична нечітка машина Тюрінга, аналогова рекурентна нейронна мережа, машина Тюрінга, що працює в нескінченому часі, релятивістські та квантові гіперобчислення.

1.10. Висновок

У цьому розділі було представлено вибірку ідей сучасної логіки, що з'явилися у період від Ляйбніца до XX століття. В основному, ці ідеї пов’язані з обчислювальними процесами, які в значній мірі і конкретно застосовувались у XX му столітті в розробках комп’ютерних наук.

У цьому розділі розглянуто логіку Ляйбніца, розроблену у XVII столітті, в якій представлено обчислювальний логічний проект логіки, на основі якого мала бути створена універсальна машина, що здійснює *числення міркувань*. Ми дослідили праці Ляйбніца, головною ідеєю яких

була розробка мови символів, "універсальної характеристики", в якій представлені концепти та ідеї, здатні полегшити процес міркування або, радше, логічний механізм обчислення. Нами було вказано на те, що Ляйбніц розглядав арифметичні та алгебраїчні мови як одні з найбільш підходящих інструментів для розробки цієї універсальної характеристики, можливості логічних доведень якої аналогічні можливостям арифметичних або алгебраїчних систем. Таким чином було узагальнено, яким чином алгебраїчна система позначень може надихнути до створення обчислювальної логічної системи.

Окремо, повнозначна система символів Ляйбніца (або система позначень) може бути використана для визначення алгебраїчної моделі логіки, яку він назвав *універсальна система логічного числення (calculus ratiocinator)*, що може бути застосована у будь-якій галузі, що має справу з процесами міркування. У цьому розділі було пояснено, що Ляйбніц вважав приклади людських дискусій чи аргументації такими, що, як правило, формують неоднозначний дискурс, з причини внутрішньої слабкості та неточності природних мов. В підсумку, ми описали яким чином універсальна система логічного числення (calculus ratiocinator) змогла надати однозначний і точний інструмент для здійснення логічних міркувань, уникаючи помилок або небажаних інтуїтивних дедукцій. З використанням такої системи, логічні висновки приймаються так само, як приймаються будь-які арифметичні чи

алгебраїчні обчислення, а філософські суперечки можуть бути формально вирішені за допомогою логічного обчислення.

У цьому розділі також було розглянуто трансцендентальну логіку Канта та діалектичну логіку Гегеля. Перша, яка посідає чільне місце серед філософських доробок, розглядається як дослідження людського розуміння, включаючи доступну силогістичну та інференційну частину людської діяльності, що стосується сприйняття. Однак, щодо діалектичної логіки Гегеля, зазначено, що вона представляє собою скоріше метафізичний доробок ніж класично-логічний. Вона представлена як діалектичний механізм для розуміння реальності та самої чистої думки (самого розуму) з використанням метафізичних принципів.

Цей розділ також представляє огляд окремих логічних здобутків часів *математичного повороту* в XIX, які в основному були реалізовані у Великобританії та Німеччині. Мова йде, перш за все, про алгебраїчну теорію логіки, в основному розроблену Булем, а також про праці Венна, Гамільтона та інших. Було зазначено, що Венн займався дослідженнями символічної логіки, в той час як Де Морган зосереджувався на логіці відношень. Буль в цей час був зосереджений на алгебрі логіки, вдосконаленій пізніше Пірсом і Шредером. Було також сказано, що більшість з цих подій відбулися напередодні надзвичайно важливого досягнення Фреге, який запропонував логіку предикатів згідну з символічною логікою. Цю логіку продовжував вдосконалювати та

популяризувати Пеано, який використовував термін "Логіка відношень" [14] а також Де Морган, використовуючи термін *математична логіка*. Після цього було прослідковано зусилля Рассела та Вайтгеда по відстоюванню програми логіцизму, що здійснювалося на основі теорії типів, способом, що був протилежним до моделі Фреге та вважався позбавленим від суперечностей. Ці значні здобутки продовжилися працями Кантора, який сформулював теорію множин, пропонуючи представляти математичні об'єкти, зокрема цілі числа, за допомогою множин; крім того у цьому розділі підкреслюється, згідно із припущенням Кантора, що з використанням цієї теорії та логіки першого порядку можна здійснити дедукцію математичних теорем.

Окремий підрозділ у цій частині був присвячений роботі Буля над алгеброю логіки, що показує, як Буль хотів перетворити логічні дедукції на механічний процес, використовуючи свою алгебраїчну логіку, подібно до будь-яких обчислень, які можна виконати за допомогою стандартної алгебри. Його логіка була виокремлена в якості провідного компоненту значних розширень, які були здійснені Булевою алгеброю у XX ^{му} столітті.

Ще одним фундаментальним проривом, дослідженим в цьому розділі є логіка предикатів Фреге. Ми дослідили її в контексті обґрунтування позицій філософського логіцизму, які, ймовірно, в подальшому суттєво вплинули майже на всі аспекти комп’ютерних наук.

Більше того, ми висвітлили логіку Рассела та Вайтгеда, описану в праці "Принципи математики" [23], що в перше вийшла друком у 1910 році, та в подальшому мала фундаментальний вплив на математичну логіку. Після цього було представлено здобутки Гільберта, в основному щодо теорії доведення, та історичний огляд основних внесків Геделя завдяки його теоремам про неповноту, які назавжди змінили історію логіки, а також історію філософії. В останньому підрозділі даного розділу здійснено узагальнення деяких історичних перетинів сучасної логіки та комп'ютерних наук.

Розділ 2. Філософія комп’ютерних наук крізь призму логіки

Філософія комп'ютингу в основному торкається питань про природу комп’ютерних наук, природу обчислень і того, що взагалі можливо вирахувати, природу інформації та її обробки. Вона також вивчає обчислювальні моделі, межі та можливі наслідки обчислюваних речей.

У цьому розділі ми визначаємо комп’ютерні науки як багатоцільову дисципліну, що ґрунтується на логіці, математиці та інженерії, підкреслюючи її можливості щодо когнітивного опрацювання. Виходячи з такої багатоаспектності, ми піднімаємо питання характеризації комп’ютерних наук в контексті її зв’язків з обробкою інформації та абстракціями, а також в контексті місця комп'ютингу в межах філософії комп’ютерних наук.

Крім того, ми приймаємо і відстоюємо позицію часткового комп’ютаціоналізма, коли йдеться про те, що в основі процесу пізнання лежать логічні механізми. Ми також висловлюємо певні сумніви щодо точки зору гіперкомп’ютаціоналізма в частині розуміння складових пізнавального процесу. Більше того, ми наводимо аргументи на користь того, щоб розглядати машини Тюрінга як моделі обчислення за замовчуванням у контексті комп'ютерної теорії розуму; ми також визначаємо роль інформації в осягненні інтелектуальних процесів.

Нарешті, ми пропонуємо власний погляд на те, в чому роботи можуть замінити людей, а в чому ні, а також щодо того, чому роботи не повинні мати морального статусу. Наприкінці, ми розглядаємо проблему прозорості обчислень та деякі наслідки цієї проблеми.

1.11. Чому комп'ютерні науки має багато аспектів?

Одним з найбільш фундаментальних питань філософії комп'ютерних наук є питання стосовно її природи. Результатом численних спроб відповісти на це питання стала поява різних точок зору, що пропонують розуміти природу комп'ютерних наук багатоаспектно. Чи є комп'ютерні науки мистецтвом, тому що "програми можуть бути прекрасними" [24, p. 670]?, чи є вона (справжньою) наукою, оскільки "Наука - це знання, які ми розуміємо настільки, що можемо навчити йому комп'ютер" [24, p. 668]? ЇЇ називають вільним мистецтвом, природничою наукою, штучною наукою, частиною математики, емпіричною (або / і теоретичною) наукою про штучність, інженерною дисципліною, поєднанням інженерії та науки. То чи є комп'ютерні науки поєднанням всього перерахованого, якоїсь частини, чи чимось зовсім іншим? Чи комп'ютерні науки «насправді важко розглядати як самостійну дисципліну» [25], "Чи вона настільки відрізняється, що ми повинні розглядати її як цілковито нову науку" [26, p. 1], або "може взагалі не можна вести мову про якусь її природу" [27, p. 109]?

Наприклад, розглянемо проблему пошуку алгоритму сортування (Quicksort, Insertsort, Heapsort тощо) незалежно від алгоритмічної складності. Яку природу матиме таке вирішення? Які пізнавальні активності залучені до пошуку такого вирішення? Чи це математика, чи логіка, інтуїція, чи інтелектуальне прозріння, чисте алгоритмічне мислення, чи щось інше?

Зазвичай для віднесення комп'ютерних наук до галузі науки використовують поняття обчислення, для того щоб констатувати її приналежність до сфери інженерії наводять поняття компонентів обладнання, а поняття формальних методів пропонується для того, щоб поставити комп'ютерні науки у сферу науки, поняття апаратних компонентів у сферу техніки, а поняття формальних методів у сферу логіки.

Як вважав Рапапорт [28], комп'ютерні науки "здається, має гібридну природу" [28]. Зокрема, «сучасні комп'ютерні науки, будучі цілісною дисципліною, має у своїй ДНК як інженерію так і науку» [28]. Нам здається, що теорії алгоритміки та математики формують одне їз її *крил*, а інженерія виступає в якості другого *крила*. Перше охоплює відкриття в арифметиці та логіці з найдавніших часів і до сучасних уявлень про формальні системи в обчисленні. Друге охоплює різноманітні машини, побудовані реально (машини Паскаля та Беббіджа) або ж абстрактно (універсальна характеристика Ляйбніца та його універсальна система логічного числення (ratiocinator)), що пізніше призвело до появи найсучасніших цифрових комп'ютерів. Розуміння

комп’ютерних наук як єдності цих двох галузей (як, звичайно, і інших компонентів) є важливим для визначення природи цієї дисципліни [28].

Для того, щоб дослідити і особливо зрозуміти науковий характер комп’ютерних наук, доречно розглянути уявлення про наукові методології та принципи комп’ютерних наук. Ми також розглядаємо наскільки виправданими є окремі запропоновані поділи та розмежування між складовими цієї дисципліни. На думку автора, причиною такого відмежування від інших дисциплін (математики, логіки, інженерії тощо) може бути *особливе використання академічних практик*. Мається на увазі практика класифікації, перегрупування (як у випадку дисциплін групи STEM[1]), що розділяє, відокремлює і навіть протиставляє ці дисципліни одна одній, як не пов'язані та незалежні одна від одної.

Отже, наше розуміння цієї дисципліни іноді впирається в відомчі перешкоди з повною відсутністю (або небажанням) поглянути на ці дисципліни, і, отже, на природу комп’ютерних наук, як на ціле, в якому всі частини мають своє значення і отримують свій справжній характер в результаті їх безперервної взаємодії.

У цьому контексті варто зазначити, що межі цих, часто штучних, поділів між дисциплінами та субдисциплінами містять найбільш важливі елементи розуміння цих областей, оскільки вони відображають

[1]Наука, Технологія, Інжиніринг, Математика

найглибші і реальні впливи між ними; ці впливи можуть дати відповіді на багато питань, включаючи ті, які пов'язані з їх справжньою природою.

Можливо, комп’ютерні науки слід розглядати як сферу, де на одному полюсі розташовані її теоретичні та математичні основи, а на іншому полюсі - фізичні та інженерні реалізації. Між цими двома полюсами всі інші комп’ютерні досягнення можуть бути розгалужені та тісно пов’язані у вигляді динамічної мережі взаємодій.

Насправді багато тем з математики, логіки, електротехнічної та електронної інженерії тощо стали частинами комп’ютерних наук та перетинаються (як у випадку штучного інтелекту) з багатьма іншими дисциплінами, такими як філософія, когнітивна наука, комунікація, нейронаука, лінгвістика тощо. Більше того, комп’ютерні рішення застосовуються майже у будь-якій царині знань.

Підводячи підсумок щодо нашого бачення, комп’ютерні науки можливо розуміти як науку про інтелектуальні процеси. Це наука про пізнавальну діяльність або просто частина когнітивної науки. Це *когнітивна обчислювальна наука* (КОН) або, інакше, когнітивні феномени для комп'ютингу. комп’ютерні науки - це набір концептів і практик, які стосуються процесів людського розуму і артефактів, які використовують міркування, абстракції, інтуїцію, формальні і неформальні логічні методи, математичні структури, фонові знання, інженерні експерименти і реалізації. Вона так само має справу з іншими

численними поняттями, механізмами і явищами як людського розуму, так і реальності, які майже неможливо осягнути або передбачити.

1.12. Яким чином можливо характеризувати комп'ютерні науки?

Дослідження характеризації комп'ютерних наук є важливим моментом прояснення сутності цієї дисципліни. Чи є комп'ютерні науки «унікальною, оскільки це дисципліна наук про інформацію» [24]? Чи використання «абстракцій, модулярності, ієрархії та вирішення проблем наділяє її основними характеристиками» [29, p. 37]? Чи вона принципово "характеризується алгоритмічним мисленням" [24]?

Чи можна вважати комп'ютерні науки просто наукою про інформацію, та чи справді існує суттєва різниця між обчисленнями та обробкою інформації, або, за словами Піччініні [30], чи справедлива думка, що "семантична інформація вимагає представлення, а обчислювальна - ні" [30]?

Що є об'єктом досліджень комп'ютерних наук? комп'ютерні науки іноді розглядають як вчення про реалізацію алгоритмів з використанням структур даних, а вчених інформатиків вважають "алгоритмічними мислителями" [31, p. 172]. Інші вважають комп'ютерні науки "вченням про інформацію", а не про алгоритми [32, p. 3] оскільки вона обробляє інформацію логічним та абстрактним способом. ЇЇ також вважають "корпусом знань, що займається проектуванням, аналізом, впровадженням, ефективністю та

застосуванням процесів, що перетворюють інформацію" [33, p. 16]. Думку, про те, що комп'ютерні науки поєднує науковий підхід із інженерними явищами також висловлюють наступним чином: «ті, хто вивчають *способи представлення та обробки* є вченими та ті, хто вивчають *машини та системи* є інженерами" [34, p. 164].

Чи є комп'ютерні науки лише наукою про алгоритми (як уже згадувалося раніше), чи вона має справу з більш загальними структурами, такими як процедури? Автор у [35] вважає " комп'ютерні науки - це природнича наука, яка вивчає процедури" [35]. Процедури тут беруться не як природні об'єкти, але як "природні явища, які можна виміряти" [35]. Більше того, алгоритми можна вважати особливими випадками процедур, які "не завжди є точними, не завжди завершуються і не завжди забезпечують правильні (або найкращі) рішення" [35]. Більше того, часто пропонують додати до цих процедур, які обробляють інформацію, апаратні компоненти реального життя.

Негативний висновок щодо того, чи є комп'ютерні науки (природничою) наукою чи ні, було представлено в результаті дослідження цього питання в [28] наступним чином: "Об'єктом вивчення комп'ютерних наук не є *чорні ящики* але *скляні ящики* оскільки не природа, а *ми* спроектували та збудували їх. Отже, ми можемо осягнути їх так, як ніколи не змогли б осягнути *природних* речей" [28].

Подібним чином, комп’ютерні науки іноді вважають синтетичною цариною інженерії, а не наукою [36, p. 61–62] тому що її асоціюють із створенням речей із інженерною діяльністю, а не з науковими дослідженнями мислення.
Ци можна сказати, що вона реалізує інжиніринг абстракцій? Або ж вона досліджує абстракції? В такому випадку, чи стає комп’ютерні науки наукою тільки тому, що вона займається вивченням чогось?

Одним з найбільш важливих питань як теорії, так і філософії обчислень є питання: "Що можливо обчислити?" [28].

Це питання передбачає розуміння обчислювальних концептів і, можливо, обчислювальних моделей, які в дійсності можуть визначити, що саме можливо обчислити. Обчислювальна модель може (повинна?) містити уявлення про способи представлення інформації, включати уявлення про те, що ми можемо робити з інформацією та як ми можемо виконувати обробку інформації, а також те, як можна визначити обчислення в такий спосіб, що окремі задачі можуть бути автоматизовано а інші ні. Цікавий варіант відповіді на ці питання був запропонований у [28], у вигляді *символічного* визначення комп’ютерних наук, а також мови і правил, які можуть бути використані для обчислень. Даний варіант відповіді представлено наступним чином:

" комп’ютерні науки - це вчення про те, які проблеми можливо вирішити, які завдання можна виконати і які особливості світу можна зрозуміти... за допомогою обчислень, тобто, використовуючи мову, в якій є лише 2 іменники ("0", "1"), 3 дієслова ("перемістити", "надрукувати", "зупинити"), 3 граматичні правила (послідовність, виділення, повторення; або просто рекурсія), і нічого іншого; це реалізується шляхом надання алгоритмів, які показують, як це можна зробити ефективно, практично, фізично та етично" [28].

Для Робінсона "обчислення є в першу чергу справою логіки, яка реалізується слідуванням формальним міркуванням, незалежно від будь-якої конкретної або апаратної реалізації" [37, p. 12]. Це питання спершу стосується застосування людських логічних правил, а потім вже спроб віднайти реальне втілення для реалізації цих правил.

Філософія комп'ютингу охоплює багато видів діяльності, включаючи ті, що були, запропоновані Соракером у [38]:

1. "Аналіз, інтерпретація та роз'яснення центральних понять в комп’ютерних науках та зв'язків між ними" [38].

2. "Аналіз, уточнення та оцінка цілей та ключових припущень комп’ютерних наук та різних її галузей та зв'язків між ними" [38].
3. “Аналіз, уточнення і оцінка методів та методологій як комп’ютерних наук в цілому так і різних її галузей” [38].
4. “Аналіз наукового статусу комп’ютерних наук та її зв’язку з іншими галузями науки” [38].
5. «Аналіз ролі та значення комп’ютерних наук для суспільства в цілому, а також для конкретних людських цілей та проектів» [38].

Здійснені обговорення та запропоновані відповіді щодо природи комп’ютерних наук можуть стати у нагоді у багатьох аспектах. Один з них є фундаментальним для розуміння *обчислювальних моделей* які визначають межі застосування комп’ютерів і можуть пролити світло на окремі філософські зв’язки між комп’ютерами та людським розумом як наприклад у комп’ютаціоналізмі. Наступний розділ стосується машини Тюрінга, тези Черча-Тюрінга та пов'язаних з ними обчислювальних моделей.

1.13. Обчислення за допомогою машин Тюрінга та інтуїція тези

Теорія обчислюваності бере свій початок в в праці Алана Тюрінга, написаної у 1936 р. в якій він надав обчислювальну основу сучасних комп'ютерів. Він описав математичну абстрактну модель обчислюваності, що була названа "машиною Тюрінга", яка визначає, що можливо обчислити за допомогою комп'ютерів, а що ні. Ця робота

характеризує обчислювальні функції, використовуючи чисто механічні можливості машин Тюрінга, і передбачається, що вони відповідні до того, що можливо механічно обчислити людським розумом.

Модель машин Тюрінга пропонує саме поняття обчислень, використовуючи формальну структуру теорії обчислень, і дає основне уявлення про можливості і межі обчислень, визначаючи вирішувані і невирішувані задачі. Ця модель представляє просту механічну, але абстрактну машину, яку можна використовувати для обчислень, що здійснюються у спосіб дуже подібний до людського.

Машини Тюрінга широко визнані в якості обчислювальної моделі, здатної дати формальну відповідь на питання, чи є функція обчислюваною чи ні. Незважаючи на те, що були запропоновані інші моделі обчислень (наприклад, рекурсивні функції, лямбда-числення і т.д.), всі ці моделі є еквівалентними, що означає, що функція обчислювана в одній моделі тоді і тільки тоді, коли вона обчислювана у всіх інших моделях. Насправді обчислювані функції Тюрінга в точності відповідають рекурсивним функціям, що їх визначив Клін. Фактично, Клін змінив сферу застосування обчислюваних функцій (що їх *всюди* визначали як домени), що Гедель називав *узагальненими рекурсивними функціями*. Алонцо Черч та Еміль Пост запропонували інші еквівалентні варіанти формалізації.

Машина Тюрінга - це не тільки найпростіша модель, але й найприродніша і найподібніша до людського механізму обчислень. Технічно машина Тюрінга - це абстрактний скінчений автомат, який має

нескінченну об'єм зовнішньої пам’яті на стрічці з скінченим алфавітом. Залежно від того, який символ зчитується із стрічки, він замінює цей символ якимось іншим (можливо, тим самим), переходить в інший стан і переміщує головку стрічки на одну позицію вліво або вправо. У будь-який момент машина Тюрінга може зупинитися. Ці дії є скінченними і зазвичай виконуються у заздалегідь визначеному порядку (детермінована машина Тюрінга), подані у формі таблиці переходів, і являють собою комп'ютерну програму, яку можна виконати для реалізації конкретного завдання. Результати, залишені на стрічці, можливо розглядати як результати обчислень, виконаних програмою. У цьому контексті функція, що обчислена машиною Тьюрінга, називається обчислюваною функцією (Тюрінга).

Кажучи простіше, Тюрінг припускав, що обчислювальні можливості цих машин еквівалентні механічним можливостям людини. Неформальний механізм машини Тюрінга може мати відповідний до людського спосіб обчислення. Наприклад, він передбачає, що людина може одноразово здійснити обмежений обсяг необхідних обчислень, перебуваючи в одному конкретному психічному стані. Зверніть увагу, що значення і області обчислюваних функцій - це символи із скінченого алфавіту, а саме обчислення - це свого роду маніпуляція символами. Характер того, що обчислюється, визначає кількість необхідних психічних станів так само, як і кількість станів машини. Крім того, операція, яку людина може виконати в конкретний момент, знаходиться в залежності від стану людини і від того, яким є результат її попередніх

дій в цей конкретний момент. Під час цього процесу людина може переключити свою увагу на інший крок обчислення або інший стан, може написати новий результат, в тому числі написати новий прямо на старому результаті, що означає замінити його чимось іншим.

Основним поняттям, що пов'язує машини Тюрінга їз поняттям обчислень, є теза Тюрінга-Черча. В цій тезі, що являє собою аналітичний інструмент розв’язування задач, стверджується, що будь-яку функцію можливо обчислити за допомогою ефективної процедури (механічний метод) тоді і лише тоді, якщо її обчислює машина Тюрінга. В понятті ефективної процедури можна віднайти і філософський зміст, якщо розглянути її еквівалентність до того, що взмозі обчислити людський розум. Іншими словами, вона визначає, що можливо обчислити алгоритмом, так само як і обчислити машиною Тюрінга (сучасними комп'ютерами) і навпаки.

Ця теза широко обговорювалося у філософії обчислень та мала великий вплив на дискусії щодо комп’ютаціоналізма. Незважаючи на точне значення функції, обчислюваної машини Тюрінга, поняття *ефективної процедури* лишається інтуїтивним, особливо коли йдеться про процедуру, механічно обчислювану людиною.

Насправді ані Тюрінг, ані Черч не надали формального доведення своєї тези, що пояснюється, з одного боку, "інтуїтивною природою речей"[14], що можуть бути обчислені людиною, а з іншого - неможливістю довести еквівалентність між точною математичною

моделлю (машиною Тюрінга) і іншою повністю інтуїтивною ідеєю обчислюваності. Проте, як уже згадувалося в даному контексті, користь від цієї тези для філософії обчислень є величезною, а питання навколо цього поняття продовжують інтригувати зацікавлених.

Крім того, дана теза є важливою при визначенні класу обчислюваних функцій а, отже і класу функцій, які неможливо обчислити, що дозволить уникнути марнування часу на пошук неіснуючого рішення.

Однак, як зазначив Ходжес: "значна частина теорії обчислюваності з часів Тюрінга була присвячена виведенню іншої інтуїції" [14]. Як наслідок, завдання пошуку іншої математичної моделі, що характеризує обчислювані функції, і створення формальної моделі, що представляє те, що може бути обчислено фізичним способом, стали основними питаннями теорії обчислюваності.

У цьому ключі можна сформулювати й інші важливі філософські питання: Чи є модель машини Тюрінга найбільш підходящою серед цих запропонованих моделей? Чи існує таке поняття обчислень, яке є більш фундаментальним, ніж машини Тюрінга? Чи існує якась загальна теорія, застосовна до будь-якого виду обчислювальної системи, і яка пояснює складові такої системи? Ці питання пов'язані з головною дискусією у філософії обчислень, яка називається "комп'ютерна теорія розуму" (комп'ютаціоналізм), дослідженню якої присвячена важлива частина цього розділу.

Було запропоновано багато концептів для формальної характеристики таких функцій Тюрінга, як *обчислювані множини* та *обчислювані (або рекурсивно) зліченні множини*. Зокрема, множина $\boldsymbol{S}$ при заданому алфавіті є обчислюваною, якщо існує така машина Тюрінга, яка на кожному введенні за цим алфавітом зупиняється і видає ТАК чи НІ стосовного того, чи належить введення до множини $\boldsymbol{S}$ чи ні. Однак множина $\boldsymbol{S}$ є зліченною, якщо існує алгоритм, який може перелічити всі члени $\boldsymbol{S}$.

Важливою перевагою зазначеного поняття є можливість працювати з цими множинами без потреби застосування вичерпних списків кодів, зазвичай необхідних у випадку машин Тюрінга.

Поняття *зліченної множини* є центральним елементом в теорії обчислюваності і використане в багатьох головних її результатах. Однією з найвідоміших проблем у цій теорії є десята проблема Гільберта яка воліє "віднайти алгоритм, щоб визначити, чи має дане Діофантове рівняння рішення, або показати, що такого алгоритму немає" [39, p. 71]. Матіясевич запропонував теорему, що дає негативну відповідь щодо проблеми Гільберта, використовуючи поняття зліченних множин. Зокрема, ця проблема, описана Ходжесом в [14, p. 490], звучить наступним чином:

"Множина натуральних чисел S є зліченною, що по суті означає, що існує така скінченна множина поліномних рівнянь $E(x, y_1, \dots, y_n)$ з цілими коефіцієнтами і змінними $x, y_1, \dots, y_n$, , при якій натуральне число nзнаходиться в S тоді і тільки тоді, коли множина рівнянь

$E(x, y_1, \ldots, y_n)$ має певне вирішення, приналежне до відносних чисел"[14, p. 490].

Окрім обчислювальних функцій є ще одна відома нерозв'язна проблема - *проблема розв'язності* сформульована Д. Гільбертом. Її суть полягає в пошуку ефективної процедури, що дозволяє вирішити, чи є даний вираз першого порядку універсально вірним, з використанням скінченої кількості інструкцій. Ця проблема, ставить задачу перевірити, чи дійсно та чи тільки є так, що дане твердження може бути виведено з аксіом, які Тюрінг і Черч незалежно один від одного визнали *нерозв'язними*. *Проблема зупинки*, або проблема вибору на основі опису машини Тюрінга та її вхідних даних, чи зупиниться машина або буде працювати вічно, також є нерозв'язною проблемою, доведеною самим Тюрінгом.

Незважаючи на формально визначені відповіді, щодо теорії обчислюваності, інші питання, що мають філософське підґрунтя, стосуються того, чи можна *вирішити* ці проблеми, використовуючи іншу модель обчислень, відмінну від моделі Тюрінга; або, знову ж таки, чи можна знайти альтернативну модель обчислень, яка може розширити межі того, що дійсно може бути обчислено.

Дивно, але *позитивні* (хоча і суперечливі) відповіді були запропоновані шляхом введення таких понять, як гіперобчислення, зазвичай з твердим наміром вийти за межі можливостей Тюрінга. Гіперобчислення вважають обчислювально потужнішими ніж машини Тюрінга, а також

породженими пізнанням. Це означає, що когнітивні системи обчислюють більше, ніж загалом обчислюється машиною Тюрінга. Однак немає доказів того, що проблема, яка не піддається обчисленню за Тюрінгом, може бути обчислена *гіперобчислювальним людським процесом*. Поняття гіперкомп'ютаціоналізма, що ставить питання про еквівалентність гіперобчислень пізнанню є питанням філософії штучного інтелекту, яке ми піднімаємо в цьому розділі.

Інша ідея, запропонована Р. Пенроузом, стверджує, що людський мозок може вирішувати задачі, нерозв'язні для універсальної машини Тюрінга, використовуючи для цього квантові поняття [40, p. 10]. Більше того, Макленнан виокремив теоретичні дослідження деяких обчислень у реальному часі, подібних до природних обчислень. Він припустив, що цей вид обчислень може бути здійснений за допомогою "аналогових комп'ютерів, навіть якщо він не є обчислюваним за Тюрінгом". [40, p. 11]

Інші важливі питання що постають у філософії обчислень, це питання *чому ми обчислюємо і що обчислюємо?* [41, p. 1].

Слово "обчислення", здається, має ширше значення, ніж здатність обчислювати за допомогою наявних технологій; обчислювальне моделювання психічних і природних процесів вплинуло на кілька нових поглядів щодо обчислень, і були запропоновані різні обчислювальні моделі для виконання такого роду обчислень. Наприклад, біологічні структури, такі як клітини і мозок, вивчалися як популярні моделі об'єктів в нейронному штучному інтелекті. Більш того, природні моделі,

такі як явища еволюції (ДНК або навіть квантові обчислення), було запропоновано розуміти як обчислювальні процеси, які керуються концептом "трансформація як обчислення з метою пристосування". [40]

Ми хочемо підняти ще одне питання в цьому контексті: чи можна розуміти ці процеси пристосування як обчислення в сенсі переходу від одного набору станів до іншого шляхом виконання певних операцій?

Важливо підкреслити використання слова "перетворення", особливо тому, що багато гіпотез вважають, майже виключно, що обчислення - це процес "маніпулювання символами, обчислення і перетворення інформації з використанням конкретної обчислювальної моделі"[40]. Очевидно, що такий погляд на обчислення є неповним; насправді, ми цілком можемо розглядати обчислювальні процеси як генератор знань, а не тільки як обробку і перетворення інформації.

Внаслідок цього обговорення, в даному дослідженні ми схильні поставити наступні теоретичні питання:

1- Чи існують формальні еквіваленти між цими наведеними моделями обчислень (МТ, біологічні моделі і т.д.)?

2- Надати докази в разі позитивної відповіді і ієрархію обчислювальних рівнів в разі негативної.

1.14. Комп'ютерна теорія розуму у філософії штучного інтелекту

Штучний інтелект (ШІ) - це область комп’ютерних наук, яка спрямована на створення комп’ютерного програмного забезпечення чи

апаратних систем, здатних виконувати завдання, які зазвичай виконуються людьми. Ці завдання можуть включати людські дії і мисленєві компоненти, такі як міркування, планування, прийняття рішень, розпізнавання образів, вирішення проблем і завдань, навчання, використання природних мов, розуміння концептів тощо.

Крім того, штучний інтелект намагається, з одного боку, вивчати людський інтелект і пов'язані з ним інтелектуальні явища, а з іншого боку, моделювати інтелектуальні процеси людини з використанням комп'ютерів. Фахівці зі штучного інтелекту, а пізніше філософи ШІ, швидко зрозуміли, що ШІ - це не тільки сфера, що займається інтелектуальними комп'ютерними системами, але також вивченням і розумінням людського інтелекту.

Інтерес до того, щоб зробити ШІ переважним *інструментом* вивчення людського інтелекту, породив безліч філософських питань. Деякими найбільш захоплюючими питаннями є наступні:

1) Чи може людське мислення бути змодельованим за допомогою інтелектуальної комп’ютерної системи?

2) Чи здатні ці запрограмовані комп’ютери мати свідомість?

3) Чи можуть психічні стани бути еквівалентними обчислювальним станам?

Фактично, переконаність в позитивній відповіді на перше питання називається сильним ШІ і безпосередньо пов'язана з позитивною відповіддю на третє питання, яка була названа радикальним *комп'ютаціоналізмом* (або, як іноді кажуть, переконанням, що пізнання

прирівнюється до обчислень) [42]. Обчислюваність буде розглянута в наступному підрозділі. Якщо говорити точніше, то в даному тексті ми розрізняємо частковий комп'ютаціоналізм, в якому стверджується, що пізнання лише частково може бути пояснено за допомогою обчислень, і повний комп'ютаціоналізм, в якому панує переконання , що пізнання є еквівалентним обчисленням і що *будь-яка* когнітивна діяльність може бути пояснена за допомогою обчислень.

Філософія ШІ досліджує фундаментальні питання, безпосередньо пов'язані зі згаданими вище, наприклад, чи можуть комп'ютери мати певний загальний інтелект, чи є рівнозначними комп'ютерний "інтелект" та людський інтелект. Крім того, в філософії ШІ етичні питання завжди посідали важливе місце. Ці теми також досліджуватимуться в наступних підрозділах.

1.14.1. Що можна запозичити з комп'ютаціоналізма?

Комп'ютаціоналізм це позиція, яку часто розуміють так, що обчисленнями можливо пояснити психіку. Іншими словами, комп'ютаціоналізм вважає пізнання і обчислення еквівалентними процесами [43, p. 1].

Потенційна співмірність між комп'ютерами і людськими когнітивними системами є початковим мотивом появи комп'ютаціоналізма.

Комп’ютаціоналізм, визначений таким чином, може надати механічне розуміння поведінки. Існує багато визначень комп’ютаціоналізма, оскільки кожне визначення пов'язано з, як правило, окремим розумінням обчислень. З цього випливає, що глибина тлумачення пізнання в поняттях обчислень безпосередньо пов'язана з типом визначення, яке ми a priori даємо обчисленням. У цьому контексті комп’ютаціоналізм може бути когерентний (чи ні) з філософськими і конкретними концептами в залежності від того, який варіант комп’ютаціоналізма ми беремо (слабкий, сильний і т.д.). Наприклад, припущення про те, що психічні стани еквівалентні обчислювальним станам, передбачає сильний варіант комп’ютаціоналізма, відстоювати який зазвичай важко. Заперечення цієї версії не означає заперечення комп’ютаціоналізма, але може привести до інакшого, більш слабкого його розуміння, яке передбачає, що обчисленнями можливо пояснити лишень тільки деякі когнітивні аспекти. [43] Саме таку позицію займаємо ми в цьому тексті, наводячи окремі факти на її підтримку.

На цьому етапі важливо навести різні погляди щодо обчислень, описані в літературі (Пуччініні), які мають прямий вплив на рівень дискусії про комп'ютаціоналізм. Обчислювальні системи іноді розглядаються як конкретні системи, забезпечені обчислювальною точністю і відповідні фізичним станам системи. Ця відповідність іноді обмежена дотриманням "причинно-наслідкових зв’язків між фізичними станами". [43, p. 5] Через незадоволеність деяких філософів таким

визначенням, був запропонований семантичний аспект обчислення. Цей аспект передбачає, що "обчислення потребують репрезентації, за якою, процеси, *котрі маніпулюють відповідним видом репрезентації відповідним чином*, вважаються обчисленнями".[43, p. 5] Це визначення, вочевидь, розглядає сучасні комп'ютери та когнітивні моделі як здатні обчислювати за допомогою репрезентацій. За тим самим визначенням, він виключає з обчислювального підходу всі інші моделі, які не здатні генерувати репрезентації. Важливо відзначити, що позиція, згідно з якою для обчислення потрібні репрезентації, не узгоджується з точкою зору теорії обчислюваності. Більше того, деяких фізиків цікавить питання, чи можуть фізичні процеси обчислювати чи ні, і які можуть бути обмеження щодо таких обчислень. Проблема тут пов'язана з припущенням, що багато фізичних систем не мають справу з репрезентацією. Це цілком очевидна проблема, оскільки ще потрібно довести, що фізичні системи дійсно здатні обчислювати. Інше тлумачення обчислень ґрунтується на *механістичному* підході, який розглядає обчислення як "особливий вид процесу, частково визначений видом пристрою, яким маніпулюють ... тільки невелика частка систем, що маніпулюють відповідними пристроями згідно з правилами, визначеними для них, вважаються обчислювальними системами".[43, p. 6]

Ці визначення або погляди щодо обчислень привели до різних поглядів щодо комп’ютаціоналізма. Деякі з них пов’язують пізнання із

процесором, що працює в мережевому середовищі. Інші вбачають в ньому автомати, особливий вид комп'ютера, універсальну машину Тьюринга або будь-який інший вид машин, які перевершують можливості класичних комп'ютерів.

Класичний комп'ютаціоналізм часто пропонував пояснення процесу пізнання засобами лінгвістичних структур або засобами *мови мислення*. Ця версія є предметом дискусій, особливо тому, що немає вагомих доказів того, що обчислення потребують лінгвістичного підґрунтя. На противагу, конекціоністська точка зору припускає, що "пізнання не має нічого спільного з мовою, і його можливо пояснити за допомогою *механізму* мозку" [40]. Інші більш сучасні конекціоністи розширили цю точку зору, але, мабуть, занадто загальним (та спільним) способом, стверджуючи, що пізнання можна пояснити за допомогою нейронних мереж, які, очевидно, складають функціонал мозку (з цієї причини ми використовували слово загальний). Однією з найдавніших позицій комп’ютаціоналізма є символістичний погляд, який проводить аналогію між людиною та комп’ютером, в тому сенсі, що обидва спираються на обробку символів. З причин, що стосуються історії цієї теми та для встановлення зв’язку з деякими логічними підходами, ми детальніше обговорюємо символістські та конекціоністські позиції в наступних рубриках підрозділу.

Символічний штучний інтелект

В основі символічного ШІ лежить ідея, що людський та комп'ютерний інтелекти постають результатом маніпуляцій із символами, з дотриманням певних правил. Ґрунтуючись на гіпотезі, що будь-яка комп'ютерна система необхідно є фізичною системою символів, Ньюел і Саймон припустили, що символьна модель (в фізичному сенсі) є еквівалентною будь-якій моделі, що володіє загальним інтелектом [40]. Тут ми використали "еквівалент" у сенсі достатності та необхідності умов. Ця ідея обмежує сферу володіння загальним інтелектом стверджуючи таку можливість виключно для фізичних систем символів. Відтак, автори вважали, що людський розум генерує фізичну систему символів, дуже схожу на комп'ютерний механізм, що водночас є вірою в комп'ютаціоналізм та позицією сильного ШІ [40]. Зауважимо, у більш м'якій формі комп’ютаціоналізма стверджується, що *символьної моделі цілком достатньо, щоб мати загальний інтелект, але вона не є для цього необхідною*. Наслідком цієї позиції є те, що різноманітні системи можуть бути наділеними загальним інтелектом і без того, щоб бути еквівалентними. Зокрема, комп’ютерні системи та людський розум, маючи загальний інтелект, не обов’язково мають бути еквівалентними[40, p. 31].

Питання комп’ютаціоналізма цікавили багатьох філософів, які пропонували аргументи за і проти цих переконань. Наприклад, Дж. Фодор, в праці *Гіпотеза мови думки* стверджує що: "Людське пізнання

складається із синтаксичних операцій із фізично реалізованими уявленнями в свідомості, які мають комбінаторний синтаксис та семантику" [40, p. 31].

На противагу цьому Джон Серль і Губерт Дрейфус виступали проти комп'ютаціоналізма, наводячи два різні аргументи. Серль використав те, що називається аргументом китайської кімнати, а Дрейфус залучив проблему здорового глузду, використовуючи одночасно ситуаційний та втілений підходи.

Докладніше, Серль вважає, що, на відміну від людей, "комп’ютери можуть маніпулювати символами, не розуміючи при цьому їх значення, що виявляє помилковість позиції комп’ютаціоналізма" [40, p. 31].

У своєму прикладі китайської кімнати Серль уявляє себе тим, хто не говорячи і не розуміючи жодного слова китайською, знаходиться у кімнаті. Він отримує китайські ієрогліфи в якості вхідних даних; Потім він механічно виконує деякі вказівки (припустимо англійську версію комп’ютерної програми), щоб на виході отримати інші китайські ієрогліфи. Отримані відповіді здаються такими, що їх продукує мовець китайської мови, що насправді не так. Серль стверджує, що він не розуміє китайської мови так само, як і комп'ютерна система, яка може його замінити. Таким чином, машина, яка не мислить, насправді не володіє розумом, а обчислення не є відповідними до процесу пізнання. Висновок, який він отримав, полягає в тому, що сильна версія комп’ютаціоналізма є хибною.

Однак Дрейфус вважає, що людське пізнання не використовує таку саму обробку інформації або, заснований на правилах підхід, що його використовують комп'ютери.

Ця ідея влучно висловлена Соакером, оскільки для Дрейфуса: "Людський інтелект є ситуаційним, тобто визначається ситуаціями, в яких опиняються люди, - і водночас втіленим, тобто виявляється у цілеспрямованих сенсомоторних взаємодіях людського тіла з навколишнім середовищем у реальному часі" [40, p. 32].

Соракер додає що: "Комп'ютери, (як стверджує Дрейфус), не мають втіленості, а обробка інформації ними не ситуаційною, а, натомість відірваною і абстрагованою від реальності, в якій вони опиняються" [40, p. 32].

Отже, автори вважають, що людський розум істотно відрізняється від комп'ютерної моделі. Більше того, Дрейфус вважає не тільки вважає комп'ютаціоналізм хибним, але і власне символічний ШІ таким, що не здатен продукувати загальний інтелект. Він стверджує, що символічний ШІ "допускає помилкові гіпотези, породжені декартовим раціоналізмом, такі, що сприйманий світ має об'єктивну формальну структуру і що можна моделювати всі типи знань" [40]. Більше того, це непорозуміння щодо символічного ШІ (за Дрейфусом) відповідає припущенню, що інтелект передбачає невтілене використання формальних правил.

У цьому контексті проблема формального представлення будь-якого типу знань породжує головну проблему. Це прямо означає (і не

приймається автором), що людські знання можуть бути представлені формально або на основі правил, що дозволяє комп'ютерні маніпуляції. У своїй аргументації Дрейфус вказує на проблему знань здорового глузду, стверджуючи, що комп’ютери, а отже і символічний ШІ, не здібні до здорового глузду, яким послуговуються люди. Він стверджує, що люди "здатні легко розрізняти змістовні та беззмістовні уявлення, використовуючи власні комплексні фонові знання" [44] і, що символічна модель ШІ до цього не здатна [44].

На основі попереднього обговорення ми піднімаємо наступні питання:

1- Яким є точне значення поняття *комплексних фонових знань*?
2- Чим саме представлений здоровий глузд в обчислювальних моделях (людських чи ні)?
3- Чи існують конкретні умови, що гарантують (або спростовують) наявність здорового глузду у обчислювальної моделі (людської чи ні)? Якщо існують, то необхідно вказати які.
4- Чи може комп'ютерна система накопичити *необхідні комплексні і широкі фонові знання* за допомогою, наприклад, комплексних моделей машинного навчання?

Штучний інтелект конекціоністів

Альтернативним до символічного ШІ є конекціоністський підхід до ШІ, розроблений у 90-х роках XX ст., в основу якого покладено моделі нейронних мереж [40, p. 32,45]. Основна ідея цього підходу передбачає, що релевантною моделлю інтелектуальних процесів є структура людського мозку, а не комп'ютери. Ця модель відкидає думку про те, що інтелект можна моделювати за формальними правилами з використанням символів. Замість цього він заснований на взаємодії зв'язків в мережі вхідних, вихідних і проміжних елементів, об'єднаних разом.

Багато задач можуть бути виконані за допомогою нейронних мереж, включаючи розпізнавання образів і координацію поведінки. Однак інші завдання, більш успішно розв'язувані за допомогою символічного ШІ, непридатні для реалізації нейронними мережами, як, наприклад, формальна діяльність, вирішення проблем і процедури міркування. Насправді символічний та конекціоністський підходи до ШІ дуже різні, але, часто здаються такими, що доповнюють один одного. Крім того, обидві парадигми є комп'ютаціоналістськими в сенсі спільності погляду щодо того, що психічна діяльність є, діяльністю з обробки інформації і, по суті, може бути описана процедурами або алгоритмами.

Незважаючи на комп'ютаціоналістський характер як символічного так і конекціоністського підходів до ШІ, між ними є значні відмінності. Наприклад, в той час як в символічному ШІ "діяльність з обробки є

послідовною, дедуктивною і централізованою з боку одиниці, що керує всіма задачами", [40] обробка в конекціоністскій мережі є "паралельною та асоціативною"[40] і досягається завдяки множині невеликих обчислювальних структур, які в сукупності, але незалежним чином, виробляють кінцевий результат[40]. На додачу, символічний ШІ в основному використовує формальні правила для маніпуляцій з інформацією, в той час як конекціоністскій ШІ використовує свого роду оцінку з точки зору вагомості елементів мережі в певний момент час [40].

Подібно до символічного ШІ, конекціоністський ШІ отримував як позитивні так і негативні відгуки з боку багатьох експертів, щодо поглядів представників напрямку на комп'ютаціоналізм. Наприклад, Дрейфус, який не погоджувався із символічним ШІ, також не погодився із конекціоністським ШІ, використовуючи той самий аргумент про знання здорового глузду. Він повторював, що в новій конфігурації потрібне використання старих знань для інтелектуального вирішення нових проблем. Цю вимогу неможливо виконати в конекціоністському ШІ, оскільки фонових знань у таких мережах просто не існує, а, отже, конекціоністський ШІ не міг би мати, на його думку, загального інтелекту.

Конекціоністський підхід a priori не суперечить комп'ютаціоналізму. Деякі конекціоністи, що є анти-комп'ютаціоналістами вірять в обчислювальну експлікацію поведінки з

використанням моделі нейронної мережі. Деякі інші, напротивагу, не вважають, що поведінка або будь-який психічний стан можуть мати в основі будь-які обчислення [43].

Такі позиції демонструють розмаїтість щодо орієнтованості на комп'ютаціоналізм навіть в рамках однієї і тієї ж групи переконань (класистської, конекціоністської, повністю анти-комп'ютаціоналістської та інших). Наприклад, щоб пояснити те, що ми вже обговорили, классистський погляд на комп’ютаціоналізм може спиратися на структури мов, щоб пояснити пізнання, з точки зору асоціанізму; конекціоністський погляд для пояснення пізнання може спиратися "на обчислювальні механізми мозку, на моделі обчислювальних нейронних мереж або ж на нейронні мережі, що не обчислюють" [43] (але працюють на інший природній основі). Також існують і інші моделі, як комп'ютаціоналізма так і анти-комп'ютаціоналізма.

Позиція часткового комп’ютаціоналізма на основі МТ

Одна з вихідних позицій радикального комп'ютаціоналізма, а саме те, що пізнання (або психічні стани) мають суто обчислювальну природу, походить від глибокої переконаності в аналогії між пізнанням та цифровими комп’ютерами. Ця точка зору сформувала багато уявлень щодо аспектів та досліджень у галузі штучного інтелекту, які навіть іноді ставлять на меті створення інтелектуальних пристроїв, що імітуватимуть людський інтелект або пізнавальну діяльність. З багатьох причин ми не обстоюємо в цьому тексті жодної з форм радикального чи

чистого комп'ютаціоналізма, в яких вважається, що *всі* психічні стани *цілковито* виявляються обчислювальними станами. Деякі з цих причин обговорюються в наступному розділі, коли наводяться аргументи проти радикального комп'ютаціоналізма. Виступаючи із критикою радикального комп'ютаціоналізма, ми водночас захищаємо, не суперечачи цій критиці, певну версію комп’ютаціоналізма. У цій позиції ми стверджуємо, що *окремі* види пізнавальної діяльності можуть бути пояснені і змодельовані за допомогою обчислювальних процесів у вигляді комп'ютеризованих моделей. Ці положення будуть детально описані в тексті.

В даному дослідженні, незалежно від усіх згаданих положень комп'ютерної теорії розуму, ми в основному ставимо під сумнів погляд комп’ютаціоналізма, підтримуючи одні й заперечуючи інші його аспекти, засновані на поняттях теорії обчислюваності, особливо що стосується універсальної машини Тюрінга (УМТ). У цій роботі ми розглядаємо модель МТ як еталон за замовчуванням, коли говоримо про поняття обчислення. У цьому контексті ми не проводимо суттєвої різниці між абстрактною моделлю МТ і сучасними цифровими комп'ютерами, навіть якщо іноді важливо враховувати їх технічні відмінності, що стосуються необмеженої пам'яті в порівнянні з пам'яттю з обмеженим об'ємом. Як і раніше, цифрові комп'ютери являють собою найбільш вірну і точну реалізацію МТ. Прийняття в даній роботі МТ (цифрових комп'ютерів) в якості моделі обчислень за замовчуванням

ґрунтувалося на багатьох теоретичних, технічних та історичних фактах. В контексті нашої дискусії про комп'ютаціоналізм, обґрунтування даного погляду має, серед іншого наступні аспекти:

- Історична важливість МТ, особливо в частині зацікавлення ШІ у створенні комп’ютерів, здатних виконувати когнітивні завдання.
- Сучасні комп’ютери, як модель обробки інформації, як і раніше є найуспішнішими моделями пояснення пізнання.
- Машини Тюрінга є найбільш точно визначеними і відповідними до сучасних комп’ютерів, які сьогодні є єдиною надійною конкретною моделлю обчислень; (Гіперобчислювальні ідеї (допоки!) не мають універсальної конкретну модель, якою вони могли б заручитися, щоб перевершити потужність МТ).
- Завдяки своїй формальній математичній моделі як корпусу теорії обчислюваності, будь-яка характеризація, що зв'язує МТ і пізнання, повинна бути пропорційно формальною і точною; це, безумовно, може допомогти в кращому розумінні аргументу комп'ютаціоналізма (або анти-комп’ютаціоналізма), заснованого на МТ.
- Логічні числення МТ здатні формалізувати, представляти та маніпулювати знаннями, частково подібними до знань людського мозку. Отже, ТМ можна вважати моделлю, що має більше переваг щодо пояснення цих частин пізнавальної діяльності.

- Машина Тюрінга, є достатньою для обчислення функцій, які обчислюються алгоритмами (якщо ми вважаємо вірним тезу Черча-Тюрінга).
- МТ має надійну математичну еквівалентність також і з іншими формальними моделями, такими як запропоновані Черчем, Постом, Кліном тощо.
- Всі досягнення в галузі ШІ (моделювання, виконання когнітивних завдань, навчання, вирішення проблем, міркування, спілкування) засновані на цифрових комп'ютерах з конкретною і експериментальною базою, яка може підтримувати нові види досвіду (біологічні, сенсорні зв'язки і т.д.).

На цьому етапі, говорячи про обчислювальний процес, ми маємо на увазі обчислювальну модель, за якою може працювати МТ. Це не обов'язково означає, що кожна поведінка піддається обчисленню, але якщо когнітивна задача може бути пояснена (описана, змодельована) за допомогою обчислювальної задачі, що використовує правила, і може бути виконана відповідним механізмом, ми вважаємо, що цей механізм спирається на машину Тюрінга.

Тюрінг передбачив кілька моделей обчислень, що відрізняються від його стандартної машини Тюрінга (МТ), включаючи реальні або випадкові числа, або машину, яка може виконувати програми нескінченно. Одна з найдивовижніших його ідей стосується штучних

нейронних мереж та його міркувань про те, чи може комп’ютер (і яким чином) бути інтелектуальним у сенсі мислення та навчання. Незважаючи на віру Тюрінга в комп’ютаціоналізм, сформульовану у тезі, що "можна побудувати машину, яка буде *дуже близько* імітувати поведінку людського розуму"[43], він висунув заперечення проти комп’ютаціоналізма, припустивши, що будь-яка створена "обчислювальна машина обмежена набором теорем, які їй можливо довести, що відрізняє її від математиків, які можуть виявити нові способи докази цих теорем"[43]. Отже, математики "можуть доводити все більше і більше теорем, ніж обчислювальні машини, і, відтак, людські когнітивні системи не є обчислювальними машинами Тюрінга" [43].

Багато філософських питань задають експерти, які цікавляться взаємозв'язками між обчислюваними функціями та *матеріалами* (людськими чи штучними), що здатні обчислювати ці функції. Наприклад, "Що означає, що якийсь фізичний об'єкт є реалізацією якогось абстрактного алгоритму, а якийсь матеріальний орган, такий як мозок, є втіленням ідеалізованої машини?" [46, p. 7]. Це питання цікаве особливо вживанням слів *реалізація* і *втілення.* Ці концепти можуть явити значно ширше коло запитань, ніж ідеї доведення (або спростування) програми комп’ютаціоналізма. Таким чином, питання еквівалентності між *тілом* психічних станів, яким є мозок, і обчислюваними функціями, що представлені комп'ютерами (або МТ),

частково виходить на рівень більш загального розуміння когнітивних можливостей людини та можливостей комп'ютерної обчислюваності.

Критика класичних аргументів на користь комп'ютаціоналізма

Вченими пропонуються різні аргументи на підтримку комп'ютаціоналізма. Наприклад, аргумент панкомп'ютаціоналізма, в якому стверджується, що майже все, і особливо Всесвіт, постає як обчислювальні системи, а отже, когнітивна система людини також є обчислювальною. Проблема цього аргументу, на наш погляд, може бути визначена як надмірна загальність позиції, яка 1) не дає ні достатньої точності (або характеризації) когнітивної системи, ні уявлення про сенс обчислень, 2) через широке розуміння обчислюваності, занадто багато понять і явищ опиняються під знаком обчислень, в той час як деякі з цих понять можуть бути краще класифіковані з використанням інших уявлень, таких як, наприклад, природні явища і т.д.

Аргумент комп'ютаціоналізма, ґрунтований на тезі Черча-Тюрінга, не є переконливим, особливо тому, що „він описує межі обчислень за допомогою ефективної процедури" [43]; це не є демонстрацією того, що кожна пізнавальна діяльність є обчислювальною [43]. Наприклад, як щодо діяльності, яка може бути здійснена без необхідності формального опису за допомогою ефективної процедури?

Двома іншими аргументами на користь комп'ютаціоналізма, що обговорюються в [43] є: "аналогія обробки інформації" [43]та "статус

когнітивної гнучкості" [43]. Перший проводить паралель між пізнанням і обчислювальними системами за властивостями семантичної інтерпретації і, в силу того, що людська маніпуляція репрезентацією подібна до обчислювального механізму, приходить до висновку про вірність позиції комп'ютаціоналізма. У цьому контексті одна з версій приходить до такого висновку, оскільки приймає точку зору, що люди і машини однаково обробляють інформацію, а інша, оскільки вважає *спільною для обох характеристикою* процеси мовної репрезентації. Деякі питання, які можна задати щодо цих аргументів, стосуються 1) важливості (або ваги) обробки інформації людиною в її системі пізнання, та 2) чи є кожен підхід до обробки інформації, що вимагає точності, справді обчислювальним чи ні.

Другий аргумент щодо когнітивної гнучкості наголошує на подібності між людським пізнанням та комп’ютерами в сенсі невизначеності меж можливостей вирішення проблем та (поведінкового) навчання. Така позиція обґрунтовується універсальністю людського підходу і підходу машин (Тюрінга), котрі проявляють гнучкість у виконанні безлічі подібних завдань та необмеженість розширення числа і типу цих завдань. Іншими словами, обоє вони слідують "механізму загального призначення" [43], виконуючи різні групи інструкцій для реалізації різних цілей. Вони можуть “розв’язувати математичні задачі, доводити логічні результати, мати справу з динамічною грою, розпізнавати паттерни, вести складне спілкування, вивчати нові концепти тощо” [43]. Ця позиція, яка

підтримує пряму аналогію між людським пізнанням і комп'ютером, хоча і заслуговує на повагу, залишає відкритим питання про деякі види людської пізнавальної діяльності, які не мають еквівалента в рамках комп'ютеризованої системи.

Підсумовуючи, слід сказати, що цілком можливо відстоювати позицію часткового комп'ютаціоналізма безвідносно будь-яких переконань щодо запропонованих пунктів. Зокрема, ми відстоюємо ідею, що *окремі* когнітивні механізми можливо пояснити за допомогою обчислень. Ми припускаємо, що ці механізми є видами *логічної* діяльності і включають, серед іншого, логічні обчислення (міркування, умовиводи і т.д.), формальні обчислення, детерміновані (скінчені) процеси, представлення знань, обробку інформації, аргументацію (в поєднанні з деякими мовними явищами) і навчання.

Критика класичних аргументів проти комп'ютаціоналізма

Численні аргументи проти комп'ютаціоналізма було представлено в [43]. У цьому тексті ми виділяємо чотири з них, які ми розширюємо особистими міркуваннями, і пропонуємо інші ідеї, що ставлять під сумнів позицію радикального комп'ютаціоналізма. Відібрані аргументи пов'язані з "інтенціональністю, свідомістю, математичним запереченням, втіленим і вкладеним поняттям і динамічним фактором" [43].

Аргумент інтенціональності, безумовно, найбільш відомий завдяки згаданому раніше досліду "китайської кімнати", що його запропонував Дж. Серль Він демонструє, як і чому обчисленню недоступне розуміння, і що обчисленням не властива інтенціональність, яка властива (людському) пізнанню. Є достатньо підстав визнати, що для розуміння когнітивних здібностей необхідно дещо більше, ніж обчислювальна інтерпретація, така як інтенціональність. Однак, питання, яке ми можемо поставити, полягає в тому, наскільки комп'ютаціоналізм може допомогти у розумінні поведінки, який механізм поведінки може бути чітко зрозумілий за допомогою (або на основі) комп'ютаціоналізма?

Цей аргумент, на наш погляд, може бути пов'язаний з аргументом свідомості у сенсі *розуміння*. Пропонований свідомістю фактор якого немає в обчисленнях вимагає, як і в випадку з інтенціональністю, певного рівня розуміння. Отже, питання тут подібне до попереднього і стосується виявлення меж обчислювальних репрезентацій, які пояснюють пізнання і не обов'язково вимагають свідомості. Насправді це питання могло б мати на меті часткове виправдання комп'ютаціоналізма або його слабкої версії.

Аргумент математичного заперечення спирається на порівняння обмеженості машин щодо доведення теорем та творчого потенціалу математиків щодо постійного винаходження нових методів доведення, якого комп’ютери не мають. В рамках цього підходу часто передбачається, що людське пізнання ґрунтується на більш досконалому

виді обчислень, який називають гіперобчисленням, що нібито здатне вирішувати більші й складніші задачі, ніж модель машини Тюрінга. Припустимо, що цей аргумент відповідає дійсності, тоді виникають наступні цілком природні питання:

1) Чи цей аргумент просто відкидає комп'ютаціоналізм, чи пропонує інший його тип із більшими можливостями, ніж пропонують класичні комп’ютери?

2) Чи можливо описати задачі, які можна обчислити за допомогою техніки гіперобчислення, які б не були обчислюваними за Тюрінгом?

Іншими словами, якщо машина Тюрінга здатна моделювати лише частину пізнання, що саме являє собою інша частина пізнання, назвемо її *гіперпізнанням*, яка не моделюється (або не обчислюється) машиною Тюрінга?

Незалежно від будь-якої відповіді на ці питання, нам здається, що аргумент гіперкомп’ютаціоналізма показує, що пізнанню - або його частині - може відповідати обчислювальна модель, навіть якщо ця модель теоретично не здатна бути відповідною пізнавальному процесу взагалі. Отже, слабка версія комп'ютаціоналізма, очевидно, не заперечується цим аргументом.

Втілена і вкладена пізнавальні позиції стверджують, що обчислення, напротивагу пізнанню, є невтіленим («відмежованим від тіла» [43]) та невкладеним (“не інтегрованим у середовище” [43]). Цей аргумент можна поставити під сумнів, твердячи, що обчислення може

бути, в деякому сенсі, втіленим в *апаратній системі* (мозку або комп'ютері), і з тієї ж причини воно постає вкладеним, оскільки мозок і комп'ютер існують в певних середовищах. [43]. Знову ж таки, це не суперечить розумінню пізнання в рамках часткового комп'ютаціоналізма.

Останній аргумент передбачає, що обчислення, на відміну від пізнання, є динамічним в тому сенсі, що воно змінюється в часі. Контраргумент часто стверджує, що в пізнанні також використовуються динамічні концепти або інструменти, наприклад, при вирішенні задач. Здається, що ні аргумент, ні контраргумент не в силах обґрунтувати анти-комп'ютаціоналізм (як і власне комп'ютаціоналізм), заявлений таким чином.

Насправді, відповідно до наших позицій і згідно із динамічним розумінням пізнання, ми наводимо такі міркування. Якщо ми визнаємо, що наше розуміння людського пізнання є динамічним, а наше розуміння комп'ютерних процесів ґрунтується на моделі машини Тюрінга, то як ми можемо обґрунтувати обчислювальну *еквівалентність* між чітко визначеною, встановленою і детермінованою моделлю (МТ) і невизначеною (або не повністю визначеною, а іноді навіть розпливчастою) недетермінованою концепцією людських психічних процесів?

Іншими словами, враховуючи наші достатньо повні знання комп’ютерних систем і обмеженість наших знань щодо численних аспектів людського пізнання, яким чином можливо довести *еквівалентність* між добре відомою (або зрозумілою) моделлю та майже невідомою (або незрозумілою) моделями? І в кінці, як це відношення еквівалентності можна було б вважати можливим за наявності величезних прогалин щодо знань про механізми людської психіки?

Це питання, так чи інакше, містить в собі аргумент проти радикального комп'ютаціоналізма, і ми називаємо таке заперечення *детерміністським (або, можливо, гносеологічним) запереченням* радикального комп'ютаціоналізма. Отже, будь-яку відповідь, на ці запитання, яка б містила в собі *обґрунтування еквівалентності* між пізнанням та обчисленням, мабуть буде дуже важко відстояти як аргумент. Водночас слабкий комп'ютаціоналізм все ще лишає можливості для обґрунтування навіть за умови існування буд-яких із перелічених переконань.

Окрім того, ми надаємо заперечення *онтологічного* типу, в якому підкреслюється *суттєва різниця* між пізнанням та обчисленням, що унеможливлює доведення їх еквівалентності.

Наприклад, важко довести, що такий ключовий компонент пізнання, як людське чуттєве сприйняття, засноване на даних первинних органів чуття, може бути змодельовано і пояснено за допомогою будь-якої форми обчислень. Крім того, лишається сумнівною правильність

розуміння того, наскільки подібна обчислювальна модель або пояснення відповідає її реальному сприйняттю/розумінню людиною. Цей сумнів породжений складністю конкретно і точно *виміряти* і *кваліфікувати* природне чуттєве сприйняття людини, яке за допомогою штучного механізму обчислень спрямоване на природу.

1.14.2. Гіперобчислення та гіперкомп'ютаціоналізм

Історично склалося так, що деякі опоненти радикального комп'ютаціоналізма пропонували моделі обчислень, які повинні були відповідати людському когнітивному представленню точніше, ніж машини Тюрінга із їх обмеженнями. Як вже було зазначено, ці моделі часто називають гіперобчисленнями. Ймовірно, найскладніша проблема, пов'язана з цим поглядом, полягає в тому, щоб паралельно з абстрактною моделлю запропонувати надійний фізичний механізм, який може конкретно вийти за межі можливостей МТ (Девіс). Сам Тюрінг серйозно сумнівався в тому, що МТ може повністю пояснити пізнання, незважаючи на згадку про деякі можливості (безумовно, обмежених) емуляції людського розуму. При цьому він був далеким від сильної версії комп'ютаціоналізма, що спирається на штучний інтелект. Класичним аргументом проти цієї радикальної позиції зазвичай є порівняння людського розуму і МТ з точки зору внутрішніх обмежень. У ньому стверджується, що людський розум, на противагу МТ, не має проблем з математичними результатами, подібних до тих, які приводили Гедель, Тюрінг і Райс щодо внутрішніх і теоретичних обмежених

можливостей. Отже, машини Тюрінга не здатні репрезентувати людський розум. У руслі цієї запропонованої *стіни*, як частково згадувалося раніше, інші ідеї наводилися на підтримку різних технічних механізмів, які могли б забезпечити більш потужні моделі, ніж МТ або конкретно сучасні комп'ютери. Всі ці моделі, що включають фізичні і динамічні системи, а також аналогові нейронні мережі, можуть відповідати принципу гіперобчислень, який: 1) містить не-тюрінгівські елементи обчислень; 2) повинен пояснювати (і моделювати) людське пізнання. Робилися спроби довести, що слабкість МТ безпосередньо пов'язана з принципами локальності та обмеженості. По-перше, це той факт, що "комп’ютер може змінювати відразу впізнавані конфігурації, лише за обмежений час. Другий означає, що комп’ютер може одразу розпізнати лише обмежену кількість конфігурацій (це означає, що кількість інформації, яка обробляється за скінченний час, має верхню межу)" [47, p. 105 Hernandez-Quiroz].

У будь-якій потенційній моделі гіперобчислень, особливо коли мова йде про її можливу фізичну реалізацію, два згадані принципи не мають місця, оскільки така моделі повинна мати здатність репрезентувати людське пізнання і виходити за межі можливостей МТ. Незалежно від будь-яких переконливих моделей гіперобчислень, наші основні сумніви щодо цього питання полягають 1) в сумнівності конкретної здійсненності або їх фізичної реалізації (ця проблема була представлена такими вченими, як М. Девіс) і 2) в тому, наскільки ці

моделі дійсно здатні моделювати і пояснювати людський розум. Іншими словами, наші два питання вимагають, для того щоб можна було дати на них зрозумілу відповідь, досить формальних і переконливих доводів, що стосуються потужності гіперобчислювальної моделі (сильнішої, ніж МТ), і, що, безумовно, є більш складним завданням, доводів щодо того, що запропоновані моделі можуть емулювати і бути відповідними кожному компоненту людського пізнання. З цього приводу можуть бути представлені і інші питання. Наприклад, чи є гіперобчислення єдиним переважаючим способом (в сенсі якогось більш потужного, ніж ТМ) уявлення і розуміння людського розуму, або існує інша модель, природним чином більш підходяща для розуміння пізнання; зокрема, щодо сторін людського пізнання, які *важко піддаються опису*, таких як сприйняття, свідомість і т.д.?

Як уже згадувалося, машини Тюрінга постають математичною моделлю, еквівалентною поняттю алгоритму, яке довгий час було неформальним поняттям обчислень. Відповідно, поняття алгоритму технічно описується машиною Тюрінга, а алгоритм (або процедура) є обчислюваним тоді, коли він може бути еквівалентно обчисленим машиною Тюрінга.

Ідея віднаходження гіперобчислювальної моделі, яка може обчислити більше функцій, ніж може зробити МТ, безумовно, має виправдання в поясненні людського розуму, але також пояснюється причинами, пов'язаними з функціями, що не обчислювані машиною Тюрінга, такими

як Діофантові функції. Теоретично кажучи, модель гіперобчислень під назвою "машина Тюрінга, що прискорюється" [47, p. 44 Franchette] була запропонована Д. Коуплендом [47, p. 44 Franchette] і претендує на вирішення Діофантової функції. Зокрема, вона пропонує подолати обмеження МТ, які полягають в тому, що будь-який алгоритм повинен виконувати кінцеве число дій; в ній дивним чином стверджується, що машина Тюрінга, що прискорюється може, принаймні теоретично, виконувати нескінченне число дій, використовуючи кінцеву кількість часу. В процесі пошуку глибшого розуміння і причин необхідності реалізації фізичної машини, відповідної до моделі прискорення Коупленда, виникло кілька запитань. Наприклад, було показано [47, p. 44] що необхідно "розрізняти за силою *отримання доступу* до результат обчислення і *обчислення* результату, виконуючи необхідні кроки" [47]. У цій праці визнається, що конкретне обчислення нескінченного числа кроків за кінцевий час для людини є складним. Однак питання доступу до обчисленого результату в цій праці характеризується по-різному. В ній також представлені можливі труднощі, що виникають із моделями гіперобчислення. Точніше кажучи, доступ (людини) до результату у внутрішньому значенні не вимагає фізичної реалізації. Однак той самий доступ (завжди з боку людини) у зовнішньому значенні вже вимагатиме фізичної реалізації. До того ж, стверджується, що деякі аспекти гіперобчислювальних моделей залежать від фізичних, а не математичних обмежень, що виправдовують їх реалізацію. Отже, завдяки цій фізичній реалізації вказані гіперобчислювальні моделі

можуть забезпечити доступ людини до їх результату у зовнішньому значенні. В контексті тези Черча-Тюрінга прийнято вважати, що можливості МТ безпосередньо залежать від природи обмежень щодо її фізичного втілення. Відповідно, ідею створення та усвідомлення фізичних обмежень будь-якої нової моделі [гіпер] обчислень слід завжди розглядати з позицій її реальної сили порівняно з її абстрактною силою [47].

Висновок щодо комп'ютаціоналізма та гіперкомп'ютаціоналізма

Позиція *радикального* комп’ютаціоналізма передбачає, що пізнання еквівалентно обчисленням, а саме, що *вся* психічна активність може бути змодельована і зрозуміла за допомогою того чи іншого виду обчислень. Сильна позиція гіперкомп’ютаціоналізма передбачає, що пізнання є еквівалентним до гіперобчислень. У попередньому підрозділі ми представили заперечення цих обидвох поглядів, одночасно відстоюючи позиції часткового комп’ютаціоналізма та гіперкомп’ютаціоналізма.

Ми зауважили, що позиція радикального комп’ютаціоналізма приречена на невдачу, надавши для цього нові докази, що стосуються аспектів *відмінності по суті* між пізнанням і обчисленнями і *відмінності за знаннями*, які стосуються рівнів розуміння, що лежать в основі цих двох концептів.

Що стосується гіперобчислювальних моделей, які б змогли подолати обмеження машини Тюрінга, то ми частково обґрунтували, чому ці моделі не можуть пояснити людський розум, використавши для цього аргументи проти повного комп'ютаціоналізма і спираючись на існуючі сумніви з приводу конкретної реалізації гіперобчислень[48], які можна перенести на філософські розмисли.

Ми погоджуємося з думкою інших, що тлумачення гіперкомп’ютаціоналізма як протиставлення комп’ютаціоналізма виглядає суперечливим. Це пояснюється тим, що гіперобчислення все ще залишаються особливим типом абстрактного, переважно радикального комп'ютаціоналізма. Крім того, ми вважаємо існує багато філософських та технічних зауважень, які можливо висловити щодо цього підходу.

1.15. Філософія інформації як частина філософії штучного інтелекту

Іншою проблемою, пов’язаною з філософією штучного інтелекту, є вивчення природи даних та інформації, а також взаємозв’язків між ними. Розширення цих взаємозв'язків розглядалося як варіант розуміння деяких процесів людського мозку, що поєднують беззмістовні символи (дані) з змістовними символами (інформацією).

Ця проблема призвела до появи багатьох тез, щодо комп'ютерів та розуміння даних. Зокрема одна із пропозицій стверджує, що "Якщо

комп'ютаціоналізм має рацію, і якщо людський розум здатний надавати значення символам, то комп'ютери, в принципі, повинні робити те саме" [49 section 4.2].

Очевидно, що ця ідея має на увазі питання трансформації беззмістовних символів у похідні змістовні символи; таким чином, природа інформації стає основоположною не лише у філософії інформації, а й у процесі розуміння природи обчислень.

Оскільки прийнято вважати, що інформація складається з належним чином оформлених і змістовних даних, і є лише перетворенням даних у щось змістовне, ми в цьому контексті намагаємося відповісти на наступні питання:

1. Яке місце посідають і яку роль відіграють у цьому перетворенні обчислення?
2. У чому подібність між перетворенням даних на інформацію за допомогою комп’ютерів та перетворенням/виведенням беззмістовних символів на змістовні ідеї у людському мозку?
3. Це має таке людське перетворення обчислювальний характер?

Інформація є дуже складним і багатогранним поняттям і стосовно її природи ведуться широкі дискусії у філософії інформації. Ми вважаємо, що розуміння концепту інформації є найважливішим завданням на шляху розуміння обчислення, для якого вона, по суті, є робочим матеріалом. Крім того, розуміння інформації має важливе

значення у філософії ШІ, особливо в частині, що розглядає інтелект як модель обробки інформації.

Дуже цікавим є опис розуміння інформації представлений у [50, p. 5] :

> Інформацію можна розуміти як набір можливостей (протилежність невизначеності); як кореляцію (і, отже, структуру), а також інформацію можна розуміти як код, як в ДНК... Крім того, інформацію можна розглядати скоріш як динамічну, а не статичну; її можна розглядати як щось, що передається і приймається, на неї можна дивитися як на те, що обробляється, або її можна розглядати як щось, що виробляється, створюється, будується. ЇЇ можна вважати яу об'єктивною так і суб'єктивною. Її можна вважати річчю, власністю або відношенням. Її можливо розглядати з позицій формальних або неформальних теорій. Їі можливо вважати синтаксичним, семантичним або ж прагматичним явищем. Можливо побачити її прояви в усіх сферах як природи так і суспільства [50, p. 5].

Пов'язуючи інформацію і інтелект між собою, Деннет поставив фундаментальне питання про те, чи можна "природний інтелект повністю проаналізувати за допомогою моделі обробки інформації на деякому рівні абстракції" [50, p. 18].

Складність цього питання полягає в основному в значенні *природного інтелекту* , який базується на різноманітних структурах обробки інформації. Фактично, природний інтелект часто асоціюється з кількома видами когнітивної діяльності, включаючи мову, пам’ять, вирішення задач, навчання тощо.

У цьому контексті абстрактний інтелект був визначений Вангом як *"будь-яка здібність людини або системи, яка автономно переносить форми абстрактної інформації між даними, інформацією, знаннями і поведінкою в мозку або системах"* [50, p. 19].

Враховуючи таку позицію, і очевидну необхідність встановлення точного значення терміну *поведінка* у обчислювальній системі ми вважаємо питання про те, чи може штучна система мати абстрактний інтелект, все ще актуальним.

1.16. Етичні питання штучного інтелекту

На наш погляд, етичний штучний інтелект займається розробкою штучних агентів або машин, які працюють в гармонії з людьми та їх оточенням. На додачу до цього, *етичний* агент повинен вести себе і приймати рішення в умовах фундаментального обмеження, що полягає

в тому, що його дії не завдають ніякої шкоди людям, тваринам і навколишньому середовищу. У зв'язку з закономірними питаннями про місце машини в житті людини та моральних відносин між ними, в зв'язку з штучним інтелектом з'явився ряд філософських питань. Зокрема, етичні аспекти стали основоположними не лише у філософії штучного інтелекту, а і в основі всіх аспектів ШІ.

В даному дослідженні ми розглядаємо штучні машини як здатні надати людині надзвичайні можливості щодо ефективного та оптимального вирішення важливих задач, яке без цих машин практично неможливо отримати. Це вочевидь пов'язано не тільки зі складністю проблем, але і з величезним обсягом інформації, яку неможливо точно обробити за доступний людині дуже обмежений час.

Незважаючи на все більш активне використання штучних сутностей в суспільстві, існує довгий перелік питань, які необхідно постійно обговорювати, що стосуються етичних аспектів машин, почуття відповідальності, обмежень і прав, що визначають відносини між людьми і машинами. У цьому підрозділі ми виокремлюємо деяких супутні проблеми та ставимо запитання, які, на наш погляд, варті більш ґрунтовного обговорення. Крім того, ми висловлюємо власну точку зору і зауваження щодо конкретних етичних питань, таких як заміна людей роботами при виконанні певних завдань, етична відповідальність, свідомість, моральний статус і закони робототехніки Азімова, особливо, коли мова йде про автономних інтелектуальних агентів.

1.16.1. Зауваження щодо заміни людей роботами

Використання робототехніки породило велику кількість філософських та етичних питань щодо бажаних і небажаних відносин між людьми та машинами. Ми прагнемо зрозуміти, чи існує якийсь можливий ризик в тому, що інтелектуальні машини виконуватимуть людські завдання. У такому випадку вбачається важливим виявити ці ризики та спробувати їх уникнути.

Наступні питання окреслюють коло нашої зацікавленості в цьому питанні:

- Наскільки комп’ютери/роботи повинні/можуть приймати рішення, особливо коли мова йде про людські життя?
- Якими є людські етичні зобов'язання, щодо делегування таких рішень роботам та щодо передбачуваних наслідків поведінки штучних агентів?
- Наскільки штучні агенти можуть підкорятися чи довіряти іншим штучним агентам? Як визначити пріоритет щодо прийняття рішень між людьми та машинами, а також між самими машинами?
- Як можливо розуміти, оцінювати і довіряти рішенню машини за відсутності розуміння поведінки непередбачуваних моделей, таких як еволюційні або конекціоністські системи? [28] Це питання піднімає проблему прозорості алгоритмів у розумінні

технічних причин та умов, за яких приймається конкретне рішення.

В ситуації, коли машини виконують конкретні завдання значно ефективніше ніж люди, особливо важливим стає поняття автономії. Таким чином, наведені раніше пункти або питання вимагають чіткого розмежування між характером завдань, які можна делегувати машинам, і тими, які не можна. У цьому тексті пропонуються окремі зауваження та умови прийняття рішень.

У цьому контексті цікаво буде розглянути відомі закони Азімова щодо робототехніки, і представити щодо них деякі закваження. Ці фіктивні, але водночас релевантні закони викладені в наступних трьох положеннях [40, p. 40]:

1) "Робот не може нашкодити людині або через свою бездіяльність, дозволити, щоб людині було завдано шкоду" [40, p. 40].

2) "Робот повинен підкорятися наказам людини, за винятком випадків, коли такі накази суперечать Першому Закону" [40, p. 40].

3) "Робот повинен захищати своє власне існування, якщо такий захист не суперечить Першому чи Другому законам" [40, p. 40].

Зверніть увагу, що закони Азімова надають пріоритет людському рішенню (роботи повинні підкорятися людям) і, очевидно, роблять людське життя пріоритетом у функціонуванні штучних систем. У цьому

контексті постає цілком природне запитання, чи достатньо цих законів, щоб захистити людські життя від роботів, які функціонують неналежним чином? Якщо це не так, як ці закони можна змінити, щоб гарантувати більший захист?

По-перше, потрібно наполягати на тому, щоб умова в пункті 1, що стосується "не завдання шкоди людині" [40, p. 40] виконувалася у будь-якому випадку, чи то прямо, чи опосередковано. Більше того, термін *через бездіяльність* має чітко передбачати дії щодо *захисту* життя людини за наявності будь-якої потенційної загрози, в тому числі, від інших машин. Крім того, такі істоти, як тварини разом із навколишнім середовищем, також мають бути враховані та захищені. Точніше, ми пропонуємо, щоб третій закон був названий четвертим, а наступний закон став третім:

3) "Робот не може заподіяти шкоду тварині або вчинити дії, які прямо або опосередковано (через бездіяльності) завдають шкоди навколишньому середовищу, якщо такий захист не суперечить попереднім законам" в основі яких лежать попередніх закони Азімова [40, p. 40].

За будь-яких обставин і враховуючи те, що неможливо запобігти *всіх* потенційних помилок в роботі штучних агентів, які можуть привести до втрати людських життів (масові руйнування, автономні роботи-вбивці, ядерна катастрофа), вкрай важливо додатково вивчити цю ситуацію, щоб вжити необхідних заходів. Більше того, що стосується останнього

закону Азімова (нині номер 4), варто обговорити низку філософський питань. Отже, ми пропонуємо наступні питання:

- У чому може полягати сенс існування машини (в моральному плані)?
- В чому сенс збереження такого існування, і в якій мірі цим агентам потрібно себе захищати?
- Чи можуть ці машини дійсно застосовувати моральні правила за умови відсутності їх розуміння і сприйняття в сенсі людської моралі?

1.16.2. Моральний статус, відповідальність, і свідомість роботів.

Ми вважаємо, що факт дозволу роботам приймати рішення або виконувати завдання, що стосуються людських життів, має прямий зв'язок із статусом та обов'язками, які ми маємо приписувати цим машинам.

Таким чином, на нашу думку корисним буде наступне:

- Розрізняти моральне існування та неморальне існування
- Визначити відповідальність у разі небажаних дій автономних машин (заподіяння шкоди або загибель людей)
- Вивчити можливості наділення етичними обов'язками за умови відсутності свідомості

- Вирішити, чи повинні штучні агенти мати моральний статус, а також визначитися, чи цей статус неминуче зменшує (або повністю знімає) людську відповідальність, зокрема відповідальність інших сторін, таких як розробники або власники відповідних інтелектуальних систем.

Незважаючи на можливі протилежні точки зору, неважко визнати, що штучні системи в цілому не мають морального статусу в тому ж сенсі, що і люди. Тут мається на увазі "здатність до феноменального досвіду і наявність вищого інтелекту і та усвідомлення" [51, p. 6]. У будь-якому випадку, приписування машинам морального статусу піднімає питання про моральну відповідальність і свідомість, які слід детально розглянути в даному тексті. Ми аналізуємо цей момент у наступних рядках.

Наприклад, ми вважаємо, що звільнення сутностей, що не мають свідомості від моральної відповідальності є центральним моментом. Мова тут йде про *свідомість подібну до людської* оскільки (принаймні частину) тварин можна вважати такими, що мають свідомість, але на них не покладається відповідальність у разі заподіяння шкоди людині. Така ситуація виникає і тоді, коли поняття покарання не може (принаймні, на наш погляд) застосовуватися до таких суб'єктів.

Етична відповідальність, згідно з визначенням у [52, p. 5], включає в себе "здатність співчутливо ставитися до ситуації інших; чинити вільно, або самостійно визначаючи свої дії згідно з прийнятними

правилами (а не просто *сліпо* виконувати вимоги, щоб залишатися в рамках цих правил)" [52, p. 5] а також "бути здатним прагнути або хотіти етично прийнятних ситуацій; мати морально доброчесний характер або схильності; проявляти відповідні моральні емоції або почуття...". [52, p. 5] Таке визначення етичних обов'язків цікаве у багатьох аспектах. Зокрема, воно імпліцитно наділяє суб'єктів *людськими якостями*, щоб вести мову про етичну відповідальність. Ці якості (які не завжди властиві лише людям) включають емпатію, бажання, чесноти, врахування ситуації інших вільно та цілеспрямовано, слідування відповідним правилам тощо [52, p. 5]. Однак, виходячи з попереднього визначення, необхідно провести більш глибоке дослідження того, якою мірою інтелектуальний агент може/повинен мати етичну відповідальність.

Ми припускаємо, що ці агенти не набувають досвіду, подібного до людського, вони не здатні зрозуміти людську етичну відповідальність, а отже, не можуть гарантувати етичних взаємин з іншими суб'єктами, що володіють моральним статусом, такими як люди. Враховуючи дане твердження, нам здається, що *володіння моральним статусом* є необхідною (але не достатньою) умовою для того, щоб приписати етичну відповідальність будь-якому суб'єкту. Більш того, якщо дотримуватися радикального погляду на моральний статус, за яким свідомість є необхідною умовою для того, щоб мати цей статус, то виходить, що володіння свідомістю є необхідною (але водночас завжди

недостатньою) умовою для прийняття будь-етичної відповідальності. Слід зауважити, що ця умова лишається недостатньою, оскільки тварини, які мають моральний статус, не можуть брати на себе багатьох видів відповідальності, як уже згадувалося раніше.

Тепер, щоб пояснити наше розуміння свідомості, наведене в попередньому абзаці, ми звернемося до думок, висловлених у [52, p. 5] наступним чином:

> [Свідомість] потребує здатності відчувати світ, жити життям свідомої, цілеспрямованої особистості, а не просто володіти окремими свідомими функціями. Отже, для того, щоб комусь мати справжнє етичне зобов'язання допомагати іншому, згідно з цим поглядом, йому потрібно бути взмозі відчувати емпатію до цієї людини. Отож, моральні дії, принаймні такого роду, засновані на можливості брати до уваги потенційний досвід інших людей так, ніби вони є частиною власної свідомості. [...] іншими словами, умовою свідомості від першої особи є емпатична ідентифікація з свідомістю інших [52, p. 5].

Іншою серйозною проблемою відповідальності машин є складність визначення деяких обов'язків при виникненні машинної помилки, особливо коли деякі види штучних, наприклад, недетермінованих реалізацій, є важко передбачуваними, оскільки вони просто використовують непередбачувані способи поведінки. Наприклад, як зазначалося в [51, p. 1], методи, в основу яких покладено складні нейронні мережі або генетичні алгоритми, можуть видавати помилки, щодо яких неможливо зрозуміти, як і чому вони виникли. Ми вважаємо, що складність виявлення дійсних причин *помилкового* рішення має прямий стосунок до приписування *правильної* відповідальності сутності, приймає участь у процесі (системі, дизайнеру і т.д.) У суспільстві Справедливості визнається, що коли хтось (або, в кінцевому рахунку, щось) заподіяв шкоду іншому, слід застосовувати певну міру відповідальності. У цьому контексті стає актуальною проблема *алгоритмічної прозорості*, особливо, коли мова йде про визначення моральної відповідальності за подібну поведінку машини.

Незважаючи на те, що деякі ідеї ставлять під сумнів те, чи можуть штучні сутності бути моральними за відсутності свідомості, інші погляди пропонують прямо приписувати моральний статус цим штучним агентам. Виходячи з наших попередніх зауважень, нам здається, що подібні атрибуції абсолютно позбавлені сенсу. Більше того, в подальшому ці атрибуції можуть бути небезпечними. Наша позиція

щодо позбавленості сенсу будь-якого приписування етичної відповідальності штучним агентам обумовлена багатьма факторами.

Для демонстрації нашої позиції наведемо простий приклад з реального життя, ми вважаємо абсурдною ситуацію, коли інтелектуальна система керування автомобілем визнається *винною* внаслідок можливої аварії, в результаті якої постраждала людина. Ця ситуація є аналогічною до тієї, в якій автономну систему керування зброєю вважатимуть *винною* за помилковий запуск ракети, що спричинив людські жертви. У зв'язку з цим можна було б вказати на багато інших юридичних і соціальний чинників, але вони не розглядаються в даній дисертації.

В цілому, в руслі попередніх зауважень про приписуванні морального статусу штучним агентам, питання про те, як люди повинні ставитися до інтелектуальних сутностей, обговорювалося в [52, p. 6]. У цьому випадку обговорюється тема *"дружніх та емпатичних відносин"* [52, p. 6] між людьми та роботами. Ці *дружні та емпатичні* відносини з роботами - це крок вперед на шляху до їх гуманізації, який відкриває широкі можливості для надання їм морального статусу, за умови якого, вони можуть брати на себе потенційну етичну відповідальність. На наш погляд, тут постає багато проблем, яких слід уникати. Ось деякі із цих побоювань: 1) ймовірність заміни відповідальності людини на відповідальність машини, що практично означає, що ніхто не несе реальної відповідальності, і 2) така ситуація, відсутності

відповідальності, дозволяє зацікавленим суб'єктам (розробникам або причетним організаціям) постійно порушувати етичні межі можливостей машин і, отже, збільшувати ризики, які можуть обернутися більшою небезпекою для людських життів. У деяких ситуаціях надання більш розширених повноважень автономним системам озброєння або надання політичних та соціальних прав інтелектуальним системам може призвести до невідомих, можливо, небажаних наслідків.

1.17. Висновок

У цьому розділі здійснено огляд окремих фундаментальних понять філософії обчислень та запропоновано суб'єктивні точки зору щодо комп'ютаціоналізма, гіперкомп'ютаціоналізма та етичних аспектів обчислень.

Філософія комп’ютерних наук, яка історично займалася осмисленням понять, пов'язаних з обчисленням і теорією чисел, охоплює також і питання, пов'язані з можливостями застосування обчислень в науці, і намагається дослідити цілий ряд проблем, що допомагає краще зрозуміти реальне значення обчислень з використанням будь-якої обчислювальної моделі і в будь-якому контексті. Ці фундаментальні питання можуть мати тісний зв'язок з математикою, логікою, свідомістю, пізнанням та їх філософським осмисленням.

Представивши огляд літератури, присвяченої дослідженням природи комп’ютерних наук, ми запропонували визначення цієї дисципліни, що поєднує її логічні, математичні та інженерні аспекти. Передбачається, що природа комп’ютерних наук має прямий зв'язок з деяким механізмом когнітивної обробки. На основі обговорення природи комп’ютерних наук в цій частині також проводиться дослідження характеризації комп'ютера. Цій першій частині розділу передували кілька елементів та запитань, які допомагають краще осмислити цю тему. Наприклад, визначення об'єктів вивчення філософії обчислень, коли мова йде про науку і сучасні технології, які визначають сенс обчислень, їх моделей та розвитку. Чи є доцільним розглядати обчислення як засновані на інформації, вирішенні задач, просторово-часових ресурсах тощо? Яким чином комп’ютинг впливає на людське пізнання, суспільство, навколишнє середовище тощо? Чи завжди обчислення мають користь? Чи є щось, що ми ніколи не повинні обчислювати?

У другій частині розділу предметом нашого розгляду була філософія штучного інтелекту, яка приділяла особливу увагу відношенням між логікою, пізнанням і обчисленнями. У цьому контексті ми виступили на захист погляду часткового комп’ютаціоналізма, який передбачає, що логічна частина пізнання може бути змодельована або відповідати одній з форм обчислень. Відповідно до цієї точки зору, в обчислювальному процесі не можливо відтворити більшість

пізнавальних форм. На додачу, ми висловили скептичну позицію щодо гіперкомп’ютаціоналізма, кладучі в її основу концептуальні елементи і лишаючи осторонь будь-які потенційні реалізації гіперобчислень.

Ми надали ряд аргументів на користь того, щоб розглядати машину Тюрінга як модель обчислення за замовчуванням у контексті комп'ютерної теорії розуму гіперкомп'ютаціоналізма. Ми також відмітили, що інформація відіграє суттєву роль в розумінні інтелектуальних процесів, а також пов'язана із рядом фундаментальних філософських питань.

Що стосується поняття етичних обчислень, і особливо ролі роботів, ми вважаємо, що роботи або будь-які інші подібні комп'ютерні системи змогли б виконувати завдання, які історично вважалися суто людськими. Наприклад, на противагу думкам, висловленим в [53], ми вважаємо цілком прийнятним, а іноді і навіть кращим, щоб деякі завдання, наприклад роботу представника служби підтримки користувачів, неозброєної поліції / митної служби тощо, могли виконувати роботи. Ми можемо помітити, що головною відмінністю при виконанні таких завдань штучними системами (а не людьми) є висока ефективність та точність внаслідок комп'ютеризації. Крім того, рішення представлене штучним агентом має переваги незацікавленості, більшої об’єктивності меншої упередженості.

В заключній частині даного розділу ми міркували щодо того, чому роботи не повинні мати жодного морального статусу незалежно від типу завдання, яке вони виконують. Ми дотримуємося такої точки зору,

оскільки існує ризик виникнення потенційних проблем в з відповідальністю. Наприкінці ми також торкнулися проблеми прозорості обчислень, і обговорили деякі її наслідки.

Проблема відповідальності роботів та проблема їх сили і спроможності виконувати завдання, на наш погляд, безпосередньо взаємопов'язані і їх слід розглядати разом.

Відстоювати можливість прийняття рішень штучними агентами або відкидати можливість прийняття ними ж інших рішень, на нашу думку, безпосередньо випливає з природи і наслідків конкретного завдання. Ми вважаємо правильним не об'єднувати всі явища і системи штучного інтелекту в один клас, особливо через велике розмаїття штучних архітектур і підходів, між якими іноді немає нічого спільного, крім назви "штучні системи". Наші оцінки таких систем повинні враховувати ці особливості і працювати з ними з урахуванням їх типів, можливостей, повноважень і застосовності.

Нарешті, і це було зазначено в [53] "питання не в цьому *що* розумні машини можуть робити, але *чи* ми повинні дозволити їм це робити" [53]. Ця позиція є багато в чому актуальною, оскільки враховувати рівень *розширення можливостей* того, що ми маємо дозволити інтелектуальним машинам робити - це людська відповідальність. Однак нам здається неоднозначним розуміння того, як штучний інтелект (який замінює деякі людські завдання чи робочі обов'язки) може «представляти загрозу для людської гідності» [53] як це висловлено в

тому ж самому джерелі. Зокрема, слово *гідність* у цьому контексті є цілком культурним та динамічним, і таким, що змінюється у просторі та часі. Чи вважають люди сьогодні використання банкомату (який по суті замінив банківського касира) загрозою людській гідності? Те саме питання можна поставити щодо сотень завдань/професій, які історично виконувалися людьми, а тепер натомість штучними інтелектуальними системи (медичні процедури, автопілот в автомобілі, переклад, державні послуги тощо). Отже, якщо машини замінюють людей в певних задачах, ми критично ставимося до *застарілих* поглядів стосовно того, що люди будуть "відчужені", "знецінені" або "розчаровані".[53]

Розділ 3. Впливи логіки на комп’ютерні науки

Цей розділ дисертації присвячено розгляду досягнень логіки, які, на думку автора, істотно вплинули на розвиток комп’ютерних наук. Маються на увазі, перш за все, логічні розробки Буля, Фреге та Рассела.

Ми даємо технічний опис алгебраїчної логіки Буля, в якій репрезентуються закони мислення, і показуємо, як класична логіка Буля призвела до появи сучасної Булевої логіки, підкреслюючи при цьому різницю у застосуванні в цих логіках законів. Крім того, ми даємо технічне пояснення, щодо застосування булевої логіки при розробці схем та у криптографії.

У другій частині ми описуємо причини, що пояснюють, на нашу думку, успіх логіки Фреге в комп’ютерних науках. Ці причини, на наш погляд, полягають у виразності логіки предикатів, відповідності її до інтерпретацій Ковальського, еквівалентності між диз'юнктами Горна та семантикою логічного програмування, а також у принциповому значенні принципу резолюцій у розробці цих мов. Ми надаємо опис всіх цих причин, на відповідному технічному та теоретичному рівнях.

На завершення ми поміщаємо теорію типів Рассела в контекст аксіоматичної системи і наводимо основні переваги теорії типів, такі як коректність, цілісність і ефективність комп'ютерних програм; крім того, ми простежуємо основні фактори успіху теорії типів в мовах

програмування, такі як абстракція даних, наслідування та поліморфізм. Ця частина включає роз'яснення щодо відповідностей Каррі-Говарда, які ми вважаємо головним досягненням, що лежить в основі успіху взаємозв’язків між логікою та теорією типів в контексті доведень.

1.18. Від Булевих *законів мислення* до комбінаційних схем та криптографії

У цьому підрозділі описується фундаментальне значення Булевої логіки 2 при проектуванні схем комп’ютерів та в криптографії. Ми розпочнемо з представлення Булевої логіки3 розробленої Джорджем Булем в 1847 р. і пізніше використаної для визначення Булевої або комутаційної логіки більш орієнтованої на комп'ютер. Насправді ця логіка є необхідною при проектуванні комбінаційних схем та у цифровій криптографії, що забезпечує безпечність обміну інформацією між комп’ютерами.

1.18.1. Створення алгебраїчної логіки

В одному із розділів праці *Закони мислення* [54, p. 50], Джордж Буль (1815-1864) описує свою алгебру логіки, наступним чином:

> … Ми можемо фактично відкинути логічну інтерпретацію символів в даному рівнянні; перетворити їх в кількісні

[2] Використовується взаємозамінно з терміном *Булева алгебра.*

[3]Використовується взаємозамінно з терміном *Алгебра Буля*.

символи, здатні мати значення тільки 0 і 1; виконати над ними як такими всі необхідні операції вирішення; і, нарешті, відновити їх логічну інтерпретацію (Буль). [54, p. 50]

Одне з найбільш фундаментальних переконань Буля щодо логіки полягає в тому, що будь-яке висловлювання повинно бути записано подібним до алгебраїчного рівняння чином, з використанням символічної алгебраїчної мови. У попередньому уривку терміном *операції вирішення* описується Булеве бажання віднайти в рамках його алгебри логіки такий механічний або автоматизований процес, що буде здатний працювати з логічними дедукціями. Цю ідею можна розуміти як процес зведення логіки до алгебри незалежно від реального переконаності Буля, чи логіка повинна бути частиною математики чи ні. Однак він чітко дав зрозуміти, що у своїх двох основних книгах віддає перевагу зв'язку логіку з математикою, а не з філософією [54,55]. Буль часто вказував на схожість між символічним використанням алгебри і способом символічного виразу логічних структур, таких як силогізми, і що навіть було більш для нього важливим, як ми можемо застосувати до них механізм обчислення [54,55]. Крім того, Буль пропонує новий погляд на математику, вважаючи її здатною обробляти концептуальні дані без використання кількісних аспектів. Такий некількісний процес іноді асоціюється з думкою, що загальна алгебра та алгебра логіки

зокрема дозволяють множинні інтерпретації. Це цітко висловлено в наступному фрагменті:

> Ті, хто знайомий із сучасним станом теорії символічної алгебри, знають, що обґрунтованість процесів аналізу не залежить від інтерпретації символів, [...] Будь-яка система інтерпретації, яка не впливає на істинність передбачуваних відносин, однаково допустима (Буль) [55, p. 3].

Таким чином, логіка стає формальною системою, яка розглядає поєднання символів, дотримуючись системи правил. Отже, інтерпретація будь-якого результату здійснюється згодом.

Історично склалося так, що багато математиків та філософів цікавилися приведенням логіки до форми математики. Наприклад, як ми вже бачили раніше, вже Г. В. Ляйбніц (1646-1716) був зацікавлений у побудові системи числення міркувань з використанням двох своїх основних ідей: універсальної мови та універсальна система логічного числення (Calculus Ratiocinator), чого, на жаль, довелося чекати аж до початку XX століття.

Цей підрозділ показує, як Булева алгебра логіки лягла в основу сучасної Булевої алгебри. Ми часто приписуємо перший вжиток терміну *Булева алгебра* Х. Шефферу, який здійснив це у 1913 році. Однак, схоже, що австрійсько-голландський фізик П. Еренфест вперше запропонував

використання булевої алгебри при проектуванні комутаційних систем ще у 1910 р. [4] У наступному підрозділі ми зупинимося на основних відмінностях між цими двома термінами. Наша мета в цьому підрозділі - висвітлити те, як в основному Булева алгебра застосовується в комп'ютерних науках. [11]

Наприклад, ми досліджуємо, як ця алгебра лягла в основу аналізу та проектування електричних схем, особливо щодо комбінаційних логічних схем. Щоб зробити це дослідження більш зрозумілим, ми почнемо із представлення базових засад логіки Буля, що потім розвинулися у Булеву логічну систему, в основі якої лежать виключно двійкові значення.

Ця система забезпечила, як зазначалося вище, теоретичну основу при проектуванні комбінаційних схем у вигляді наборів пов'язаних логічних вентилів, що репрезентують логічні операції (заперечення, диз'юнкція, кон'юнкція тощо).

У цьому контексті такі схеми, керуючи операторами комутації для обчислення булевої функції, роблять термін "Булева алгебра" еквівалентним терміну "алгебра комутації". Розширення цієї сфери безпосередньо пов'язане із схемами VLSI (схеми надвеликого рівня

[4]Еренфест, Пауль. *Review of L. Couturat, The Algebra of Logic,* Jour. Російський фіз. та хім. Soc., sec. 2, vol. 42, no 10. 1910.

інтеґрації), що використовуються в будь-якій сучасній електронній машині.

Більше того, в цьому розділі розглядається ще одне масштабне застосування Булевої алгебри. Мова йде про криптографію та теорію коректуючого коду. Криптографія спрямована на створення безпечної цифрової комунікації, при використанні незахищеної платформи. Ця теорія використовує методи шифрування та дешифрування на основі Булевих функцій. Призначенням коректуючого коду є виявлення та виправлення можливих помилок при передачі даних через канал зв'язку.

1.18.2. Як перетворити думки на рівняння

В основу даного підрозділу покладено структуру та позначеннях, які використовуються в нашій роботі [56] та інших цитованих у тексті джерелах. Початковий варіант логіки Буля можна коротко описати наступним чином [11,54–56]:

- Класи $x, y, z, \ldots$ які можуть репрезентувати множини, поєднані операторами: *заперечення*, *об'єднання*, і *перетин*.
- $\bar{x}$ - це заперечення класу x і репрезентує елементи області, яка не належить до класу x.
- $x + y$ є об’єднанням двох класів x і y репрезентує елемент у x або у y.
- xyxy - є перетином двох класів x і y репрезентує елементи в x і y.

- (-) - символ віднімання. $x - y$ репрезентує елементи x але не y. Він застосовується лише якщо клас y є підкласом x.
- (=) дорівнює символу різниці ($'! =' \ or \ ' \neq '$).
- *Універсум дискурсу* (або *область дискурсу)* є визначеним. Цей універсум можна розглядати як множину, до якої належать вказані змінні (тварини, цілі числа, яблука тощо). Також визначено порожній клас (null), який нічого не містить. Універсум позначається як 1, а порожній клас як 0.

Примітка: Питання щодо знання, у випадку коли даний елемент належить до універсуму, завжди є істинним (або 1), а у випадку якщо він належить до порожнього класу, завжди є хибним (або 0).

- $1x = x$1x = x та 0x = 0 $0x = 0$. Ці два спеціальні класи, універсум і порожній клас, вважаються оригінальністю в алгебрі Буля. [11]
- Закон, що постійно повторюється $xx = x$: перетин між класом x і самим собою є справедливим x.
- $x^n = x$. Цей закон є ідемпотентним і означає, що перетин класу x із самим собою n разів лишається x.

Примітка: Операція ділення в алгебрі Буля є позбавленою сенсу. Однією причин цього є те, що ділення на x обидвох частин рівняння $x^n = x$ робить його рівним $x^{n-1} = 1$ що є не має сенсу, оскільки перетин класу

x із самим собою повторюється $n-1$ не обов'язково буде 1 (універсум), за винятком випадків, коли x репрезентують цей універсум [11,56].

- Операція об'єднання є взаємовиключною. Це робить операцію об’єднання між двома класами x і y представленою $x+\bar{x}y$ замість $x+y$ що виключає будь-який член, що міститься в x і y в той самий час [11]. Зверніть увагу, що це позначення, що виражає об'єднання, є основною концептуальною відмінністю між цією первинною алгеброю Буля та новою розширеною алгеброю, яку зазвичай називають Булевою алгеброю.
- Негативний знак (-) представляє оберненість знаку додавання і виражає компліментарність класу. $1-x$ позначає доповнення x або елементи універсуму без елементів x.

Примітка: Позначення $\bar{x}$ пізніше було використано для заміни доповнення x. Негативний знак (-) дозволив виразити порожнечу перетину між класом та його доповненням за допомогою рівняння $x(1-x)=0$ [11].

Наразі, чотири основні ситуації традиційної логіки стало можливим записати за допомогою Булевої системи позначень [11,54–56]:

- Кожен x є y: $x(1-y)=0$.
- Жоден x є y: $xy=0$.

- Деякі x є y: $xy! = 0$.
- Деякі x не є y: $x(1 - y)! = 0$.

Одним з найбільш важливих символів, введених Булем у свою алгебру логіки, є символ v. Цей символ означає *деякі*. Це дозволяє уникати використання символу нерівності $(! =)$ що з'явився в останніх двох ситуаціях. Речення *деякі* x *є* y означає, що існують деякі члени x які одночасно є членами y. Ці спільні члени позначаються спеціальним непорожнім класом v. Використовуючи цей спеціальний клас, останні дві ситуації *деякі* можна записати наступним чином [11,54–56]:

- Деякі x є y:$xy = v$
- Деякі x не є y: $x(1 - y) = v$

Використовуючи попередні визначення, основні закони логіки Буля, написані у відповідності із традиційною алгеброю репрезентують наступні властивості [11,54–56]:

- $xy = yx$
- $x + y = y + x$
- $x(y + z) = xy + xz$
- $x(y - z) = xy - xz$
- $(x = y) \rightarrow (xz = yz)$

- $(x = y) \rightarrow ((x + z) = (y + z))$
- $(x = y) \rightarrow \big((x - z) = (y - z)\big)$
- $x(1 - x) = 0$

Зверніть увагу, що за законами алгебри остання властивість дійсна лише тоді, коли клас x є або порожнім класом, або класом універсуму (0 або 1). Це зауваження не свідчить про двійковий характер системи Буля. Насправді, первинна логіка Буля є досить загальною, що дозволяє вказаним класам мати і інші значення окрім 0 та 1. Однак завжди можна створити систему для особливих випадків, обмеживши допустимі значення меншим набором, наприклад, використовуючи лише двійкові значення. У цьому випадку умова $(either\ x = 0\ or\ x = 1)$ та деякі інші умови дають змогу прийти до конкретної алгебри, яка називається Булевою алгеброю. [11,56]

Ці правила скоріш за все, мають своїм корінням традиційну чи, інакше, числову алгебру. Після необхідних числень правила слід застосовувати подібно до того, як їх застосовують в класичній алгебрі. Таким чином, різниця між різними численнями полягає не в синтаксичній частині мови, але у видах інтерпретацій, які ми можемо викласти згодом.

У цьому контексті важливою метою цього процесу є здатність генерувати логічні висновки без урахування багатьох можливих інтерпретацій, які можна висловити до закінчення числення. Для Буля

генерування висновків має бути механічним процесом, який можливо порівняти з розробкою виконувану аналогічним чином при вирішенні лінійного рівняння або виразу, що включає функції [11,56].

У наступному підрозділі показано, як Булева логіка впливає на розробку цифрових схем.

1.18.3. Коли Булева функція стає комбінаційною схемою

Особливим типом алгебри логіки, розробленої Дж. Булем, є Булева алгебра в якості двійковою істинність-функціональної логіки. Нижче ми подаємо три основні відмінності цієї логіки від алгебри логіки, що була описана вище [56] :

1. Будь-яка змінна в Булевій логіці має значення або 0 або 1.
2. Додавання в Булевій логіці є інклюзивним.
3. У булевій логіці символи *доповнення* (-) та *не дорівнює* (≠) не використовуються.

Примітки: Це обмеження в пункті 1 позбавляє необхідності користуватися поняттям *деякі*, а отже, значення v не існує в Булевій алгебрі. Подібним чином, через використання двійкових значень, у використанні символу (≠) також не має сенсу.

Зверніть увагу, що оператори *доповнення (не), додавання (або) та множення (та)* відповідають запереченню, диз'юнкції та кон'юнкції в Булевій алгебрі. Ці операції, як правило, позначаються символами ¬, ∨ та ∧ відповідно.

Від транзисторів до вентилів

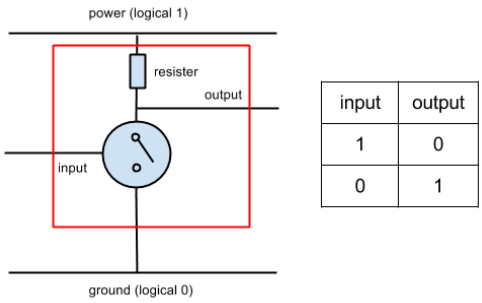
У цьому дослідженні ми виявили, що основною областю застосування Булевої логіки, як в реальному житті так і особливо в комп'ютерних науках, є аналіз схем. Такий успіх розпочався з теоретичного доказу можливого використання Булевої логіки в електронних схемах, здатних обчислювати двійкові функції. На це доведення, представлене Клодом Шенноном у 1938 році в дисертації [57], завжди посилаються, коли йдеться про проектування комп'ютерних схем. Ця ідея була і залишається фундаментальною при розробці будь-якого сучасного обчислювального пристрою.

Булева функція - це будь-яка функція, яка має позитивне число булевих вхідних значень і лише одне значення 0 -1 на виході. Основними операторами Булевої функції є заперечення, диз'юнкція та кон'юнкція (відповідно $\neg, \vee, and \wedge$). Будь-яка Булева функція може бути представлена таблицею істинності, її істинні значення залежать від комбінації значень її змінних. Практично, як тільки ці три основні оператори фізично реалізовані, будь-яка Булева функція може бути відтворена за допомогою комбінації одного або декількох базових

операторів для того, щоб спроектувати повну фізичну схему, яка відповідає функції.

Технічно, основні булеві оператори реалізовані за допомогою транзисторів, які є крихітними комутаційними пристроями, спроектованими як заземлені конструкції і використовуються багаторазово для побудови цифрової схеми.

Оператор НЕ може бути реалізований простим транзистором, і це показано на малюнку 1. 5



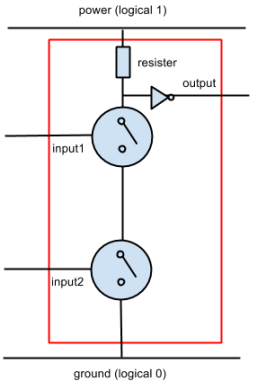
input	output
1	0
0	1

Рисунок 1. (Вентиль "Не") [58]

Функціонування транзистора базується на подачі високої та низької напруги, яким можуть відповідати Булеві значення 1 та 0. Якщо на вході з'явиться висока напруга, відкритий затвор автоматично закриється, і встановиться з'єднання між землею і виходом, який прийматиме низьку напругу 0. В протилежному випадку, якщо вхідний

[5]Рисунки 1, 2 та 3 запозичені з ресурсу en.wikibooks.org

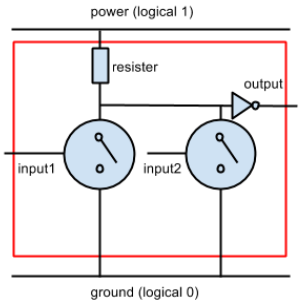
сигнал дорівнює 0, відкритий затвор залишатиметься відкриттм; вихід, в свою чергу, отримуватиме високу напругу крізь його підключення до джерела напруги за допомогою резистора. Отже, значення 1 передається на вихід (у верхній частині рисунка). На цьому рисунку показані два випадки, в яких представлено вентиль "НЕ". Можна легко розширити цей вентиль до "І" та "АБО", як показано на рисунках 2 та 3 [56,58].



input1	input2	output
0	0	0
0	1	0
1	0	0
1	1	1

Рисунок 2. (Вентиль "І") [58]

Якщо розглядати вентиль "І", два затвори чітко замикаються, коли вхід1 та вхід2 дорівнюють 1. Це передає значення 0 (земля) у напрямку виходу, раніше ніж буде здійснено заперечення до значення 1 в якості остаточного результату. В іншому випадку значення 1 проходить через резистор, перш ніж буде заперечено до 0 в якості остаточного результату [58].



input1	input2	output
0	0	0
0	1	1
1	0	1
1	1	1

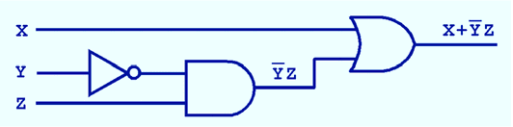
Рисунок 3. (Вентиль "АБО") [58]

Вентиль "АБО", в свою чергу, передає значення 0 (земля) в тому разі, коли принаймні або вхід1 або вхід2 дорівнюють 1. В подальшому, передане значення 0 заперечується безпосередньо перед тим, як прийме значення 1 на виході. Напротивагу цьому, значення 1 (зверху) слід заперечити до 0, перш ніж передавати його як остаточний результат [58].

Від вентилів до схем

На основі трьох основних конструктивних елементів, які ми тільки що обговорили ("НЕ", "І", "АБО"), тепер можливо фізично реалізувати будь-яку комбінаційну логічну схему. Такі схеми є, по суті, з'єднаними між собою блоками вентилів, які приймають двійкові значення змінних функції і видають унікальне двійкове значення, відповідне оцінці функції. Приклад комбінаційної схеми, яка обчислює функцію *F* наведено на рисунку 4. Ця функція використовує три основні вентилі:

Трикутник, що упирається в коло, репрезентує вентиль "НЕ", середній - це вентиль "ТА", а самий правий вентиль "АБО" продукує вивід функції *F* [59].



$F(X,Y,Z) = X+\overline{Y}Z$

Рисунок 4. (Приклад комбінаційної схеми, що обчислює функцію) [59]

У галузі проектування схем однією з найважливіших задач є створення оптимальних схем шляхом зменшення кількості використовуваних вентилів. Досягнення цієї мети дозволяє зменшили вартість і збільшити швидкість реалізованої схеми.

З цих причин експерти створили ще один набір розширених конструктивних блоків, які поєднують більш складні логічні операції в одному вентилі. Деякі з найбільш часто використовуваних розширених вентилів - це вентиль XOR ("виключне АБО"), вентиль NAND ("І-НЕ") та NOR ("АБО-НЕ") [56]. З іншого боку, кращим рішенням для мінімізації використання вентилів є, з початку, спростити початкову функцію, яку необхідно обчислити, а потім збудувати її фізично як комбінаційну схему. Метод Карно - один із найбільш часто використовуваних методів для отримання оптимального Булевого виразу. Цей метод застосовує повторюваний механізм спрощення щодо заданої Булевої функції, яка спочатку змодельована в таблиці істинності. За допомогою методу Карно можна утворити мінімізовані стандартні

форми, такі як сума-продуктів (SOP) або продукт-сум (POS), які разом підходять як для ефективної оцінки булевої функції, так і для систематичного проектування її цифрової схеми [59].

Наступний підрозділ присвячено важливості використання Булевої функції в криптографії та коректуючому коді.

1.18.4. Логічні функції для використання у криптографії та коректуючому коді

В комп’ютерних науках існує дисципліна, головним призначенням якої є забезпечення захисту цифрової комунікації між двома людьми. Це означає унеможливлення доступу сторонніх осіб до змісту повідомлення (відкритого тексту) навіть за умови доступності шифрованого повідомлення (шифротексту). Традиційна криптографія являє собою перетворення оригінального тексту в зашифрований текст за допомогою закритого ключа; відкрита криптографія, навпаки, використовує відкритий ключ у процедурі шифрування. Процедуру відновлення зашифрованого тексту з використанням секретного ключа називається розшифровкою або розкодуванням. Багато в чому подібний механізм, під назвою коректуючий код, намагається виявити та виправити помилки, які можуть виникнути під час цифрового зв'язку. Нагадаймо, що Блоковий шифр є криптографічним алгоритмом, що виконує операції шифрування в усьому блоці даних фіксованого розміру одночасно. Потоковий шифр, в свою чергу, є альтернативним методом шифрування, який опрацьовує по одному біту даних за раз [56,60].

Є багато причин, чому ми наголошуємо на важливості використання Булевих функцій в криптографії та у коректуючому коді. Ми маємо на увазі функції, які отримують ряд Булевих значень і дають унікальне Булеве значення ($F: \{0,1\}^n \rightarrow \{0,1\}$). Наприклад, як потоковий шифр, так і блоковий шифр використовують композиції нелінійних Булевих функцій під час криптографічних перетворень. Більше того, S-блоки (блоки підстановки), які, в блокових шифрах належать до алгоритмів підстановки, можливо цілком вважати векторними Булевими функціями, а операція "виключне АБО"(XOR) разом із іншими безпосередньо використовуються в деяких алгоритмах шифрування. Наприклад, алгоритми chacha20 та Blake використовують операцію додавання-чередування-XOR (ARX). Теоретично, для поліпшення ефективності використовуваних методів багато основоположних Булевих принципів в криптографічних системах, такі як кореляційний імунітет, витривалість і нелінійність, в основному застосовуються з використанням, наприклад, дистанції таблиць істинності Булевих функцій. [56].

Більше того, ступінь стійкості до деяких відомих атак у криптографічних моделях можна виміряти, користуючись основними властивостями використовуваних Булевих функцій, які, як правило, пов'язані з концептами "плутанини" та "поширення". Ці два принципи, визначені К. Шенноном, є одними з найбільш фундаментальних у криптографії. Зверніть увагу, що мета принципу плутанини полягає в

тому, щоб зробити якомога більше складною відповідність між повідомленням, що шифрується і його ключем. Метою принципу дифузії є розсіювання частоти статистики шифротекста. Це означає, що навіть незначні зміни в ключі або у вхідному тексті породжують істотні зміни в результативній частині [60].

1.19. Логіка предикатів Фреге: як логіцизм вплинув на появу мов програмування

Мета цього підрозділу, який в основному, ґрунтується на матеріалах нашої публікації [61], полягає в тому, щоб обґрунтовано пояснити чому логіка Фреге так істотно вплинула на розвиток комп'ютерних наук. Ми наводимо докази того, що ця логіка була центральним теоретичним ядром багатьох областей комп'ютерних наук, таких як логічне програмування, автоматизоване доведення теорем (АДТ) і штучний інтелект в цілому. Можна навести в якості прикладу також і багато інших областей комп'ютерних наук таких як бази даних та теорія описової складності, але вони не є предметом даного дослідження.

У цьому дослідженні ми в основному зосереджуємось на мові "Пролог" та системах АДТ.

У своїй праці *Begriffsschrift* [62] (або *Числення понять*), Готтлоб Фреге (1848-1925) розробив формальну мову логіки для обчислення істинності функцій. Це дозволяє аналізувати висловлювання як

функцію. Крім того, аргумент використовується як предикат. Ця модель включає основні квантори (спільності та імпліцитно існування) і використовує форму логічної формули для отримання дедукції.

Більше століття потому Дж. А. Робінсон, який створив "правило резолюцій" (ПР), розглядатиме те, що ми маємо, і те, що нам потрібно, як таке, що схоплюється логікою першого порядку [37, p. 2]. Він наголошує на тому, що ЛПП посідає центральне місце у логіці ще починаючи з епохи греків. Він відмітив, що: "ЛПП може бути використана для перетворення на теорії першого порядку значної кількості *інших логік* таких як модальна логіка, логіка вищого порядку, темпоральна логіка, …, квантова логіка та багато інших" (Робінсон) [63, p. 2].

Його точка зору полягає в тому, що всі ці логіки засновані на одній логіці: логіці першого порядку; він зазначає:

> *Інші логіки* - це просто системи позначень, що виражають синтаксично підсолоджені визначення понять або обмежень, які можливо формалізувати в рамках ЛПП. Існують певні універсальні схеми міркувань, засновані на тому, як насправді працює наш розум, і вони охоплені ЛПП (Робінсон) [63, p. 3].

Робінсон вважає, що логіка першого порядку не тільки є важливою для розвитку логіки, але й поширює її вагомість у інші сфери, зокрема у філософію математики. Ця ідея має тісний зв'язок із головною метою Фреге, так званою, *програмою*, в якій стверджується, що, застосовуючи його формальну систему, можливо звести всю арифметику до логіки, включаючи навіть дедуктивні умовиводи, що використовуються в математиці. Також, однією з головних цілей Фреге було досягнути точності дедукцій разом із уникненням будь-яких висновків, заснованих на інтуїції.

Робота Фреге призвела до появи синтаксичної системи, метою якої було зменшення або навіть усунення помилок, які може дедукувати система. Таким чином, ці концепти, включаючи логічні умовиводи, тепер можливо виконувати виключно механічним способом. Однак, треба визнати, що його програма виведення всієї арифметики в рамках такої системи виявилася суперечливою. З іншого боку, її вплив на розвиток комп'ютерних наук все ще лишається вагомим, навіть через 140 років після першої публікації.

У цій дисертації термін ЛППП вживається еквівалентно із термінами *логіка предикатів* (ЛП) та *числення предикатів першого порядку* ЧПППП. Цей тип логіки що містить дуже строгі граматичні правила вважається головним родоначальником комп'ютерних мов [4, p. 53].

Поняття логіки предикатів з її формальною системою, синтаксисом, правилами умовиводу та дедуктивних міркувань дали

початок створенню механічних обчислювальних процесів, які є *raison d'être* в комп’ютерних науках. У цій дисертації наводиться багато факторів успіху логіки предикатів в комп’ютерних науках. Ми узагальнюємо ці фактори наступним чином: У 1965 році Дж. А. Робінсон у праці [64] визначив у диз'юнктивній формі, *специфічну* підмножину ЛПП, з одним правилом висновування. Після більш ніж двох десятиліть розвитку практичної комп’ютерних наук, *вдосконалене* правило резолюцій Робінсона став основним прогресивним моментом в мовах логічного програмування. Це правило має прямий стосунок до розробки Пролог (програмування за допомогою логіки) Кольмерое та співавт. [65]. Вказані досягнення, включаючи інтерпретацію Ковальського [66] зіграли вирішальну роль у впровадженні (індуктивних) моделей машинного навчання з використанням поняття індуктивної логіки, розробленого Магглтоном [67]. Більше того, правило резолюцій вважається фундаментальним у багатьох системах автоматизованого доведення теорем (АДТ), які, по суті, є логічними програмами, реалізованими з метою перевірки або доведення важливих, як правило, складних гіпотез чи проблем. Більшість цих новітніх комп’ютерних понять постали внаслідок появи логіки Фреге.

У цьому підрозділі ми вводимо основні елементи логіки предикатів Фреге, використовуючи її сучасний синтаксис, і наголошуючи на високій виразності, що контрастує з багатьма іншими типами логіки, розробленими раніше. У другій частині ми спочатку

досліджуємо інтерпретацію цієї логіки Ковальським, в якій вона розглядається в якості мови програмування, а потім ми показуємо, як поняття диз'юнктів Горна та правила резолюцій стали основними складовими досягнень прологу. Ці два поняття відповідно обговорюються в четвертому розділі. На останок, ми згадуємо потенційні дослідження ролі ЛПП в інших областях комп'ютерних наук.

1.19.1. Сила виразності логіки предикатів

У цьому підрозділі ми наводимо основну термінологію, що використовується в логіці предикатів. Деякі приклади, написані мовою сучасною логікою предикатів, наведені, щоб показати високий рівень її виразності чого не можливо було досягти до появи концептуальних позначень Фреге. Ця виразність є головним елементом успіху Фреге.

Терміном може бути ім'я об'єкта ('1'), *комплексний термін*, що представляє об'єкт ('2-1'), або правильно сформовану формулу як послідовність складних термінів. Комплексні терміни - це часто *функції* які отримують *аргументи*. Будь-який концепт є *n-місною* функція, що замінює аргумент значенням *істинності* і продукує значення істинності [61].

Більше того, додаткові комплексні вирази, що вимагають заперечення та умовності є можливими в системі Фреге. Таким чином, знак рівності використовується для відображення еквівалентності двох

виразів, який у сучасних текстах зазвичай замінюється символом еквіваленції.

З іншого боку, ще одним фундаментальним принципом у логіці Фреге є використання *квантифікацій*. Зокрема, присутні у логіці Фреге концепти "кожен" і "деякі" дозволяють використовувати більш складні правила висновування. Зауважте, що квантор існування відкрито не вводився в оригінальну систему Фреге. Однак його можна замінити квантифікацією спільності та запереченням [61].

У цьому контексті доречно зауважити, що відношення M формул $\exists y, M(y, item)$ або $\exists y, M(item, y)$ є різними схемами одного і того самого правила висновування. Саме такі два приклади використовуються у наступній формулі: $M(b_1, b_2, \dots, b_n) \rightarrow \exists y, M(b_1, \dots, y, \dots, b_n)$ де M має n постійних аргументів $b_1, b_2, \dots, b_n$ [61].

Іншими словами, змінна x може знаходитися як попереду так і після констант у списку аргументів, зберігаючи, при цьому, структуру правила висновування незмінною. З іншого боку, змінна x є *зв'язаною* що означає, що її область значень складається з операторів прив'язки змінних *деякий x є таким, що*.

Більше того, в логіці Фреге можливе використання таких формулювань як, *кожна функція g являє собою дещо за певних умов* і *деяка функція g являє собою дещо за певних умов*. Це є представленням логіки другого порядку, квантифікація здійснюється як щодо функцій та і щодо об'єктів. Наприклад, якщо аргументи a і b перебувають під

концептуальною функцією g, тоді її форма другого порядку стає наступною: $\forall g, g(a) \leftrightarrow g(b)$.

На додачу, дуже ефективною можливістю в логіці Фреге є використання вкладених кванторів. Квантори можна комбінувати в одній і тій же формулі, щоб представити в символьній формі складні факти чи поняття. Наприклад, такі речення: (*Кожна дівчина любить колір)* і (*Є деякий колір, який подобається кожній дівчині)* точно формалізуються за допомогою двох вкладених кванторів. До Фреге просто не було можливості представити саме ці речення в символьному або формалізованому вигляді. У логіці Фреге ці речення, відповідно, виражаються наступним чином із використанням сучасних символів [61]:

$(\forall a)(\exists b)\ (girl(a) \rightarrow (color(b) \wedge Likes(a, b)))$ [61]
$(\exists b)(\forall a)\ (color(b) \wedge (girl(a) \rightarrow Likes(a, b))$ [61]

Потужна виразність ЛПП відкрила можливість дуже точно представити численні визначення математики з використанням вкладених кванторів. В численні ми часто згадуємо типовий приклад, щодо різниці між 1) неперервністю функції та 2) рівномірною неперервністю функції. Ці два визначення можна точно представити наступним чином, використовуючи вкладені квантори та сучасні символи логіки Фреге [61]:

Неперервність g: $(\forall\varepsilon)(\forall m)(\exists d)(\forall n)(|m-n|<d \rightarrow |g(m)-g(n)|<\varepsilon)$ [61]

Рівномірна неперервність g: $(\forall\varepsilon)(\exists d)(\forall m)(\forall n)(|m-n|<d \rightarrow |g(m)-g(n)|<\varepsilon)$ [61]

Повнота та *коректність* є фундаментальними якостями Фреге, які заслуговують уваги. Повнота існує в дедуктивній системі, якщо кожен правильний вираз логічно отриманий як висновок інших дедукцій в рамках тієї самої системи. Крім того, повнота означає, що серія дедукцій є виконуваною, якщо висновок є виконуваним. Зверніть увагу, що виконуваність означає, що можна здійснити інтерпретацію компонентів формули в рамках значень "істина-хиба", що робить всю формулу істинною.

І навпаки, коректність означає, що в такій системі є можливо довести тільки правильні вирази. Неформально не існує неправильних результатів, які можна було б отримати з розглянутої системи.

На щастя, логіка Фреге є *коректною* і їй властива *повнота щодо спростовності* (мається на увазі, що з будь-якого *невиконуваного* набору формул можливо отримати хибне значення). У цьому випадку, якщо дана формула ЛПП не має інтерпретації, що робить її істинною, тоді система може отримати *хибу* в якості оцінки цієї формули. Властивість повноти спростовності є ключовою в автоматизованих системах доведення теорем, про які йтиметься в подальшому. Ці дві головні властивості ЛПП були вперше доведені в теоремах Еербрана-Геделя [61].

В наступному підрозділі ми спочатку розглянемо ці основні визначення, а потім зупинимося на *інтерпретації Ковальського*. В даному дослідженні ми дотримуємося точки зору, що ця інтерпретація має фундаментальне значення для успіху логіки в комп'ютерних науках; мова йде про синтаксичні і семантичні відповідності між ЛПП і (логічними) мовами програмування.

1.19.2. Семантична форма логіки предикатів

Завдяки інтерпретації логіки предикатів Ковальським показано, як цей тип логіки можна розглядати в якості мови програмування. Цю інтерпретацію можна семантично та синтаксично *представити* в термінах мови програмування. Наступний підрозділ представляє це тлумачення разом і подальшим дослідженням правила Робінсона.

Відповідність між логікою предикатів та інтерпретацією Ковальського

В своїй праці [66], Р. Ковальський інтерпретує логіку предикатів як мову програмування, засновану на імплікації $(A_1 \wedge A_2 \wedge ... \wedge A_n) \rightarrow B$.. Це є описом процесдури, де B позначає назву процедури, а $A_1 to\ A_n$ - це звернення до процедури або тіло процедури.

Така інтерпретація вважає вирішення задачі еквівалентним до доведення теореми. У цьому контексті доведення - це набір обчислень,

згенерований механізмом доведення теорем, який виконує програму, представлену в аксіомах.

Інтерпретація Ковальського базується на *диз'юнктивній формі*, що може представляти собою вираз або речення, що містить набір диз'юнктів. Наступна імплікація символічно представляє диз'юнкт як два набори літералів: [61]:

$$A_1, A_2, \dots, A_n \rightarrow B_1, B_2, \dots, B_m \quad (1)$$

Атомарна формула є предикатом j термінів формули $H(a_1, a_2, \dots, a_j)$. Кожен термін є змінною *або* функцією з багатьма термінами як параметрами. Зверніть увагу, що символи функцій, предикатів та термінів є унікальними [61].

Семантично набір речень можна розглядати як кон'юнкцію $C_1 \wedge C_2 \wedge \dots \wedge C_n$, а кожен диз'юнкт, що має структуру (1) зі t змінними $x_1, x_2, \dots, x_t$ є імплікацією під квантором спільності. Цей диз'юнкт представлений наступним чином: $\forall x_i, i = 1 \dots t: A_1 \wedge A_2 \wedge \dots \wedge A_n \rightarrow B_1 \vee B_2 \vee \dots, \vee B_m$[61].

Зверніть увагу, що нейтральний елемент у лівій частині імплікації є істинним, а хибний знаходиться у правій частині. Цей факт стане в нагоді нижче по тексту, коли буде необхідно здійснити аналіз різних прикладів диз'юнктивних формул [61].

Ця форма може бути записана еквівалентно з використанням диз'юнкції термінів, оскільки $(p \to q) \leftrightarrow (\neg p \vee q)$. Кон'юнктивна нормальна форма - це кон'юнкція диз'юнкцій [61].

Диз'юнктивна форма може мати відповідність з природною мовою, яка виражає (частково) людську думку відповідно до того, що Робінсон вже ефективно представив щодо принципу резолюцій.

Раніше в тексті ми подавали синтаксис мови, щоб представити форму, в якій логічний вираз може бути записаний фаховою мовою логіки предикатів. Значення такого виразу або його семантика з використанням термінології мови програмування є ще одним важливим аспектом у розумінні цієї фахової мови. Саме це є метою наступного підрозділу, в якому ми спробуємо скласти список відповідностей між кожною формою виразів логіки предикатів і особливостями їх реалізації мовою логічного програмування.

Ці вирази записані у вигляді форм диз'юнктів Горна, які розробив А. Горн у 1951 році [68], і які посіли центральне місце в логічних мовах програмування.

Поєднання логіки предикатів, диз'юнктів Горна та семантики програмування

У цьому підрозділі ми докладно описуємо всі можливості застосування диз'юнктивної форми, представленої у диз'юнктах Горна, що використовується в комп'ютерному програмуванні. Це важливо для

розуміння деяких функціональних можливостей Пролог, які будуть представлені в подальшому.

Диз'юнкт називається *диз'юнктом Горна* якщо в його висновку не більше одного позитивного літералу. Іншими словами, в диз'юнктивній формі може щонайбільше бути один позитивний літерал. Цим описується фахова мова логіки предикатів, в якій виникають рівно чотири види диз'юнктів Горна.

Ці види можуть бути семантично інтерпретовані як мова програмування та викликають зацікавлення в двох контекстах. Перший пов'язаний з теоретичною *відповідністю* між диз'юнктами Горна в фаховій мові першого порядку та мовою програмування. Другий - це значний внесок у розробку мови логічного програмування Пролог.
Детальніші розвідки щодо Прологу буде представлено у наступному підрозділі.
У наступних пунктах наведено опис видів диз'юнктів Горна[61]:

1. *Нульовий диз'юнкт*: літерали не вводяться. Це момент зупинки або момент виконання мети. Це пустотіла процедура [61].
2. Форма $\boxed{\rightarrow C,}$ або $\boxed{true \rightarrow C}$ або $\boxed{\neg true \vee C}$: Вона представляє факт С. Це "безтілесна" процедура. *Істина* є нейтральним значенням у кон'юнктивній формі. [61].
3. Форма$(L_1 \wedge L_2 \wedge \ldots \wedge L_t) \rightarrow;$ або$(L_1 \wedge L_2 \wedge \ldots \wedge L_t) \rightarrow false.$ Це інтерпретується як інструкція виконання звернення $L_1, L_2, \ldots, L_t$. Основна процедура не має імені [61].

4. Форма$(L_1, L_2, \dots, L_t) \rightarrow C$: Це *визначений диз'юнкт*. Правий бік (С) - це ім'я процедури. Множина викликів t - це тіло процедури. [61].

Тепер слід обговорити правило резолюцій щоб краще зрозуміти попередні пункти щодо використання мови Пролог. Мова йде про ефективний спосіб отримати вирішення з використанням диз'юнктивний формул спочатку розглянувши два ідентичні предикати, а потім змінивши змінний термін константою. Насправді операції співставлення/конкретизації є одними з найважливіших обчислювальних функцій Пролог.

Використання правила резолюцій (ПР) в мовах логічного програмування

Інтерес до застосування правила резолюцій в мовах логічного програмування обумовлений, по-перше, тим, що воно було розроблене як конкретний вид обчислень з використанням підмножини логіки першого порядку, що пропонує потужний механічний процес в системах доведення теорем. Правило резолюцій було вдосконалене Робінсоном і посідає важливе місце в розглянутій вище інтерпретації Ковальського.

В основному, правило резолюцій використовується в доведенні теорем у вигляді спростування як правила висновування. Для того, щоб довести правило, береться його заперечення і здійснюється спроба віднайти суперечність щодо цього заперечення. Ця ідея стосується речень сформульованих у наступній (запропонованій Сколемом) формі: $\forall b_1, \dots, \forall b_n, B$, де B - це формула КНФ без кванторів [61].

Зауважимо, що Modus Ponens можна розглядати в пропозиційній логіці як окремий випадок правила резолюцій: $((p \to q) \wedge p) \to q$ або інакше $((\neg p \vee q) \wedge p) \to q$. Одне з правил висновування приймається як загальний випадок. Починаючи з двох диз'юнктів форми (а∧¬ а), отримується новий диз'юнкт[61].

Однак, в рамках ЛПП можливе і більш загальне застосування диз'юнктів. Застосування правила резолюцій у ЛПП базується на *уніфікації*, ключовій техніці цього правила, яка є певним видом операції співставлення. Вона отримує два терміни та інстанціює їхні змінні, щоб поставити їх.

Враховуючи, що $[\forall e, \big(D(e) \to C(e)\big) \wedge D(t)]$, теорема $C(t)$ випливає із застосування правила резолюцій та визначення e через t, то її можна формально записати як $[e: t]C(t)$.

Як правило, в такому виді запису, квантори зникають, що в нашому випадку дає: $\big((\neg D(e) \vee C(e)\big) \wedge D(t)) \to C(t)$[61].

Ідея отримання *факторів* під час впровадження механізму уніфікації дає змогу зменшити надмірність термінів.

Наприклад, $R(e) \vee \neg(S(h(e), b)) \vee R\big(g(k)\big)$ факторизується наступним чином: $R\big(g(k)\big) \vee \neg(S\big(h\big(g(k)\big), b\big)$. В новій формі термін R має меншу надмірність.

Цей принцип процедурної семантики використовується для створення подальших процедур, починаючи з першого основного звернення. Фактично, припустимо, що ми маємо:

1) D_jщо відповідає мітці процедури C
2) (Підстановочна) функція h
3) Дві імплікації $(D_1, D_2, \dots, D_n) \rightarrow D$ та $(C_1, C_2, \dots, C_m) \rightarrow C$.

Виводиться наступна форма:

$(D_1, D_2, \dots, D_{j-1}, C_1, C_2, \dots, C_m, D_{j+1}, \dots, D_n)h$ [61].

Правило резолюцій у поєднанні з формами диз'юнктів Горна відіграє вирішальну роль в обчислювальній логіці та є особливо корисним в автоматизованих системах доведення теорем.

Зверніть увагу, що використання диз'юнктів Горна є ефективним щонайменше з наступних причин:

1) Результативний диз'юнкт двох диз'юнктів Горна є також диз'юнктом Горна.
2) При наявності цільового твердження і визначеного твердження, резольвент також є цільовим твердження.

Девіс і Патнем вже вводили правило резолюцій у 1960 році, але, на жаль, їх алгоритм був розроблений для вичерпної інстанціації виразів, що спричинило експоненціальне зростання. Рішення Робінсона дало більш ефективний результат при використанні алгоритму уніфікації, особливо тому, що інстанціаційний процес був активним під час здійснення доведення до моменту виявлення суперечності. Цей *осучаснений* процес дозволяє уникнути будь-якого потенційного експоненціального зростання або того, що іноді називають *комбінаторним вибухом*.

Таким чином, переглянуте Робінсоном правило резолюцій, засноване на логіці першого порядку Фреге, повністю сформувало теоретичні основи для створення мови логічного програмування Пролог. У цьому ж зв'язку ми показали основні технічні кроки, які дозволили ефективно застосовувати логіку Фреге в програмуванні.

1.20. Теорія типів Рассела: від парадоксу до обчислень

Спочатку Рассел і Вайтгед запропонували теорію типів як математичну модель для вирішення відомого парадоксу, що був виявлений в роботі Фреге, який намагався обґрунтувати програму логіцизму в "Численні понять". У цьому підрозділі ми пояснюємо, чому ця теорія є дуже впливовою в комп’ютерних науках. Ми показуємо, як ця теорія посприяла, серед іншого, впровадженню лямбда-числення, яке є ключовим у парадигмі мов функціонального програмування. Більше того, ми відзначаємо важливість відповідності Каррі-Говарда [69], і

обговорюємо, яким чином її досягнути, доводячи ізоморфізм між логічними структурами і об’єктами теорії типів та програмуванням.

В цілому, цей підрозділ, в основу якого покладено наше дослідження [70], покликаний показати фундаментальний внесок теорії типів Рассела в розвиток мов програмування і підкреслити роль відповідності Каррі-Говарда для автоматизованих засобів доведення теорем.

Крім того, цей підрозділ опосередковано показує, як філософська теорія, метою якої було доведення позиції логіцизму, посприяла величезному реальному прогресу в галузі комп’ютерних наук. Цей приклад є аргументом "за" у класичній дискусії щодо практичного використання та застосовності філософії.

1.20.1. Місток між "Численням понять" та "Principia Mathematica"

Виявивши парадокс у роботі Фреге, Б. Рассел та А. Вайтгед запропонували нову математичну модель, яка повинна була подолати згаданий парадокс і тим самим довести правильність позиції логіцизму. Ця програма зазнала невдачі після публікації *теорем про неповноту* [17, p. 3] Геделем у 1930 році. Незважаючи на невдачу логіцизму, розвиток комп’ютерних наук в подальшому, підтвердив значимість теорії типів у багатьох областях комп’ютерних наук, особливо в розробці мов програмування. Гілліс з цього приводу пише наступне:

> Ієрархія типів до тепер відіграє важливу роль в основах теорії множин, в той час як сильна типізація - ні. Виявилося, що вона може добре функціонувати зі змінними, котрі охоплюють множини будь-якого типу. Отже, найбільший внесок Рассел зробив у розвиток мов програмування! (Гілліс) [71, p. 595].

Теорія типів широко використовується в логіці, математиці та комп’ютерних науках. Ми показуємо, як можливо програмувати, використовуючи логічні об'єкти і яким чином це дозволяє краще зрозуміти обчислювальні процеси у логічному вимірі. Точніше, ця теорія використовує логічні засоби для налагодження міркувань щодо програм. Наприклад, вона надає велику підтримку в розробці компіляторів мов, які допомагають створювати більш коректні та ефективні програми. Отже, такі програми врешті є досить надійними та безпечними завдяки виявленню помилок перед виконанням таких програм [70].

Після опису теорії типів ми відстоюємо численні застосування цієї теорії в комп’ютерних науках, особливо в розробці мов програмування.

1.20.2. Теорія типів: впливовий підхід в комп’ютерних науках

У цій теорії об’єкти ієрархічно представлені шляхом присвоєння кожному об’єкту числа, що представляє його тип. Наприклад, числа присвоюються типу 0, множини чисел - типу 1, множини множин чисел - типу 2 тощо. [70].

Цей вид організації використовує принцип об'єднання, згідно з яким множини утворюються виключно над уже утвореними об'єктами. Таким чином, якщо береться множина типів *i*, її елементи повинні бути одного типу і належати до типу меншого ніж *i*. Отже, тип такої множини неодмінно відрізняється від типу її елементів, і відома множина Рассела, що містить "усі множини, які не є елементами їх самих" [70] не має місця в цій системі, що свідчить про те, що згаданий парадокс (очевидно) подолано [70].

Виявляється, що ідея розглядати об’єкти як різні типи виявилася дуже корисною в мовах програмування. Коли вираз пропонується людині, зазвичай легко зіставити кожен об’єкт із правильним типом. Однак, коли мова йде про комп'ютери, встановлення такої відповідності стає дуже складним через відсутність у них інтуїції, і в комп'ютерному нетипізованому вирішенні часто з'являються неправильні типи. Типова помилка виникає при заміні цілого числа на дійсне число, що є наслідком проблеми точності. Інший приклад може стосуватися, що більш важливо, заміни адреси на вміст простору пам'яті, що повертається функцією. У цьому контексті типізована мова

програмування контролює відповідність типу, наприклад, між об'єктом повернення та параметрами функції, які допомагають уникнути неправильного призначення типів. Це складає важливу мету будь-якої системи типів, яка підсилює програми, змушуючи їх застосовувати заздалегідь визначені правила типів. Зверніть увагу, що ці правила є моментом розробки мови програмування, що гарантує, з одного боку, сумісність між типом даних та його фізичним представленням в пам'яті, а також цілісністю та виконанням коду, з іншого боку [70].

Таким чином, теорію типів можна розглядати як процес приведення даних та виразів до певних типів типи та керування їх взаємозв'язками щодо інтерпретації типів. Це свого роду алгоритми перевірки, вже розроблені в компіляторах типізованих мов програмування. Відповідне визначення системи типів можливо подати наступним чином:

> Система типів - це форма контекстно-залежної граматики, яка накладає обмеження на формування програм, щоб гарантувати неможливість виникнення значного класу помилок, що є наслідком неправильної інтерпретації значень. Прикладами таких помилок є: застосування функції для цілих чисел до булевого аргументу; розгляд цілого числа як покажчика на структуру даних або область виконуваного коду [72, p. 220].

В тому самому джерелі є і інший опис цієї теорії: «Теорія типів виникла як уніфікуюча концептуальна основа для проектування, аналізу та реалізації мов програмування» [72, p. 220].

Іншими словами автори вказують, що:

> Теорія типів допомагає прояснити такі тонкощі як абстракція даних, поліморфізм та успадкування. Вона забезпечує основу для розробки логіки поведінки програми, що є важливим, коли йдеться про програми. У ній пропонуються нові методи впровадження компіляторів, які можуть покращити ефективність та цілісність коду [72, p. 220].

Цілком можливо розглядати теорію типів як область, яка формально вивчає системи типів, реалізовані в мовах програмування. Її використовують як основний інструмент при розробці та реалізації мов програмування [73, p. 516]. Вона також відіграє фундаментальну роль при виборі із багатьох парадигм програмування, таких як абстракція, успадкування та поліморфізм [74, p. 2].

1.20.3. Теорія типів у витоків мов функціонального програмування

Важливо бачити тісний зв'язок між теорією типів та мовами функціонального програмування. Роль теорії типів очевидна в основах мов функціонального програмування, таких як Lisp та ML. У цьому ключі Черч, розробкою нової моделі теорії типів розпочав традицію розглядати концепцію "програмування за функціями" [70] основною частиною дизайну будь-якої функціональної мови. Такі мови, в основах яких лежать системи типів, є дуже корисними для аналізу різних мов програмування.

Насправді, система позначень лямбда-числення [75, p. 60] розроблена А. Черчем у 1932–1933 рр. сьогодні вважається частиною теорії типів. У цій системі позначень найпростішим об’єктом є функція, а не множина, як у Рассела та Вайтгеда. Лямбда-числення швидко набуло визнання в якості семантичної моделі для функціональних мов та доведення теорем.

У системі позначень лямбда-числення основні об’єкти називають термінами, які можуть бути змінними, функціями або додатками.

Терміні повинні відповідати умовам типу, що встановлені правилами типів, для того щоб їх можна було вважати коректними. Функція $f: t_1 \to t_2$ показує, що тип вводу t_1 і тип виводуt_2 є єдиними коректними правилами типів [70]. Теорія лямбда-числення постала основою для розвитку такої мови програмування як Lisp, а також інших

функціональних мов. У цій парадигмі програмування ми, як правило, розрізняємо нетипізовану (яка зазвичай неявно типізована) функціональну мову та типізовану мову. Алгол - приклад типізованої мови, що змушує об'єкти програмування (змінні, функції тощо) бути чітко віднесеними до конкретного типу. Таким чином, теорія типів забезпечує більшу точність як щодо типу даних, так і щодо областей та кооблстей функцій. Лямбда-числення, яке посприяло проектуванню функціональних мов, можна розглядати як розширення теорії типів Рассела. В таких мовах, що використовують поняття функцій, які підлягають оцінці, множини виступають одним із типів наряду з іншими. Слід зауважити, що робота з типами замість множин є значно продуктивнішою. Наприклад, різні типи абстрактних структур даних (графіки, дерева, зв’язані списки) можна легко визначити та розглядати як типи даних [70]. У 1958 році Маккарті розробив "Лісп", одну з найперших і найбільш використовуваних мов функціонального програмування, в якій атоми (по іншому, терміни), списки та функції є найпростішими об'єктами. Ця мова характеризується набором вдосконалених можливостей, таких, наприклад, як принцип динамічного типу. Вона стала популярною не лише в галузі штучного інтелекту, а й у розробці багатьох інших мов програмування. Наприклад, ACL2 (A Computational Logic for Applicative Common Lisp) розроблена на основі Common Lisp як мови проектування та вводу. Насправді ACL2 мала великий успіх в автоматизованих системах міркувань вцілому, у формальній верифікації зокрема. ML (скор. від англ. Meta Language -

"мета-мова") є ще одним важливим прикладом мови, що походить від мови "Лісп", має в основі сильну систему типів і широко використовується при розробці формалізованих мов та систем доведення теорем. Ця система, яка використовує поліморфний підхід, гарантує високий рівень безпеки типів [76, p. 353].

1.20.4. Відповідність Каррі-Говарда між логікою та типами

У цьому контексті ми розглядаємо відповідність Каррі-Говарда як головне досягнення, що поєднує логіку і теорію типів. Його основний ізоморфізм демонструє еквівалентність концептів програми та доведення.. У конструктивному або інтуїціоністському сенсі можна використовувати доведення певного об’єкта як програму, яка створює екземпляри цього об’єкта. Таким чином, доведення висловлювання *P* можна розглядати як приклад, що належить до типу *P*. Цей результат відповідності, в основному використовуваний при автоматизованому доведенні теорем, показує еквівалентність між типом і висловлюванням, між доведенням та коректно типізованою програмою, а також між не порожнім типом і істиною. Таблиця 1, представлена нижче, містить інші еквівалентності, пов'язані з цією відповідністю [70].

type theory		*logic*	
τ	type	ϕ	proposition
τ	inhabited type	ϕ	theorem
e	well-typed program	π	proof
$\rightarrow$	function space	$\rightarrow$	implication
$*$	product	$\wedge$	conjunction
$+$	sum	$\vee$	disjunction
$\forall$	type quantifier	$\forall$	2nd order quantifier
B	inhabited type	$\top$	truth
void	uninhabited type	$\bot$	falsity

Таблиця 1: *Відповідність між теорією типів та логікою. Джерело: Propositions as Types, Curry-Howard Isomorphism; Klaus Ostermann Aarhus University; Educational slides.*

Прикладом розширенням лямбда-числення в область систем доведення теорем є система *Coq*, яка використовує логічну структуру, що називається *числення індуктивних конструкцій*. Це передбачає, як уже згадувалося раніше, що висловлювання є еквівалентом типу. Інші системи доведення, такі як HOL, Twelf та Agda, можна назвати системами, що забезпечують конструктивні доведення правильності та верифікації систем, а також доведення математичних висловлювань. [70].

Крім того, ми наголошуємо, що багато систем автоматизованого доведення теорем, підтверджують успішне використання первинної теорії Рассела, яка посприяла розширення теорії простих типів Черча. Як наголошується в [10, с.9], розповсюдження принципу *висловлювання як тип* і зв'язків між логікою та системами типів складають ядро значущості праці Рассела. Такі концепти відкривають шлях для

глибоких обчислювальних інтерпретацій логіки. Це точно висловлено в наступному фрагменті: "Цим інтелектуальним напрацюванням відповідає елегантна обчислювальна та інформаційна технологія, яка поєднала в собі мови програмування, інтерактивні засоби доведення, засоби перевірки моделей і бази даних формалізованих знань" [77, p. 28].

Наступний підрозділ стосуватиметься ролі теорії типів у ключовій парадигмі мов програмування: об'єктно-орієнтовані мови.

1.20.5. В який момент теорія типів вплинула на об’єктно-орієнтовану парадигму

Іншим важливим досягненням, на яке вплинула теорія типів, є парадигма об'єктно-орієнтованого програмування. Ми вважаємо, що такі принципи, як поліморфізм, абстракція даних та успадкування, були створені безпосередньо на основі розробок теорії типів. Поліморфні типи стосуються значень, які можуть приймати різні типи залежно від обраних операцій, а поліморфні функції отримують параметри, які можуть мати різні типи (наприклад, ціле число, дійсне тощо). Фактично, значення може мати параметричний поліморфізм або поліморфізм включення, приймаючи можливість мати декілька типів. У параметричному поліморфізмі функції мають явні або неявні параметри типу; в поліморфізмі включення об’єкти пов’язані з багатьма інклюзивними класами [73, p. 474]. Принцип абстрагування даних в об’єктно-орієнтованій парадигмі стосується створення та організації складних структур даних у "чорній скрині", що надає доступ ззовні лише

до найнеобхіднішої інформацію, а всю іншу інформацію приховує. Принцип успадкування, особливий вид поліморфізму, дозволяє підкласам (класам нижчого порядку) успадковувати властивості та функції від суміжних супер класів (класів вищого порядку). Отже, об’єктно-орієнтовані мови, об’єднавши можливі способи керування та обробки, запропоновані цими трьома принципами (типи, абстракція даних та поліморфізм), мають безпосередній стосунок до основних понять в теорії типів [70].

У цьому контексті поняття *обмеженої квантифікації* корисним чином втручається в процес успадкування і може бути точно визначено наступним чином [73, p. 480]:

“ Тип :: = ... | QuantifiedType

QuantifiedType :: =

∀A. Type | Universal Quantification

∃A. Type | Existential Quantification

∀A⊆Type. Type | ∃A⊆Type. Type Bounded Quantification” [73, p. 480] [70].

Це визначення показує, як систему позначення лямбда-числення можна посилити обмеженою квантифікацією, яка зв'язує параметричний поліморфізм із механізмом підтипування за допомогою кванторів для ряду підтипів. Таким чином, типи, створені за допомогою обмеженої

квантифікації, вважаються супертипами, які можуть успадкувати інші підтипи.

1.21. Висновок

Цей розділ був розбитий на три основні частини, присвячені найбільш впливовим, на нашу думку, логічним складовим комп'ютерних наук. В основу кожної із частин відповідно було покладено праці Буля, Фреге, Рассела та Вайтгеда.

У першій частині ми показали, як Булева логіка, що бере свій початок з алгебри логіки Буля у 19 столітті, істотно вплинула на різні галузі комп'ютерних наук. Ми представили алгебру Буля як сукупність законів мислення. Це передбачає маніпуляції з думками, представленими в людською мовою, за допомогою алгебраїчного механізму і міркувань, які оперують символами. Було підкреслено, що ці символи класифікуються за своїм значенням для виконання логічних операцій. Зокрема, будь-яке пропозиційне числення можливо записати за допомогою Булевої логіки.

Провівши формальне розмежування між початковою логікою Буля та Булевою алгеброю, ми встановили та дослідили основні застосування Булевої алгебри в комп'ютерних науках. Зокрема, ми розглянули проектування комбінаційних схем та застосунки в криптографії як дуже показовий успіх практичного використання Булевих функцій в комп'ютерних науках. Крім того, ми підкреслили,

що завдяки схемам VLSI з Булевими функціями в основі, апаратне проектування у 70-х роках вийшло на новий рівень комплексного розвитку.

У зв'язку з цим, ми хотіли підтвердити, що теоретичні аспекти розвитку Булевої алгебри та прогрес цієї логіки, розширений і реалізований Джевонсом, Пірсом і Шредером, зіграли вирішальну роль у розвитку логіки Фреге. Крім того, та сама алгебра Буля має першочергове значення в статистиці та теорії множин. Наприклад, Булева алгебра знаходиться у відповідності із алгеброю множин. На додачу, поле множини ізоморфне Булевій алгебрі (з використанням доведення Стоуна) [56,61].

Більше того, відносно інших теоретичних впливів алгебри Буля, ми відзначили, що область Булевої схеми включає теоретичний зв'язок між складністю часу та складністю схеми, а перша виявлена NP-проблема є, по суті, задачею здійсненності Булевих формул (SAT), яка вважається референційною проблемою теоретичної комп’ютерних наук [56].

Другим завданням даного розділу було встановити основні причини значущості логіки Фреге для комп’ютерних наук. Окремі причини є наступними: 1) високий рівень виразності логіки предикатів 2) відповідність між логікою предикатів і інтерпретацією Ковальського 3) семантичні еквівалентності між диз'юнктами Горна та логічним

програмуванням 4) важливість принципу резолюцій в мовах логічного програмування.

Останнє завданням цього розділу полягало у з'ясуванні значення теорії типів, створеної Расселом і Вайтгедом, для комп'ютерних наук. Ми простежили історичну появу теорії типів, яка з'явилася як інструмент для підтвердження обґрунтованості програми логіцизму, шляхом подолання суперечливої ситуації, виявленої в роботі Фреге про аксіоматичні системи. Потім ми з'ясували елементи практичного застосування та переваги цієї теорії для комп'ютерних наук. В якості прикладу окремих переваг ми запропонували правильність, ефективність та цілісність комп'ютерних програм. Ми докладно показали, як ці елементи допомагають у створенні більш надійного програмного забезпечення, шляхом проведення аналізу програм та виявлення помилкових процесів перед виконанням коду. Вони також допомагають на семантичному рівні проектування компіляторів, надаючи алгоритми перевірки [74, p. 4]. Окрім цього, ми показали роль теорії типів у абстрагуванні даних, поліморфізмі та успадкуванні. Ми також пояснили, як типізовані мови програмування, аналіз та оптимізація програм на основі типів, а також механізми захисту, що мають типізовану підтримку, безпосередньо виграли від застосування теорії типів [74, p. 2].

Наостанок, ми надали опис фундаментальних теоретичних результатів впровадження ізоморфізму Каррі-Говарда. Ми також

приділили особливу увагу тому, як теорія типів запропонувала математичну структуру для розробки типізованого лямбда-числення та інтуїціоністської теорії типів, запропонованої П. Мартіном-Лофом [78].

Розділ 4. Логічні основи штучного інтелекту

Дисципліна штучного інтелекту, яка спрямована на розробку *інтелектуальних* систем, що імітують людський інтелект чи поведінку, широко застосовує теорії та концепції логіки. Сфера застосування логіки є значною, що робить її однією із фундаментальних засад дисципліни штучного інтелекту. Мова йде, наприклад, про такі застосування, як управління автономними інтелектуальними системами, експертними системами та моделями перекладу.

У цьому розділі ми прагнемо показати, що в основі розвитку штучного інтелекту лежать принципи логіки. Перша частина розкриває вплив принципів логічного програмування, що використовуються в мові Пролог, на автоматизоване доведення теорем (АДТ) та на доведення формул за допомогою логічних концептів. Після цього ми представляємо численні успішні програми АДТ як докази логічних основ штучного інтелекту.

У другій частині ми простежуємо різні зв'язки між індуктивним логічним програмуванням (ІЛП) та програмами машинного навчання. Згодом ми з'ясували, яким чином ІЛП може значно вдосконалити багато технік штучного інтелекту, таких як методи пошукових систем, комбінаційні ігрові стратегії та мобільні програми що мають справу з поведінкою користувачів.

Остання частина, більшою мірою стосується, теорії аргументації. Ми обговорюємо немонотонну логіку та розкриваємо її використання як логіки виправданих висновків (defeasible logic) у штучному інтелекті, особливо щодо теорії аргументації. Крім того, ми показуємо різні застосування теорії аргументації в області штучного інтелекту [21,79].

1.22. Вплив мови логіки на автоматизоване доведення теорем

Пролог (або **Про**грамування в **Лог**іці) - це мова, що безпосередньо ґрунтується на логіці предикатів. Це яскравий приклад ролі логіки Фреге в комп'ютерних науках. Ця мова була розроблена в 1972 році [65] з метою створення декларативної мови. Програма Пролог виражається з використанням фактів та правил. Запит у Пролозі має форму запитання для формування послідовності обчислень за цими фактами та правилами. Це можна розглядати як формули, записані як диз'юнкти Горна, які вважаються істинними та відповідають засновкам у ЛПП. Використовуючи правила резолюцій та уніфікації, Пролог створює дерево пошуку, відповідаючи на запит. Процедура семантики Пролог прагне реалізувати низку цілей, формуючи список істинності (або хибності) цих цілей та відповідних їм уніфікацій (або інстанційювань). Пролог використовує автоматичний механізм зворотного відстеження, щоб знаходити все більше і більше вирішень. Це наріжна ідея у розвитку логіки Фреге, яка просувалася від мети створення формальної системи, здатної перевірити істини, до іншої - створення механічного процесу, здатного генерувати нові зростаючі істини із сукупності умов і правил

висновування. Цей підрозділ в своїй основі має матеріали нашої публікації [61].

1.22.1. Застосування Пролог в штучному інтелекті

У Пролозі визначене твердження поводиться як процедура редукції цілей, і його синтаксис виражається у оберненій формі порівняно з формою Ковальського. Насправді твердження $B \leftarrow (A_1 \wedge A_2 \wedge ... \wedge A_n)$ означає "довести B, $(A_1 \wedge A_2 \wedge ... \wedge A_n)$ має бути доведено ". Це підкреслює той факт, що для того, щоб отримати результат B, літерали $(A_1 \wedge A_2 \wedge ... \wedge A_n)$ повинні бути попередньо виконані. Ця форма еквівалентна семантичній інтерпретації Ковальського де B відповідає імені процедури а A_1 відповідає A_n на виклики процедури. Той самий диз'юнкт Горна можливо записати мовою Пролог наступним чином [61]:

B: - A1, A2,..., A_n. [61]

Як уже згадувалося, доказ спростуванням - це основна ідея функціонування мови Пролог. Наприклад, коли цільове твердження C має бути доведено в програмі P, Пролог вставляє заперечення C $(\neg C)$ в P і починає віднаходити суперечності, що в результаті дають *хибу* в кінці. Таким чином, Пролог здійснює доведення C спростуванням $\neg C$. Вирішуючи проблему квантора існування в списку позитивних літералів

($\exists x(a \wedge b \wedge ... \wedge k)$), доведення спростуванням ставить квантор спільності над списком від'ємних літералів форми ($\forall x(\neg a \vee \neg b \vee ... \vee \neg k)$) або ($\forall x(false \leftarrow (a \wedge b \wedge ... \wedge k))$) і записується мовою Пролог як ": - a, b,…, k" [61].

У цьому контексті несподіваною є паралель між попереднім екзистенційним диз'юнктом Горна, заміненим квантором спільності, та вихідною системою Фреге, де квантор існування неявно виражається квантором спільності над запереченням твердження.

Ще раз зауважимо, що доведення теорем, що ґрунтується на резолюції, спирається на доведення спростуванням. Це пояснюється тим, що зазвичай ефективніше досягти суперечності, коли до системи додається заперечення речення. Отже, домінуюча кількість автоматизованих систем доведення теорем використовує доведення методом спростування, та інші обчислювальні логічні програми.

У наступному підрозділі докладно описано обчислювальне функціонування мови Пролог після отримання множини засновків, до яких застосовується правило резолюцій з метою доведення конкретного речення.

1.22.2. Логіка предикатів як обчислювальна модель бази знань

У цьому розділі ми демонструємо на прикладах функціонування правила резолюцій у мові Пролог [61]. Також ми надаємо, обчислювальне порівняння на основі ідеї, розробленої Робінсоном [64].

Йдеться про порівняння людського та машинного механізмів обчислення в рамках даної дедуктивної системи. Наприклад, ми демонструємо високу ефективність та актуальність обчислювального аспекту правила резолюцій при використанні їх в комп'ютеризованій системі.

Відповідно до згаданого факту, поміщаючи форму диз'юнктів Горна разом із принципом резолюції / уніфікації в центр обчислювального процесу мови Пролог, ми описуємо доведення формули або алгоритм доведення резолюції наступним чином [61]:

1. Маючи набір теорем, записаних у КНФ як знання (БЗ), і формулу R для доведення, Add not (R) додану до БЗ.
2. Повторювати, допоки не буде вичерпано всі твердження для резолюції:
    - Застосувати операції співставлення-інстанціації до розв'язних тверджень (якщо вони знайдені); додати резольвенту до БЗ.
    - Оголосити формулу R істинною, якщо резольвента має значення NIL; зупинити виконання.
3. Оголосити формулу R хибною. [61]

Зауважте, що попередній алгоритм прийняття рішень Істина / Хиба не дає жодного можливого вирішення. Наступний приклад, заснований на [80], йде далі порівняно із даним алгоритмом, надаючи список всіх можливих вирішень. На початку вводиться БЗ з основною

гіпотезою запиту що містить змінну. Метою процедури є підтвердження гіпотези, а також отримання всіх існуючих вирішень. Рішення віднаходиться кожного разу, коли значення може бути інстанційоване до згаданої змінної. У цьому прикладі запит **Любить (Джон, k)** починає пошук усіх можливостей k і кого Джон любить [61]. Він представлений наступним чином: ∃ k: Любить (Джон, k)

Теореми є наступними:

1. ¬Бачить (Джон, c) ∨ Любить (Джон, c)
2. Бачить (Джон, Ліза)
3. Бачить (m, Джейн)
4. Бачить (n, Мати (n))

Заперечення запиту може бути додане як показано, що дає:

5. ¬Любить (Джон, k) (Зверніть увагу, що ¬ ∃k, P еквівалентно ∀ k: ¬P)

Розв’язуємо 1 і 5 і замінюючи k на c, отримуємо:

6. ¬Бачить (Джон, c) [61]. Той самий підхід можна застосувати до наступного:

Між 2 і 6: c = Ліза. Рішення - "Джон любить Лізу".

Між 3 і 6: m = Джон і c = Джейн. Рішення - "Джон любить Джейн"

Між 4 і 6: n = Іван та c = Мати (n). Рішення: "Джон любить свою матір" [61]:

Попередній приклад показує два важливі моменти. Першим моментом є те, що правило резолюцій використовується виключно в

цьому обчисленні. Другий момент стосується стабільності цього механічного дедуктивного процесу, який залишається абсолютно однаковим навіть у випадку набагато складніших речень. Таким чином, ідея використання комплексних правил за одним і тим же механізмом є дуже зручною для машинно-орієнтованого використання при генеруванні нових дедукцій. Зауважте, що можливо отримати більш ефективні результати, якщо деякі алгоритми пошуку (наприклад пошук з вертанням) реалізуються як допоміжні або доповнюючи процедури до дедуктивної системи [61].

Як зазначив Робінсон [64], орієнтована на людину дедукція зазвичай використовує прості кроки послідовним способом. Кожен із цих кроків легко верифікується людиною. Ця верифікація спирається на засновки, правила висновку та вже обчислені кроки. Інтуїтивно-людський механізм завжди може впоратися із великою кількістю кроків, щоб отримати кінцевий дедуктивний результат, навіть якщо це займає більше часу.

З багатьох причин це простий, але потребуючий численних кроків механізм, абсолютно непридатний для комп'ютерного застосування. Одна з причин полягає в тому, що комп’ютерам бракує інтуїції, яка дозволяє людям швидше *переключатися* між простими правилами БЗ, щоб обирати / виключати правильний / неправильний шлях, що веде до вирішення. Навпаки, коли між *різними* правилами умовиводу використовується значна кількість кроків, дедуктивний механізм запускає *комбінаторний вибух*, що означає низький ступінь

ефективності. Водночас, використання невеликої кількості, але набагато складніших правил висновування, дуже підходить для комп'ютерного застосування. Це є важливою причиною успіху застосування правила резолюцій в комп'ютеризованій системі, яке дозволяє реалізовувати більш ефективні дедукції без появи комбінаторного вибуху. Правило, резолюцій, вдосконалене Ковальським та Кюнером у формі лінійних дедукцій, дало можливість підвищити продуктивність багатьох застосунків, таких як лінійна резолюція у *Единбурзькій лінійній пересічно-структурній* системі доведення теорем" [61]

Загалом, логіку часто використовують для аналізу різних властивостей, пов'язаних, наприклад, з висновуваннями, міркуваннями та дизайном. Зокрема, логіка допомагає у процесі специфікації (логічного) програмування, незалежно від кодування нижчого порядку. Однак, у зв'язку із щойно обговореними, заснованими на знанні моделями, може бути доречним ще більше наголосили на зв'язку між логікою та репрезентацією знання. Наприклад, експерти, такі як Р.К. Мур, виділяють три основні способи застосування логіки в ШІ: 1) "концептуальний інструмент аналізу 2) репрезентація знань і 3) теоретичні основи для мов програмування" [14, p. 849].

Зверніть увагу, що в задачах репрезентації знань, механізмів міркувань та маніпулювання доказами можна використовувати незалежні, відповідні, логічні стратегії та методології.

Моделі репрезентації знань по суті займаються пошуком стратегій та інструментів для репрезентації інформації в комп’ютеризованому середовищі. Ці стратегії зазвичай мають справу з логічними принципами та механізмами, такими як факти, правила, докази тощо. Теми, включаючи логічне програмування, формалізми міркувань, немонотонну логіку, логіку опису та когнітивну логіку, є одними з найбільш пов’язаних із застосунками в рамках репрезентації знання.

В експертних системах, наприклад, вважається, що інтелектуальні моделі принципово ґрунтуються на знаннях, маючи справу із фактами, правилами та машинами висновування (як правило, з мовами програмування) з метою репрезентації та вирішення проблеми експертної системи. Вирішення подібних завдань тісно пов’язане з автоматизованим міркуванням або доведенням теорем, що традиційно застосовує багато логічних інструментів в якості резолюційного підходу. В цьому контексті логіка предикатів першого порядку виступає однією з найбільш вживаних формальних логік. Насправді, як вже було зазначено, ця логіка є потужним формальним інструментом, що забезпечує високий рівень виразності знань. Вона здатна моделювати складні поняття реального життя та забезпечувати механізм висновування для вирішення задач (автоматизоване доведення) або отримання нових знань [14, p. 849] [61].

У наступному підрозділі ми зупинимося на потужних комп'ютерних програмах, які називаються системами автоматизованого доведення теорем (АДТ). Ці системи є одним з найбільш успішних

застосувань ЛПП в комп’ютерних науках. Крім того, доведення спростуванням та принцип резолюцій, про які йшлося раніше, широко представлені як окремі техніки АДТ.

1.22.3. Роль логіки предикатів у автоматизованому доведенні теорем.

Системи АДТ, широко застосовані в комп’ютерних науках, є результатом розвитку логіки предикатів. Ми досліджуємо взаємозв'язок між цими системами та логікою першого порядку перед тим, як перейти до дослідження використання таких програм для вирішення теоретичних та промислових задач. Крім того, ми наводимо короткий перелік успішних систем АДТ в окремих дисциплінах, щоб показати ключову роль логіки у штучному інтелекті [61].

Опис автоматизованого доведення теорем

Системи АДТ намагаються довести припущення, які вважаються істинними або логічно випливають із засновків із використанням правил. Якщо доведення завершено, припущення називають теоремою. Логіка першого порядку є переважаючою логічною мовою, що використовується для представлення припущень та засновків. Для цього також можуть бути використані і інші логічні мови, такі як логіка вищого порядку.

Використання ЛПП (або логіки вищого порядку) становить суттєву потужність систем АДТ. Завдяки своїй високій виразності, ЛПП дозволяє надати точний вираз засновків та правил в системі АДТ. Задачі опрацьовані в такий спосіб набувають точності виразу, а такі принципи, як КНФ, резолюція та уніфікація, систематично застосовуються для пошуку вирішення.

Коли припущення потрапляє до системи АДТ на перевірку, система часто надає деталі доказу припущення, що включає опис кроків, що ведуть до наявного вирішення, за умови, якщо воно існує. Потім користувач може виконати ці кроки, якщо потрібна додаткова інформація для кращого простеження висновку, отриманого із використання правил та засновків. Крім того, для деяких типів програм, детальний опис кроків, викликає більше зацікавлення в контексті задачі, ніж просто рішення у вигляді Так / Ні. Наприклад, коли у стратегічній грі потрібно знати виграшну можливу конфігурацію, наведені кроки, починаючи з початкової конфігурації, являють собою спосіб просування до отримання остаточного виграшного кроку. Очевидно, що цікавішим є отримання цих деталей, ніж проста позитивна відповідь, що виграшне рішення існує. Зауважте, що такий спосіб доведення забезпечує повну алгоритмічну прозорість порівняно з іншими обчислювальними підходами. Така прозорість є перевагою у випадку будь-якої потенційної відповідальності, яку ми повинні покласти на інтелектуальну систему у значенні, яке ми обговорили у другому розділі [61].

Підсумовуючи, системи АДТ загалом вважаються здатними вирішувати доволі складні проблеми за ефективної часової складності. Існують різні типи технік АДТ, але найпопулярнішими є 1) резолюція першого порядку з уніфікацією. 2) елімінація моделі. 3) та уніфікація вищого порядку [61].

Застосунки автоматизованого доведення теорем

Системи АДТ, які в цьому тексті іноді називаються просто *системами доведення*, були успішно використані в різних дисциплінах, особливо в логіці, математиці, комп'ютерних науках та інженерії. Історично системи автоматичного доведення залучалися до доведення математичних припущень. В даний час сфера застосування цих систем поширилася і на інші області, але в основному вони застосовуються для цілей формальних методів при перевірці та проектуванні програмного забезпечення та перевірці апаратного забезпечення. Насправді комерційний інтерес до систем АДТ є значним, оскільки він передбачає зменшення дороговартісних помилок, які з часом можуть виникати у багатьох застосунках. В якості прикладів апаратного застосування систем автоматичного доведення, можна назвати інтегральні схеми в медичному обладнанні, космічних компонентах, робототехніці та атомних електростанціях. [61].

У наступному розділі наведено різні приклади успішних систем автоматичного доведення у галузі математики, програмного забезпечення та обчислювальної техніки.

Пропозиційні системи автоматичного доведення проти систем першого порядку

Як уже зазначалося, логіка першого порядку є набагато виразнішою, ніж пропозиційна логіка. Отже, рівні виразності є пропорційними між пропозиційними системами доведення та системами доведення першого порядку.

У цьому дослідженні ми зосередилися на двох категоріях систем АДТ. Пропозиційна система доведення, яку також називають SAT-розв'язувачем, по суті є заснованим на пропозиційній логіці, розв'язувачем здійсненності булевих формул, що дозволяє записувати проблему в Булевих термінах. Як правило, такі типи розв'язувачів (наприклад, zChaff) демонструють корисність, але вони мають два основних обмеження, пов'язані з використанням Булевих виразів. Вони не завжди виразні, і їх розміри швидко збільшуються. Дійсно, першою визнаною NP-повною проблемою є SAT. Зазвичай для вирішення потрібна експоненціальна часова складність.

На противагу цьому, система доведення першого порядку є набагато виразнішою, ніж АДТ SAT, не тільки завдяки своїм кванторам та предикатним можливостям, а також завдяки правилу резолюцій Робінсона, яке дозволяє здійснювати ефективний механічний пошук етапів у сформованому доведенні. Таким чином, стає можливим природно висловити складні проблеми використовуючи формалізовані експресивні специфікації та потужні обчислювальні принципи. Ці та

інші переваги поставили системи доведення першого порядку серед найбільш зрілих та найдосконаліших [61].

Однак багато задач можливо краще виразити, коли до логічної мови додається рівність. Як наслідок, кілька сучасних систем автоматичного доведення додали рівність для покращення продуктивності. Система автоматизованого доведення теорем "E" є прикладом системи заснованій на рівності. На додаток до систем позначення рівності, деякі системи АДТ отримують більшу потужність інтегруючи різні евристики пошуку, відповідні сучасним програмам машинного навчання. Ця інтеграція надає комп’ютерним системам здатність діяти та вчитися без явного та попереднього програмування.

Як уже згадувалося, логічні системи автоматизованого доведення першого порядку широко використовуються для перевірки програмного забезпечення, особливо для перевірки правильності програм, кодованих багатьма мовами програмування, такими як Ada, Pascal або Java. "*Стенфордський верифікатор Паскаля"* [61], що використовує правило резолюцій є одним із них. Він розроблений Девідом Лукхемом у Стенфордському університеті.

Історична перспектива щодо систем автоматизованого доведення теорем

У цьому пункті ми представляємо перелік систем автоматизованого доведення ЛПП, що історично демонструють успішність їх застосування [81].

- SETHEO: 1989; одна із найстаріших систем доведення; Технічний університет Мюнхена; високопродуктивна; цілеорієнтована модель [61].
- Vampire: 1994; Манчестерський університет; широке коло задач; пристосована для перевірки програмного та апаратного забезпечення; Математичні та міркувальні доведення [61].
- "E": 1998; Мюнхенський університет; висока продуктивність; система доведення ЛПП із рівністю; міркувальні доведення [61].
- EQP: система доведення ЛПП із рівністю; Національна лабораторія Аргонни; доведено припущення Роббінса (1998). Характеризується “асоціативно-комутативними операціями уніфікації та узгодження” [61].
- Waldmeister: 1998; Інститут Макса Планка; "Одинично-еквівалентнісна система доведення ЛПП" [61]. Висока продуктивність часової та просторової складності.

- Prover9: 2004; Національна лабораторія Аргонни; наступник системи доведення ЛПП "Otter"; об'єднаний із системою Mace4 [61].
- SPASS: 2010; система доведення ЛПП із рівністю[61]; Інститут Макса Планка; інтегрує факти реляційних баз даних [61].

Протягом останніх 50 років, для численних застосувань в математиці та інженерії, були розроблені і впроваджені такі логічні системи автоматизованого доведення вищого порядку, як ACL2, Coq, HOL та Nqthm [81] [61].

Наступний пункт даного підрозділу показує велику кількість *галузей* що успішно використовують системи АДТ, завдячуючи логіці предикатів Фреге, яка перетворила ці поширені системи на важливі компоненти комп’ютерних наук.

Коли системи автоматизованого доведення теорем стали успішними?

Спочатку системи АДТ були запропоновані для доведення складних задач математики, а також перевірки програмного та апаратного забезпечення. У цьому пункті наведено перелік вибраних прикладів за дисциплінами, що показують широкий спектр галузей, які

використовують одну або кілька систем АДТ [81]. Зокрема, без *систем доведення* ЛПП стало неможливим обійтись при доведенні багатовартісних важливих проблем [61].

Математика

- Otter: представила докази в квазігрупах та при пошуку мінімальних множин аксіом.
- ICOT: використовується для вирішення деяких задач квазігруп.
- EQP: з допомогою цієї системи було вирішено задачу Роббіна.
- Geometry Expert: використовувалася для розв’язання евклідових геометричних задач.
- NUPRL: було розгорнуто в контексті парадоксу Жерара, а також леми Хігмана [61].

Розробка програмного забезпечення

- *Amphion*: інжиніринг програмного забезпечення на основі БЗ, розроблена для автоматизації підвищення якості та продуктивності [61].
- KIDS: використовуються в алгоритмах планування і розробки кодів на основі специфікацій [61].

Перевірка програмного забезпечення

- PVS: програми планування, які в основному застосовуються в завданнях, пов'язаних з управлінням польотами.
- *KIV:* перевірка програмного забезпечення; операції з графіками; Перевірка кодів мови Пролог за допомогою WAM [70].
- *Key:* об'єктно-орієнтовані перевірки парадигми [61].

Апаратна перевірка

- Bell laboratories: використання системи логіки вищого порядку ЛВП для перевірки промислового обладнання доведенням теорем в інтерактивному середовищі [70].
- Доведення коректності різних кодів для мікропроцесора AMD щодо розподілу чисел з рухомою комою. Їх було протестовано за допомогою системи ACL2. Nqthm здійснив інші перевірки коректності мікропроцесора FM9001 [61].

Ми щойно виокремили успішні використання систем АДТ у дослідженнях, а також у вирішенні реальних життєвих задач. Різні сучасні онлайн-системи АДТ доступні для широкого типу користувачів. TPTP - Thousands of Problems for Theorem Provers (тисячі задач для систем доведення теорем) - типовий приклад інтерфейсу Інтернет-бібліотеки, розробленого в Університеті Маямі, що містить різні програми АДТ [81] [61].

1.23. Удосконалення машинного навчання з використанням індуктивного логічного програмування

У цьому підрозділі, говорячи актуальність індуктивного логічного програмування (ІЛП) у штучному інтелекті ми використовуємо матеріали нашого дослідження вміщеного в [82]. Ця парадигма має справу із програмуванням логіки предикатів в цілому, та індуктивним машинним навчанням зокрема. В основному ми обираємо три особливо успішні застосунки, що використовують метод ІЛП. Ці застосунки мають справу із методами пошукових систем, ігровими стратегіями та розумінням поведінки користувачів у мобільних галузях.

У штучному інтелекті навчання - це пошуковий підхід, що має на меті довести дану гіпотезу (H), дотримуючись набору обмежень або критеріїв. У цьому контексті використовується парадигма машинного навчання, відома як індуктивне логічне програмування (ІЛП). Її використовують для вдосконалення або розвитку структури бази знань за логічними правилами. Таким чином, ІЛП - це реляційне контрольоване машинне навчання з використанням логіки, яке іноді може навчатися, отримуючи лише один навчальний приклад. ІЛП безпосередньо пов'язане із використанням логічного програмування, яке рівномірно репрезентує фон, гіпотезу та приклади. Отже, самі вводи та виводи перетворюються на логічні програми в середовищі ІЛП [82].

У типовій логічній програмі правила відрізняють позитивні та негативні приклади. Як уже згадувалося раніше, методи ІЛП прагнуть

пояснити або вивести приклади з гіпотези H та фонових знань B. Породжена гіпотеза, яка здатна пояснити виключно позитивні приклади, є однією з важливих цілей підходу ІЛП. [82].

Виходячи з визначення Лаврака та Джерскі, програма машинного навчання розуміється у функції "структури її простору, її пошукової стратегії та евристики пошуку" [83, p. 147]. Слід зауважити, що логічні програми визначають простір пошуку в техніці ІЛП.

У цьому підрозділі, як показано в [82], ми представляємо короткий огляд машинного навчання. Потім ми обговорюємо три згадані сучасні програми ІЛП, які виявились дуже успішними в машинному навчанні. Наприкінці цього підрозділу ми подаємо висновок, що включає деякі зауваження та роздуми стосовно розвитку техніки ІЛП.

1.23.1. Огляд індуктивного логічного програмування

В часи ранніх розробок ІЛП як логічного інструменту штучного інтелекту найбільш цитованими були імена таких дослідників-експертів, як Плоткін (1969), Шапіро (1983), Саммут, Банерджі (1986) та Магглтон (1991), серед інших, є найбільш цитованими експертами в цій галузі. Вони першими серед вчених переконалися у перевагах використання ІЛП. Крім того, інші вчені і поняття, представлені в ці роки, істотно вплинули на цю галузь, окремі з них використовуються і до сьогодні. Такими поняттями, зокрема, є "інверсивна резолюція" (Магглтон і Бантайн), "сатурація" (Рувейрол і Пюдже) та "предикатна інвенція"

(Магглтон і Бантайн). Крім того, інші суміжні концепції також впливали на формування сфери машинного навчання, такі як "PAC (probably approximately correct) learnability" [82] впроваджена Веліантом в 1984 році, та пізніше допрацьована Когеном в контексті програм k-local у 1993 році та концепція Джероскі щодо "*ij*-determinate logic programs" [84] [82] представлена у 1993 році.

Інші відповідні концепції імовірнісної логіки можна згадати в якості новіших, порівняно із попередніми, розробок в галузі машинного навчання. До цих концепцій відносять Байєсівські програми логіки (2001), логічні мережі Маркова (2006) та інші ймовірнісні системи ІЛП (ЙІЛП). У цьому контексті такі системи ІЛП, як FOIL (1990), Golem (1990), LINUS (1991), Progol (1995, 2000), TILDE (1997), PRISM (2005) та ProbLog (2007), значною мірою застосовуються в рамках специфічних систем штучного інтелекту, таких як прогнозування структури білка, супутникова діагностика, біохімічна галузь, робототехніка, шахові бази даних та вивчення структур лікарських засобів (препарати від хвороби Альцгеймера, кишкової палички тощо) [84] [82].

Ми спостерігаємо, що здатність до нарощення в системах машинного навчання є основною властивістю таких систем. Ця властивість природно присутня в системі, керованій моделями, що використовує побудову правил, розробку предикатів та деякі інші

концепти логічного програмування, які забезпечують нарощення її знань.

Наприклад, застосунок CONFICIUS, впроваджений Когеном у 1970-х роках, використовував цю властивість нарощення і застосовував її у багатьох галузях. Ідея полягає в тому, що нещодавно вивчена поняття безпосередньо використовується як ввід для створення нових інших понять [82]. Система *Марвін*, яка була створена Саммутом у 1980-х роках і являє собою розширення CONFICIUS, продовжувала працювати в цьому напрямку, значною мірою, виграла від використання диз'юнктів Горна у логічній програмі. Ці диз'юнкти дозволяють дуже ефективно узагальнювати поняття в навчальному середовищі. Зокрема, загальний опис поняття, записаного мовою Пролог, використовується для перевірки хибності чи істинності інстанцій (прикладів понять). Як правило, цей крок виконується під час конструювання процесу навчання при отриманні застосунком позитивних і негативних прикладів. Інші поняття, що мають стосунок до індукції диз'юнктів Горна, також використовуються у багатьох системах. Серед інших прикладів можна навести концепцію категоризації, що була запропонована Бантаймом [84,85] [82].

1.23.2. В чому причина успішності ІЛП в машинному навчанні?

У штучному інтелекті можна зустріти багато, нещодавно впроваджених і успішних застосунків ІЛП. В якості прикладів можна

навести виконання сценаріїв, функціонал табличних редакторів та розумних т'юторів. Навчання на прикладах є основною метою застосунків для роботи з електронними таблицями. Це дозволяє користувачеві уникнути вичерпного повторення завдань, особливо коли знаннями одночасно діляться користувач і середовище, що розглядається. Більше того, вивчення прикладів є особливо корисним у цьому випадку через відсутність у користувача досвіду щодо скриптової мови, реалізованої в цих системах. Прикладом такої технології є Microsoft Excel 2013 із використанням функції Flash Fill, яка автоматично виводить програми обробки рядків із декількох представлених прикладів. Цей ІЛП застосунок, серед інших, зробив можливим вивчення комплексних програм за допомогою невеликої кількості прикладів і виявився успішним завдяки їх загальності та гнучкості. Ми можемо згадати також і інші пов’язані застосунки, розроблені для специфічних роботів, для використання на смартфонах та настільних комп'ютерах, наприклад, виконання сценаріїв PowerShell [82].

Інший застосунок, що працює на основі прикладів, має стосунок до освіти та до будь-якого процесу навчання. У цій галузі повторювані та організовані завдання можуть бути автоматично змодельовані за допомогою підходу ІЛП на основі навчання за прикладами. Завдання, подібні до зворотного зв'язку, вправам і віднаходженню рішень, доцільно обробляти за допомогою цього методу навчання. Такий тип

автоматичного підходу цілком придатний до використання в галузях, математики, природничих наук та інжинірингу. Для таких субдоменів, як програмування, теорія чисел, теорія мови та алгебра, застосування генерації вправ / рішень із використанням парадигми ІЛП на основі прикладів є прийнятним [82].

Удосконалення технік пошукових систем за допомогою ІЛП

Одним з успішних прикладів ІЛП є "модель машинного навчання англійської мови" [82], що була створена у Пролозі [86]. Вибір мови Пролог є актуальним в тому сенсі, що вона ефективно генерує нові зміни слів, які йдуть далі, ніж просто класифікація або рішення щодо того, чи є даний парний термін-кандидат дійсним чи ні. У цьому застосунку система вивчає нові правила, які обчислюють коректні зміни. Таким чином, використання логічних правил зменшило традиційну високу складність генерації таких змін. Отже, ця модель ІЛП виявилася доречною щодо модулів розширення запитів щодо її здатності виводити відповідних кандидатів.

Більше того, застосунок генерував факти різних функцій, які використовували всі позитивні та негативні зміни. Ці факти містились у своєрідних наборах фонових знань. Іншим критерієм, що підвищив ефективність цієї системи, є її детерміністичний підхід, який дає рівно один результат для даного кандидата або вводу. Ця перевага характерна для таких систем ІЛП, як Aleph та FOIL. Згадана реалізація англомовної моделі використовувала розширення "Aleph 5 і FOIL 6.4" [86].

Виконання цієї програми у Aleph та FOIL показало, що Aleph демонструє загальність, а FOIL особливість в сенсі точності. Крім того, використані правила продемонстрували задовільне узагальнення з причини подібності між групою тестових прикладів та групою навчальних прикладів [82].

Десятки слів в якості вхідних даних кандидатів, що генерують правила, були протестовані по лексичним і граматичним категоріям (іменники, дієслова і т.д.). Підхід ІЛП, використаний у цьому застосунку, виявився прийнятним у процесі генерації нових змін англійської мови. Було отримано високу точність прогнозування, і можливість створити правильні зміни [86].

Інший успішний приклад ігрової стратегії ІЛП подано в наступному розділі.

ІЛП в оптимізації комбінаторних ігрових стратегій

В контексті категорії комбінаторних ігор було показано, що повна виграшна стратегія ІЛП можлива і, фактично, її було практично реалізовано [87]. У цій реалізації використано класифікатор для *попереднього гравця* (P-позиції), що дозволяє обрати принаймні один оптимізований мінімаксний виграшний хід. Ця можливість сама по собі є хорошою перевагою в тому сенсі, що дозволяє уникнути вивчення

виграшних ходів, враховуючи різні ігрові стани. Виграшна стратегія отримується завдяки комбінації класифікатора та генератора ходів.

Дивно, що використаний метод ІЛП реалізував криві точності прогнозування, коли прийшов час вивчати позиції для виграшних стратегій. Крім того, ця точність була застосована до *класу* комбінаторних ігор, і 100 % точний прогноз отримано для шести комбінаторних ігор за допомогою 26 випадкових прикладів та 10 вибіркових випробувань [87].

Більше того, порівняно з іншими підходами, було помічено, що криві точності прогнозування щодо знання позицій виграшних стратегії гри Nim були менш ефективними в трьох основних техніках, відмінних від підходу ІЛП. Насправді, жодна з технік, що використовуються в методі опорних векторів, штучних нейронних мережах і міркуваннях на основі прецедентів (CBR) не досягла 100 % точності прогнозування навіть на 200 прикладах [87].

Nim - це комбінаторна гра, де двоє гравців по черзі вибирають предмети з безлічі груп. Єдине застосоване правило зобов'язує гравців брати принаймні один предмет в свою чергу, при цьому вони можуть вибрати більше одного об'єкта за один хід, якщо всі вони знаходяться в одній групі. Перемагає гравець, який взяв останній предмет.

Гра Nim є прикладом неупередженої комбінаторної гри, яка еквівалентна будь-якій іншій грі цього типу. Іншими словами, будь-яка

неупереджена гра ізоморфна грі Нім ("теорія Шпрага-Гранді" [82] [87] представлена Берлекемпом).

Розглянута програма ІЛП повинна вивчити класифікатор позицій N-P, щоб забезпечити виграшний хід. Ці позиції N-P є єдиними двійковими станами в неупередженій грі (P-позиція для попереднього гравця та N-позиція для наступного гравця). Той самий прийом, що використовується в цьому застосунку, може бути застосований і до неупереджених комбінаторних ігор, особливо коли концепт позицій N-P реалізований для генерації виграшних стратегій.

Згенеровані приклади (32 позитивних і 18 негативних) були отримані з використанням підходу Minimax в Progol 4.5, і результат показав високу прогнозовану точність SVM в порівнянні з методом ANN. Ця точність прогнозування, як правило, була вищою при використанні двійкового подання даних, порівняно із десятковим поданням [87] [82].

ІЛП як глибинний аналіз даних щодо поведінки мобільних користувачів

Третій із застосунків, який ми розглядаємо в цьому підрозділі, стосується правил поведінки користувачів у сфері мобільних телефонів [88]. Така реалізація постає ще один ілюстративним прикладом успіху немонотонного підходу ІЛП, який зробив можливим автоматичне добування даних про поведінку користувачів.

У цьому дослідженні були протестовані дві моделі. Перший набір даних був частиною групи Reality Mining, а другий пов’язаний з мобільним застосунком ULearn. Головною метою цієї останньої моделі є отримання реалістичної оцінки точності вихідних даних.

Використання підходу ІЛП для цієї реалізації виявилось дуже продуктивним. Фактично, цей немонотонний підхід, супроводжуваний іншими евристиками для оптимізації діапазону пошуку шляхом зменшення часової складності, заснований на техніці TAL (Top-directed Abductive Learning), зміг адаптувати існуючі правила з 80 % точністю.[88].

Важливо підкреслити переваги використання підходу TAL у мобільній сфері. Деякі з цих переваг є наступними [88] [82]:

- Його висока виразність у процесі навчання логічним програмам.
- Його поводження із запереченнями, та здатність вивчати немонотонні гіпотези,
- Завершеність в плані пошуку рішення, у випадках, якщо потенційне рішення існує.
- Крім того, це впровадження TAL, яке було першим у представленні передумов та гіпотез як логічних програм, крім таких переваг, має ще можливість виведення правил та

здійснення глибокого пошуку в суттєво більшому наборі даних. [88].

Великі об'єми інформації, щодо мобільності та поведінки користувача, згруповані таким чином, що дозволяє користувачеві гнучко користуватися певною мовою та уточнювати результати навчання, вставляючи обмежені правила щодо обраної мови. Більше того, правила вдосконалено за допомогою спеціального набору даних та застосуванням процесу перевірки, що покращує точність. Наприклад, той факт, що користувач приймає або відхиляє вхідний дзвінок, напряму інтегровано як нове правило, що впливає на поведінку користувача. Слід зауважити, що вибір евристики, при впровадженні ІЛП істотно впливає на точність результатів навчання [88] [82].

В цілому, модель TAL демонструє прийнятну застосовність щодо поведінки користувачів мобільних пристроїв. Вона показує гарні результати навіть при опрацюванні великих наборів даних. Немонотонні моделі ILP з їх вузьким колом обмежень витримують шум системи і гарантують масштабованість типу і розміру області [88].

Друга модель, що використовується в цьому мобільному застосунку, це - ULearn, яка є інструментом клієнт-сервер. Це робить правила доступними для користувача у вигляді вихідних результатів фонових знань та навчальних прикладів. Роль користувача в цьому

відношенні є вирішальною, особливо при виборі виду правил і визначенні кількості обмежень на діапазон пошуку даних [88].

1.24. Використання теорії аргументації в штучному інтелекті

Дослідження в галузях теорії аргументації та штучного інтелекту були тісно взаємопов’язані протягом останніх двох десятиліть. Цей двосторонній обмін був вигідним для обох сфер. Наприклад, теорія аргументації надавала концепти міркувань, які можна обробляти цифровим способом, а штучний інтелект використовував ці концепти для перевірки їх дійсності. Таким чином, співпраця між цими двома галузями дозволила перевірити на практиці обсяг правил і концептів, наданих теорією аргументації.

Аргументація штучного інтелекту використовує різні теоретичні та практичні основи. У теоретичному плані він надихається формальними моделями аргументації, які беруть свій початок, наприклад, в сучасній і філософській логіці. З практичної точки зору аргументативні завдання реалізуються за допомогою комп'ютерних програм. З іншого боку, природність системи забезпечує природність форми дослідження. Маються на увазі аспекти системи, що пов'язані з людським розумом або неформальною логікою [19,21,22]. Зв'язок між цими різними концептами теорії аргументації покращує рівень досліджень цієї теорії в ШІ, підкреслюючи актуальність формальних та обчислювальних аспектів [79].

1.24.1. Немонотонна логіка та немонотонне міркування

Історично класична аристотелівська логіка оперує логічними твердженнями (і дедукціями), які є *стабільно* правильними. Такий вид незмінних наслідків із припущень, зазвичай пов'язаний з монотонною логікою, в рамках дедуктивної системи може створювати конфлікти між системою і міркуваннями здорового глузду. Немонотонна логіка пропонує в певному сенсі уникати цього конфлікту. Результати, отримані із формул в монотонній логіці *лишаються незмінним*, в той час як множина формул збільшується. У немонотонній логіці деякі з цих результатів *змінюються* коли набір формул стає більшим. Цей підхід, на відміну від формальної монотонної теорії, виражається в нехтуванні деякою інформацією в системі через слабку ймовірності її появи або неможливості її використання на практиці [14]. Отже, немонотонна логіка часто пов’язується з такими явищами, як "невизначеність, обмежена раціональність, нормальність та ін." [14].

Очевидно, немонотонна логіка пов’язана з довгостроковими (або стратегічними) цілями ШІ, але, що важливіше, в ній зацікавлені розробники застосунків, особливо, коли мова йде про моделі аналізу міркувань.

Історично склалося, що головні немонотонні застосунки, такі як “Belief revision, Closed-world reasoning, та Planning”[14], мали велике значення на початку її розвитку. Belief revision займається впровадженням системи підтримки достовірності (СПД), яка дозволяє

оперувати переконаннями знання шляхом використання інструментів корекції з метою динамічного оновлення цих переконань. Як наслідок, механізм СПД впливав на практичний ШІ, який, до прикладу, наголошував на використанні логічного аналізу. Крім того, за допомогою СПД, немонотонна логіка могла впоратися із висновками, які використовують поняття можливості поліпшення. Міркування замкнутого світу в основному стосуються аналізу баз даних з використанням логічних парадигм, особливо в уявленні і міркуваннях щодо знань. Однією з найбільш важливих властивостей в цих дедуктивних принципах базах даних є припущення про замкненість світу, згідно з яким, твердження вважається помилковим, якщо неможливо довести його істинність в рамках системи. Ця ідея в інших джерелах фігурує під назвою *заперечення як відмова* принцип, який ми також обговорюємо в цьому тексті. Можна просто побачити, що коли істинність твердження не вдається довести у замкненому середовищі (світі), воно буде вважатися хибним, що означає, що його заперечення є істинним. Третій застосунок немонотонної логіки, планування, вивчає факти в темпоральних міркуваннях, щоб отримати формалізовані репрезентації в ситуаціях планування. У цих темпоральних станах об’єкти зберігають свій статус, коли немає необхідності або відсутнє обґрунтування для переходу в інший статус. При цьому зберігається розуміння того, що перехід на інший статус завжди можливий, коли з’являється нове обґрунтоване знання (або подія) [20]. Перш ніж продовжувати розгляд немонотонної логіки та інших пов'язаних із нею

елементів, ми коротко представимо концепцію *заперечення як відмови*. Зробити це на даному етапі ми вважаємо за необхідне, оскільки вона показує одну технічну, але водночас центральну роль штучного інтелекту в логіці.

1.24.2. Заперечення як відмова: новий логічний концепт

У ранніх розробках підходящої для штучного інтелекту мови логічного програмування характерним є використання концепції заперечення як відмова (ЗЯВ) в протилежність традиційному використанню заперечення. У більш загальному сенсі, розроблена комп'ютерних наук немонотонна логіка використовує, серед інших концепцій, принцип (ЗЯВ), особливо в області штучного інтелекту. Мова Пролог є типовим прикладом, в якому застосовується принцип ЗЯВ в середовищі логіки першого порядку і вважається однією з найбільш успішних мов, що застосовують немонотонну логіку. Ідею ЗЯВ можна розглядати як частину аспектів управління логічного програмування, яка являє собою більш загальну характеристику способу продукування умовиводів. [71]. Важливим елементом управління в Prolog є інструкція вирізання, яка дозволяє зупинити пошук у базі знань, одночасно доводячи теорему. Ця інструкція (позначається як "!") технічно використовується для того, щоб уникнути конкретних непотрібних пошуків з вертанням і відповідає ідеї відкидання деякої інформацією в ситуації міркування здорового глузду. Очевидно, цей тип *технічного управління*, якого не існує в класичній логіці, дозволяє

економити цінний час виконання та уникає зайвих дедуктивних результатів. У цьому контексті керуюча інструкція вирізання в Пролог, що додається до відмови доведення конкретного твердження S, представляє заперечення як принцип відмови в немонотонній логіці, демонструючи відмінність між цією логікою (реалізованою в Пролог) і класичною. Отже, немонотонна логіка (Пролог) - це нова, більш потужна версія (або своєрідне розширення) класичної логіки в процесі побудови дедуктивних доказів (Гілліс), що вводить розкіш повного управління механізмом пошуку доведення [71].

Як вже згадувалося раніше, механізація логічних концептів, що набула оформлення у застосунках штучного інтелекту, є центральною частиною розробки нових ідей та методологій як загалом у логіці, так і в логічному ШІ, зокрема. Таким чином, експерти запропонували широку групу концептів, що стосуються, зокрема, підходів до міркувань для дій і змін, питань формалізму планування, а також застосування роботів. В цілому, з новими удосконаленнями традиційних логічних методологій, логіка стала формальною дисципліною, міцно поєднаною із комп'ютерними науками та математикою [14].

У більш історичному аспекті штучний інтелект завжди був *об'єднуючою* областю, не тільки для логіки та комп'ютерних наук, а й для філософської логіки та математичної логіки, особливо щодо питань міркувань. Цей спільний інтерес до штучного інтелекту був

вирішальним для експертів ШІ для розробки нових логічних ідей на формальній та математичній основі, а також з точки зору їх реалізації.

Крім того, логічна основа в ШІ була одним із перших підходів, особливо застосовуваних у застосунках, що задіюють міркування, які вимагають формальних технік. Насправді, використання штучного інтелекту на основі логіки дає краще розуміння та репрезентацію проблеми міркувань, яку бажають реалізувати. На традиційному перетині філософської логіки та логіки штучного інтелекту, були здійснені значні досягнення у формалізації *здорового глузду*, потрібні для вирішення реальних ситуацій. Наприклад, Маккарті запропонував формалізацію "Розуміння наративів, діагностики, міркувань щодо ставлення інших агентів тощо" [14].

Отже, нові логічні горизонти та техніки, які були створені, виявились корисними як для логіки, так і для штучного інтелекту. "Стратегії планування, управління автономними інтелектуальними моделями та теорія аргументації" [20] є типовими прикладами в цьому контексті. Ця частина стосується ролі теорії аргументації у штучному інтелекті і базується на джерелі [20], яке ми вважаємо основним у цій сфері.

1.24.3. Немонотонна логіка як виправдане висовування

Як ми вже згадували, серед всіх галузей штучного інтелекту, пов'язаних із аргументацією, найбільше розвивається саме немонотонна

логіка. Висловлюючись точніше, логіку називають немонотонною в протилежність монотонній класичній логіці, коли висновок, отриманий із засновків, перестає бути дійсним, коли додаються деякі інші засновки. Це дозволяє приймати висновок тимчасово, доки не буде додано нову інформацію в систему, що призводить до відкликання висновку. Іншими словами, поточний висновок приймається за відсутності інформації про протилежне, і як тільки нова додана інформація суперечить йому, передбачуваний висновок відкликається.

Як вже згадувалося раніше, у Пролог принципи логічних програм, що описують факти та правила щодо конкретного домену, безпосередньо пов'язані з немонотонною логікою в основному через ідею заперечення як відмови. Твердження в логічній програмі вважається істинним, коли його можна вивести в рамках програми, і хибним, коли не можливо. Наприклад, щоб довести істинність твердження P, ми можемо спробувати вивести не (P). Якщо це неможливо (трапляється відмова), ми можемо зробити висновок що його заперечення є істинним, що означає, що P є істинним (заперечення не (P)). Відповідно до ідеї немонотонної логіки, виведене положення в логічній програмі за принципом заперечення як відмови може втратити статус виведеного при додаванні нового факту до системи. Область досліджень, що називається "семантика стійких моделей логічної програми" [20] забезпечує формалізацію для інтерпретації логічних програм із запереченням як відмовою. Крім того, дослідження

немонотонної логіки в основному мало успіх в рамках *комп'ютерного програмування на основі логіки* зокрема на згаданій раніше мові Пролог та в експертних системах, що використовують немонотонні міркування загалом.

Немонотонна логіка виявилася особливо ефективною, коли 1) досліджувана система є “матеріально адекватною” [20] у тому сенсі, що вона може виразити велику кількість відповідних прикладів, 2) система “формально адекватна” [20] маючи формальні властивостями відповідно до очікувань, 3) система є “обчислювально адекватною” [20] завдяки можливості обґрунтовано обчислити деякі види умовиводів.

Теорія виправданого мислення, описана Поллоком у 1987 р. і тісно пов’язана з немонотонною логікою, змінює уявлення про ідею розуму як основи аргументації. Зазвичай беруть до уваги два типи раціональності висновування: 1) безсумнівне висновування, коли висновок отриманий логічним способом і 2) коли дається імплікація та твердження, що робить висновок *недійсним*, в результаті чого, висновок вважають спростованим [20]. Ми вважаємо важливим підкреслити відповідність між цією ідеєю та немонотонною логікою, згаданою раніше. Додане висловлювання у виправданих міркуваннях точно відповідає новій інформації, доданій до системи в немонотонній логіці, а спростований висновок відповідає його відкликаному висловлюванню. Згідно з підходом Поллока, сильна теорія міркувань дозволяє реалізувати її за допомогою комп’ютерної програми, якій можливо надати оцінку для

кращого розуміння успіхів і невдач теорії. Більше того, Поллок розглядає міркування як процес, на основі причин. Цей процес контролюється правилами, що представляють "процедурні знання" [89], дозволяючи здійснювати коректні міркування.

Крім того, Поллок [89] [20] визначає кілька класів особливих причин:

1. "Дедуктивні причини"[89]: Які подібні до тих, що вивчаються в класичній логіці, як висновуючі.

2. “Сприйняття”[89]: Стан сприйняття, о основі якого лежить переконання, що "дещо, в чому вбачається причина дає підставу вважати, що це і є причина" [89]. В даному контексті вважається, що аргументи проти надійності є загальним типом спростувальних аргументів, які справедливі для всіх причин *першого враження*.

3. "Пам’ять" [89]: За словами Поллока, люди пам’ятають переконання, отримані шляхом міркування, і, як правило, "не пам’ятають причин, що спонукали їх до створення цих переконань" [89].

4. "Статистичний силогізм" [89]: Статистичний силогізм вбачається як простий випадок імовірнісних міркувань. Цей вид силогізму повинен враховувати будь-яку інформацію, пов’язану з ним, і часто може "змінити ймовірність того, що факт станеться" [89].

5. "Індукція" [89]: Поллок вказує два типи індукції. Перший тип - це перелічувальна індукція, що відповідає класичній індукції, яка

узагальнює факт на основі багатьох конкретних фактів. Другий тип - статистичний, який під час узагальнення враховує ймовірність того, що той чи інший факт відбувся, і "на основі ймовірності того, що подібні факти вже мали місце" [89].

1.24.4. Аргументи та правила

Враховуючи, що будь-яка формалізація аргументації повинна бути побудований на змістовних підставах, специфікація типу причин та схеми аргументів, досліджуються в рамках теорії аргументації. На противагу правилу modus ponens, яке описується як "абстрактне, строге і універсальне" [20], схеми аргументів вважаються "конкретними, виправданими та залежними від контексту" [20]. Наприклад, в засновку, що містить твердження будь-якої особи, істинність висновку з використанням цього засновку залежить, наприклад, від статусу цього твердження (помилка, хиба і т.д.) [20].

При розробці схем аргументації засновки схеми та критичні питання уточнюються шляхом додавання або прийняття нових схем. Схеми аргументації формалізовані Верхейєм як виправдані правила умовиводу. Він запропонував операції зі схемами аргументації, виходячи з "висновку, засновків, умов використання та винятків" [20].

У цьому контексті розвивається штучний інтелект у праві, і приділяється особлива увага теорії аргументації. Наприклад, Праккен

розглядає класичні та немонотонні логіки з метою моделювання періодичної форми норм у праві, що містить винятки [20].

Крім того, Хейдж поширив логіку першого порядку на те, що називається *логікою на основі причин* в якій причинам, як застосуванню норм, відводиться фундаментальна роль. Оперуючи причинами індивідуально, властивості правил краще виражаються [20].

1.24.5. Застосунки теорії аргументації в ШІ

Тревор Бенч-Капон у 2003 році розробив модель цінностей, яка передувала аргументам. У цій моделі за основу взята точка зору Перельмана і Ольбрехтса-Тітека, яка виражається наступним чином: "Якщо люди протистоять один одному щодо прийнятого рішення, то це не тому, що вони роблять якусь логічну або обчислювальну помилку. Вони обговорюють застосовне правило, цілі, які необхідно враховувати, значення, яке слід надавати цінностям, інтерпретацію і характеристику фактів " [20, p. 654].

В реальних життєвих ситуаціях рішення може бути остаточним в силу їх специфіки. В цьому напрямку Бенч-Капон прагне побудувати розширення формальної аргументації, беручи до уваги цінності аудиторії, до якої ми звертаємося. Він, врешті, приймає абстрактну модель аргументації Данга, в якій до кожного аргументу додається відповідне значення. Таким чином, прийняття аргументу тягне за собою додавання до нього значення. Нарешті, Бенч-Капон та Сартор

інтегрували модель, засновану на цінностях, при обробці правових міркувань, поєднуючи в їх основі норми і ситуації. Зверніть увагу, що юридичні міркування утворюються шляхом побудови та використання теорії, здатної обґрунтувати рішення з точки зору розглянутих цінностей [20].

Діалогічна конфігурація аргументації часто пов'язана з темою зобов'язання представлення доказів. В такому випадку учасник діалогу наводить свої аргументи в умовах процедурних або матеріальних обмежень. Перше обмеження виникає, наприклад, при наданні контраргументу до попереднього аргументу. Друге обмеження може статися, коли аргумент є сильним з точки зору інших попередніх аргументів [20].

На рівні застосунків ці докази є доцільними в праві, враховуючи важливість сильного докази або більш ефективної контратаки в цій області. Отже, область "зобов'язання надання доказів" [20] додатково виявила зацікавлення у теорії аргументації загалом, а також до ШІ, що застосовується до права, зокрема.

Для посилення аргументу також може бути використаний кількісний підхід. Як правило, в системі аргументації може використовуватися *умовна ймовірність*. Наприклад, "чим ймовірніше те, що гіпотеза за якихось умов матиме місце, тим вище ймовірність прийняття цієї гіпотези за таких умов" [20]. Зазвичай це називають як байєсівською гносеологією. Подібно до того, що робив Поллок, інші

автори наводять аргументи проти імовірнісного підходу аргументації. Вони в основному стверджують, що, використовуючи такий підхід, "переконання передують обґрунтуванням" [20].

Однак інші автори, такі як Цукерман, Рівере і Верхей, намагалися підтримати байєсівський підхід, демонструючи успішні випадки (Цукерман і Рівере) або створюючи "формальну теорію виправданої аргументації" [20, p. 663] підкреслюючи той факт, що логічні та імовірнісні характеристики тісно пов'язані.

Крім того, часто експериментальні та емпіричні доведення включаються в модель аргументації для посилення запропонованого факту, особливо якщо аргумент має основною метою встановлення істини. Наявні докази можуть ведуть до висновків, що пояснюють факти. Побудова міркувань на основі наявних доказів відповідає *абдукційній логіці* розробленій Пірсом. Такий спосіб міркувань дає висновування для *найліпшого доступного пояснення*. Наступна ситуація, наведена автором, ілюструє випадок такого типу міркувань (Джозефсон і Джозефсон) [20]:

1- А - це набір фактів, спостережень тощо.
2- Б може пояснити А.
3- Не існує інших даностей, які можуть пояснити А, так, як це може зробити В.
4- Отже, В, мабуть, є істинним" [20, p. 663].

Слід зазначити, що подібні аргументи, які використовують правила доказовості, як правило, є аргументами виправдувального типу. Таким чином, вони вважаються істинними, допоки не з'явиться додаткової суперечливої інформації.

Успіх теорії аргументації в галузі штучного інтелекту головним чином пов'язаний з теоретичним прогресом, який підтримується численними застосунками та дослідженнями конкретних випадків, такими як обробка природної мови. Насправді теорії та системи були перевірені та отримали оцінку за допомогою тематичних досліджень. Стосовно такого підходу в юридичній сфері, Маккарті описує ситуацію наступним чином:

> Правові поняття не можуть бути адекватно представлені визначеннями, що встановлюють необхідні та достатні умови. Натомість юридичні поняття є неодмінно *відкрито-текстурними*. Правові норми не є статичними, а натомість динамічними. Оскільки їх застосовують до нових ситуацій, вони постійно змінюються щоб бути *придатними* для нових *фактів*. Таким чином, важливим процесом у правових міркуваннях є не застосування теорії, а її побудова. В такому процесі побудови теорії немає єдиної *правильної відповіді*. Однак існують правдоподібні аргументи різного

ступеня переконливості для кожної альтернативної версії правила в кожній новій фактичної ситуації (Маккарті) [20, p. 666].

На сьогодні є очевидним, як зазначається в [20], що поглиблення співпраці між штучним інтелектом та теорією аргументації дасть плідні результати для багатьох дослідницьких проблем. Не дивлячись на численні позиції, що підкреслюють протилежність людей і машин, історичний та природний зв'язок між ними показує необхідність співпраці між ними.

На наш погляд, така співпраця має враховувати не тільки формальні, або засновані на логіці частини аргументації, але також емоційні і риторичні аспекти аргументу.

1.25. Висновок

В цьому розділі ми показали як застосунки Пролог, індуктивне логічне програмування, та теорія аргументації виступили основою досягнень штучного інтелекту.

Враховуючи ключову роль логіки предикатів Фреге в основах Пролог, яку ми представили в розділі 3, ми деталізували як ця мова перетворилася на дієвий інструмент впровадження автоматизованого доведення теорем. Для цього ми надали деталізований алгоритм розроблений для доведення логічних формул з використанням мови

програмування Пролог. На додачу, ми надали перелік застосунків автоматизованого доведення теорем для різних галузей, підкреслюючи при цьому успішність зв'язку між використанням у Пролог логіки предикатів та цими застосунками.

У другій частині ми показали успішні застосунки індуктивного логічного програмування (ІЛП) в машинному навчанні, що включає індуктивне навчання. Ми показали, що машинне навчання засноване на логіці предикатів постає специфічним типом застосунків штучного інтелекту, що працюють без вичерпного програмування. В цьому контексті ми показали, що ІЛП в змозі мати справу із логічними програмами як із вводом так із виводом, використовуючи принцип індукції, що індукує програму з різними прикладами вводу та виводу.

Ми розглянули численні сучасні застосунки ІЛП в машинному навчанні, демонструючи при цьому, що фундаментальним підґрунтям їх досягнень є логіка предикатів. Ми окремо показали як використання ІЛП може істотно поліпшити техніки пошукових систем, стратегію комбінаційних ігор та видобуток даних щодо поведінки користувачів мобільних пристроїв.

Ці результати підкреслили як підхід ІЛП в машинному навчанні зокрема і в цілому у штучному інтелекті дає багато переваг. Серед багатьох переваг використання моделі ІЛП ми виокремили наступні: 1) рівень виразності; 2) включення фонових знань; 3) інтерпретованість та можливість нарощення знань утворенням предикатів; 4) придатність

застосунку в широкому колі різних областей, в основі яких лежать правила. Вказані переваги, тим чи іншим чином, забезпечують більшу продуктивність системи, краще розуміння та більшу гнучкість порівняно із іншими парадигмами машинного навчання. Ми також показали найвагомішу причину успіху підходу ІЛП, яка полягає в можливості індукувати програму, використовуючи виключно приклади вводу і виводу. Зокрема, концепт індукованості програми можна застосувати розпочавши вирішувати прості задачі, а потім використати отримані рішення щоб вирішувати інші складніші задачі.

Більше того, існує довгий перелік застосунків чи навіть галузей, що можуть ефективно використовувати техніку ІЛП. Прикладами таких застосунків є мультипроцесор, який налаштовується, багатозначне ІЛП навчання, навчання залежним концептам, застосування теорії графіків (стиснення, сітчаста модель) і стратегії роботів.

В останній частині цього розділу ми дослідили теорію аргументації. Ми показали, як немонотонна логіка використовується в якості виправданого типу міркувань в теорії аргументації. Більше того, ми виокремили декілька застосунків теорії аргументації у штучному інтелекті.

Фактично ми показали, як немонотонна логіка виступила прикладом того, як штучний інтелект спонукає розвиток логіки. Ми відзначили, що теоретичні дослідження механізмів міркування, дедукції і доведення були підтримані практичними комп'ютерними розробками

для представлення, маніпулювання і тестування цих концептів за допомогою мов програмування. Відповідно до цієї ідеї, багато комп'ютерних мов, таких як Пролог, що використовуються для реалізації конкретних завдань, застосовують немонотонну логіку. Ми вважаємо цей аспект взаємовигідними відносинами між ШІ та логікою через теорії аргументації. Ми багато в чому використовували наші матеріали щодо цієї теорію представлені в [20], та щодо різних напрямів цієї теорії наведених в [79]. Ми відводимо важливе місце концепції *заперечення як відмови* і її взаємозв'язку з теорією аргументації, яка збагатилася численними нещодавніми теоретичними розробками в логіці і штучному інтелекті.

Ми вважаємо не зовсім точним твердження, що логіка відіграла фундаментальну роль у *кожній* області штучного інтелекту. Однак ми стверджуємо, що деякі формальні і неформальні логіки мали значний вплив на *багато* важливих областей цієї дисципліни, особливо що стосуються представлення знань, розуміння проблем, автоматизованих міркувань і доказів, а також теорії аргументації.

ВИСНОВКИ

У цій дисертації ми в основному досліджували вплив логіки на комп’ютерні науки, а також сучасні проблеми філософії комп’ютерних наук та штучного інтелекту. Ми виявили, що для розуміння і вирішення цих проблем необхідно поєднання філософського дослідження з технічними зусиллями, які поєднують логіку і комп’ютерні науки. Досліджувані нами поняття логіки почали розвиватися ще з середини XIX століття, але і сьогодні їх застосування в комп’ютерних науках не втратило актуальності. Наші філософські роздуми були присвячені природі комп’ютерних наук і взаємовідносинам між людьми і інтелектуальними машинами. Ми задалися питаннями і запропонували свій погляд на свідомість, моральний статус і моральну відповідальність, що можуть бути пов'язані з такими машинами. Ми були зацікавлені у вирішенні багатьох питань, серед яких наступні:

- У чому полягає справжня природа комп’ютерних наук і як її можна охарактеризувати?
- Яка обчислювальна модель найкраще підходить для розуміння комп'ютаціоналізма?
- Наскільки ми повинні довіряти рішенням, прийнятим комп’ютеризованою системою?
- За яких умов ми повинні надавати перевагу рішенню, прийнятому комп’ютером над людським рішенням?

- Які галузі комп’ютерних наук найбільше виграли від співпраці із логікою?
- Як можливо теоретично визначити ці переваги відповідно до їх застосунку в комп’ютерних науках?
- Чи можливо встановити причини такого *успіху*?
- Чи існують філософські ідеї, що лежать в основі розвитку логіки, які вплинули на комп’ютерні науки?
- Як поняття обчислення стало центральним пунктом логіки?

Взявши на себе завдання дати відповідь на зазначені питання, ми отримали наступні основні результати відповідно до поставлених в дисертації завдань:

- Ми визначили комп’ютерні науки як пов'язаний з обчисленнями когнітивний феномен, а також як набір концепцій, що стосуються логічних методів і інженерних експериментів і реалізацій. Ми стверджували, що характеристика комп’ютерних наук повинна враховувати обробку інформації, процедурні міркування і абстракції.
- Ми стверджували, що модель машини Тюрінга слід розглядати як шаблон обчислення за замовчуванням, особливо в рамках нашого дослідження комп'ютаціоналізма. Наш список аргументів, на користь такого припущення, включає конкретну відповідність МТ сучасним

комп'ютерам, її математичний формалізм, і її еквівалентність до різних обчислювальних моделей (тобто до рекурсивних функцій).

- Ми відстоювали позицію часткового комп’ютаціоналізма, спираючись на *логічні* явища, що пов'язують обчислення та пізнання.
- Ми навели аргументи проти радикального комп’ютаціоналізма, а саме аргументи *відмінності по суті* та *проблеми знання*.
- Ми представили скептичну позицію по відношенню до гіперкомп'ютаціоналізма, ґрунтуючись на наших міркуваннях про комп'ютаціоналізм, разом із питаннями про потенційні реалізації нетюрінгівского обчислювального підходу.
- Ми стверджували, що розумні агенти не повинні мати морального статусу, та (як наслідок) не можуть нести моральної відповідальності. Цей аргумент ґрунтувався на необхідності мати свідомість, щоб володіти цими моральними елементами. Ми описали небажані наслідки протилежної позиції.
- Ми припустили, що діяльність, *заснована на логіці*, може вплинути на нашу довіру до прийняття рішень роботами. Цей аргумент враховував прозорість обчислень.

- Ми виявили, що логіка предикатів Фреге, логіка Буля та теорія типів Рассела найбільше серед інших логічних розробок вплинули на комп’ютерні науки.
- Ми визначили основні причини впливовості логіки предикатів в комп’ютерних науках: 1) її висока виразність як символічного мови 2) відповідність між фаховою мовою логіки предикатів (диз'юнкти Горна) і семантикою логічного програмування 3) фундаментальні досягнення в програмуванні на базі логіки предикатів, такі як інтерпретації Ковальського і правило резолюцій.
- Причини впливу логіки Буля на комп’ютерні науки ми визначили шляхом її адаптації в Булеву логіку. Потім ми показали, як ця Булева логіка виявила свою придатність для 1) проектування комбінаційних схем за допомогою булевих функцій і фізичної реалізації булевих логічних операторів 2) застосування булевих функцій в криптографії та коректуючому коді.
- Ми визначили причини, що вплинули на теорію типів Рассела в комп’ютерних науках з позиції цілісності даних, безпеки програм та ефективності як у функціональному, так і в об'єктно-орієнтованому програмуванні. Крім того, серед цих впливів, ми підкреслили значення відповідності Каррі-Говарда між типами та логікою.

- Ми стверджували, що мови логічного програмування, автоматизоване доведення теорем і індуктивне логічне програмування є одними з тих областей штучного інтелекту, які отримали найбільшу користь від логіки предикатів. На користь цього аргументу ми представили детальне вивчення теоретичних та практичних впливів логіки на ці сфери.

- Всупереч нашому основному інтересу в даній дисертації, ми показали також, як комп'ютерні науки вплинула на логіку, розвиваючи немонотонну логіку (або виправдані міркування) за допомогою парадигм логічного програмування. Врешті, ми продемонстрували важливість немонотонної логіки для теорії аргументації, застосованої у штучному інтелекті.

- Ґрунтуючись на попередньому пункті, ми прийшли до висновку, що 1) штучний інтелект не міг би існувати (принаймні, в його першому рамковому розвитку) без логіки Фреге; 2) проектування і аналіз комбінаторних схем, а також основна частина криптографії не могли б існувати без Булевої логіки.

- Ми дослідили історичний та обчислювальний розвиток логіки. У цьому дослідженні ми виокремили основні логічні

теорії, які, серед інших успішно призвели до появи комп’ютерних наук як самостійної дисципліни.

- Ми дійшли висновку, що ці теорії спираються на *обчислювальні компоненти* які виникають починаючи з часів логіки Ляйбніца і до часів Тюрінга. Між цими двома віхами постали також теорії Буля, Де Моргана, Фреге, Гамільтона, Рассела, Вайтгеда, Гільберта, Геделя та Черча. Таке історичне дослідження дозволило виявити, як ідея обчислення стала центральним моментом сучасної логіки. У цій частині дослідження ми показали, як спроба обґрунтувати логіцизм, філософське положення про засади математики, побічно привела до розробки найважливіших логічних принципів в комп’ютерних науках.

Багато проблем і питань, пов'язаних з цією роботою, заслуговують подальшого дослідження. Наприклад, ми вважаємо доцільним розширити історичні та теоретичні дослідження додаткових структур логіки, в яких ідея обчислення є центральною. Це дозволить більш глибоко вивчити роботи таких логіків, як Гамільтон, Шредер, Тарський, Пірс, Хербранд, Пост, Черч та Клін.

Ми вважаємо також корисним розширення досліджень зв’язків між пізнанням, логікою та обчисленням. Таке завдання може допомогти краще зрозуміти, наприклад, природу когнітивної діяльності, яку можливо пояснити за допомогою обчислень або логічного механізму. У

цьому контексті ми вважаємо, що подальше дослідження обчислювальних моделей, відмінних від МТ, може надати краще розуміння ролі логіки у формуванні комп'ютаціоналізма та гіперкомп’ютаціоналізма.

Інший напрямок подальших досліджень може бути пов'язаний із етичними питаннями штучного інтелекту. Йдеться про встановлення застосунків штучного інтелекту, використання яких матиме важливі соціально-етичні наслідки. Мова також йде про те, щоб обдумати можливі відповіді. Крім того, може бути доцільним складання списку етичних критеріїв, що ведуть до чіткої практики моральної відповідальності в застосунках штучного інтелекту.

Також, перспективним напрямком подальших досліджень може бути подальше дослідження впливу логіки також і на інші області комп’ютерних наук, представлені в даній дисертації, такі як описова складність, теорія знання і бази даних.

Насамкінець ми пропонуємо розширити вивчення взаємозв'язку між логікою і комп’ютерними науками, розглянувши роль теорії мов (або, можна сказати, лінгвістичних аспектів) в цьому взаємозв'язку.

Список використаних джерел

1. Aarhus University. (2011). The Computational Turn: Past, Presents,Futures? *Proceedings IACAP*.

2. Abramsky, S. (2014). Two puzzles about computation. *ArXiv Preprint ArXiv:1403.4880*.

3. Asimov, I. (2004). *I, Robot*. Bantam Books.

4. Awwad, M. (2018a). Influences of Frege‟s Predicate Logic on Some Computational Models. *Future Human Image*, *1*(9), 5–19.

5. Awwad, M. (2018b). The role of inductive logic programming in some machine learning applications. *Humanities Studies*, *16* (Issue 31), 1– 10.

6. Awwad, M. (2019). From Philosophy to Computation: How Russell‟s Type Theory Impacts Programming Languages. *EESA Journal*, *1*(47 (part 7)), 70–74.

7. Awwad, M. (2021). From Boole‟s Logic to Boolean Applications in Computer Science. *Educational Discourse: Collection of Scientific Papers*, *32*(4), 18–25.

8. Bechtel, W., & Abrahamsen, A. (1991). *Connectionism and the mind: An introduction to parallel processing in networks.* Basil Blackwell.

9. Bevir, M. (2000). The logic of the history of ideas. *Rethinking History*, *4*(3), 295–300.

10. Birtwistle, G., & Subrahmanyam, P. A. (2012). *Current Trends in HardwareVerification and Automated Theorem Proving*. Springer Science & Business Media.

11. Black, E., Modgil, S., & Oren, N. (2018). *Theory and Applications of Formal Argumentation: 4th International Workshop, TAFA 2017, Melbourne, VIC, Australia, August 19-20, 2017, Revised Selected Papers*. Springer.

12. Blackmon, J. C. (2007). *Computationalism and the Putnam-Searle Thesis*. University of California, Davis.

13. Boddington, P. (2017). *Towards a Code of Ethics for Artificial Intelligence*. Springer.

14. Boole, G. (1844). On a general method in analysis. *PhilosophicalTransactions of the Royal Society of London*, *134*, 225–282.

15. Boole, G. (1847). *The mathematical analysis of logic* (Philosophical Library). Philosophical Library.

16. Boole, G. (1957). *The laws of thought*. Dover, New York (original edition1854).

17. Bostrom, N., & Yudkowsky, E. (2014). The ethics of artificial intelligence. *The Cambridge Handbook of Artificial Intelligence*, *1*, 316 334.

18. Boyer, R. S., & Moore, J. S. (2014). *A Computational Logic*. Academic Press.

19. Brady, G. (2000). *From Peirce to Skolem: A neglected chapter in the historyof logic*. Elsevier.

20. Brey, P., & Søraker, J. H. (2009a). Philosophy of computing and information technology. In *Philosophy of technology and engineering sciences* (pp. 1341–1407). Elsevier.

21. Brey, P., & Søraker, J. H. (2009b). Philosophy of computing and

information technology. In *Philosophy of technology and engineering sciences* (pp. 1341–1407). Elsevier.

22. Bringsjord, S., & Zenzen, M. (2012). *Superminds: People Harness Hypercomputation, and More*. Springer Science & Business Media.

23. Brooks Jr, F. P. (1996). The computer scientist as toolsmith II. *Communications of the ACM*, *39*(3), 61–68.

24. Brown, F. M. (2012). *Boolean Reasoning: The Logic of Boolean Equations*. Courier Corporation.

25. Burgess, J. P. (2016). Logic & Philosophical Methodology. In *The Oxford Handbook of Philosophical Methodology*. Citeseer.

26. Buss, S. R. (1998). *Handbook of Proof Theory*. Elsevier.

27. Cardelli, L., & Wegner, P. (1985). On understanding types, data abstraction, and polymorphism. *ACM Computing Surveys (CSUR)*, *17*(4), 471– 523.

28. Chang, C.-L., & Lee, R. C.-T. (2014). *Symbolic Logic and Mechanical Theorem Proving*. Academic Press.

29. Church, A. (1940). A formulation of the simple theory of types. *The Journal of Symbolic Logic*, *5*(2), 56–68.

30. Coeckelbergh, M. (2020). *AI Ethics*. MIT Press.

31. Colmerauer, A., Kanoui, H., Pasero, R., & Roussel, P. (1973). *Un systeme de communication homme-machine en francais*. Technical report, Groupe de Intelligence Artificielle Universite de Aix.

32. Constable, R. L. (2011). The trimuph of types: Creating a logic of computational reality. *Principia Mathematica Anniversary Symposium*, 28–

44.

33. Copeland, B. J., Posy, C. J., & Shagrir, O. (2015). *Computability: Turing, Gödel, Church, and Beyond*. MIT Press.

34. Couturat, L. (1901). *La logique de Leibniz* (Vol. 2). Georg Olms Verlag.

35. Crama, Y., & Hammer, P. L. (2010). *Boolean models and methods in mathematics, computer science, and engineering* (Vol. 2). Cambridge University Press.

36. Davis, M. (2000). *The Universal Computer: The Road from Leibniz to Turing*. New York, W.W. Norton & Company.

37. Davis, M. (2001). *Engines of Logic: Mathematicians and the Origin of the Computer*. W. W. Norton & Company.

38. Davis, M. (2004). The myth of hypercomputation. In *Alan Turing: Life and legacy of a great thinker* (pp. 195–211). Springer.

39. Davis, M. (2013). *Computability and Unsolvability*. Courier Corporation. Denecker, M., & Kakas, A. (2002). Abduction in logic programming. In *Computational logic: Logic programming and beyond* (pp. 402–436). Springer.

40. Dennett, D. C. (2014). *Intuition Pumps And Other Tools for Thinking*. W. W. Norton & Company.

41. Denning, P. J. (1985). What is computer science. *American Scientist*, *73*.

42. Denning, P. J., & Tedre, M. (2019). *Computational Thinking*. MIT Press.

43. Dodig Crnkovic, G., & Hofkirchner, W. (2011). Floridi‟s “open

problems inphilosophy of information", ten years later. *Information*, *2*(2), 327– 359.

44. Dreyfus, H. L., Dreyfus, S. E., & Athanasiou, T. (1986). *Mind Over Machine: The Power of Human Intuition and Expertise in the Era of the Computer*. NY Free Press.

45. Eemeren, F. H. van, & Garssen, B. (2012). *Topical Themes in Argumentation Theory: Twenty Exploratory Studies*. Springer Science & Business Media.

46. Eemeren, F. H. van, & Garssen, B. (2019). *From Argument Schemes to Argumentative Relations in the Wild: A Variety of Contributions to Argumentation Theory*. Springer Nature.

47. Eemeren, F. H. van, Garssen, B., Krabbe, E. C. W., Henkemans, F. A. S., Verheij, B., & Wagemans, J. H. M. (2014). *Handbook of Argumentation Theory*. Springer Netherlands.

48. Fitting, M. (2012). *First-Order Logic and Automated Theorem Proving*. Springer Science & Business Media.

49. Floridi, L. (2002). *Philosophy and Computing: An Introduction*. Routledge.Floridi, L. (2013). *The Philosophy of Information*. OUP Oxford.

50. Fodor, J. (1975). *The Language of Thought. Cambridge, MA: Harvard*. University Press.

51. Forsythe, G. E. (1968). What to do till the computer scientist comes. *TheAmerican Mathematical Monthly*, *75*(5), 454–462.

52. Franchette, F. (2011). *Why to Build a Physical Model of Hypercomputation*. N. A. Publishing.

53. Frege, G. (1879). Begriffsschrift, a formula language, modeled upon that of arithmetic, for pure thought. *From Frege to Gödel: A Source Book in Mathematical Logic, 1931*, 1–82.

54. Frege, G. (1980). *The Foundations of Arithmetic: A Logico-Mathematical Enquiry Into the Concept of Number*. Northwestern University Press.

55. Frege, G., & Carnap, R. (2004). *Frege&s lectures on logic*. Open Court Publishing.

56. Gabbay, D. M., & Woods, J. (2004). *The Rise of Modern Logic: From Leibniz to Frege, tom 3 z serii Handbook of the History of Logic*. Elsevier.

57. Gensler, H. J. (2017). *Introduction to logic*. Routledge.

58. Gillies, D. (2002). Logicism and the development of computer science. In *Computational Logic: Logic Programming and Beyond* (pp. 588–604). Springer.

59. Gödel, K. (1931). *On formally undecidable propositions of Principia Mathematica and related systems: Vol. 38: 173–198.* (Monthly Bulletin of Mathematics and Physics Reprinted and translated in Gödel (1986, 144-195).).

60. Grattan-Guinness, I., & Bornet, G. (2013). *George Boole: Selected Manuscripts on Logic and its Philosophy*. Birkhäuser.

61. Haaparanta, L. (ed.). (2009). *The development of modern logic*. OUP USA. Haaparanta, L., & Hintikka, J. (2012). *Frege Synthesized: Essays on the Philosophical and Foundational Work of Gottlob Frege*. Springer Science & Business Media.

62. Halpern, J. Y., Harper, R., Immerman, N., Kolaitis, P. G., Vardi, M. Y., & Vianu, V. (2001). On the unusual effectiveness of logic in computer science. *Bulletin of Symbolic Logic*, *7*(2), 213–236.

63. Hart, W. D. (2010). *The Evolution of Logic*. Cambridge University Press.

64. Hartmanis, J. (1993). Some observations about the nature of computer science. *International Conference on Foundations of Software Technology and Theoretical Computer Science*, 1–12.

65. Hartmanis, J., & Lin, H. (1992). *Computing the future: A broader agenda for computer science and engineering*. National Academies Press.

66. Horgan, T., & Tienson, J. (2012). *Connectionism and the Philosophy of Mind*. Springer Science & Business Media.

67. Horn, A. (1951). On sentences which are true of direct unions of algebras. *The Journal of Symbolic Logic*, *16*(1), 14–21.

68. Howard, W. A. (1980). The formulae-as-types notion of construction. *To HB Curry: Essays on Combinatory Logic, Lambda Calculus and Formalism*, *44*, 479–490.

69. Huth, M., & Ryan, M. (2004). *Logic in Computer Science: Modelling and Reasoning about Systems*. Cambridge University Press.

70. Kant, I. (1908). Critique of pure reason. 1781. *Modern Classical Philosophers, Cambridge, MA: Houghton Mifflin*.

71. Kaushik, S. (2007). *Logic And Prolog Programming*. New Age International (P) Ltd.

72. Kennedy, J. (2015). *Kurt Gödel*. Stanford Encyclopedia.

https://plato.stanford.edu/entries/goedel/

73. Khomenko, I. (2013). Informal logic: Between Logic and Epistemology. *Philosophy Study*, *11*, 1051–1060.

74. Khomenko, I. (2016). Informal logic and real argument. *Філософська Думка*, *3*, 34–46.

75. Khomenko, I. (2018a). A Look at Informal Logic. *Future Human Image*,*1*(9), 52–62.

76. Khomenko, I. (2018b). Empiricalization as a Trend in Argumentation Study. *Future Human Image*, *10*, 20–28.

77. Kneale, W., & Kneale, M. (1962). *The development of logic*. Oxford University Press.

78. Knuth, D. E. (1974). Computer programming as an art. In *ACM Turingaward lectures*.

79. Knuth, D. E. (1985). Algorithmic thinking and mathematical thinking. *TheAmerican Mathematical Monthly*, *92*(3), 170–181.

80. Kowalski, R. (1979). *Logic for Problem Solving*. Ediciones Díaz de Santos. Kowalski, R. (2011). *Computational Logic and Human Thinking: How to Be Artificially Intelligent*. Cambridge University Press.

81. Kowalski, R. (1974). Predicate logic as programming language. *IFIP Congress*, *74*, 569–544.

82. Kuipers, B., McCarthy, J., & Weizenbaum, J. (1976). Computer power andhuman reason. *ACM SIGART Bulletin*, *58*, 4–13.

83. Lavrac, N., & Dzeroski, S. (1994). Inductive Logic Programming. *WLP*,146–160.

84. Leibniz, G. (1703). Explication de l‟arithmétique binaire (Explanation ofBinary Arithmetic). *Mathematical Writings VII. 223.*

85. Leibniz, G. W. (1679). De Progressione Dyadica, Pars I. *Manuscript Dated,15*, 46–47.

86. Leibniz, G. W. (1989). Dissertation on the Art of Combinations. In *Philosophical Papers and Letters*. Springer.

87. Leibniz, G. W. F. (1840). *God. Guil. Leibnitii Opera philosophica quae exstant latina, gallica, germanica omnia* (Vol. 2). sumtibus G. Eichleri.

88. Lipson, H. (2011). *Foundations of Artificial Intelligence* [Academic]. https://www.cs.cornell.edu/courses/cs4700/2011fa/lectures/16_FirstOrderLogic.pdf

89. Loveland, D. W. (2016). *Automated Theorem Proving: A Logical Basis*. Elsevier.

90. Macbeth, D. (2009). *Frege's Logic*. Harvard University Press.

91. Mahoney, M. S. (2011). *Histories of computing* (Edited by Thomas Haigh). Harvard University Press, Cambridge.

92. Markitanis, A., Corapi, D., Russo, A., & Lupu, E. C. (2015). Learning user behaviours in real mobile domains. In *Latest Advances in Inductive Logic Programming* (pp. 43–51). World Scientific.

93. Martin-Löf, P., & Sambin, G. (1984). *Intuitionistic type theory* (Vol. 9). Bibliopolis Naples.

94. Matiiasevich, Y. V. (1993). *Hilbert's tenth problem*. MIT press.

95. Metakides, G., & Nerode, A. (1996). *Principles of Logic and Logic Programming*. Elsevier.

96. Milner, R. (1978). A theory of type polymorphism in programming. *Journal of Computer and System Sciences*, *17*(3), 348–375.

97. Minsky, M. (1979). Computer Science and the Representation of Knowledge. *In Dertouzos, L. and Moses, J., Editors, The Computer Age: A Twenty Year View*.

98. Muggleton, S. (1991). Inductive logic programming. *New Generation Computing*, *8*(4), 295–318.

99. Muggleton, S., De Raedt, L., Poole, D., Bratko, I., Flach, P., Inoue, K., & Srinivasan, A. (2012). ILP turns 20. *Machine Learning*, *86*(1), 3–23.

100. Muggleton, S. H., & Xu, C. (2015). Can ILP Learn Complete and Correct Game Strategies? In *Latest Advances In Inductive Logic Programming* (pp. 3–10). World Scientific.

101. Müller, V. C. (2018). *Philosophy and Theory of Artificial Intelligence 2017*. Springer.

102. Munoz, C. (2007). Type theory and its applications to computer science. *Quarterly News Letter of Institute for Computer Application in Science and Engineering (ICASE)*, *8*(4).

103. Nahin, P. J. (2017). *The Logician and the Engineer: How George Boole and Claude Shannon Created the Information Age*. Princeton University Press.

104. Newborn, M. (2012). *Automated Theorem Proving: Theory and Practice*. Springer Science & Business Media.

105. Nilsson, U., & Maluszynski, J. (1995). *Logic, Programming and Prolog*. Wiley.

106. Nye, A. (2019). *Words of power: A feminist reading of the history of logic*. Routledge.

107. Olszewski, A., Wolenski, J., & Janusz, R. (2013). *Church's Thesis After 70Years*. Walter de Gruyter.

108. Omodeo, E. G., & Policriti, A. (2017). *Martin Davis on Computability, Computational Logic, and Mathematical Foundations*. Springer.

109. Paleo, B. W. (2016). *Leibniz's Characteristica Universalis And CalculusRatiocinator Today*.
https://www.researchgate.net/publication/311456139

110. Peltier, N., & Sofronie-Stokkermans, V. (2020). *Automated Reasoning: 10thInternational Joint Conference, IJCAR 2020, Paris, France, July 1-4, 2020, Proceedings, Part I*. Springer Nature.

111. Piccinini, G. (2009). Computationalism in the Philosophy of Mind. *Philosophy Compass*, *4*(3), 515–532.

112. Piccinini, G. (2015). *Physical computation: A mechanistic account*. OUPOxford.

113. Pollock, J. L. (1987). Defeasible reasoning. *Cognitive Science*, *11*(4), 481–518.

114. Powers, T. M. (2017). *Philosophy and Computing: Essays in Epistemology,Philosophy of Mind, Logic, and Ethics*. Springer

115. Pylyshyn, Z. W. (1984). *Computation and cognition*. MIT press Cambridge,MA.

116. Rapaport, W. J. (2020). *Philosophy of computer science*. https://cse.buffalo.edu/~rapaport/Papers/phics.pdf

117. Robinson, J. A. (1965). A machine-oriented logic based on the resolution principle. *Journal of the ACM (JACM)*, *12*(1), 23–41.

118. Robinson, J. A. (1994). Logic, computers, Turing, and von Neumann. In *Machine intelligence 13: Machine intelligence and inductive learning* (pp. 1–35). Clarendon Press, Oxford.

119. Robinson, J. A. (2000). Computational logic: Memories of the past and challenges for the future. *International Conference on Computational Logic*, 1–24.

120. Russell, B. (1902). *Letter to Frege, in van Heijenoort (1967)*. Harvard University Press.

121. Russell, B. (1908). Mathematical Logic as Based on the Theory of Types. *American Journal of Mathematics*, *30*(3), 222–262.

122. Russell, B. (1958). *The philosophy of Leibniz*. London: George Allen and Unwin.

123. Russell, B., Slater, J. G., & Frohmann, B. (1992). *Logical and PhilosophicalPapers, 1909-13*. Psychology Press.

124. Russell, B., & Whitehead, A. N. (1910). Principia Mathematica. *Cambridge University Press, Cambridge*.

125. Sammut, C. (1993). The origins of inductive logic programming: A prehistoric tale. *Proceedings of the 3rd International Workshop on Inductive Logic Programming*, 127–147.

126. Santos, J. C. A., & Ribeiro, M. F. de S. B. (2014). Improving search engine Query Expansion techniques with ILP. In *Latest Advances In Inductive Logic Programming* (pp. 27–34). World Scientific.

127. Scheutz, M. (2002). *Computationalism: New Directions*. MIT Press.

128. Scott, J., & Bundy, A. (2015). Creating a new generation of computational thinkers. *Communications of the ACM*, *58*(12), 37–40.

129. Searle, J. R. (1984). *Minds, Brains and Science: The 1984 Reith Lectures*. British Broadcasting Corporation.

130. Searle, J. R. (2008). *Philosophy in a New Century: Selected Essays*. Cambridge University Press.

131. Shannon, C. E. (1938). A symbolic analysis of relay and switching circuits. *Electrical Engineering*, *57*(12), 713–723.

132. Shapiro, S. C. (2001). *Computer science: The study of procedures*. http://www.cse.buffalo.edu/~shapiro/Papers/whatiscs.pdf

133. Shramko, Y., & Wansing, H. (2005). Some Useful 16-Valued Logics: How aComputer Network Should Think. *Journal of Philosophical Logic*, *34*(2), 121–153.

134. Shramko, Y., & Wansing, H. (2011). *Truth and Falsehood: An Inquiry into Generalized Logical Values*. Springer Science & Business Media.

135. Siegelmann, H. T. (2012). *Neural Networks and Analog Computation: Beyond the Turing Limit*. Springer Science & Business Media.

136. Stankovic, R. S., & Astola, J. (2011). *From Boolean Logic to Switching Circuits and Automata: Towards Modern Information Technology*. Springer Science & Business Media.

137. Sterling, L., & Shapiro, E. Y. (1994). *The Art of Prolog: Advanced Programming Techniques*. MIT Press.

138. Sullivan, A. (2003). *Logicism and the Philosophy of Language:*

Selections from Frege and Russell. Broadview Press.

139. Syropoulos, A. (2008). *Hypercomputation: Computing Beyond the Church- Turing Barrier*. Springer Science & Business Media.

140. Teuscher, C. (2013). *Alan Turing: Life and Legacy of a Great Thinker*. Springer Science & Business Media.

141. Torrance, S. (2006). The ethical status of artificial agents–with and without consciousness. *Ethics of Human Interaction with Robotic, Bionic and AI Systems: Concepts and Policies. Istituto Italiano per Gli Studi Filosofici, Napoli*, 60–66.

142. Trendelenburg, F. A. (1856). *Über Leibnizens Entwurf einer allgemeinen Charakteristik*. Druckerei der Königl. Akademie der Wissenschaften.

143. Van Heijenoort, J. (1999). *Frege and Gödel: Two Fundamental Texts in Mathematical Logic—PhilPapers*.

144. Van Heijenoort, J. (2002). *From Frege to Gödel: A Source Book in Mathematical Logic, 1879-1931*. Harvard University Press.

145. Van Heijenoort, J. (1967). Logic as calculus and logic as language. *Proceedings of the Boston Colloquium for the Philosophy of Science 1964/1966*, 440–446.

146. Wang, H. (1997). *A Logical Journey: From Gödel to Philosophy*. MIT Press.

147. Whitesitt, J. E. (2012). *Boolean Algebra and Its Applications*. Courier Corporation.

148. Wikibooks. (2017). *Foundations of Computer*

Science/ComputingMachinery.https://en.wikibooks.org/wiki/Foundations_of_Computer_Science/Co mputing_Machinery

149. Wikipedia. (2021). *Automated theorem proving* (p. https://en.wikipedia.org/w/index.php?title=Automated_theorem_proving&oldid=1013812888).

150. William, D. (1984). *On Some Fundamental Distinctions of Computationalism*. University of Western Ontario, Centre for Cognitive Science.

151. Yang, K. (2009). *Boolean Algebra and Digital Logic*. https://www2.southeastern.edu/Academics/Faculty/kyang/2009/Spring/CMPS375/ClassNotes/CMPS375ClassNotesChap03.pdf