МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЛЬВІВСЬКА ПОЛІТЕХНІКА»

Кваліфікаційна наукова
праця на правах рукопису

ЛЕНЬКО ВАСИЛЬ СТЕПАНОВИЧ

УДК 510.6.16+004.89

ДИСЕРТАЦІЯ

МЕТОДИ ТА ЗАСОБИ УПРАВЛІННЯ ПЕРСОНАЛЬНИМИ ЗНАННЯМИ В ІНТЕЛЕКТУАЛЬНИХ СИСТЕМАХ

124 – «Системний аналіз»
12 – «Інформаційні технології»
Подається на здобуття наукового ступеня доктора філософії

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

Ленько В.С.

Науковий керівник:
Пасічник Володимир Володимирович
доктор технічних наук, професор

Львів – 2020

АНОТАЦІЯ

Ленько В.С. Методи та засоби управління персональними знаннями в інтелектуальних системах. – Кваліфікаційна наукова праця на правах рукопису.

Дисертація на здобуття наукового ступеня доктора філософії за спеціальністю 124 «Системний аналіз» (12 – Інформаційні технології). – Національний університет «Львівська політехніка», Львів, 2020.

Зміст анотації

У дисертаційній роботі здійснено системний аналіз методів та засобів управління персональними знаннями як необхідних компонентів для раціональної поведінки інтелектуального агента. Методами системного аналізу досліджено концепт «знання», властивості та ролі поняття «подання знання», еволюцію та множину найпоширеніших моделей подання знання, а також їхні обчислювальні можливості, що втілюють процес міркування. У дослідженні запропоновано нові підходи до застосування формальних логічних моделей, а саме описових логік і теорій типів. Запропоновано метод подання онтологій в програмному середовищі доведення теорем Coq і розроблено моделі баз знань з використанням зазначених логічних формалізмів для управління знаннями в інформаційних системах, мас-медіа та інфраструктурних мережах.

Робота складається зі вступу, чотирьох розділів, висновків, списку посилань та додатків. В першому розділі подано ґрунтовний аналіз релевантних темі дослідження наукових публікацій, проаналізовано епістемологічну структуру поняття знання та акцентовано увагу на зв’язку між істинністю та обґрунтованістю переконання. В цьому ж розділі здійснено порівняльний аналіз найпоширеніших моделей подання знання – фреймів, семантичних мереж, продукційних правил та логічних моделей. Із врахуванням п’яти основних способів подання знань та пов’язаних з ними властивостей, зокрема, виразності і розв’язності, основну увагу дослідження сфокусовано на логічних моделях подання знань.

У другому розділі досліджено логічний формалізм «числення індуктивних конструкцій», який зокрема використовувався для доведення теореми про чотири фарби. Рушієм цього формалізму є теорія типів із залежними типами, яка згідно з ізоморфізмом Каррі-Говарда, втілює різновид систем «природня дедукція» в теорії доведення. З'ясовано, що основними перевагами числення індуктивних конструкцій є його висока виразність та базування на некласичній конструктивній логіці, де поняття істинності виразу ґрунтується на можливості побудови цього виразу з примітивів. Крім того визначено, що теорія типів значною мірою уникає парадоксів класичної логіки та сприяє «механізації» як логічного виведення, так і подання виразів, що не володіють властивістю істинності. Недоліком зазначеного формалізму є його нерозв'язність, яка унеможливлює повну автоматизацію перевірки істинності довільного виразу. Цей недолік вимагає участі дослідника в доведенні теореми, але значну частину перетворень можна автоматизувати за допомогою доступних «тактик». Подання і обґрунтування логічних виразів з використанням числення індуктивних конструкцій здійснюється в інтерактивному асистенті доведення теорем Coq.

Третій розділ подає аналіз описових логік, які є розв'язними фрагментами логіки першого порядку, та широко застосовуються в галузі семантичного вебу, як формальна основа мови онтологій OWL 2. Досліджується структура базової описової логіки *ALC*, допустимі логічні задачі та методи їх розв'язання. З'ясовано, що атрибутивна мова з доповненнями, *ALC*, розширюється новими конструкторами понять та ролей, які формують основу описової логіки $\mathrm{SROIQ}^{(\mathrm{D})}$ – формального рушія мови OWL 2. Формалізм *ALC* використано для подання тверджень з вибраних предметних областей, а для обґрунтування їхньої логічної сумісності застосовано табло-алгоритм. Оскільки описові логіки є підмножинами логіки першого порядку, то їм властиві певні недоліки класичної логіки, зокрема монотонність матеріальної імплікації, яка дозволяє формувати логічно коректні, проте контекстно незв'язні арґументи.

У четвертому розділі проаналізовано особливості проектування системи взаємодії віртуальних наукових колективів, що функціонують з метою побудови баз знань спеціального та загального призначення. Такі спільноти зазвичай розділені географічно та інституційно, тому виникає потреба в системі, яка дозволить надійно та зручно узгоджувати напрацювання учасників віртуального наукового колективу, з можливістю підтвердження історії та авторства будь-якого твердження бази знань. Оскільки, спеціалізація колективу може значно варіюватися, то для подання знань доцільно вибрати мову з високою виразністю, а саме числення індуктивних конструкцій. Для формування і доведення істинності тверджень бази знань пропонується використовувати інтерактивного асистента доведення теорем Coq, який містить мову Gallina та набір тактик для доведення логічних тверджень. Композицію спільної бази знань з напрацювань учасників віртуального наукового колективу доцільно здійснювати за допомогою системи розподіленого контролю версій Git, яка де-факто є стандартом для управління історією програмного коду. Оскільки порядок змін в Git-сховищі можливо редагувати, то для підтвердження авторства важливих змін в базі знань, а також поширення актуальної версії проєкту, пропонується використання технології Blockchain як розподіленого реєстру записів.

Актуальність дослідження арґументується невпинним зростанням обсягів та ускладненням структури інформаційних потоків. Прогрес в галузі технологій числення забезпечив формування нейромережевих моделей фіксації знання, проте їх здатність до абстракції в інформаційному суспільстві є суттєво обмеженою. При цьому, методи концептуалізації та дедуктивні логічні системи пропонують надійні та обґрунтовані формалізми для подання, синтезу та перевірки істинності знання. Розвиток логічних систем уможливив «механізацію», зокрема автоматизацію, процесу міркування над знанням, що сприяє можливості критично аналізувати та проєктувати сучасні інформаційні потоки.

Основні наукові результати дисертації отримані у формі прикладних досліджень:

- розроблено онтологічно-орієнтований метод подання бази знань в середовищі асистента доведення теорем Coq, що сприяє формалізації нових та існуючих онтологій, а також уможливлює застосування логік вищих порядків як інструменту підтримки прийняття рішення в інтелектуальних системах;
- розроблено архітектуру та програмну реалізацію інтелектуальної системи для управління персональними та колективними базами знань, яка володіє властивостями версійності, розподіленості, автономності та забезпечує можливість надійного підтвердження авторства;
- розроблено приклади формального доведення істинності описових знань, зокрема, для задач проектного управління в IT-підприємстві, узгодженості матеріалів в мас-медіа та моделей інфраструктурних мереж, що сприяло перевірці гіпотез в напівавтоматичному режимі;
- здійснено порівняльний аналіз моделей подання знань, описової логіки та формальних систем теорії типів, а також особливостей їх застосування для подання знань різної природи, зокрема, описових речень природньої мови, логічних, математичних теорій та виразів;
- здійснено системний аналіз понять «знання» та «істинність знання», і аргументовано використання конструктивної логіки з метою усунення проблем Ґетьє, які компрометують традиційні методи обґрунтування істинності знання.

Ключові слова

системний аналіз, знання, логіка, числення конструкцій, Coq, описова логіка, онтологія, блокчейн, розподілена система, Git

ABSTRACT

Lenko V.S. Methods and tools for personal knowledge management in intelligent systems. – Qualifying scientific work on the rights of a manuscript.

The dissertation for obtaining a scientific degree of the Doctor of Philosophy on the specialty 124 “System analysis” (12 – Information technologies). – Lviv Polytechnic National University, Lviv, 2020.

Abstract content

The dissertation is devoted to system analysis of methods and tools for personal knowledge management, which is a necessary component for the rational behavior of an intelligent agent. By the methods of system analysis, it investigates the concept of “knowledge”, the properties and roles of “knowledge representation” concept, an evolution of the most common knowledge representation models, as well as their computational capabilities, which resemble the process of human reasoning. The conducted research proposes new approaches to the application of formal logic models, namely descriptive logics and type theories. The method for ontologies representation in the environment of Coq proof assistant is proposed, along with the examples of knowledge base models for the domains of information systems, mass-media, and transport networks.

The thesis consists of an introduction, four chapters, conclusions, a list of references, and appendices. The first section presents a thorough analysis of related scientific publications, analyzes the epistemological structure of knowledge concept, and emphasizes the role of the relationship between truth and validity of belief. The same section provides a comparative analysis of the most widespread knowledge representation models: frames, semantic networks, production rules, and logic-based models. Taking into account five major roles of knowledge representation and their distinguished properties, such as expressiveness and decidability, the main focus of the research is on logic-based models of knowledge representation – the ones accompanied by formal semantics.

The second section investigates the logical formalism called the “calculus of inductive constructions”, which in particular was used for proving the four color theorem. The driving force behind this formalism is the dependent types theory, which according to the Curry-Howard isomorphism syntactically embodies a kind of systems in proof theory named “natural deduction”. It is found that the main advantages of the calculus of inductive constructions are its high expressiveness and the adherence to nonclassical constructive logic principles, according to which the concept of the truth of an expression is based on the ability to construct this concept from primitives. Additionally, the type theory was designed to avoid the paradoxes of classical logic and facilitates the “mechanization” of logical inference, as well as the representation of expressions that do not possess the property of truth. The main drawback of this formalism is undecidability, which makes impossible the full automation of truth verification for an arbitrary expression. This shortcoming requires the participation of a researcher in the process of theorem proving, but still, a significant part of the syntactic transformations can be automated by existing “tactics”. Representation and justification of the logical expressions are carried out in the interactive proof assistant Coq, using the provided implementation of the calculus of inductive constructions.

The third section presents the analysis of description logics, which are decidable fragments of first-order logic and are widely used in the semantic web as a formal basis for the OWL 2 ontology language. It considers the structure of a basic description logic *ALC*, common reasoning problems and method for establishing knowledge base consistency. It is found that the attributive language with complements, *ALC*, can be extended with a set of concepts and roles constructors that form the basis of a highly expressive descriptive logic $\mathrm{SROIQ}^{(\mathrm{D})}$ – the formal core of OWL 2 language. While *ALC* is used to represent the statements, the tableaux algorithm deals with reasoning problems. Since description logics are subsets of first-order logic, they inherit certain shortcomings of classical logic, including the monotonicity of material implication, which allows the formation of logically correct, but contextually incoherent arguments.

The fourth section specifies the design of a collaboration system for virtual scientific communities that aim to build knowledge bases for dedicated and general use. Nowadays, these communities are typically divided geographically and institutionally, so there is a need for a system that will reliably and conveniently reconcile the work of the virtual scientific community members, with the ability to confirm the history and authorship of any expression in the knowledge base. The specialization of a research unit may vary greatly, so it is advisable to choose a language with high expressiveness, namely calculus of inductive constructions. Coq interactive proof assistant provides an implementation of the aforementioned formalism as Gallina language and comes with a set of useful "tactics" for logical inference, so it is proposed to use it for knowledge base engineering. Construction of the shared knowledge base that consists of the parts contributed by the members of a virtual scientific community should be carried out using Git, which is de facto a standard for software code versioning. Since it is technically possible to rewrite the history of commits in a Git repository, it is proposed to establish the authorship of the important knowledge base expressions and share project metadata by the means of Blockchain technology as a distributed ledger system.

The research relevance is justified by the constant growth of information flow volumes and their structural complexity. Advances in computational technology have made it possible to use artificial neural network models for knowledge discovery, but as of today their ability to abstract is severely limited. In contrast, conceptualization methods and logic-based systems offer reliable and sound formalisms for knowledge representation, synthesis, and verification. Curry-Howard isomorphism established the cornerstone relationship between logic-based and computation models and facilitated the "mechanization" of reasoning using verified software-based theorem proving. The ability to detect inconsistencies in multiple statements significantly increases the reliability of the knowledge base, which in turn drastically improves the quality of the input data for the process of intellectual analysis and saves time and efforts of the decision-maker.

The main results of the dissertation are obtained in the form of fundamental and applied research:

- the method for ontologies representation in the environment of Coq proof assistant has been developed, which contributes to the formalization of new and existing ontologies and enables the application of higher-order logics as a reliable decision-support tool;
- the architecture and software implementation of an intelligent system for personal and shared knowledge bases management have been created, which has the properties of being autonomous, distributed, supports versioning and ensures reliable authorship confirmation;
- the examples of knowledge bases and formal reasoning over them have been provided, in particular for the domains of information systems, mass media, and infrastructure networks, which facilitate a semi-automatic procedure of hypotheses testing;
- the comparative analysis of knowledge representation models, description logic, type theories, their characteristic features, and applications to the encoding of the knowledge of the various kinds, including natural language sentences, logical and mathematical expressions, has been conducted;
- the system analysis of the concepts “knowledge”, “truth of knowledge” and the argument on constructive logic adoption for the sake of Gettier problems elimination, which compromise traditional methods of knowledge justification, have been presented.

Keywords

systems analysis, knowledge, logic, calculus of constructions, Coq, description logic, ontology, blockchain, distributed system, Git

Список публікацій здобувача

Наукові праці, в яких опубліковані основні наукові результати дисертації:

1. Кунанець Н. Е., Ленько В. С., Пасічник В. В., Щербина Ю. М. Персональні бази даних та знань віртуальних дослідницьких спільнот. *Науковий вісник НЛТУ України*. 2017. Вип. 27(6). С. 185–191. DOI: https://doi.org/10.15421/40270638.
2. Lenko V., Kunanets N., Pasichnyk V., Shcherbyna Yu. Decentralized Blockchain-based platform for collaboration in virtual scientific communities. *Econtechmod*. 2019. Vol. 8 (1). P. 21–26.
3. Lenko V. S., Pasichnyk V. V., Shcherbyna Y. M. Knowledge representation models. *Вісник Національного університету «Львівська політехніка»*. Серія: Комп'ютерні науки та інформаційні технології. 2017. № 864. С. 157–168.
4. Табачишин Д. Р., Ленько В. С., Кунанець Н. Е., Пасічник В. В., Щербина Ю. М. Експертне оцінювання «розумності міста» із застосуванням нечіткої логіки. *Штучний інтелект*. 2017. № 1 (75). С. 102–110.
5. Ленько В. С., Щербина Ю. М. Застосування методів штучного інтелекту до сегментації графічного образу. *Вісник Національного університету «Львівська політехніка»*. Серія: Інформаційні системи та мережі. 2011. № 715. С. 194–203.
6. Lenko V., Pasichnyk V., Kunanets N., Shcherbyna Y. Knowledge representation and formal reasoning in ontologies with Coq. *Advances in Computer Science for Engineering and Education*. 2018. Vol. 756. P. 759–770. DOI: https://doi.org/10.1007/978-3-319-91008-6_74.
7. Lenko V. Type-theoretical foundations of the derivation system in Coq / V. Lenko, V. Pasichnyk, N. Kunanets, Y. Shcherbyna // *Proceedings of the 2018 IEEE First International Conference on System Analysis & Intelligent*

Computing (SAIC). – IEEE, 2018. – С. 220-225. DOI: https://doi.org/10.1109/SAIC.2018.8516885

Наукові праці, які додатково відображають наукові результати дисертації:

8. Matsiuk O. The procedures of processing of geolocation data on urban underground spaces / O. Matsiuk, N. Kunanets, V. Pasichnyk, V. Lenko, Y. Shcherbyna, A. Rzheuskyi // *Proceedings of the 2019 9th International Conference on Advanced Computer Information Technologies (ACIT).* – IEEE, 2019. – С. 500-503. DOI: https://doi.org/10.1109/ACITT.2019.8780085

Наукові праці, які засвідчують апробацію матеріалів дисертації:

9. Ленько В. С. Теоретичні аспекти логічного міркування у середовищі Coq / В. С. Ленько, В. В. Пасічник, Н. Е. Кунанець, Ю. М. Щербина // *Інтелектуальні системи прийняття рішень і проблеми обчислювального інтелекту: матеріали міжнародної наукової конференції ISDMCI-2018*, 21-27 трав. 2018 р., Залізний Порт, Україна / Херсонський нац. технічний ун-т. – Херсон, 2018. – С. 174-176.
10. Ленько В. С. Проектування відкритих децентралізованих реєстрів з використанням технології Blockchain / В. С. Ленько, В. В. Пасічник, Н. Е. Кунанець, Ю. М. Щербина // *Математика. Інформаційні технології. Освіта: тези доповідей VII міжнар. наук.-практ. конф. МІТО-2018*, 3-5 черв. 2018 р., Луцьк-Світязь, Україна / Східноєвропейський нац. ун-т ім. Л. Українки. – Луцьк-Світязь, 2018. – С. 72-74.
11. Ленько В. С. Міркування в онтологіях з використанням теорії типів / В. С. Ленько, В. В. Пасічник, Ю. М. Щербина, Н. Е. Кунанець // *Теоретичні та прикладні аспекти побудови програмних систем: матеріали XIV міжнар. наук. конф. TAAPSD'2017*, 4-8 груд. 2017 р., Київ, Україна / Київський нац. ун-т ім. Т. Г. Шевченка. – Київ, 2017. – С. 138-141.

12. Ленько В. С. Децентралізована комунікація у віртуальних наукових колективах / В. С. Ленько, Н. Е. Кунанець, В. В. Пасічник, Ю. М. Щербина // *Практичне застосування нелінійних динамічних систем в інфокомунікаціях: матеріали VI міжнар. наук.-практ. конф.*, 9-11 лист. 2017 р., Чернівці, Україна / Чернівецький нац. ун-т ім. Ю. Федьковича. – Чернівці, 2017. – С. 118-119.
13. Кунанець Н. Е. Подання онтологічних моделей знань в системі інтерактивного доведення теорем Coq / Н. Е. Кунанець, В. С. Ленько, В. В. Пасічник, Ю. М. Щербина // *Інформаційні технології та взаємодії: матеріали доповідей IV міжнар. наук.-практ. конф. IT&I-2017*, 8-10 лист. 2017 р., Київ, Україна / Київський нац. ун-т ім. Т. Шевченка. – Київ, 2017. – С. 169-170.
14. Кунанець Н. Е. Подання онтологій з використанням теорії типів / Н. Е. Кунанець, В. С. Ленько, В. В. Пасічник, Ю. М. Щербина // *Системи та засоби штучного інтелекту: тези доповідей Міжнародної наукової молодіжної школи AIIS'2017*, 18 жовт. 2017 р., Київ, Україна / Київський нац. ун-т ім. Т. Шевченка. – Київ, 2017. – С. 114-117.
15. Ленько В. С. Подання знань та логічні міркування у формальних онтологіях/ В. С. Ленько, В. В. Пасічник, Ю. М. Щербина // *Комп'ютерне моделювання та програмне забезпечення інформаційних систем і технологій. (КМПЗ-2017) : збірник наукових праць (тези доповідей і вибрані статті) III Всеукраїнської міжнародної науково-практичної конференції*, 28-30 вересня 2017 р., Рівне, Україна / Національний університет водного господарства та природокористування. – Рівне, 2017. – С. 94-95.
16. Ленько В.С. Проект системи управління персональними знаннями / В. С. Ленько, В. В. Пасічник, Ю. М. Щербина // *Управління проектами: стан та перспективи. (ІТПРР-2017) : матеріали XIII міжнародної науково-*

практичної конференції, 12-15 вересня 2017 р., Миколаїв, Україна / Національний університет кораблебудування імені адмірала Макарова. – Миколаїв, 2017. – С. 61-62.

17. Lenko V. S. On personal knowledge management systems / V. S. Lenko, V. V. Pasichnyk, Y. M. Shcherbyna // *Математика. Інформаційні технології. Освіта: тези доповідей VI міжнар. наук.-практ. конф. МІТО-2017*, 5-7 черв. 2017 р., Луцьк-Світязь, Україна / Східноєвропейський нац. ун-т ім. Л. Українки. – Луцьк-Світязь, 2017. – С. 57-59.
18. Ленько В. С. Побудова моделі «Розумного соціополісу» на основі знань експертів / В. С. Ленько, Н. Е. Кунанець, В. В. Пасічник, Ю. М. Щербина // *Інформаційні технології, економіка та право: стан та перспективи розвитку. (ІТЕП-2017) : матеріали міжнародної науково-практичної конференц*ії, 27-28 квітня 2017 р., Чернівці, Україна / Буковинський університет. – Чернівці, 2017. – С. 141-143.
19. Lenko V. S. Knowledge Representation Models / V. S. Lenko, V. V. Pasichnyk, Y. M. Shcherbyna // *Теоретичні та прикладні аспекти побудови програмних систем: матеріали XIII міжнар. наук. конф. TAAPSD'2016*, 5-9 груд. 2016 р., Київ, Україна / Київський нац. ун-т ім. Т. Шевченка, Ін-т післядипломної освіти. – Київ, 2016. – С. 143-152.
20. Ленько В. С. Методи та засоби управління персональними знаннями / В. С. Ленько, В. В. Пасічник, Ю. М. Щербина // *Інформаційно-обчислювальні технології, автоматика та електротехніка: матеріали Міжнар. наук.-практ. конф. ІТАЕ-2016*, 10-11 листоп. 2016 р., Рівне, Україна / Нац. ун-т водного господарства та природокористування. – Рівне, 2016. – С. 143-144.
21. Ленько В. С. Відновлення графічних образів за допомогою карт Кохонена / В. С. Ленько, Ю. М. Щербина // *Сучасні проблеми прикладної математики та інформатики: збірник наукових праць конф. APAMCS-*

2016, 5-7 жовт. 2016 р., Львів, Україна / Львівський нац. ун-т ім. І. Франка. – Львів, 2016. – С. 108-111.

22. Ленько В. С. Застосування методів штучного інтелекту до сегментації графічних образів / В. С. Ленько // *Чотирнадцята всеукраїнська (дев'ята міжнародна) студентська наукова конференція з прикладної математики та інформатики СНКПМІ-2011: тези доповідей*, 5-6 трав. 2011 р., Львів, Україна / Львівський нац. ун-т ім. І. Франка. – Львів, 2011. – С. 60-61.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

БЗ	база знань
FOL	логіка першого порядку
PAT	висловлювання як типи
Coq	асистент доведення теорем
Gallina	мова специфікації в Coq
CIC	числення індуктивних конструкцій
MTT	сучасні теорії типів
OWL	мова подання онтологій
SROIQ	виразна описова логіка
ALC	атрибутивна мова з доповненнями
TBox	термінологічна частина бази знань
ABox	декларативна частина бази знань
IPFS	розподілена файлова система
IPLD	модель даних контентно-адресованої мережі
Git	розподілена система контролю версій
DID	децентралізований ідентифікатор
P2P	мережа рівноправних вузлів
CID	ідентифікатор контенту

ВСТУП

Актуальність теми. В сучасному інформаційному суспільстві громадяни щоденно зіштовхуються з необхідністю аналізу великої кількості інформації. З джерел різної природи надходить як структурована, так і неструктурована інформація, яку необхідно опрацювати та, у випадку корисності, зберегти для подальшого використання. Варто зазначити, що процес аналізу інформації є достатньо енерго- та часозатратним, тому людина значною мірою використовує механізм пам'яті, задля мінімізації зусиль на повторний аналіз вже опрацьованого матеріалу. Тим не менш, фізіологічні особливості людини не дозволяють повноцінно зберігати та використовувати усі результати аналізу, які вона накопичує впродовж своєї життєдіяльності. Звідси виникає потреба у допоміжних засобах, які б зберігали ці результати, надавали доступ до них та, у вигляді доданої вартості, забезпечували їхню несуперечність.

В дисертаційній роботі досліджуються моделі та методи подання знання, структура знання в контексті епістемології, а також системи отримання логічного висновку як засобу інтеґрації множини тверджень. Визначаються та аналізуються суттєві ознаки систем баз знань, теоретичні та практичні аспекти їхньої організації, властивості та принципові обмеження моделей подання знання, а також логічного міркування. Отримані результати застосовуються до актуальних проблем в галузях інфраструктурних мереж, проєктного управління та мас-медіа, зокрема для аналізу потенційно суперечливих тверджень.

Зв'язок роботи з науковими програмами, планами, темами. Тема дисертації відповідає науковому напрямку кафедри інформаційних систем та мереж Національного університету «Львівська політехніка»: «Дослідження, розроблення і впровадження інтелектуальних розподілених інформаційних технологій та систем на основі ресурсів баз даних, сховищ даних, просторів даних та знань з метою прискорення процесів формування сучасного інформаційного суспільства».

Дисертаційне дослідження виконано в межах держбюджетних науково-дослідних робіт:

- «Методи та засоби функціонування систем підтримки прийняття рішень на основі онтологій» (номер державної реєстрації 0118U000269; терміни виконання роботи: 01.2018-12.2019 рр.);
- «Система підтримки прийняття рішень розпізнавання мультиспектральних образів на основі технологій машинного навчання та онтологічного підходу» (номер державної реєстрації 0120U102203; терміни виконання роботи: 04.2020-12.2021 рр.).

Мета і завдання дослідження. Метою дисертаційного дослідження є розвиток методів та засобів для управління персональними знаннями в інтелектуальних системах, які забезпечать ефективне подання, узгодження, актуалізацію та розширення баз знань, з метою їх практичного застосування в системах підтримки прийняття рішення.

Для досягнення поставленої мети вирішено такі завдання:

— здійснено системний аналіз понять «знання», «істинність знання» та «міркування» в контексті епістемології та формальної логіки;

— здійснено порівняльний аналіз властивостей найпоширеніших моделей подання знання та міркування;

— досліджено методи та засоби структурування і логічного узгодження знання різної природи;

— розроблено онтологічно-орієнтований метод подання бази знань в середовищі асистента доведення теорем Coq;

— спроєктовано систему управління персональною та колективною базами знань, що ґрунтується на технологіях децентралізації та контентної адресації;

— подано інтерпретацію результатів дисертаційного дослідження у вигляді висновків та рекомендацій.

Об’єктом дослідження є процес управління персональними знаннями в інтелектуальних системах.

Предметом дослідження є методи та засоби управління персональними знаннями в інтелектуальних системах.

Методи дослідження. Для розв’язання сформульованих задач застосовано методи: системного аналізу, порівняльного аналізу, концептуального моделювання, дедукції, синтезу та абстракції. Методами системного та порівняльного аналізу досліджено моделі подання знання, системи теорій типів, базова описова логіка ALC, на предмет їх структури та застосування, з метою виявлення їхніх переваг та недоліків. Концептуальне моделювання застосовано під час побудови онтологій предметних областей, з використанням формалізмів описової логіки і теорій типів. Методи синтезу та абстракції використано для побудови архітектури та програмної реалізації розподіленої бази знань.

Наукова новизна отриманих результатів. Новизна основних результатів дисертації формується завдяки теоретико-практичному підходу до проєктування баз знань з використання формального апарату теорії типів та ґрунтованому аналізу понять «знання», «істинність знання» та «міркування» в контексті епістемології, формальної логіки та концептуального моделювання.

Вперше одержано онтологічно-орієнтований метод подання бази знань в середовищі асистента доведення теорем Coq, що сприяє формалізації нових та існуючих онтологій, а також уможливлює застосування логік вищих порядків як надійного інструменту підтримки прийняття рішення.

Удосконалено методи та засоби подання, управління, зберігання та поширення знання в межах систем персональних та колективних баз знань.

Удосконалено методологію вибору формальних систем для подання та міркування над знаннями, яка ґрунтується на аналізі ознак виразності і розв’язності.

Удосконалено трактування понять «знання», «міркування» та «істинність знання» в контексті методології проектування баз знань.

Набули подальшого розвитку підходи до використання теорії типів і конструктивної логіки для формального подання описового знання та обчислення логічного висновку в середовищі асистента доведення теорем Coq.

Практичне значення отриманих результатів дисертаційного дослідження полягає у можливості формалізації довільної онтології знань, а також надійному обчисленні логічного висновку завдяки дедуктивній системі асистента доведення теорем Coq. Результати дослідження можуть бути використані у різних галузях народного господарства, зокрема для:

- будівництва та обслуговування інфраструктурних мереж, зокрема для перевірки коректності наявних план-схем на предмет узгодження з'єднань та властивостей структурних елементів мережі;
- проєктного управління в галузі інформаційних технологій, зокрема для визначення узгодженості майбутніх модулів системи з властивостями існуючих модулів;
- перевірки інформаційних новин на достовірність, зокрема для виявлення інформаційних повідомлень, які не узгоджуються з наявною базою знань користувача;
- проєктування надійної інформаційної системи для співпраці віртуального наукового колективу, з використанням сучасних технологій розподіленого зберігання та доступу до знання.

Особистий внесок здобувача. Дисертація є самостійною науковою працею, в якій автором особисто розроблено нові наукові ідеї та результати, що дозволили вирішити наукове завдання формального подання та управління персональними знаннями. Робота містить прикладні положення та висновки, сформульовані дисертантом особисто. Ідеї, положення чи гіпотези інших авторів, які присутні в дисертації, мають відповідні посилання і використані лише для підкріплення ідей та результатів здобувача.

Апробація результатів дисертації. Результати дисертаційного дослідження апробовано на міжнародних наукових та науково-практичних конференціях, наукових школах та консорціумах, семінарах:

- 9th International Doctoral Consortium “Informatics Engineering Education Research”, 5-9 December 2018, Druskininkai, Lithuania.
- The First International Conference on Computer Science, Engineering and Education Applications (ICCSEEA2018) 18-20 January 2018, Kyiv, Ukraine.
- IEEE First International Conference on System Analysis & Intelligent Computing (SAIC) 08-12 October 2018, Kyiv, Ukraine.
- 9th International Conference on Advanced Computer Information Technology (ACIT'2019) 5-7 June 2019, České Budějovice, Czech Republic.
- VII Міжнародна науково-практична конференція «Математика. Інформаційні технології. Освіта» 3-5 червня 2018 р., Луцьк-Світязь, Україна.
- XIV міжнародна наукова конференція «Інтелектуальні системи прийняття рішень та проблеми обчислювального інтелекту», 21-27 травня 2018 р., Залізний Порт, Україна.
- XIV міжнародна науково-практична конференція «Теоретичні та прикладні аспекти побудови програмних систем (TAAPSD'2017)», 4-8 грудня 2018 р., Київ, Україна.
- VI міжнародна науково-практична конференція «Практичне застосування нелінійних динамічних систем в інфокомунікаціях», 9-11 листопада 2017 р., Чернівці, Україна.
- IV міжнародна науково-практична конференція «Інформаційні технології та взаємодії (IT & I 2017)», 8-10 листопада 2017 р., Київ, Україна.
- Міжнародна наукова молодіжна школа «Системи та засоби штучного інтелекту (AIIS’2017)», 17-19 жовтня 2017 р., Київ, Україна.

- III Всеукраїнська міжнародна науково-практична конференція «Комп'ютерне моделювання та програмне забезпечення інформаційних систем і технологій (КМПЗ- 2017)», 28-30 вересня 2017 р., Рівне, Україна.
- XIII міжнародна науково-практична конференція «Управління проектами: стан та перспективи (ІТПРР-2017)», 12-15 вересня 2017 р., Миколаїв, Україна.
- VI Міжнародна науково-практична конференція «Математика. Інформаційні технології. Освіта» 5-7 червня 2017 р., Луцьк-Світязь, Україна.
- Міжнародна науково-практична конференція молодих науковців та студентів «Інформаційні технології, економіка та право: стан та перспективи розвитку (ІТЕП- 2018)», 27-28 квітня 2017 р., Чернівці, Україна.
- XIII міжнародна науково-практична конференція «Теоретичні та прикладні аспекти побудови програмних систем (TAAPSD'2016)», 5-9 грудня 2018 р., Київ, Україна.
- Міжнародна науково-практична конференція молодих науковців, аспірантів та студентів «Інформаційно-обчислювальні технології, автоматика та електротехніка (ІТАЕ-2016)», 10-11 листопада 2016 р., Рівне, Україна.
- XXII Всеукраїнська наукова конференція «Сучасні проблеми прикладної математики та інформатики (АРАМСS-2016)», 5-7 жовтня 2016 р., Львів, Україна.
- XIV Всеукраїнська (IX міжнародна) студентська наукова конференція з прикладної математики та інформатики СНКПМІ-2011, 5-6 травня. 2011 р., Львів, Україна.
- Наукові семінари кафедри інформаційних систем та мереж (2017-2020 рр.).

Публікації. У 22 наукових публікаціях повністю відображені основні результати дисертації, з них: 1 стаття у науковому фаховому виданні України, яке включено до міжнародної наукометричної бази (Index Copernicus); 1 стаття у науковому періодичному виданні іншої держави; 3 статті у наукових фахових виданнях України; 3 матеріали у збірниках конференцій, які включено до міжнародної наукометричної бази (Scopus); 14 наукових публікацій у збірниках матеріалів та тез конференцій.

Структура й обсяг дисертації. Дисертаційна робота викладена на 151 сторінці та складається з анотації, змісту, переліку скорочень, вступу, чотирьох основних розділів, в яких міститься 19 рисунків та 6 таблиць, списку використаних джерел з 106 найменувань, а також 2 додатків. За структурою, мовою та стилем викладення дисертація відповідає вимогам МОН України. Робота написана грамотною українською мовою з використанням сучасної наукової термінології, а стиль викладення матеріалу є послідовним та логічним.

РОЗДІЛ 1. СПОСОБИ ПОДАННЯ ЗНАНЬ В ІНТЕЛЕКТУАЛЬНИХ СИСТЕМАХ

Знання як усвідомлена інформація є одним з ключових факторів розвитку суспільства. З позицій прагматизму, істинне знання сприяє досягненню бажаних та уникненню небажаних станів, де стани та їхня класифікація формуються суб'єктом еволюційно, зокрема під час життєдіяльності. Джерелами набуття знання є власний досвід, що формується з відчуттів та умовиводів, а також інформація, отримана від інших суб'єктів в символьній, графічній, звуковій та інших формах подання. Варто зазначити, що будь-яка одиниця інформації зазвичай породжує для різних суб'єктів відмінні за структурою та функцією знання, оскільки її інтерпретація залежить від контексту – множини знань та цілей (бажаних і небажаних станів), які властиві суб'єкту при здійсненні інтерпретації.

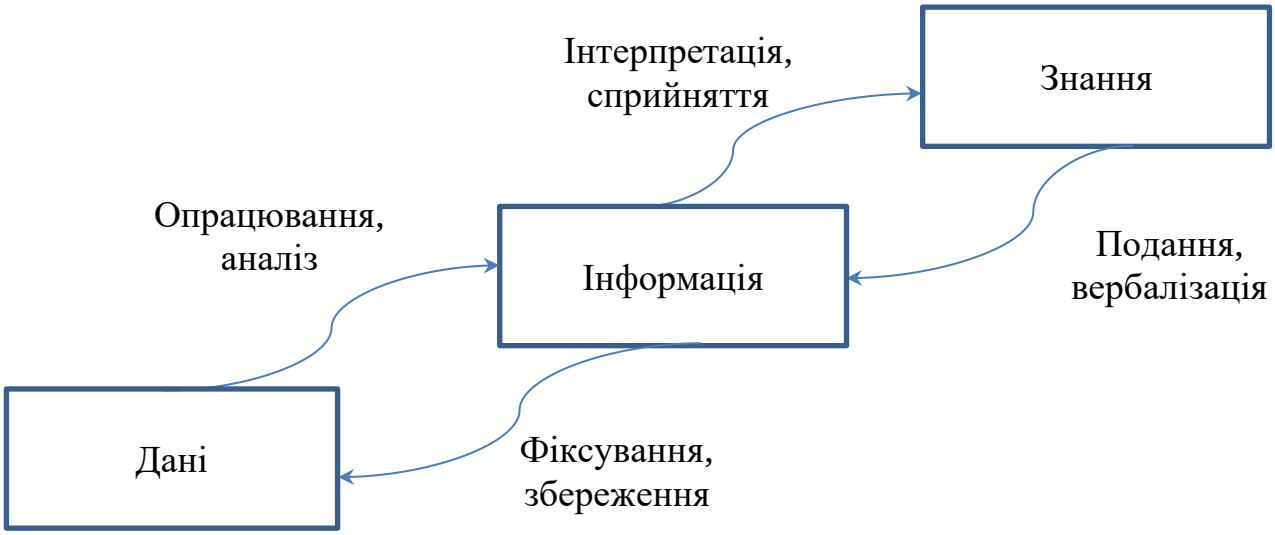

Рис. 1.1. Співвідношення понять «знання», «інформація» та «дані» [1]

Втілення амбітних цілей потребує використання значної кількості ресурсів за відносно обмежений період часу, тому співпраця між зацікавленими суб'єктами є важливим фактором для їхньої реалізації. Співпраця суб'єктів з метою досягнення деякої цілі передбачає наступні етапи: 1) формулювання мети; 2) формулювання

завдань; 3) планування послідовності дій; 4) розподіл завдань між зацікавленими суб'єктами; 5) виконання завдань, з активним використанням зворотнього зв'язку. Необхідною умовою для успішного втілення цілі в умовах співпраці є наявність спільної, недвозначної інтерпретації мети та завдань, які здебільшого подаються у вигляді інформації певної форми. Оскільки результат інтерпретації залежить від контексту суб'єкта, то для узгодженого розуміння інформації і, як наслідок, ефективної координації дій, необхідно зменшити вплив контексту суб'єкта на результат інтерпретації.

Розглянемо можливість обмежити вплив контексту суб'єкта на результат інтерпретації інформації. Нижче наведені спостереження дозволяють припустити існування такої можливості. По-перше, контекст для інтерпретації інформації можна задавати явно – означення понять, відношень, позначень, які присутні в описових твердженнях, варто означати за допомогою додаткової інформації. Такий підхід надає суб'єкту пріоритетне джерело для визначення контексту існування інформації; як правило, деталізація контексту зменшує кількість допустимих моделей інтерпретації [2]. Методи та засоби онтологічної інженерії [3] найкраще використовуються для практичного втілення зазначеного підходу. По-друге, застосування інформації, яку складно інтерпретувати в межах еволюційно набутого контексту, є можливим лише за умови явного подання самого контексту. Ця риса спостерігається в теорії формальних мов, яка використовується, зокрема, для проєктування та аналізу мов програмування. В результаті, забезпечення інформації явним контекстом і використання формальних мов для його подання, дозволяють обмежити вплив контексту суб'єкта на результат інтерпретації.

Основою для побудови виразів формальної мови є формальна граматика – набір аксіом для створення та перетворення синтаксичних виразів мови. Форма цих правил безпосередньо впливає на можливість побудови множини допустимих синтаксичних виразів і характеризується поняттям «експресивність» мови подання. Ієрархія формальних граматик за Н. Чомскі [4] передбачає наявність чотирьох типів

граматик, кожен з яких розширює попередній тип в ієрархії, але жоден з них не можна порівняти за «експресивністю» з природньою мовою. Формальні мови компенсують цю різницю наявністю вагомих обчислювальних властивостей, зокрема, можливістю повної або часткової автоматизації процесу розв'язання деяких задач міркування в системах баз знань.

1.1. Гіпотеза подання знання

Фундаментальною парадигмою в проєктуванні баз знань прийнято вважати гіпотезу подання знання, яку в 1982 році у своїй дисертаційній роботі сформулював філософ Б. Сміт [5]:

Будь-який механічно втілений інтелектуальний процес буде складатися з структурних інгредієнтів, які:

1) *ми, як зовнішні спостерігачі, природно приймаємо для представлення пропозиційного аспекту знання, що демонструє загальний процес,* та
2) *незалежно від зовнішнього смислового приписування, відіграють формальну, але причинно-наслідкову і істотну роль у породженні поведінки, що проявляє це знання.*

Приймаючи цю гіпотезу, можна вважати системи, засновані на знаннях, такими, які задовольняють її за будовою. Згідно з поданою гіпотезою, структури в системах, заснованих на знаннях, мають задовольняти дві основні вимоги [6]:

1) наявність можливості інтерпретувати їх як висловлювання, які в сукупності становлять усі знання системи (структури повинні бути виразами мови, якій властива теорія істинності, причому синтаксичні вимоги до їх форми, наприклад у вигляді речення, не встановлюються);
2) відігравання символьними структурами причинно-наслідкової ролі в поведінці цієї системи (коментарі у мовах програмування не задовольняють зазначену вимогу).

Формулювання гіпотези подання знань є відображенням уявлення про спосіб подання знання та його вагому роль у формуванні поведінки інтелектуального агента. Оптимальні моделі подання та методи оперування знанням для формування ефективної поведінки агента є важливими проблемами, які досліджуються ще з часів Сократа. Варто зазначити, що результатами наукових досліджень зазначених проблем є положення, принципи, теорії, гіпотези, моделі, методи та спостереження, які належать до різних галузей наук, зокрема філософії, штучного інтелекту, математичної логіки, теорії ігор, економіки, психології. В дисертаційній роботі основний фокус зосереджено на методах та засобах подання і управління знаннями в контексті систем штучного інтелекту.

1.2. Поняття «знання» та його структура

Під час системного аналізу поняття знання, визначається суть «знання». Варто зазначити, що це фундаметальна проблема в епістемології, і наразі немає чіткого шляху її вирішення. Епістемологія – це одна з чотирьох основних галузей філософії, яка досліджує об’єкт «знання» на предмет його природи, структури, джерел, обмежень і зв’язків, зокрема з таким поняттям як «істина». Філософи розрізняють три основні типи знання [7]:

1) Декларативне знання («знати-що»): описові твердження – висловлювання (propositions), які описують певний аспект дійсності. Декларативні знання також називають «описовими» знаннями;
2) Процедурне знання («знати-як»): знання як здатність щось зробити, навик;
3) Знання через знайомство: це знання сформоване з особистого досвіду чи шляхом пізнання об’єкта завдяки безпосередній взаємодії.

Значна частина сучасної та античної епістемології зосереджена на дослідженні декларативного знання, через можливість його поширення між суб’єктами. Гіпотеза подання знання передбачає наявність саме декларативного знання, тому доцільно розглянути його властивості детальніше.

Декларативні знання зручно подавати за допомогою конструкції «*S* знає що *p*», де *S* позначає суб'єкта, що має знання, а *p* – висловлювання, що є відоме [8]. Постає питання, які умови є необхідними і достатніми для *S*, щоб *знати*, що *p*? Згідно з традиційним підходом до відповіді на це питання, початок якого сягає часів Платона, *знати що p* – це «істинне обґрунтоване переконання». Відповідно «*S* знає що *p*», якщо виконуються наступні умови:

1. висловлювання *p* є істинним;
2. *S* переконаний що *p*;
3. переконання *S* що *p* є обґрунтоване.

В сукупності ці умови приймаються як необхідні та достатні для визначення знання. Якщо вони задовольняються, тобто *p* є істинним, і *S* переконаний що *p*, і це переконання є обґрунтоване, тоді «*S* знає що *p*». Крім цього, якщо «*S* знає що *p*», то *S* має істинне обґрунтоване переконання що *p*. Таким чином стверджується, що декларативне знання є еквівалентне істинному обґрунтованому переконанню.

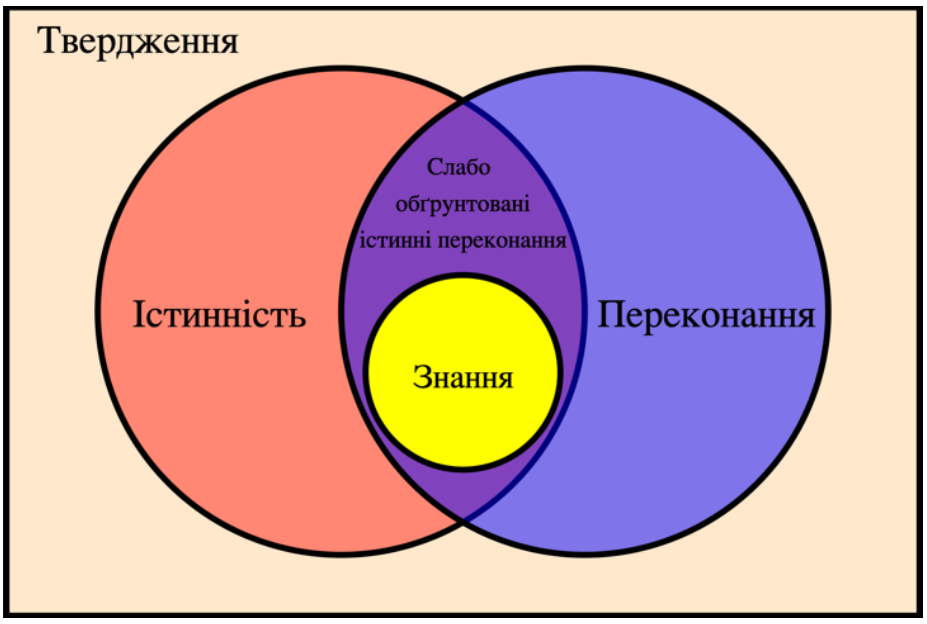

Рис. 1.2. Знання подане у вигляді кіл Ейлера

Проаналізуємо роль кожної з умов в контексті відмінності знання від переконання. Перша з умов стверджує, що висловлювання *p*, яке *S* знає, є істинним,

тобто не є хибним. Звісно людина може *думати*, що знає *p*, а згодом виявляється, що *p* хибне; проте це не знання, а хибна думка – людина помилялася, коли думала що знає *p*. Друга умова стверджує: *S* має бути переконаним що *p* для того, щоб знати що *p*. Хоча деякі філософи вважають, що знання і переконання є абсолютно різними ментальними станами, описана ситуація з хибною думкою дозволяє стверджувати про існування переконання, яке помилково приймалося за знання. Таким чином їхня подібність дозволяє обґрунтовано вважати, що знання є видом переконання.

Третя умова вимагає, щоб переконання було обґрунтоване – це дозволяє відрізнити знання від істинного переконання, яке існує завдяки успішному збігу обставин. Згідно з традиційним підходом, переконання *S* у тому, що *p* є істинним, не є успішним збігом обставин, коли воно є розумне або раціональне з точки зору *S*, для того щоб прийняти *p* за істину. У теорії очевидності (evidentualism) *володіння доказами* робить переконання обґрунтованим. Основна ідея полягає в тому, що переконання є обґрунтованим в тій мірі, в якій воно задовольняє докази *S*. У нетрадиційному підході обґрунтування гарантує, що переконання *S* має високу об’єктивну ймовірність істини і саме тому не є успішним збігом обставин. Одна з провідних ідей полягає в тому, що високої об’єктивної ймовірності істини можливо досягнути лише тоді, коли переконання зароджується в *надійних пізнавальних процесах* та *джерелах*. Ця позиція відома як надійність (reliabilism).

Варто зазначити, що обидва способи обґрунтування істинного переконання не є достатніми для знання, оскільки їм властиві проблеми, сформульовані Ґетьє. Їх суть полягає в тому, що існують приклади істинного обґрунтованого переконання, де істинність є успішним збігом обставин, тобто три необхідні умови не є достатніми. Проблеми Ґетьє наочно демонструють, що зв’язок між тим, що робить висловлювання істинним і тим, що обґрунтовує його істинність, потребує кращої деталізації. В науковій роботі «Intuitionistic Epistemic Logic» [9] запропоновано інтуїціоністський підхід до трактування поняття «знання», для якого, згідно з твердженнями авторів, не властиві проблеми Ґетьє. Це дозволяє розглянути

істинність знання в контексті інтерпретації Брауера-Гейтінга-Колмогорова, а також здійснити побудову конструктивного доведення його істинності з використанням ізоморфізму Каррі-Говарда та виразних формальних систем типізованого лямбда-числення.

1.3. Способи подання знань

Інтелект, що властивий людині, є феноменом, який сучасна наука досі не може точно пояснити. В галузі штучного інтелекту, одне з численних означень має наступний вигляд: інтелект – це процес логічного висновку (міркування), що застосовується до збережених знань [10]. Важливим у визначенні є те, що воно підкреслює існування двох категорій, знання та міркування, які взаємопов'язані; крім цього наявність знань є необхідною для здійснення процесу міркування. Таким чином, задача набуття, зберігання і оновлення знання є першочерговою в процесі побудови інтелектуальної системи.

Після набуття знань їх доцільно зберігати для подальшого використання в міркуваннях. База знань (БЗ), як правило, складається з описових тверджень, які суб'єкт вважає істинними. Структура тверджень може змінюватися, від простого декларативного речення до складної онтології. Спосіб організації тверджень у БЗ суттєво впливає на механізми отримання висновків, що зазвичай застосовуються до збережених знань. Системний аналіз моделей та методів подання знань досліджує їхні структурні характеристики, які зокрема впливають на якість концептуалізації предметних областей та властивості методів міркування. Серед множини критеріїв, що використовуються для порівняння моделей та методів подання знання, варто виокремити виразність подання та можливість отримання надійних висновків.

Найпоширеніші способи подання знань прийнято поділяти на чотири класи: логічні моделі, семантичні мережі, фреймові моделі та продукційні правила [11]. Логічні моделі є формальними системами, що містять мову подання та дедуктивну систему – інструменти для опису знання і міркувань над ним. В поточному розділі

детально розглянуто логіку першого порядку і теорії типів, які відомі також як логіки вищого порядку. Семантичні мережі використовують наочний графовий підхід до подання знання, де вузли графа відповідають поняттям, а дуги позначають зв'язок між ними. Найпоширенішими видами семантичних мереж є: означальна, стверджувальна, наслідкова, виконавча, навчальна та гібридна [12]. Фреймові моделі базуються на об'єктно-орієнтованому підході до подання знань, в якому знання структуровані навколо поняття «об'єкт». Наявність «об'єкта» дозволяє групувати пов'язані властивості та процедури, що полегшує опис та управління знаннями. При цьому особливий вид фреймів – сценарії – використовуються для опису стереотипної послідовності подій в певному контексті [13]. Продукційні правила подають знання як сукупність правил «якщо-то» та завдяки своїй структурі, забезпечені методами автоматизованого та напівавтоматизованого міркування. Найвідомішим втіленням моделей, що засновані на продукційних правилах, є експертні системи.

Сучасні дослідження в галузі подання знання та міркування (KR&R) здійснюються в напрямках онтологічної інженерії і теорії типів. Онтологія як метод структурування знань поєднує в собі риси вищезазначених моделей подання знання та забезпечена множиною програмних реалізацій, зокрема Protege, Cyc, Dublin Core та інші. Деякі оптимізаційні рішення щодо розширення онтології пропонуються в науковій праці «Задачі оптимізації структури та змісту онтології та методи їх розв'язування» [14]. Теорії типів є важливим різновидом формальної логіки. З моменту відкриття ізоморфізму Каррі-Говарда теорії типів викликають активний інтерес наукової спільноти. Зусилля науковців спрямовані на дослідження в області гомотопічної теорії типів [15].

1.3.1. Логіка першого порядку

Логіка першого порядку (FOL) – це сукупність формальних систем, що складаються з формальної мови та дедуктивної системи, для подання знання та

міркування. Існує багато різновидів FOL, зокрема конструктивна (інтуїціоністська), багатосортова, модальна, тощо. Найпоширеніше використання властиве класичній логіці предикатів, тому доцільно проаналізувати її детальніше. FOL є стандартом формалізації математичних теорій, а також відіграє важливу роль у поданні знань довільної природи. Логіка предикатів є еволюцією логіки висловлювань, яка не має достатньої виразності для подання знань дійсності. Основна різниця між логікою першого порядку та логікою висловлювань полягає в онтологічному зобов'язанні: FOL, як і природна мова, припускає, що світ містить об'єкти, зв'язки та функції, в той час як логіка висловлювань оперує виключно атомарними твердженнями. FOL використовується для подання логічних взаємозв'язків та міркувань, а також дозволяє розділити логічне твердження на терми [16]. Як приклад, розглянемо два логічні твердження «Черч – математик» і «Тюрінг – математик». В логіці висловлювань ці твердження розглядаються як не пов'язані між собою, а в логіці предикатів можна ввести предикат «*Математик(x)*», який робить зв'язок між твердженнями явним.

Формальна мова FOL має синтаксис та забезпечена стандартною семантикою, теорією моделей. Синтаксис визначає, які послідовності символів з алфавіту є допустимими виразами, а семантика присвоює значення цим виразам. Алфавіт – це набір символів, з яких можуть утворюватися вирази формальної мови. В логіці предикатів символи алфавіту поділяються на дві групи: логічні та нелогічні; різниця між ними полягає в тому, що логічні символи завжди мають однакове значення, тоді як значення нелогічних символів залежить від інтерпретації. Логічні символи в FOL представлені кванторами ($\forall$, $\exists$), логічними сполучниками ($\wedge$, $\vee$, $\Rightarrow$, $\Leftrightarrow$, $\neg$), пунктуацією (дужки), змінними (нескінченна множина), рівністю (=). Нелогічні символи складаються з предикатів та функцій з валентністю (кількість аргументів) більшою або рівною нулю. Предикат валентності 0 відповідає змінній в логіці висловлювань, а функцію валентності 0 називають постійною. Правила формування у FOL визначають два типи допустимих синтаксичних виразів, терми та формули.

Терми представлені змінними та функціями, тоді як формули складаються з предикатів (приймають терми як аргументи), символу рівності (оперує термами), логічних сполучних елементів і кванторів. Змінна у формулі може бути обмеженою або вільною (або в обох станах), залежно від того, чи входить вона в область квантора. Формула без вільних змінних називається реченням першого порядку. Речення першого порядку – це формули, які мають чітко визначені значення істинності в інтерпретації, і саме вони використовуються для подання знання; все інше присутнє для підтримки синтаксичного механізму. Семантика логіки першого порядку означається інтерпретацією, яка задає предметну область і трактування нелогічних символів.

Міркування в FOL здійснюються в рамках надійної дедуктивної системи, яка використовується для того, щоб показати, що певна формула є логічним наслідком іншої формули. Здебільшого дедуктивні системи складаються з логічних аксіом (можуть бути відсутні) та правил виведення, які зокрема можна застосовувати для породження теорем. Ці системи мають важливу рису – вони здійснюють виключно синтаксичні перетворення, які забезпечують перевірку коректності формул, без необхідності означення інтерпретації. Існує різноманіття дедуктивних систем для логіки першого порядку, а саме: Гільбертові системи, природна дедукція, числення послідовностей, табло-метод, метод резолюцій. Дедуктивна система називається обґрунтованою, якщо вона породжує виключно логічно істинні формули – ті, які гарантовано будуть логічним наслідком. Дедуктивну систему називають повною, якщо вона гарантує синтаксичне виведення усіх логічно істинних формул з аксіом. Розв’язність дедуктивної системи забезпечує уникнення зупинки роботи алгоритму, що розв’язує проблему коректності логічного висновку. Жоден автоматизований процес міркувань в FOL не може бути обґрунтованим та повним, оскільки задача міркування «чи є довільне речення логіки предикатів істинним» є загалом нерозв’язна [17, 18].

Логіка першого порядку є інструментом подання знань та міркувань, оскільки їй притаманні багато металогічних властивостей, які відсутні в сильніших логіках. За теоремою Ліндстрема, FOL – це максимальна логіка, яка задовольняє низхідні властивості теореми Левенгейма-Сколема та властивості зліченної компактності [19]. Разом з тим, FOL має суттєві обмеження. По-перше, їй властивий парадокс Сколема, який стверджує, що нескінченні структури (тобто лінія, натуральні числа) не можуть бути категорійно аксіоматизовані в логіці першого порядку; ця проблема вирішена в більш виразних логіках, таких як логіка другого порядку. По-друге, виразності FOL недостатньо для подання деяких частин речення природної мови, зокрема дієслівних присудків та відносних прикметників, а також властивостей, що потребують квантора над предикатами. Зрештою, обчислювальна складність FOL є одним з головних факторів, що породжує необхідність розгляду альтернативних варіантів подання знань та міркувань.

1.3.2. Теорії типів (логіки вищих порядків)

Початково представлені Б. Расселом, теорії типів відіграють фундаментальну роль як в логіці, так і в математиці [20]. Вони з’явилися як відповідь на парадокс, виявлений Б. Расселом у наївній теорії множин, який демонструє, що належність до «множини всіх множин, які не є елементами себе», є суперечністю. Після того, як парадокс був відтворений у системі Фреже шляхом використання розширення предикатів як аргументів до предикатів, Г. Фреже зазначив, що вираз «предикат предикований на собі» не є точним. Г. Фреже відзначив різницю між предикатами та об’єктами, вважаючи, що предикат може застосовуватися лише до об’єкта і не може приймати предикат як аргумент. Наявності відмінності між об’єктами, предикатами та предикатом предикатів достатньо, щоб заблокувати парадокс. Цю ієрархію Б. Рассел називав «розширеною ієрархією типів», а її необхідність була визнана наслідком парадоксу Рассела [21].

Фундаментальна робота з формальної логіки «Principia Mathematica» [20] визначає структуру типу наступним чином:

- i – тип індивідів, () – тип тверджень;
- якщо $A_1 \dots A_n$ типи, тоді $(A_1 \dots A_n)$ є n-арне відношення між об'єктами цих типів.

Оскільки предикат класифікується як n-арне відношення типу $(A_1 \dots A_n)$, очевидно, що він не може приймати аргументи типу (A), зокрема предикати, тим самим уникаючи парадоксу Рассела. Відповідно до зазначеної структури можна також визначити типи двійкових відношень (i, i), двійкові сполучники $((), ())$ та квантори над індивідами $\big((i)\big)$. А. Черч запропонував власне формулювання теорії типів, яке ґрунтується на просто-типізованому λ-численні; воно дало можливість вводити функції як примітивні об'єкти та використовувати їх в аргументах і результатах. Типи формальної системи Черча визначаються наступним чином [22]:

- i – тип індивідів, o – тип тверджень;
- якщо A, B типи, тоді $A \to B$ є типом функцій з A в B.

Ця структура дозволяє сформулювати наступні типи: $i \to i$ (функції), $(i \to i) \to i$ (функціонали), $i \to o$ (предикати), $(i \to o) \to o$ (предикати предикатів).

Відкриття ізоморфізму Каррі-Говарда дозволило встановити взаємозв'язок між двома сімействами формалізмів, системами доведень і моделями числення, а саме інтуїціоністською логікою висловлювань та просто-типізованим λ-численням. В роботі [23] В. Говард описує відповідність між сполучниками висловлювань і типами виразів в лямбда-численні, а також описує поняття «залежних типів», які відповідають кванторам предикату. Введення залежних типів дозволяє представити доведення у логіці предикатів за допомогою типізованого λ-числення. З'являються нові концепції програмування: «висловлювання як типи», «доведення як програми» та «спрощення доведень як оцінка програм» [24]. Їхній розвиток дав поштовх для розробки нового виду програмного забезпечення з назвою «асистенти доведення теорем», до яких належать такі середовища як «Coq», «NuPRL», «Agda», «Idris».

Співвідношення Каррі-Говарда-Ламбека поєднало теорію доведень, теорію типів і теорію категорій через ізоморфну інтерпретацію понять інтуїціоністської логіки, типізованого λ-числення та декартових замкнутих категорій. Найактивніші дослідження здійснюються в гомотопічній теорії типів (HoTT), яка пов'язує топологію з «твердженнями як типами».

1.3.3. Семантичні мережі

Семантична мережа – це модель подання знань, яка використовує графову структуру для опису знань предметної області. Поняття, індивіди та властивості відповідають вузлам графа, а зв'язки між ними задаються орієнтованими дугами. Формально семантична мережа задається кортежем $< I, R_1, R_2 \dots R_n, M >$, де I позначає множину інформаційних одиниць, які пов'язані між собою певними типами зв'язків $R_1, R_2 \dots R_n$, а M відповідає карті, що встановлює їх інтерпретацію [14]. До найпоширеніших видів семантичних мереж належать: стверджувальна, означальна, наслідкова, виконавча, навчальна та гібридна [12]. Незважаючи на суттєві відмінності в проєктуванні та застосуванні зазначених видів семантичних мереж, усі вони використовують декларативно-графічний підхід до подання знання.

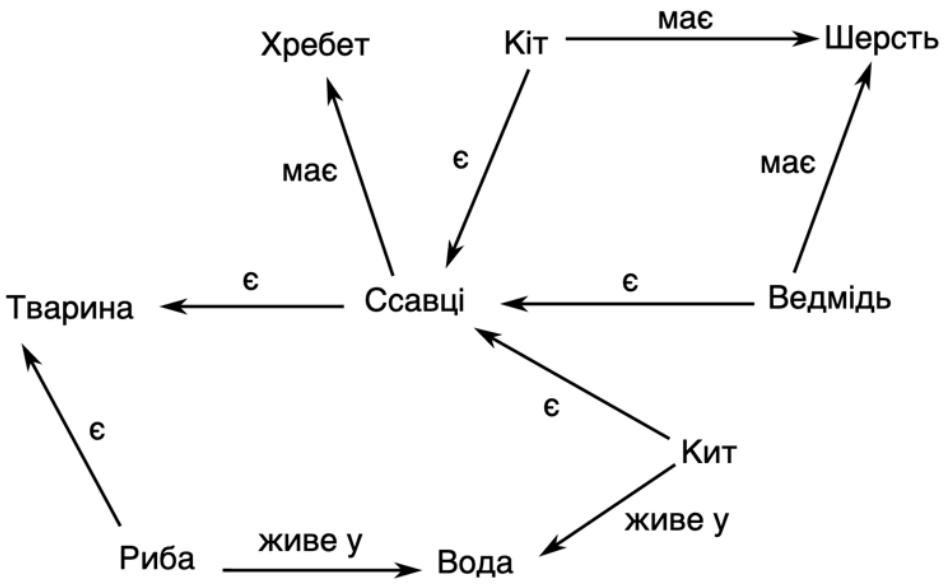

Рис. 1.3. Приклад означальної семантичної мережі

Розглянемо детальніше структурні і функціональні властивості кожного виду. Означальні мережі застосовуються для опису понять, індивідів та відношень між ними. Здебільшого вони використовують відношення виду «є», «має», «екземпляр», щоб сформувати таксономію знань, яка є необхідною для можливості здійснення міркування. Семантична мережа припущень подає структуру логічних тверджень графічно, тому її елементам властиві семантичні ролі, що відповідають логічним операторам, таким як квантор існування, кон'юнкція, диз'юнкція, заперечення, тощо. Семантична мережа наслідків є підвидом пропозиційної семантичної мережі, яка використовує імплікацію як основне відношення для з'єднання вузлів; інші відношення можуть бути вкладені всередину пропозиційних вузлів, але вони ігноруються процедурами виведення. Виконавчі семантичні мережі забезпечені механізмами, що можуть спричинити структурні перетворення власної мережі. До найпоширеніших виконавчих механізмів належать: процедури поширення сигналу, вкладені процедури і процедури перетворення графа. Навчальні семантичні мережі, з метою забезпечення ефективної роботи у зовнішньому середовищі, корегують власну структуру під час аналізу нової інформації [25]. Корегування навчальної мережі можна здійснити трьома способами: запам'ятовуванням, реструктуризацією та зміною ваг. Нарешті, гібридні мережі, одночасно поєднують декілька різновидів семантичних мереж, з метою використання їхніх переваг чи усунення недоліків певного виду мережі.

Однією з основних переваг семантичних мереж є наочність подання знань, яка забезпечується графовою структурою. По-перше, графове подання показує прямі зв'язки між пов'язаними поняттями, тоді як лінійне подання покладатися на повторювані випадки змінних чи імен, щоб показати ті самі зв'язки. По-друге, графове подання часто має евристичну цінність, допомагаючи інженерам виявляти закономірності, які важко або неможливо побачити в лінійній формі; це досягається кластеризацією пов'язаних знань [26]. При аналізі недоліків семантичних мереж, важливо підкреслити відсутність стандартизації значень вузлів та дуг. Крім цього,

деяким видам семантичних мереж властиві функціональні проблеми, такі як помилки множинного успадкування, змішування різних рівнів абстракції та відсутність формальної семантики. «Наочність» подання забезпечила поширеність семантичних мереж як способу подання знань.

1.3.4. Фрейми

Фрейм – це об’єктно-орієнтована модель подання знань, в якої онтологічне зобов’язання передбачає наявність об’єктів, властивостей, значень та відношень між ними. Як наслідок, його часто називають поданням трійки «об’єкт-властивість-значення». Об’єкти є інтуїтивною абстракцією для впорядкування знань про уявні та матеріальні сутності [27]. В контексті міркування важливою є надана об’єкту можливість групувати процедури для визначення власних властивостей, а також властивостей пов’язаних об’єктів. Об’єктно-орієнтоване подання поняття, формується шляхом об’єднання в спільну структуру множини суттєвих властивостей об’єктів певного виду. У FOL твердження про властивості поняття є розкиданими, тому складно контролювати узгодженість та повноту бази знань.

Фрейм можна означити як структурне та реляційне подання об’єкта, оскільки в системах, що засновані на фреймах, спостерігається наступне співвідношення: «об’єкту, якому притаманні властивості, що мають значення» відповідає «фрейм, який характеризується слотами, що мають заповнювачі». Кожен фрейм має назву та складається з множини слотів з відповідними заповнювачами. Заповнювач може належати до примітивних типів (число, стрічка, дата), складних типів (множина, список) чи бути посиланням на інший фрейм. В залежності від рівня абстракції, розрізняють два види фреймів – індивідуальний та загальний. Індивідуальний фрейм подає певний об’єкт, який є екземпляром деякого поняття, а загальний фрейм використовується для опису абстрактної категорії. Індивідуальні фрейми мають спеціальний слот під назвою «INSTANCE-OF», де заповнювач – це ім’я (чи вказівник) загального фрейму. Загальні фрейми можуть мати спеціальний слот під

назвою «IS-A» із заповнювачем, який вказує на більш загальний фрейм. Наявність цих спеціальних слотів є принциповою, оскільки вони забезпечують механізм успадкування властивостей та процедур.

Процес узгодження знань зазвичай контролюється за допомогою процедур, які пов'язані зі слотами в загальних фреймах. Існує два види процедур – «IF-NEEDED» та «IF-ADDED». Перший виконується, коли заповнювач слоту відсутній, а його значення є необхідним; другий виконується, коли заповнювач слотів додано, а його значення впливає на інші фрейми. Найпоширенішими випадками застосування цих процедур є: встановлення значень за замовчуванням в порожні слоти, розрахунок значень заповнювачів динамічних слотів, підтримка узгодженості. Процедури та заповнювачі більш загального фрейму застосовуються до певного фрейму через механізм успадкування. Вони впливають на міркування в фреймових системах, яке зазвичай має наступний порядок виконання: 1) користувач створює екземпляр нового фрейму, щоб означити існування якогось об'єкта; 2) заповнювачі слотів успадковуються там, де можливо; 3) застосовуються успадковані процедури "IF-ADDED", які ініціюють процес створення пов'язаних фреймів та заповнення слотів; 4) якщо користувачеві чи будь-якій процедурі потрібен заповнювач слотів, але він порожній, то запускається успадкована процедура "IF-NEEDED".

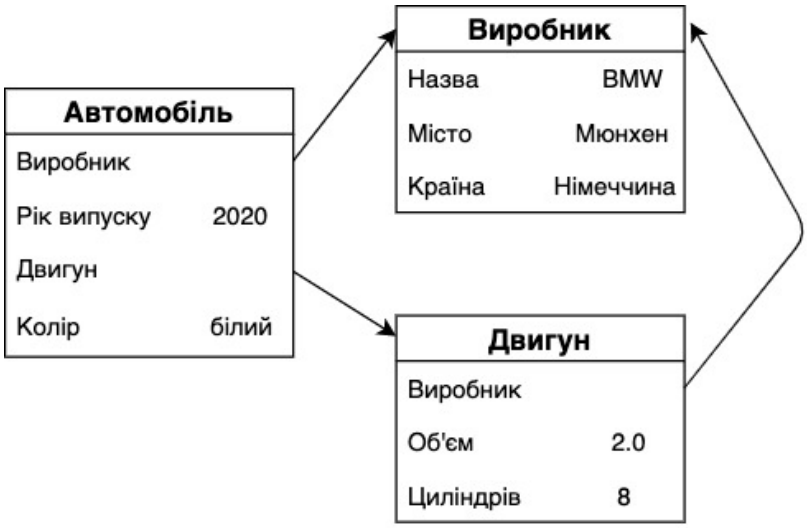

Рис. 1.4. Множина фреймів, що частково описує автомобіль

Фреймова модель має свої переваги і недоліки, основні переваги забезпечені її структурними особливостями. Об’єктно-орієнтований підхід до подання знань дозволяє групувати властивості та процедури, пов’язані з об’єктами певного виду, спрощуючи процеси міркування та підтримки узгодженості. Заповнювачі слотів, які є посиланнями на інші фрейми, асоціюють фрейми між собою, структуруючи пов’язані об’єкти у формі кластера. Нові властивості і відношення додаються до об’єкта за допомогою нових слотів у фреймі, а відповідні процедури допоможуть поширити заповнювач за замовчуванням для доданих слотів в існуючі фрейми. Механізм процедур робить динамічні (обчислювані) властивості явними, а також дозволяє запобігати потенційно непотрібним розрахункам. Фрейми передбачають декларативний і процедурний контроль процесу міркуваннння, що зменшує загальні витрати ресурсів на концептуальне моделювання предметної області. Елегантна реалізація механізму успадкування підкреслює виразність фреймової моделі. Серед основних недоліків фреймів слід назвати: відсутність формального механізму міркування; відсутність стандартів для значень заповнювачів слотів; множинне наслідування може породжувати конфліктуючу поведінку успадкованих процедур.

1.3.5. Сценарії

Сценарій – це структурована схема подання знань [13], що описує стереотипну послідовність подій в певному контексті. Подія відповідає залежній від часу дії, яка може вказувати на причинно-наслідкові зв’язки. Сценарії дуже схожі на фрейми, за винятком того, що значення, які заповнюють слоти, повинні бути впорядковані в часі, тому фрейми вважаються більш загальним способом подання знань. Сценарії застосовуються в інтелектуальних системах аналізу природної мови для організації бази знань про ситуації, які система має розуміти. Вони дають знання та очікування про перебіг конкретних подій і можуть бути застосовані до нових ситуацій. Сценарії є досить ефективним інструментом подання знання, оскільки подіям в дійсності досить часто властива стереотипність.

Для опису послідовності подій сценарій використовує набір слотів, що містять інформацію про елементи, які утворюють контекст або беруть участь у подіях. Зазвичай сценарій має назву та складається з наступних слотів: шлях, реквізити, ролі, вхідні умови, сцени та результати. Шлях використовується для фіксації конкретного різновиду певного сценарію. Ролі описують акторів, які беруть участь в сценарії, а реквізити позначають об’єкти, що використовуються в послідовності подій. Вхідні умови визначають обставини, які є необхідними для того, щоб події в сценарії відбулися. Сцени описують фактичну послідовність подій, які впорядковані у часі. Результати сценарію описують стан дійсності, який існує після того, як відбулися події. Варто зазначити, що деякі сцени у сценарії можуть бути необов’язковими, що збільшує універсальність сценарію як способу подання знань довільної предметної області.

Знання, що подаються сценарієм, зазвичай зберігаються у символьній формі. Основна функціональність сценарію полягає у відповіді на питання, що стосуються ролей, об’єктів та результатів його виконання в контексті заданої ситуації. Процес міркування у сценарії передбачає застосування методів пошуку шаблонів та їхнього узгодження з сценарієм заданої ситуації. Хоча деякі ситуацій не спостерігалися, сценарій дозволяє передбачити, що з ким і коли станеться. Після запуску сценарію користувач може задавати запитання та отримувати відповіді з обмеженими або відсутніми початковими знаннями.

Сценарій дозволяє отримати цілісну картину розвитку ситуації з множини спостережень. Такий підхід забезпечує здатність суб’єкту прогнозувати результати знайомих та схожих ситуацій. Стереотипний характер сценаріїв сприяє їх застосуванню до різноманітних ситуацій. Основні обмеження цього способу подання знань виникають при формуванні складних понять та відношень; крім цього, не всі знання можна подати у формі сценарію. Також іноді складно вибрати оптимальну структуру сценарію, яка б забезпечила ефективне міркування. Як вже зазначалося, сценарії можна розглядати як частковий випадок фреймів.

Сценарій: Похід в ресторан Шлях: Звичайний ресторан Реквізити: Їжа, Стіл, Меню, Гроші Ролі: Власник, Клієнт, Офіціант, Касир	Сцена 1: «Вхід в ресторан» Клієнт заходить в ресторан Оглядає столи Вибирає найкращий Вирішує там сісти Йде туди Займає місце
	Сцена 2: «Замовлення їжі» Клієнт просить меню Офіціант приносить меню Клієнт гортає меню Вибирає їжу Замовляє їжу
Вхідні умови: Клієнт голодний Клієнт має гроші Власник має їжу	Сцена 3: «Поїдання їжі» Офіціант приносить їжу Клієнт її оглядає Клієнт їсть їжу
Результати: Клієнт не голодний Власник має гроші Клієнт має менше грошей Власник має менше їжі	Сцена 4: «Оплата рахунку» Клієнт просить рахунок Офіціант приносить рахунок Клієнт оплачує рахунок Офіціант дає готівку касиру Касир дає решту офіціанту Офіціант приносить решту Клієнт залишає чайові Клієнт виходить з ресторану

Рис. 1.5. Псевдо-форма сценарію в ресторані [13]

1.3.6. Продукційні правила

Продукційні правила, також відомі як правила «ЯКЩО-ТО», є одним з найпоширеніших способів подання знань, зокрема, в експертних системах [28] та системах підтримки прийняття рішень [29, 30]. Вони характеризуються високою модульністю, яка забезпечує гнучкість процесу створення і управління базою знань. Кожне правило подає відносно невелику та незалежну одиницю знання; крім цього, його можна самостійно додати чи видалити з множини існуючих правил. Простота синтаксису та інтуїтивний спосіб подання знань сприяє інтерпретації і аналізу знання, що подане у зазначеній формі. Будь-яке продукційне правило складається з засновку та висновку. Засновок відповідає передумовам (речення «ЯКЩО»), а висновок – результату (речення «ТО»). З метою здійснення процесу міркування над знанням, правила групуються у продукційну систему, яка має свої структурні та функціональні особливості.

ЯКЩО комп'ютер справний
ТА існує доступ до мережі Інтернет
ТА встановлено застосунок Zoom
ТО приєднатися до конференції

Рис. 1.6. Приклад продукційного правила, що описує онлайн-комунікацію

Продукційна система складається з двох компонент: БЗ та механізму висновку, яким відповідно властиві продукційні правила двох видів – декларативні та процедурні. Декларативні правила описують поняття та властивості предметної області, а процедурні правила описують способи розв'язання задач міркування для різних операційних контекстів. Вони також відрізняються місцем зберігання: одні зберігаються в базі знань, а інші інтеґруються в механізм висновку. БЗ продукційної системи має визначену структуру, яка поділяє її на дві області: робочу пам'ять та базу правил. Робоча пам'ять містить набір фактів, які здебільшого є твердженнями

про предметну область, але можуть також бути структурами даних. Синтаксис та форма подання тверджень не є строго формалізованими, тому можуть відрізнятися в різних базах знань. База правил утворена множиною правил «ЯКЩО-ТО» визначеної структури, де засновок має форму кон'юнкції гіпотез, а висновок – послідовності дій. Структура довільного запису в базі правил задається формулою $p_1 \wedge p_2 \wedge \ldots \wedge p_n \Rightarrow a_1, a_2, \ldots a_k$. Члени засновку можуть мати форму заперечуваних або незаперечуваних предикатних тверджень про поняття і властивості передметної області. Висновки, як правило, містять наступні дії: «ДОДАТИ» факт до робочої пам'яті, «ВИДАЛИТИ» факт з робочої пам'яті, «ЗМІНИТИ» значення атрибуту, «ЗАПИТ» користувачеві на введення додаткових фактів чи команди; наприклад, $A(x) \wedge B(x) \wedge C(y) \Rightarrow$ ВИДАЛИТИ: $D(x)$, ДОДАТИ: $B(y)$.

Механізм висновку у продукційних системах найчастіше реалізується методом прямого виводу. Він розпочинає роботу з набору фактів, що доступні в робочій пам'яті, та застосовує до них «активні» правила з бази знань, для того, щоб здобути більше фактів, які потенційно ведуть до мети. Правило вважається активним, якщо множина фактів в робочій пам'яті робить його засновок істинним; це необхідна умова для застосування дій з висновку. Поширеною є ситуація, коли кілька правил є активними одночасно, що створює необхідність вказувати порядок їх виконання. Насправді, порядок виконання є важливим, оскільки дії ДОДАТИ та ВИДАЛИТИ можуть змінити набір активних правил, так що наступне активне правило в черзі може бути деактивоване попереднім. Стратегія вибору правила для виконання серед наявних кандидатів називається вирішенням конфліктів. Популярні стратегії вирішення конфліктів перелічено нижче:

- Уникнення дублювання – не застосовувати одне і те ж правило двічі;
- Актуальність – вищий приорітет у привил, що оперують недавно доданими в робочу пам'ять фактами;
- Спеціалізація – спеціалізовані правила мають вищий пріоритет ніж правила загального призначення;

- Рівні пріоритету – правилам присвоюються рівні пріоритету, де правила з вищим рівнем пріоритету виконуються першочергово.

Ще одна невизначеність, яка інколи проявляється в продукційних системах під час процесу міркування, пов'язана з операцією об'єднання. Деякі правила можуть потребувати значної кількості об'єднань, зокрема якщо вони містять змінні, що перебирають усіх індивідів, для яких є активні правила. Крім того, множини гіпотез засновків деяких правил можуть перетинатися, що призводить до багаторазового повторення одинакових об'єднань. Для мінімізації ефекту проблеми неефективного об'єднання та вирішення конфліктів було розроблено алгоритм Рете [31]. Основні принципи його поведінки полягають в наступних кроках: правила компілюються в мережу, яка об'єднує засновки кількох правил разом, уникаючи таким чином дублювання гіпотез; поширюються лише істинні об'єднання; переоцінюються лише змінені засновки.

Підсумовуючи, продукційні правила як модель подання знання забезпечують модульність вмісту та гнучкість управління базою знань. Вони характеризуються виразністю, інтуїтивністю причинно-наслідкової інтерпретації та укомплектовані формальним механізмом висновку, що доступний в продукційних системах. Крім цього, їхня корисність доведена успіхом великих комерційних експертних систем. Серед недоліків продукційних правил, важливо зазначити, що не всі знання можна подати у формі правил. Вони рідко застосовуються для подання структурованого декларативного знання, а також незручні для відстеження ієрархій. Великі системи, як правило, містять тисячі правил, що ускладнює консолідацію знань та суттєво знижує швидкодію механізму висновку.

1.3.7. Онтології

Онтології є сучасним способом подання знань, що ґрунтується на процесі концептуалізації. Концептуалізація – це абстрактний, спрощений погляд на деяку вибрану частину дійсності, яка містить об'єкти, поняття та інші сутності, а також

відношення між ними, що, як передбачається, існують у певній предметній області [32]. Оскільки процес концептуалізації предметної області не є формалізованим, то його результат може втілюватися у вигляді різних онтологій. Згідно з поширеним означенням, поняття «онтологія» – це явна специфікація концептуалізації. Поруч з онтологією з'являється поняття «онтологічне зобов'язання». Вважається, що теорія онтологічного зобов'язання до певного об'єкта діє лише тоді, коли цей об'єкт є загальним для всіх онтологій, що задовольняють цю теорію. Отже, онтологічне зобов'язання можна означити як об'єкт чи об'єкти, спільні для онтологій, що задовольняють деяку теорію.

Формально онтологія складається з понять, організованих в таксономію, їхніх властивостей чи атрибутів, пов'язаних аксіом та правил виведення [33]. Онтологія разом із набором екземплярів класів становить базу знань. Для опису елементів онтології зазвичай використовують наступні позначення:

- клас (концепт) – подає поняття з предметної області;
- екземпляр (індивід) – подає конкретний екземпляр певного класу;
- слот (властивість, роль) – подає ознаку чи атрибут концепту або індивіда;
- аспект (обмеження ролі) – задає обмеження слоту на значення;
- відношення – подає зв'язки між класами, екземплярами;
- правила – речення «якщо-то», що описують допустимі логічні виведення;
- аксіоми – логічні твердження, що формують загальну теорію предметної області.

Онтологічна інженерія передбачає поєднання структурних елементів чотирьох найпоширеніших способів подання знань: фрейми використовуються для подання концептів та їх атрибутів; семантичні мережі застосовуються для подання відношень між елементами онтології; логічні моделі є підґрунтям для аксіом теорії предметної області, а продукційні правила подають допустимі правила виведення. Процес побудови онтології складається з кількох етапів: визначення класів в онтології; впорядкування класів у таксономію; визначення слотів та опис їхніх

допустимих значень; заповнення значень слотів для індивідів. Правила міркування та аксіоми можуть бути додані на третьому кроці, при необхідності.

Існує достатньо прикладів успішних застосувань онтологій в науці та бізнесі. По-перше, вони поширюють принцип явного подання контексту інформації, який мінімізує вплив персонального контексту на результат інтерпретації та сприяє формуванню спільного, узгодженого трактування. По-друге, наявність існуючих онтологій та відповідного доступу, дозволяє суб'єкту повторно використовувати перевірені та структуровані спільнотою знання, зокрема для побудови інтеґрованої, розширеної чи спеціалізованої онтологій. Розрізняють онтології, що орієнтовані на предметну область, на задачу та високорівневі онтології. Онтології, що орієнтовані на предметну область, містять знання про конкретну предметну область, в той час як високорівневі онтології є їхньою основою та надають означення концептів загального призначення. Онтології, що орієнтовані на задачу, можуть містити знання одразу з кількох предметних областей і уникають обмежень спеціалізованих онтологій [33]. По-третє, онтології зручно застосовувати для логічного узгодження знання, його структурованого розширення та здійснення інтелектуальних запитів, які, як правило, реалізуються за допомогою функціонального інтерфейсу «СКАЖИ І ЗАПИТАЙ», коли користувач взаємодіє із системою знань, висловлюючи логічні припущення (СКАЖИ) і формуючи інтелектуальні запити (ЗАПИТАЙ). Крім цього, онтології мають такі структурні властивості, як відокремлення знань предметної області від операційних знань, строге формальне подання теорії предметної області у вигляді множини аксіом, можливість повторного використання знання з інших онтологій, тощо.

Проєктування онтології з метою вирішення актуальних прикладних проблем вимагає значних ресурсів – експертів предметної області, інженерів баз знань, часу. Оскільки розбудова та актуалізація онтології є ітеративним процесом, то витрати на її підтримку є також неминучими. Повторне використання існуючих онтологій сприяє зменшенню витрат на її підтримку, проте не усуває необхідність залучення

висококваліфікованих кадрів в цей процес. Тому варто розвивати автоматизовані чи напівавтоматизовані методи побудови та підтримки онтологій, які б зменшили витрати ресурсів до мінімуму [34]. Вивченням онтології зазвичай називають частково автономний процес побудови структури онтології та заповнення її екземплярами. Існує багато способів вивчення онтології. Найпоширенішим є вивчення онтології у контексті; при цьому генерується таксономія за допомогою аналізу тексту та пошуку інформації. Інші способи, такі як пошук пов'язаних даних, вивчення концептів для описових логік та OWL 2, краудсорсинг онтологій також активно розвиваються.

Не менш важливим завданням, що постає під час вивчення онтології, є оцінка її якості. На практиці, якість вивченої онтології часто залежить від ступеня автоматизації: вища залученість експерта в напівавтоматичній побудові онтології призводить до кращої якості. Якщо онтологія є недостатньо якісною для певної задачі, це вимагатиме значної кількості ресурсів для подальшого опрацювання результатів аналізу.

1.4. Висновки до розділу 1

В першому розділі дисертації здійснено системний аналіз понять «знання» та «істинність знання», а також найпоширеніших способів подання знань. Результати аналізу дозволяють сформулювати наступні наукові положення, висновки та рекомендації:

- трактування поняття «знання» в контексті проєктування систем баз знань доцільно здійснювати на основі його епістемологічної структури, відповідно до якої знання є істинним обґрунтованим переконанням, де обґрунтованість явно пов'язана з істинністю та має означені форми;
- проблеми Ґетьє висвітлюють недоліки «денотаційної» інтерпретації знання та створюють перспективи для «операційної» інтерпретації, де поведінка виразу краще корелює з істинністю ніж його значення;

- системний аналіз способів подання знань сформував розуміння структурних та функціональних можливостей найпоширеніших різновидів подання знань, що є надійною основою для здійснення порівняльного аналізу;
- порівняльний аналіз способів подання знань встановив їхні основні переваги і недоліки, а результати аналізу створили підґрунтя для формування методології вибору оптимальної моделі подання знань, що використовується для концептуалізації довільної предметної області;
- сучасні онтології поєднують в собі надійні структурні елементи інших способів подання знань та характеризуються високою універсальністю, виразністю, можливістю повторного використання існуючих онтологій, а також логічного узгодження знань.
- логічні моделі подання знань характеризуються наявністю формальної семантики, що гарантує однозначність інтерпретації синтаксичних виразів, перевірених форм дедуктивного міркування і виразних формальних мов подання, які зменшують вплив суб’єктного контексту на інтерпретацію та міркування.

РОЗДІЛ 2. СПОСОБИ ПОДАННЯ ТА ЗАСОБИ УПРАВЛІННЯ БАЗАМИ ЗНАНЬ В КОНТЕКСТІ ТЕОРІЇ ТИПІВ

2.1. Теорія типів

Теорія типів бере свій початок у відомій праці «Principia Mathematica» [20], де використовується для подолання різноманітних парадоксів формальної логіки та систем перепису, зокрема парадоксу Рассела [21]. У розгалуженій теорії типів поняття типу застосовується для розмежування математичних сутностей (об’єкта, предиката, предиката предикатів) та сприяє організації типів в ієрархії, з метою уникнення проблеми «зачарованого кола» [35]. Труднощі формалізації індуктивних доказів потребували введення аксіоми зводимості, яка певним чином нівелювала парадигму уникнення вищезазначеної проблеми. Л. Чвістек та Ф. Рамзі зауважили, що без принципу Рассела ієрархія рівнів типів може бути порушена [36]. Отримана логіка називається «простою теорією типів» і вона еквівалентна розгалуженій теорії типів з аксіомою зводимості.

У 1940-х роках А. Черч сформулював просту теорію типів, що базується на функціях замість відношень та використовує спеціальний запис для побудови і застосування функцій [37]. Оскільки теорія типів Черча допускає неконструктивні принципи міркування, вона вважається класичною логікою вищого порядку. У 1972 році П. Мартін-Леф запропонував конструктивну форму теорії типів [38]; згодом ізоморфізм Каррі-Говарда [39] пов’язав доведення теорем в конструктивній логіці з численням синтаксичних виразів в теоріях типів. Сучасні дослідження теорії типів в комп’ютерних науках присвячені здебільшого класичній та конструктивній логіці у стилі А. Черча.

В мовах програмування статична перевірка типів є важливим механізмом, що здійснює семантичну перевірку коректності програми. Його мета полягає в тому, щоб визначити, чи відповідають синтаксичні вирази у вихідному коді програми очікуваній поведінці. Програма вважається добре-типізованою, якщо кожному

виразу можливо присвоїти тип, дотримуючись правил наявної системи типів. З'ясуємо детальніше поняття «тип» та його призначення. Загалом «тип» – це властивість, присвоєна мовному терму [40]. Як властивість, він приписує терму декілька корисних функцій з денотаційного та операційного погляду. Зокрема, тип позначає набори значень; крім цього, він застосовується для обмеження області змінної, контролю за коректністю формування синтаксичних виразів і класифікації виразів за їх значеннями.

2.1.1. Просто-типізоване лямбда-числення, λ→

Просто-типізоване лямбда числення або λ→ – це формальна система в теорії типів, яка призначена для опису абстрактної поведінки «функцій в області визначення». Як і нетипізоване лямбда числення [41], цей формалізм передбачає наявність нескінченної множини змінних, двох правил побудови термів та одного правила усунення. Типи, представлені нескінченною множиною простих типів, що складається зі змінних та стрілочних типів. Маючи нескінченну множину змінних типів $\mathbb{V} = \{\alpha, \beta, \gamma, \dots\}$, множина всіх простих типів $\mathbb{T}$ означається наступним чином:

- Змінна типу: якщо $\alpha \in \mathbb{V}$, то $\alpha \in \mathbb{T}$;
- Стрілочний тип: якщо $\sigma, \tau \in \mathbb{T}$, то $(\sigma \to \tau) \in \mathbb{T}$.

В лямбда численні синтаксичні вирази називаються λ-термами. Терм M вважається придатним до типізації, якщо існує тип σ, такий що $M : \sigma$, де знак двокрапки «:» позначає *відношення типізації*, яке присвоює тип σ терму M. При нескінченній множині змінних $V = \{x, y, z, \dots\}$ множина типізованих термів $\Lambda_{\mathbb{T}}$ означається наступним чином:

- Змінна: якщо $x \in V$ та $\sigma \in \mathbb{T}$, тоді $x{:}\,\sigma \in \Lambda_{\mathbb{T}}$;
- Абстракція: якщо $x{:}\,\sigma$ та $M{:}\,\tau$, тоді $\lambda x.M{:}\,\sigma \to \tau \in \Lambda_{\mathbb{T}}$;
- Застосування: якщо $M{:}\,\sigma \to \tau$ та $N{:}\,\tau$, тоді $MN \in \Lambda_{\mathbb{T}}$.

Вираз абстракції $\lambda x.M$ позначає функцію, яка відображає довільне вхідне значення x у вихідне значення M. Спеціальний символ λ, відомий як зв'язувач

змінної, використовується для підкреслення абстрактної ролі змінної x у виразі терму. Наявність зв'язувача в λ-термі класифікує процес появи змінних на три категорії: вільні, зв'язані і зв'язуючі. Наприклад, λ-терм $(x\ y)$, який формується з x та y правилом застосування, містить дві вільні вхідні змінні. Абстракція над x породжує терм $(\lambda x.\, x\ y)$, де перший x є зв'язуючим, другий x є зв'язаним, а y є вільним. Подальше абстрагування над y породжує терм $(\lambda y.\, \lambda x.\, x\ y)$, де перші y і x є зв'язуючими, другі x і y є зв'язаними, а вільні змінні є відсутніми. У лямбда численні функція може приймати лише один аргумент; для подання функції з кількома аргументами використовується метод обчислення значення функції, що називається каррінг (curring). Згідно з ним, терм $(\lambda y.\, \lambda x.\, x\ y)$ обчислюється як $(\lambda y.\, M)$, де $M \coloneqq (\ \lambda x.\, x\ y)$.

Типізація λ-терма починається з типізації його змінних. Передбачається, що кожна змінна має унікальний тип, тому якщо $x\colon \sigma$ та $x\colon \tau$, то $\sigma \equiv \tau$. Існує два способи присвоєння типів змінним: явна типізація (á la Church) та неявна типізація (á la Curry). Перший полягає у присвоєнні типу кожній змінній після її оголошення. Типи зв'язаних змінних задаються одразу з появою зв'язуючої змінної після символу λ, тоді як типи вільних змінних задаються в контексті. У випадку неявної типізації типи змінних не вказуються, тому λ-терми, що можуть бути типізовані, знаходяться процесом пошуку.

Правило усунення під назвою β-скорочення встановлює відношення на області λ-термів, які мають форму $((\lambda x\colon \sigma.\, M)\ N)$, та є формалізацією *процесу обчислення функції*. Однокрокове β-скорочення означається наступним чином [42]:

(а) $(\lambda x\colon \sigma.\, M)\ N \to_\beta M[x \coloneqq N]$;

(б) Якщо $M \to_\beta N$, то $(M\ L \to_\beta N\ L)$, $(L\ M \to_\beta L\ N)$, і $(\lambda x.\, M \to_\beta \lambda x.\, N)$.

Базове правило (а) вказує, що терм застосування функції можна «обчислити» за допомогою заміни $M[x \coloneqq N]$ на інший λ-терм, в той час як правила сумісності (б) використовуються, коли $((\lambda x\colon \sigma.\, M)\ N)$ є частиною більшого терму.

Заміна складається з набору правил перепису, що застосовуються до λ-терма, для отримання нового λ-терму. Більшість правил є однокроковим перетворенням, за винятком терму абстракції $(\lambda x{:}\,\sigma.\,M)[x \coloneqq N]$, де заміна здійснюється в два кроки. Перший крок захищає вільні змінні в N від ненавмисного зв'язування у цільовому термі. Це досягається правилом заміни, що має назву α-перетворення, і здійснює перейменування зв'язуючих змінних у цільовому термі. В лямбда численні назва зв'язуючої змінної не є істотною, тому α-перетворення є коректним та корисним відношенням. Після того, як перший крок виконано, другий крок полягає у переписі змінних. Формальне означення заміни виглядає наступним чином:

(а) $x[x \coloneqq N] \equiv N$ та $y[x \coloneqq N] \equiv y$, якщо $x \not\equiv y$;

(б) $(P\ Q)[x \coloneqq N] \equiv (P[x \coloneqq N])(Q[x \coloneqq N])$;

(в) $(\lambda y{:}\,\sigma.\,P)[x \coloneqq N] \equiv \lambda z{:}\,\sigma.\,(P^{y\to z}[x \coloneqq N])$, якщо

- відсутні вільні появи змінної z в N;
- відсутні вільні або зв'язуючі появи змінної z в P;
- $P^{y\to z}$ позначає результат заміни кожної вільної появи y в P на z.

Для оголошення типів вільних змінних у термі використовується поняття контексту. Контекст Γ – це список оголошень змінних у вигляді $x{:}\,\sigma$. Він може складатися з декількох оголошень, розділених комою, одного оголошення або ж навіть бути пустим. Вираз, що має форму $\Gamma \vdash M{:}\,\sigma$, називається *судженням*. Усі предметні змінні, що оголошені в контексті судження Γ, зв'язують відповідні вільні змінні в λ-термі $M{:}\,\sigma$. Загалом, в теоріях типів існують три різновиди задач, що пов'язані із судженнями [42]:

(1) Допустимість типізації: ? $\vdash$ терм : ?

(1а) Присвоєння типу: контекст $\vdash$ терм : ?

(2) Перевірка типу: контекст $\vdash^{?}$ терм : тип

(3) Пошук терму: контекст $\vdash$? : тип

(3а) Доведення типу: $\emptyset$ $\vdash$? : тип

Розв’язання цих задач формує основу користувацького сервісу в базах знань, що використовує теорію типів для подання та керування описовими знаннями. Задача допустимості типізації терму полягає в доведенні можливості його типізації шляхом побудови відповідного контексту, типу або визначення кроку, де продовження його побудови є неможливе. В підзадачі присвоєння типу контекст є заданим, тому необхідно побудувати лише тип. Задача перевірки типу містить визначений контекст, терм, тип та вимагає підтвердження того, що заданий терм має вказаний тип. Задачі пошуку терму чи доведення типу мають за мету побудувати λ-терм вказаного типу у зазначеному контексті. Підзадача доведення типу – це проблема, що зустрічається в системі доведення «природня дедукція».

Хорошою властивістю $\lambda\rightarrow$ є те, що всі три види задач є розв’язними в цій системі, тобто існує загальний метод (чи алгоритм) обчислення розв’язку задачі. Побудова розв’язку в задачах допустимості типізації та пошуку типу починається з розділення терму, в якого тип є невідомим, на структурно простіші частини. Потоки цього процесу виконуються рекурсивно та зупиняють розділення у випадку, коли отримана шляхом розділення частина є типізованим λ-термом. На зворотному шляху рекурсії здійснюється типізація розділених термів через обчислені типи їх частин, аж до моменту типізації вихідного терму.

Зазначивши приблизну форму послідовності обчислень, доцільно визначити правила, що використовуються для побудови і розділення структурно складних термів. Система логічного виведення в $\lambda\rightarrow$ складається з трьох правил, що подаються у формі «засновки-висновок», де перелік засновків вказується над горизонтальною лінією, а висновок під нею:

(1) Змінна терму:
$$\frac{x:\sigma \in \Gamma}{\Gamma \vdash x:\sigma}$$

(2) Абстракція:
$$\frac{\Gamma, x:\sigma \vdash M:\tau}{\Gamma \vdash \lambda x:\sigma.\, M:\sigma \rightarrow \tau}$$

(3) Застосування:
$$\frac{\Gamma \vdash M:\sigma \rightarrow \tau \qquad \Gamma \vdash N:\sigma}{\Gamma \vdash M\ N:\tau}$$

Ці правила є формальною специфікацією для типізації λ-термів різних форм, таких як змінна, абстракція і застосування. Правило (1) виражає те, що кожне оголошення змінної, яке знаходиться в деякому контексті, є логічно вивідним з цього контексту. Привила (2) і (3) визначають відповідно умови типізації абстракції і її застосування в судженнях. Усі правила логічного виведення є універсальними, оскільки вони застосовні з довільними значеннями параметрів Γ, σ, τ, x, M, N.

Просто-типізоване лямбда-числення, λ→, задовольняє багатьом бажаним властивостям абстрактної поведінки функцій. Зокрема, ця система запобігає застосуванню функцій до самих себе, наявності точки нерухомості в усіх функціях та появі нескінченних послідовностей скорочень. Разом з тим, система цього виду є занадто слабкою, щоб виражати всі обчислювані функції, оскільки рекурсія не є можливою в λ→. Більше того, структура типів містить мало інформації про терми, тому є необхідність розширити λ→ до потужніших систем лямбда-числення з складнішими типами. Платою за таке розширення є зазвичай (крім системи $\lambda\underline{\omega}$) нерозв'язність задачі пошуку терму.

2.1.2. Типізоване лямбда-числення другого порядку, λ2

Типізоване лямбда-числення другого порядку – це формальна система в теорії типів, яка розширює просто-типізоване лямбда-числення механізмом абстракції типу терму. В λ→ терм $x : \alpha$ можна абстрагувати до $\lambda x{:}\, \alpha . x : \alpha \to \alpha$, відповідно утворюється терм, що залежить від терму. У $\lambda 2$ можна піти далі та абстрагувати отриманий терм до вигляду $\lambda \alpha{:}*. \lambda x{:}\, \alpha . x : \Pi \alpha{:}*. A \to \alpha$, відповідно утворюється терм, що залежить від типу. Таким чином, маючи заданий терм $M : A$ абстракція над типом α породжує новий терм $\lambda \alpha{:}*. M : \Pi \alpha{:}*. A$, де $*$ позначає «тип усіх типів», а Π – це зв'язувач змінної типу. Π-тип $\Pi \alpha{:}*. (\alpha \to \alpha)$ є типом усіх функцій, що відображають довільний тип α в λ-терм з типом $\alpha \to \alpha$. Абстракція над кількома змінними типу породжує $\lambda \beta{:}*. \lambda \alpha{:}*. M : \Pi \beta{:}*. \Pi \alpha{:}*. A$. Розширення, що надається системою $\lambda 2$, дозволяє формувати поліморфні терми, зокрема поліморфні функції.

Введення Π-типів у системі $\lambda 2$ впливає на вигляд множини типів, множини попередньо-типізованих термів, α-перетворення, однокрокого β-скорочення та систему логічного виведення. Маючи нескінченну множину змінних типу $\mathbb{V} = \{\alpha, \beta, \gamma, \dots\}$ і нескінченну множину об'єктних змінних $V = \{x, y, z, \dots\}$, абстрактний синтаксис $\lambda 2$-типів $\mathbb{T}2$ та $\lambda 2$-термів $\Lambda_{\mathbb{T}2}$ записується наступним чином [42]:

$$\mathbb{T}2 = \mathbb{V} \mid \mathbb{T}2 \to \mathbb{T}2 \mid \Pi\mathbb{V}{:}*.\,\mathbb{T}2$$

$$\Lambda_{\mathbb{T}2} = V \mid \Lambda_{\mathbb{T}2}\ \Lambda_{\mathbb{T}2} \mid \Lambda_{\mathbb{T}2}\ \mathbb{T}2 \mid \lambda V{:}\,\mathbb{T}2.\ \Lambda_{\mathbb{T}2} \mid \lambda\mathbb{V}{:}*.\ \Lambda_{\mathbb{T}2}$$

Тип $\Pi\mathbb{V}{:}*.\,\mathbb{T}2$ і терми $(\lambda\mathbb{V}{:}*.\ \Lambda_{\mathbb{T}2})$, $(\Lambda_{\mathbb{T}2}\ \mathbb{T}2)$ з'являються в розширені $\lambda 2$. Ці нові вирази відповідають правилам абстракції та застосування другого порядку:

(1) Абстракція 2-го порядку: $$\frac{\Gamma,\ \alpha{:}* \vdash M{:}\,A}{\Gamma \vdash \lambda\alpha{:}*.\,M{:}\,\Pi\alpha{:}*.\,A}$$

(2) Застосування 2-го порядку: $$\frac{\Gamma \vdash M{:}\,\Pi\alpha{:}*.\,A \quad \Gamma \vdash B{:}*}{\Gamma \vdash MB{:}\,A[\alpha \coloneqq B]}$$

Вигляд однокрокового β-скорочення є також зміненим з метою опрацювання терму застосування другого порядку:

(а) $(\lambda x{:}\,\sigma.\,M)\ N \to_\beta M[x \coloneqq N]$;

(б) $(\lambda y{:}\,\sigma.\,P)[x \coloneqq N] \equiv M[\alpha \coloneqq T]$;

(в) Правила сумісності є такими ж як і в $\lambda\to$.

Перейменування змінної типу в α-перетворенні має наступний вигляд:

(а) $\lambda\alpha{:}*.\,M =_\alpha \lambda\beta{:}*.\,M[\alpha \coloneqq \beta]$, якщо β не з'являється в M;

(б) $\Pi\alpha{:}*.\,M =_\alpha \Pi\beta{:}*.\,M[\alpha \coloneqq \beta]$, якщо β не з'являється в M.

Висловлювання в системі $\lambda 2$ мають вигляд $M{:}\,\sigma$ або $\sigma : *$, тому оголошення – це вираз із змінною терму або змінною типу. Оскільки константи типу з $\lambda\to$ стали змінними типу в $\lambda 2$, всі вільні змінні типу терму в судженні повинні бути оголошеними в контексті $\lambda 2$ перед тим, як вони можуть бути використані. Для вирішення задач коректної типізації, перевірки типу або пошук терму, в $\lambda 2$ передбачена наступна система логічного виведення:

(1) Змінна терму: $\dfrac{\Gamma \text{ - } \lambda 2\text{-контекст} \quad x{:}\,\sigma \in \Gamma}{\Gamma \vdash x{:}\,\sigma}$

(2) Застосування 1-го порядку: $\dfrac{\Gamma \vdash M{:}\,\sigma \to \tau \qquad \Gamma \vdash N{:}\,\sigma}{\Gamma \vdash M\ N{:}\,\tau}$

(3) Абстракція 1-го порядку: $\dfrac{\Gamma, x{:}\,\sigma \vdash M{:}\,\tau}{\Gamma \vdash \lambda x{:}\,\sigma.\, M{:}\,\sigma \to \tau}$

(4) Змінна типу: $\dfrac{\Gamma \text{ - } \lambda 2\text{-контекст} \quad B \in \mathbb{T}2}{\Gamma \vdash B{:}*}$ всі вільні змінні з B оголошені в Γ

(5) Застосування 2-го порядку: $\dfrac{\Gamma \vdash M{:}\,\Pi\alpha{:}*.\, A \quad \Gamma \vdash B{:}*}{\Gamma \vdash MB{:}\,A[\alpha \coloneqq B]}$

(6) Абстракція 2-го порядку: $\dfrac{\Gamma, \alpha{:}* \vdash M{:}\,A}{\Gamma \vdash \lambda\alpha{:}*.\, M{:}\,\Pi\alpha{:}*.\, A}$

Варто зазначити, що правило (4) вводиться в систему логічного виведення для того, щоб уможливити використання правила (5), в якому другий засновок має вигляд судження $\Gamma \vdash B{:}*$.

2.1.3. Просто-типізоване лямбда числення з операторами типу, λ$\underline{\omega}$

Система $\lambda 2$ дозволила побудову узагальнених термів, що абстраговані як від змінних типу, так і від змінних терму. Теорія λ$\underline{\omega}$ розширює λ→ механізмом абстракції типу від змінних типу і вводить поняття тип, що залежить від типу. У формальній системі λ$\underline{\omega}$ тип $\alpha{:}\ *$ може бути абстрагований до конструктора типів $(\lambda\alpha{:}\ *.\, A){:}\ * \to *$, який сам по собі є не типом, а є функцією, що оперує типом як значенням. Відповідно, конструктор типу може представляти сімейства типів зі складнішою формою $(\lambda\alpha{:}*.\,\lambda\beta{:}*.\,\alpha \to \beta){:}\ * \to * \to *$ або ж абстрагувати над видами $(\lambda\alpha{:}*\to*.\,\alpha){:}\,(* \to *) \to (* \to *)$. Видом називають тип конструктора типів, зокрема, усі вирази з формою $*$, $* \to *$, $* \to * \to *$, $(* \to *) \to (* \to *)$ – це види. Абстрактний синтаксис для множини усіх видів записується наступним чином:

$$\mathbb{K} = * \mid \mathbb{K} \to \mathbb{K}\,.$$

Оскільки вид може використовуватися як зв'язуюча змінна в конструкторах типу, то він повинен бути коректно типізованим. Для цього вводиться *тип усіх видів*, а саме □. Таким чином, відношення типізації для видів може набувати вигляду $* : \square$, $* \to * : \square$, $(* \to *) \to (* \to *) \ : \square$.

Поява видів і □ породжує чотири рівні синтаксису: терми, конструктори типів, види та тип усіх видів. Це дозволяє записувати ланцюжки судження форми $t : \sigma : * : \square$. Варто зазначити, що обидва вирази $(\lambda\alpha{:}*.\, \alpha) : * \to *$ та $\alpha{:}*$ типізовані видами, тому обидва є конструкторами типів, проте перший не є типом і не є заселеним. Відповідно ланцюжок судження буде коротшим $\sigma{:}* \to *{:}\square$. Для позначення тих конструкторів типу, які не є типами, застосовується поняття *належні конструктори типу*. Термін *сорт* використовується для позначення або $*$, або □, тому множина сортів має вигляд $\{*, \square\}$.

Оскільки типи в $\lambda\underline{\omega}$ є складнішими, важливо перевірити допустимість їхньої форми. Тип в судженні є допустимим, якщо його можна формально вивести. В результаті, перевірка вивідності контекстних оголошень поєднується з побудовою належного контексту, що власне показано у правилах 2-3. Система виведення в $\lambda\underline{\omega}$ складається з наступних правил:

(1) Сорт: $$\emptyset \ \vdash * : \square$$

(2) Змінна: $$\frac{\Gamma \vdash A : s}{\Gamma, x : \mathrm{A} \ \vdash x : \mathrm{A}}$$ якщо x ∉ Γ

(3) Послаблення: $$\frac{\Gamma \vdash A : B \qquad \Gamma \vdash C : s}{\Gamma, x : \mathrm{C} \ \vdash A : B}$$ якщо x ∉ Γ

(4) Формування: $$\frac{\Gamma \vdash A : s \qquad \Gamma \vdash B : s}{\Gamma \vdash \mathrm{A} \to \mathrm{B} : s}$$

(5) Застосування: $$\frac{\Gamma \vdash M : A \to B \qquad \Gamma \vdash N : A}{\Gamma \vdash M\ N : B}$$

(6) Абстракція: $$\frac{\Gamma, x : \mathrm{A} \vdash M : B \qquad \Gamma \vdash A \to B : s}{\Gamma \vdash \lambda x : \mathrm{A}.\, M : A \to B}$$

(7) Перетворення: $$\frac{\Gamma \vdash A : B \qquad \Gamma \vdash B' : s}{\Gamma \vdash A : B'}$$ якщо $B =_\beta B'$

Правило (1) встановлює коректну сформованість виду $*$ через його типізацію $\square$. Наступне правило (2) формалізує оголошення та виведення нової змінної типу або терму, що типізована деяким сортом або типом A відповідно. Це правило є менш загальним ніж його відповідник в λ→, оскільки воно не дозволяє отримати попередні оголошення, зокрема $\alpha : *, x : \alpha \vdash \alpha : *$. Зазначена проблема вирішується введенням правила (3), яке дозволяє додати будь-яке коректно сформоване оголошення в кінець контексту. Правило (4) допускає формування стрілочних типів та видів, тоді як правила (5) та (6) формалізують вирази застосування і абстракції для конструкторів типів і термів. Правило (7) зазначає, що якщо типи або види є β-перетворюваними, $B =_\beta B'$, то з $\Gamma \vdash A: B$ можна вивести $\Gamma \vdash A: B'$. Як наслідок отримуємо лему про «унікальність типів до перетворення»: типи не повинні бути унікальними буквально, але вони унікальні до перетворення. Варто зазначити, що присутність мета-теоретичної властивості про збереження типу в $\lambda\underline{\omega}$ можна довести без правила перетворення.

2.1.4. Просто-типізоване лямбда числення із залежними типами, $\boldsymbol{\lambda P}$

Формальна система λP розширює λ→ ще в одному напрямку, дозволяючи формувати *тип, залежний від терму*, так що конструктор типу може залежати від терму: $(\lambda x: A. B): \Pi x: A. B$. Оскільки змінна $x: A$ може виявитися вільною в конструкторі типу $B: s$, то її можна абстрагувати. Отриманий вираз ставить у відповідність кожному $x: A$ тип $B(x)$, тобто є «залежно типізованою функцією». Цей формалізм дозволяє задавати багатозначні функції та предикати, які є основною для здійснення логічного числення засобами теорії типів. Наприклад, подання елементів теорії множин і логіки першого порядку має наступний вигляд:

- Індексований тип: $(\lambda n: nat. S_n)$, $S_n = \{0, n, 2n, 3n, \dots\}$;
- Предикат: $(\lambda n: nat. P_n)$, $P_n =$ «n є просте число».

β-скорочення терму $(\lambda n{:}\, nat.\, P_n)\ 5$ з типом $*$ відповідає висловлюванню «5 є просте число». PAT-інтерпретація логіки з фундаментальними поняттями «висловлювання як типи», «доведення як програми» та «спрощення доведень як оцінка програм» є основоположною ідеєю теорії типів, як інструменту для формального доведення властивостей програм [24, 43].

Система виведення λP вельми нагадує систему виведення $\lambda\underline{\omega}$, за винятком правил формування, застосування та абстракції, що оновлені з →-типів до Π-типів. Крім цього, правило формування в $\lambda\underline{\omega}$ виключає можливість побудови типів, що залежні від типів. Таким чином, система правил виведення λP має наступний вигляд:

(1) Сорт: $$\emptyset \vdash * : \square$$

(2) Змінна: $$\frac{\Gamma \vdash A : s}{\Gamma, x{:}\,\mathrm{A} \vdash x{:}\,\mathrm{A}}$$ якщо x ∉ Γ

(3) Послаблення: $$\frac{\Gamma \vdash A : B \qquad \Gamma \vdash C : s}{\Gamma, x{:}\,\mathrm{C} \vdash A : B}$$ якщо x ∉ Γ

(4) Формування: $$\frac{\Gamma \vdash A : * \qquad \Gamma, x{:}\,\mathrm{A} \vdash B : s}{\Gamma \vdash \Pi x{:}\, A.B : s}$$

(5) Застосування: $$\frac{\Gamma \vdash M : \Pi x{:}\, A.B \qquad \Gamma \vdash N : A}{\Gamma \vdash M\ N : B[x \coloneqq N]}$$

(6) Абстракція: $$\frac{\Gamma, x{:}\,\mathrm{A} \vdash M : B \qquad \Gamma \vdash \Pi x{:}\, A.B : s}{\Gamma \vdash \lambda x{:}\,\mathrm{A}.M : \Pi x{:}\, A.B}$$

(7) Перетворення: $$\frac{\Gamma \vdash A : B \qquad \Gamma \vdash B' : s}{\Gamma \vdash A : B'}$$, якщо $B =_\beta B'$

Правило (4) дозволяє формувати і типізувати Π-тип, що також відомий як «тип добутку». Нехай A є скінченним типом з двома екземлярами a_1 та a_2, тоді тип, що залежить від терму $\Pi x{:}\, A.B$ відповідає $B[x \coloneqq a_1] \times B[x \coloneqq a_2]$. Варто зазначити, що Π-тип може розглядатися як узагальнення декартового добутку та простору функцій [42].

Подання предикатів в теорії типів та PAT-інтерпретація логіки створюють можливість для задання «мінімальної логіки предикатів». Ця логіка складається з

двох операторів: імплікації ⇒ та квантора загальності ∀, які застосовуються до множин, висловлювань та предикатів на множинах. Згідно з PAT-інтерпретацією «висловлювання як типи», множина S і висловлювання A відповідають типам $S : *$ та $A : *$ відповідно, тоді як предикат P над множиною S задається як тип функцій $P: S \to *$. Елементи множини S заселяють тип $S : *$, а тип $A : *$ заселений «об'єктами доведення» висловлювання A. Стосовно операторів, то логічна імплікація $A \Rightarrow B$ відповідає типу функції $A \to B$, а квантор загальності $\forall_{x \in S} P(x)$ задається як тип добутку $\Pi x: S. P\ x$.

2.1.5. Числення конструкцій, λC

Три розширення λ→ ввели ряд понять, які суттєво сприяють виразності теорії типів. Числення конструкцій (CoC) пов'язує ці поняття в межах власної системи виведення [44], тим самим дозволяючи формувати «терми і типи, що залежні від термів і типів». Ця важлива властивість досягається розширенням λP з незначною модифікацією правила формування, так що сорти s_1 та s_2 можуть вибиратися незалежно:

$$\frac{\Gamma \vdash A : s_1 \quad \Gamma, x: \mathrm{A} \vdash B: s_2}{\Gamma \vdash \Pi x: A. B : s_2}$$

Тип $\Pi x: A. B : s_2$ успадковується від своєї основи $B : s_2$, однак у більш загальних системах теорії типів, зокрема, «системах чистих типів» (PTS), він може мати окремий тип s_3, внаслідок чого можливими є вісім комбінацій [45] .

Усі три розширення λ→ є взаємно незалежними системами типів, тому їх можна візуалізувати як три перпендикулярні напрямки розширення λ→. В сукупності вони породжують систему типів λC, тому $\lambda C = \lambda 2 + \lambda\, \underline{\omega} + \lambda P$. Крім цього, λ→ можна також комбінувати з двома із трьох розширень; в результаті отримаємо системи типів $\lambda\omega$, $\lambda P2$, $\lambda P\underline{\omega}$. Усі вісім систем типів зручно розміщувати в межах єдиного каркасу, що має назву *λ-куб*.

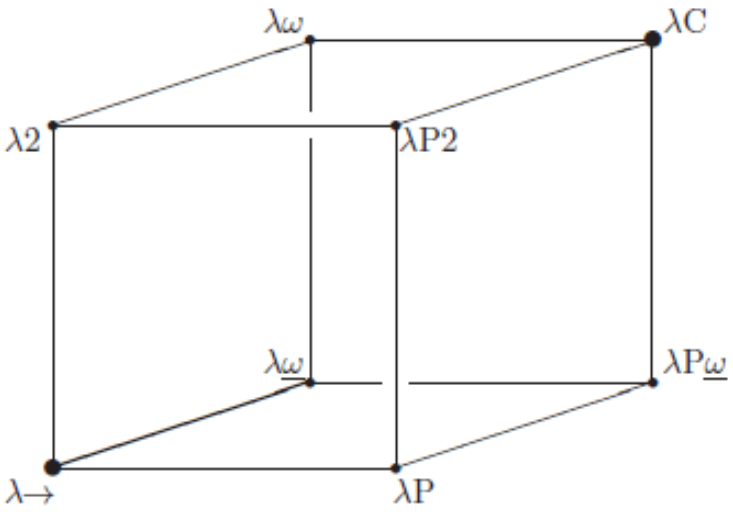

Рис. 2.1. Лямбда-куб (λ-куб) [42]

Дослідження Барендреґта [46] в теорії типів створили уніфіковану систему виведення для усіх восьми систем типів в λ-кубі:

(1) Сорт: $\emptyset \vdash * : \square$

(2) Змінна: $\dfrac{\Gamma \vdash A : s}{\Gamma, x : \mathrm{A} \vdash x : \mathrm{A}}$ якщо x $\notin$ Γ

(3) Послаблення: $\dfrac{\Gamma \vdash A : B \qquad \Gamma \vdash C : s}{\Gamma, x : \mathrm{C} \vdash A : B}$ якщо x $\notin$ Γ

(4) Формування: $\dfrac{\Gamma \vdash A : s_1 \qquad \Gamma, x : \mathrm{A} \vdash B : s_2}{\Gamma \vdash \Pi x : A . B : s_2}$

(5) Застосування: $\dfrac{\Gamma \vdash M : \Pi x : A . B \qquad \Gamma \vdash N : A}{\Gamma \vdash M\ N : B[x := N]}$

(6) Абстракція: $\dfrac{\Gamma, x : \mathrm{A} \vdash M : B \qquad \Gamma \vdash \Pi x : A . B : s}{\Gamma \vdash \lambda x : \mathrm{A} . M : \Pi x : A . B}$

(7) Перетворення: $\dfrac{\Gamma \vdash A : B \qquad \Gamma \vdash B' : s}{\Gamma \vdash A : B'}$ якщо $B =_\beta B'$

Контекст Γ є коректно сформованим, якщо він є частиною вивідного судження: тобто існують A і B такі, що $\Gamma \vdash A : B$. Вираз M в λC є допустимим, якщо існують Γ і N такі, що M є типізованим термом $\Gamma \vdash M : N$ або заселеним типом $\Gamma \vdash N : M$. Абстрактний синтаксис множини усіх виразів в λC, $\mathcal{E}$, записується у вигляді:

$$\mathcal{E} \;=\; V \mid \square \mid \;*\; \mid (\mathcal{E}\;\mathcal{E}) \mid (\lambda V{:}\,\mathcal{E}.\mathcal{E}) \mid (\Pi V{:}\,\mathcal{E}.\mathcal{E}).$$

Числення конструкцій має сукупність важливих властивостей, зокрема сильну нормалізацію, збіжність, унікальність типів до перетворення, розв'язність задач перевірки типу та допустимості типізації. Задача пошуку терму не є розв'язною в λC, що власне задовольняє вимогам теореми про відсутність загального алгоритму доведення чи спростування довільного логічного висловлювання. Формулювання задач та пошук розв'язків все ще потребують участі експертів, але комп'ютерні програми, такі як асистент інтерактивного доведення теорем Coq [47], забезпечують суттєву підтримку для управління синтаксичною складністю процесу отримання логічного висновку та надають гарантію його коректності.

2.1.6. Числення індуктивних конструкцій, CIC

Числення індуктивних конструкцій – це формалізм, що програмно втілений в асистенті інтерактивного доведення теорем Coq. Він розширює теорію числення конструкцій індуктивними означеннями, що покликані забезпечити ефективне подання типів даних. Індуктивні означення дають інтуїтивне уявлення про такі поняття як досяжність та операційна семантика, які означаються за допомогою правил виведення [48]. CoC є достатньо виразним для подання індуктивних означень, проте воно має суттєві недоліки, зокрема низьку ефективність обчислення функцій на індуктивних типах даних, а також присутність деяких властивостей, які неможливо довести. Тому виникає необхідність розширення числення конструкцій шляхом введення примітивних індуктивних означень.

Для кожного оголошеного індуктивного типу даних Coq автоматично створює та доводить *принцип індукції* [49]. Узагальнений шаблон подання індуктивного означення в Coq виглядає наступним чином:

Inductive *Name Parameters* **:** *Arity* **:=**
Constructor1 : *Type1*
| *Constructor2* : *Type2*

Використання індуктивного означення в процесі виведення вимагає наявності правил його введення та усунення. Детальний аналіз оголошення індуктивного означення виявляє, що правила введення вже задані списком його конструкторів. Правила усунення використовують два різні підходи: 1) правило відповідності шаблону, розширене для залежних типів; 2) конструкцію нерухомої точки для рекурсивних означень [50]. В Coq конструкція перевірки виразу на відповідність шаблону задана командою «match». Маючи індуктивне означенння типу даних «Vector», що залежить від довжини вектора n : *nat* та параметризований типом елемента A : *Set*

```
Inductive Vector (A : Set) : nat -> Set :=
    vnull: Vector A 0
  | vcons: forall n, A -> Vector A n -> Vector A (S n).
```

Правило усунення, яке перетворює вектор натуральних чисел в суму його елементів, оголошується наступним чином:

```
Fixpoint vsum (n : nat)(v : Vector nat n) : nat :=
        match v with
          vnull => 0
        | vcons n x v' => x + (vsum v')
end.
```

Числення індуктивних конструкцій має сукупність важливих властивостей, зокрема розв’язність задачі перевірки типу. Разом з тим, означення рівності в CIC все ще є предметом активних досліджень у відносно новій формальній системі, гомотопічна теорія типів, HoTT [15].

2.2. Інтерактивний асистент доведення теорем Coq

Coq – це програмна реалізація формальної системи керування доведеннями. Він забезпечує середовище, що полегшує процеси явного подання контексту та перевірки гіпотез. Coq постачається з високоекспресивною мовою специфікації *Gallina*, мовою команд *The Vernacular*, основними і стандартними бібліотеками,

множиною атомарних тактик і виразів для обчислення доведень, мовою тактик L_{tac} та розширенням *SSReflect*. Керування процесом доведення відбувається в середовищі Coq IDE, яке здійснює управління ресурсами та надає користувачеві графічний інтерфейс та допоміжні інструменти, такі як навігація вперед та назад у Vernacular файлі. Coq система є гнучкою: її налаштування за замовчуванням можна легко розширити за допомогою користувальницьких модулів, а підказки для обчислення доведень можна зберігати та використовувати повторно через бази даних HintDb. Поточна версія Coq 8.10.2 постачається в сукупності з CoqIDE у вигляді файлів встановлення для операційних систем Windows 32-біт (i686), Windows 64-біт (x86_64) та macOS. Встановлення Coq та CoqIDE в операційній системі Linux виконується за допомогою інструменту командного рядка для управління пакетами «apt».

Мова специфікації в Coq, Gallina, має розвинутий синтаксис для розробки формалізованих теорій, що складаються з логічних об’єктів, таких як аксіоми, гіпотези, параметри, леми, теореми, означення констант, функцій, предикатів та множин [47]. Вона пропонує базовий синтаксис виразів числення індуктивних конструкцій (CIC), який можна розширити різними примітивами та налаштувати відповідно до уподобань користувача. Для забезпечення логічної коректності всі об’єкти в Coq є типізованими, що дозволяє створювати доведення специфікацій програм з використанням Gallina. З метою забезпечення логічного виведення вищого порядку, мова специфікацій оснащена трьома видами Prop, Set, Type і вимагає, щоб кожен тип був коректно сформованим. Побудова та скорочення термів здійснюються згідно з правилами системи виведення в численні індуктивних конструкцій. Поєднання з середовищем виконання здійснюється за допомогою використання команд Vernacular. Наявність модульної системи дозволяє ефективно використовувати існуючі специфікації і поширювати нові специфікації без ризику створення конфліктів, зокрема у масштабних специфікаціях, до розробки яких залучено багато експертів.

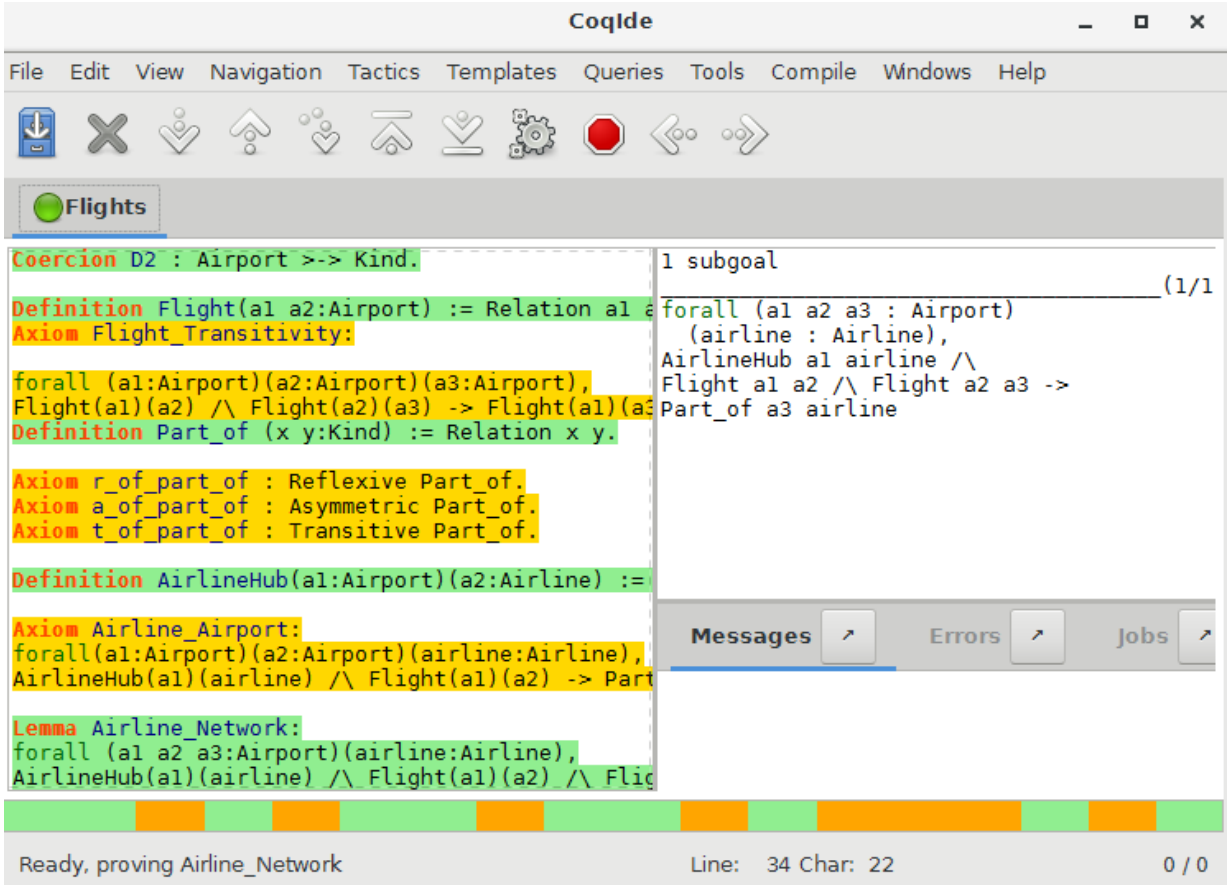

Рис. 2.2. Етап доведення леми в середовищі CoqIDE

Механізм доведення в Coq представлений множиною тактик, мовою тактик L_{tac} та мовою доведень *SSReflect*. Тактики – це дедуктивні правила, які втілюють процес зворотного міркування від висновку (ціль) до засновків (проміжні цілі). Застосована до висновку, тактика замінює ціль на підцілі, які вона породжує. Тактики формуються шляхом поєднання атомарних тактик і тактичних виразів. Одними з найпопулярніших тактик є *intros*, *destruct*, *apply*, *assumption*, тощо. Мова тактик L_{tac} надає синтакс та механізми опрацювання помилок, циклів, розгалужень в тактиках. Вона здебільшого використовується під час процесу доведення, але також може бути використана в глобальних означеннях. Розширення *SSReflect* є відносно новим елементом системи Coq, що втілює надійну методику доведення теорем *small-scale reflection*. Хоча множина тактик *SSReflect* перетинається з множиною тактик Coq, це розширення розроблене незалежно від Coq і має декілька істотних відмінностей,

зокрема, інший підхід до керування гіпотезами, наявність кроків рефлексії, тощо. Загалом, доведення, що створені в *SSReflect*, мають зовсім інший вигляд ніж ті, що використовують виключно тактики Coq.

2.2.1. Приклади логічних міркувань в Coq

Класичним прикладом застосуванням інтерактивного асистента доведення теорем Coq можна вважати формальне подання математичних теорій та програмних специфікацій мовою Gallina, для «сертифікованого» обчислення істинності їхніх теорем і положень інструментами теорії типів CIC. Відомим прикладом зазначеного застосування є доведення теореми з теорії графів про розфарбування планарного графа в чотири кольори, яке у 2005 році успішно «механізували» в середовищі Coq [51]. Процес доведення будь-якого математичного чи декларативного твердження вимагає його попереднє подання мовою Gallina, або його формалізацію засобами теорії типів, з подальшим перетворенням у вирази мови Gallina. Відповідність між цими системами подано нижче [52, 53]:

Таблиця 2.1.

Кодування виразів теорії типів мовою Gallina

Вирази теорії типів	***Конструкції мови Gallina***
Типи	
A	A
A → A	A -> A
$\Pi x{:}\,A.\,B$	forall x : A, B
□	Type
∗	Set | Prop | K : Type
∗ → ∗	Set -> Set | Prop -> Prop | K -> K
∗ → □	A -> Set | A -> Prop | A -> K
∗ : □	K : Type

Вирази теорії типів	***Конструкції мови Gallina***
$\alpha : *$	A : Set | A : Prop | A : K
$(* \to *)\ \alpha$	(fun x : K => x -> x) A
$\Pi x{:}\, A.\, B\ x$	forall x : A, B x
Терми	
$x{:}\,$A	x : A
$\lambda x{:}\,$A$.\, x$	fun x:A => x
$(\lambda x{:}\,$A$.\, x)\ z$	(fun x:A => x) z
$\lambda A : *.\, \lambda x : A.\, x$	fun {A : Type} (a : A) => a
$(\lambda A : *.\, \lambda x : A.\, x)$ B x	(fun {A : Type} (a : A) => a) x

Приклад 1. Довести істинність одного із законів класичної логіки – закону Пірса, який формулюється наступним чином $\big((P \to Q) \to P\big) \to P$.

Розв'язання. Зауважимо, що закон Пірса в класичній логіці є еквівалентним аксіомі виключення третього $A \vee \neg A$. Відповідно до ізоморфізму Каррі-Говарда, числення теорії типів є еквівалентне конструкціям інтуїціоністської логіки, в якій принцип виключення третього (LEM) не є аксіомою. Тому для доведення допустимості закону Пірса необхідно також сформулювати означення LEM.

Definition LEM := forall a : Prop, a ∨ ~a.

Definition Peirce := forall p q : Prop, ((p -> q) -> p) -> p.

Theorem Peirce_Eq_LEM: Peirce <-> LEM.

Proof.

unfold Peirce, LEM.

split; intros.

apply H with (q := ~ (a ∨ ~a)); tauto.

destruct (H p); tauto.

Qed.

Наведений код містить в собі вирази мови Gallina, а також доведення теореми з використанням атомарних тактик і тактичних виразів. Формалізацію елементів логіко-математичних теорії засобами теорії типів доцільно здійснювати згідно з наступним співвідношенням:

Таблиця 2.2.

Відповідність елементів теорії типів і логіко-математичних теорії [15]

Типи	***Логіка***	***Множини***	***Гомотопії***
A	висловлювання	множина	простір
$a: A$	доведення	елемент	точка
$B(x)$	предикат	сім'я множин	розшарування
$b(x): B(x)$	умовне доведення	сім'я елементів	секція
$\mathbf{0}, \mathbf{1}$	$\bot, \top$	$\emptyset, \{\emptyset\}$	$\emptyset$,$*$
$A + B$	$A \vee B$	диз'юнктне об'єднання	кодобуток
$A \times B$	$A \wedge B$	множина пар	простір добутків
$A \to B$	$A \Rightarrow B$	множина функцій	простір функції
$\sum_{(x:A)} B(x)$	$\exists_{(x:A)} B(x)$	диз'юнктна сума	визначений простір
$\prod_{(x:A)} B(x)$	$\forall_{(x:A)} B(x)$	добуток	простір секцій
Id_A	рівність =	$\{(x, x) \mid x \in A\}$	простір шляхів A^I

Варто зазначити, що заперечення $\neg A$ в теорії типів відповідає типу $A \to \mathbf{0}$, а еквівалентність $A \leftrightarrow B$ типу $(A \to B) \times (B \to A)$. Позначення логічних операцій, що присутні в прикладі, зокрема ~a та Peirce <-> LEM, введені у стандартній бібліотеці Coq.Init.Logic:

Inductive False : Prop :=.

Definition not (A:Prop) := A -> False.

Notation "~ x" := (not x) : type_scope.

Definition iff (A B:Prop) := (A -> B) /\ (B -> A).

Іншою важливою областю практичного застосування асистента доведення теорем Coq є перевірка коректності поведінки програми та властивостей деяких її компонент, згідно з специфікацією. Для виконання цього завдання необхідно подати специфікацію і реалізацію оцінюваної програми мовою Gallina, причому специфікація буде представлена довільним типом S, а реалізація – термом, $t{:}S$. Розв'язання задач теорії типів, зокрема перевірки типу і пошуку терму, дозволяє встановити відповідність реалізації програми до її специфікації, або ж навіть сконструювати цю реалізацію. Варто зазначити, що однією з фундаментальних властивостей системи Coq є можливість перетворити код мови Gallina в код таких функціональних мов як Haskell, OCaml та Scheme.

Відомим прикладом формальної перевірки коректності поведінки програми з використанням Coq є доведення безпекових властивостей технології Java Card [54], якій у 2007 році присвоєно навищий рівень безпеки та надійності, EAL7, згідно з міжнародним стандартом сертифікації комп'ютерної безпеки Common Criteria. Поява технології Blockchain та розвиток концепції Smart Contracts актуалізували потребу залучення методів формального підтвердження коректної поведінки розумних контрактів, зокрема, після втрати коштів смарт контрактом DAO [55]. У 2016 році провідні технічні університети США розпочали проект «The Science of Deep Specification», який покликаний створити систему формально перевіреного програмного та апаратного забезпечення [56].

Приклад 2. Створити поліморфний тип структури даних «Список», який підтримує асоціативну операцію об'єднання кількох списків.

```
(* Оголошення нового модуля *)
Module List.
    (* Індуктивне оголошення поліморфного типу «Список» *)
    Inductive list (X : Type) : Type :=
     | nil : list X
     | cons : X -> list X -> list X.
```

```
(* Оголошення змінної типу даних неявним аргументом *)
Arguments nil {X}.
Arguments cons {X} _ _.
(* Оголошення рекурсивної функції об’єднання списків *)
Fixpoint app {X : Type} (l1 l2 : list X) : (list X) :=
  match l1 with
  | nil {X} => l2
  | cons h t => cons h (app t l2)
end.
(* Синтаксичне позначення операції об’єднання списків *)
Notation "x ++ y" := (app x y) (at level 60, right associativity).
(* Теорема про асоціативність операції об’єднання списків і її доведення *)
Theorem app_associativity : forall A (l m n : list A),
   l ++ m ++ n = (l ++ m) ++ n.
Proof.
  intros.
  induction l as [|v l' IHl'].
  - simpl. reflexivity.
  - simpl. rewrite -> IHl'. reflexivity.
Qed.
End List.
```

2.2.2. Логічні міркування над природною мовою

Отримання логічних висновків з описових речень природної мови є відомою та актуальною задачою в контексті опрацювання природньої мови і розвитку наукового напряму штучний інтелект. Використання формальних систем для логічного міркування над твердженнями природньої мови вимагає задання формальної семантики, яка обґрунтовано відображає елементи речення природньої

мови у відповідні вирази формальної мови. Історично найвідомішою системою для логічного міркування над природньою мовою є логіка предикатів, а також її розв'язні підмножини – описові логіки, які гарантують можливість автоматичного обчислення логічного висновку. Відображення тверджень природної мови у вирази логіки предикатів є можливим завдяки формальній семантиці Е. Монтегю [57], оскільки ці формалізи спільно використовують семантику теорії моделей. Істотними недоліками логіки предикатів є обмежена виразність, а також можливість формування семантично некоректних виразів, що ставить під сумнів суб'єктивне сприйняття доведеного твердження як знання.

Вагомий розвиток формалізмів теорії типів, а також визначні результати в доведенні нетривіальних математичних теорем активізували науковий інтерес щодо можливості обчислення логічного висновку над твердженнями природньої мови, в контексті дедуктивної системи теорії типів. У 2014 році в науковій праці «Natural language inference in Coq» [58] запропоновано формальну семантику природньої мови, яка відображає елементи твердження у вирази «сучасної теорії типів» (MTT) – теорії з залежними типами і примусовою підтипізацією, яка ґрунтується на «уніфікованій теорії залежних типів» (UTT). Формалізм UTT є вельми подібним до предикативного числення індуктивних конструкцій, за винятком відсутності ко-індуктивних типів, тому асистент доведення теорем Coq реалізує MTT. Формальна семантика в MTT є водночас модельно-теоретичною та доказово-теоретичною, оскільки синтаксичні вирази MTT використовуються як для подання колекцій об'єктів і ситуацій, так і для формування суджень, що беруть участь в процесі доведення.

MTT-семантика інтерпретує різноманітні лінгвістичні категорії природньої мови відповідно до наступних правил [58]:

- Речення (S) інтерпретується як висловлювання типу $Prop$;
- Загальний іменник (CN) інтерпретується як тип;
- Змінений загальний іменник (MCN) інтерпретується як Σ-тип;

- Неперехідне дієслово (IV) інтерпретується як предикат над типом D, який інтерпретує область дієслова (функція типу $D \to Prop$);
- Прикметник (ADJ) інтерпретується як предикат над типом D, який інтерпретує область прикметника (функція типу $D \to Prop$).

Інтерпретація складніших лінгвістичних категорій, зокрема прислівників (VP), підрядних прикметників (ADJ_{subs}) та кванторів, здійснюється за допомогою конструкторів залежних типів – Σ та П. Їхнє застосування потребує наявності простору загальних іменників $CN:Set$. Ідея полягає в тому, що прислівники, підрядні прикметники та квантори інтерпретуються як поліморфні предикати над простором загальних іменників CN:

- Прислівник (VP): $\Pi A{:}\, CN.\,(A \to Prop) \to (A \to Prop)$;
- Підрядний прикметник (ADJ_{subs}): $\Pi A{:}\, CN.\,A \to Prop$;
- Квантор: $\Pi A{:}\, CN.\,(A \to Prop) \to Prop$.

Ієрархія типів реалізується механізмом примусової підтипізації, **Coercion**, який надається системою Coq.

Приклад 3. Наведемо спосіб подання лінгвістичних категорій природньої мови в середовищі Coq:

Definition CN := Set.

Parameters Engineer Manager Employee Person: CN.

Axiom ma : Engineer -> Employee. **Coercion** ma : Engineer >-> Employee.

Axiom ea : Manager -> Employee. **Coercion** ea : Manager >-> Employee.

Axiom ae : Employee -> Person. **Coercion** ae : Employee >-> Person.

Parameter Vasyl : Engineer.

Parameter can_program: Employee -> Prop.

Parameter experienced : forall A:CN, A -> Prop.

Definition all := fun A:CN => fun P:A->Prop => forall x:A, P(x).

Definition some := fun A:CN => fun P:A->Prop => exists x:A, P(x).

Definition no := fun A:CN => fun P:A->Prop => forall x:A, not(P(x)).

Логічні міркування над твердженнями природної мови в середовищі Coq є інтуїтивно зрозумілими. Оскільки семантичний наслідок відповідає відношенню імплікації між двома різними семантичними структурами, то його істинність можна підтвердити шляхом доведення теореми в системі Coq: якщо відношення імплікації між двома семантичними структурами є доведеною теоремою, то семантичний наслідок – коректний.

Приклад 4. Довести істинність твердження природної мови: «Василь є інженером. Василь вміє програмувати. Деякі працівники вміють програмувати».

```
Theorem sem_entail: can_program Vasyl -> exists e : Employee, can_program e.
Proof.
    intros.
    exists Vasyl.
    assumption.
Qed.
```

Оскільки теорема доведена, то семантичний наслідок є істинним. Зауважимо, що декларативний спосіб подання тверджень природньої мови дозволяє формувати базу тверджень, а механізм формулювання теорем створює можливість перевірки їх логічної сумісності. Незважаючи на те, що пошук терму (доведення теореми) в CIC є нерозв'язною задачею, користувацькі тактики гарно зарекомендували себе в процесі автоматизації доведення теорем. Описана функціональність властива «базам знань» – класу програмного забезпечення, який реалізує інтерфейс для подання, зберігання, зміни і використання знання, з можливістю формулювання логічних запитів.

2.3. Онтологічно-орієнтований метод подання знання в Coq

Онтологічно-орієнтований підхід до впорядкування знання сприяє повторному використанню існуючих онтології. Онтологію формально можна визначити як кортеж $\langle C, R, F \rangle$, де C – це множина понять предметної області, R: $C \to C$ – скінченна

множина відношень між поняттями, F – скінченна множина функцій інтерпретації. Структура онтології складається з понять, що впорядковані у вигляді таксономії, їхніх властивостей та ознак, правил виведення, аксіом та обмежень предметної області. Для онтологій характерним є мереологічне відношення «частина-ціле», яке використовується для опису понять, які можуть виконувати роль абстракції понять нижчого порядку. Онтологія разом із сукупністю екземплярів класів формує базу знань. Виразність мови специфікацій в Coq є більш ніж достатньою для подання основних компонент структури будь-якої онтології [59, 60]:

Таблиця 2.3.

Подання онтологічної структури в середовищі Coq

Елемент онтології	**Вираз в Coq**
Поняття	Class C : Type.
Екземпляр	Instance X : C.
Властивості, правило виведення	Class C (att1:nat) (att2:bool) : Type := { f1: attr1 −> bool; }.
Бінарне відношення	Parameter R : C −> C −> Prop.
Наслідування понять	Parameter D1 : SubClass_G −> C. Coercion D1 : SubClass_G >−> C.
Відношення «частина-ціле»	Definition PartOf (x y : C) := R x y. Axiom A1 : Reflexive PartOf. Axiom A2 : Asymmetric PartOf. Axiom A3 : Transitive PartOf.
Квантори ∃,∀	exists X : C, forall X : C.

Базовий структурний елемент, що подає довільний факт онтології, називається «трійкою». Трійка, це твердження у формі ⟨суб’єкт, присудок, об’єкт⟩, що подає зв’язок, означений присудком між суб’єктом і об’єктом. У трійці суб’єкт і присудок – це URI, що вказують на веб-ресурси, тоді як об’єктом може бути або URI, або ім’я, що подає значення. Множину трійок зручно візуалізувати у вигляді орієнтованого графа, де кожна трійка позначає зв’язок між суб’єктом та об'єктом.

Розробка програмного забезпечення є однією з предметних областей, для якої онтологічне моделювання знань і можливість формальної перевірки гіпотез, є важливими компонентами забезпечення надійності кінцевого продукту. В останні роки життєвий цикл розробки програмного забезпечення передбачає взаємодію з десятками допоміжних інструментів, які зокрема призначені для: контролю версій, CI/CD, управління проєктами, аудиту безпеки, забезпечення якості коду, моніторингу журналів подій, тощо. Коли працівник компанії приєднується до проєкту, отримання допусків до всієї інфраструктури проєкту може тривати кілька годин або днів. Натомість, онтологія управління доступом із сертифікованою поведінкою в Coq не лише економить людино-години, але й зменшує ризик встановлення неправильних допусків.

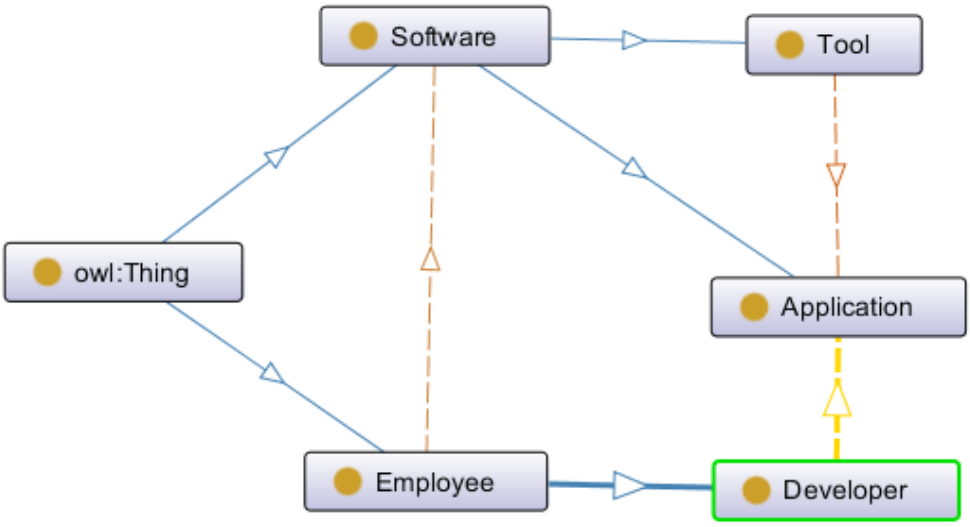

Рис. 2.2. Фрагмент онтології управління доступом до програмного забезпечення проєкту

Приклад 5. Фрагмент онтології «Управління доступом до програмного забезпечення проєкту», де онтологія $O = \langle C, R, F \rangle$, C = {Software; Employee; Tool; Application; Developer}; R = { IsAppDeveloper; IsAppTool; AccessGranted }; F = { Employee_App_Access; App_Developer} подаються згідно з наведеним онтологічно-орієнтованим методом подання знань в Coq:

1. Завантажуємо необхідні Coq модулі:

Require Import Coq.Classes.RelationClasses.

2. Означаємо сутності відношення та базового поняття:

Definition Kind := Type.

Parameter Relation : Kind -> Kind -> Prop.

3. Оголошуємо таксономію понять:

Class Software: Type.

Parameter D1: Software-> Kind.

Coercion D1: Software>-> Kind.

Class Tool: Type.

Parameter D2 : Tool -> Software.

Coercion D2 : Tool >-> Software.

Class Employee: Type.

Parameter D4 : Employee-> Kind.

Coercion D4 : Employee>-> Kind.

Class Developer: Type.

Parameter D5 : Developer-> Employee.

Coercion D5 : Developer >->Employee.

Class Application: Type.

Parameter D3 : Application-> Software.

Coercion D3 : Application>-> Software.

4. Оголошуємо відношення між поняттями:

Definition isAppTool(t:Tool)(a:Application) := Relation t a.

Definition accessGranted(e:Employee)(s:Software) := Relation e s.

Definition isAppDeveloper(d:Developer)(a:Application) := Relation d a.

5. Оголошуємо функції інтерпретації:

Axiom App_Developer :

forall (d:Developer)(a:Application),

isAppDeveloper(d)(a) -> accessGranted(d)(a).

Axiom Employee_App_Access :

forall (e:Employee)(a:Application)(t:Tool),

accessGranted(e)(a) /\ isAppTool(t)(a) -> accessGranted(e)(t).

Фрагмент онтології встановлює взаємозв’язки між поняттями «Працівник компанії», а саме «Програміст», і захищеним «Програмним забезпеченням», що включає «Застосунок» та «Інструмент». Коли працівник приєднується до проєкту як розробник програми, isAppDeveloper, йому надається доступ до застосунку, accessGranted, згідно з аксіомою App_Developer. Оскільки процес розробки застосунків зазвичай передбачає взаємодію з набором допоміжних інструментів, isAppTool, аксіома Employee_App_Access надає доступ до цих інструментів усім, хто має доступ до застосунку. Якщо «Програміст» чи «Інструмент» покидають проєкт, відповідні права доступу чи приналежності також припиняються.

Приклад 6. Запишемо та доведемо відповідну лему онтології «Управління доступом до програмного забезпечення проєкту» в Coq:

Lemma Tool_Dev_Access :
forall (d:Developer)(a:Application)(t:Tool),
isAppDeveloper(d)(a) /\ isAppTool(t)(a) -> accessGranted(d)(t).
Proof.
intros d a t H1.
destruct H1 as [H1 H2].
apply App_Developer in H1.
assert(H3:accessGranted(d)(a) /\ isAppTool(t)(a))..
split; assumption.
apply Employee_App_Access in H3.
assumption.
Qed.

Доведення леми засвідчує очікувану поведінку онтології, коли кожному розробнику застосунку автоматично надається доступ до кожного інструменту, що використовується в межах застосунку.

2.4. Висновки до розділу 2

Теорія типів представлена розмаїттям формальних систем, які зазвичай відрізняються виразністю та обчислювальними властивостями. Оскільки просто-типізоване лямбда числення, $\lambda\!\rightarrow$, характеризується низькою виразністю, зокрема не допускає рекурсивних функцій, то розглянуто три напрямки його розширення – поліморфними термами, конструкторами типів і залежними типами. Числення конструкцій *CoC* об'єднує ці три напрямки в межах єдиної системи виведення і розширюється індуктивними означеннями до *CIC*. Більшість задач теорії типів є розв'язними в *CIC*, за винятком задачі пошуку терма, яка згідно з ізоморфізмом Каррі-Говарда відповідає задачі доведення теореми. Ця поведінка узгоджується з теоремою Черча-Тюрінга, тому доведення теорем в *CIC* може бути лише частково автоматизованим.

Інтерактивний асистент доведення теорем Coq втілює теорію типів *CIC* і надає мову специфікацій Gallina для подання знань. Обгрунтування поданих знань здійснюється шляхом формулювання теорем та їх доведення за допомогою атомарних тактик і мови L_{tac}. Формалізації та доведенню в Coq піддаються знання різної природи: математичні та логічні теорії, програмні специфікації і твердження природньої мови.

Основні наукові результати розділу отримані у формі прикладних досліджень:

- здійснено порівняльний аналіз формальних систем теорії типів та особливостей їх застосування з метою подання знань різної природи, зокрема описових речень природньої мови, логічних і математичних теорій та виразів;
- розроблено онтологічно-орієнтований метод подання бази знань в середовищі асистента доведення теорем Coq, що сприяє формалізації нових та існуючих онтологій, а також уможливлює застосування логік вищих порядків як інструменту підтримки процесу прийняття рішення;

- розроблено приклади формального доведення істинності описових знань, зокрема для задачі управління правами доступу до програмних проєктів, що спрощує і пришвидшує процес перевірки гіпотез в напівавтоматичному режимі;
- обґрунтовано застосування числення індуктивних конструкцій для подання та міркування над знаннями різної природи, а також відображено механізм реалізації його виразів формальною мовою Gallina в асистенті доведення теорем Coq;
- досліджено семантику формалізації речень природної мови в середовищі асистента доведення теорем Coq, яка базується на сучасних теоріях типів, і уможливлює міркування над формальним поданням різновидів тверджень природної мови.

РОЗДІЛ 3. ПОДАННЯ ТА МІРКУВАННЯ НАД ЗНАННЯМИ З ВИКОРИСТАННЯМ ОПИСОВИХ ЛОГІК

Описові логіки (DL) відіграють фундаментальну роль в сучасних онтологіях, що засновані на стандартах OWL та OWL 2. Їхня роль полягає у формуванні основи для прямої теоретико-модельної семантики [61] OWL онтологій, які де-факто є стандартом проєктування баз знань. Історична еволюція онтологій від фреймової моделі подання знання до моделей, що засновані на логіці, пояснюється проблемами неоднозначної інтерпретації фреймових синтаксичних конструкцій та обмеженої універсальності методів міркування на фреймах [2]. Описові логіки як розв'язні фрагменти логіки першого порядку, пропонують розвинену теоретико-модельну семантику та обґрунтовані методи (форми) коректного міркування над синтаксичними виразами.

Для OWL 2 найвідповіднішою семантикою серед описових логік є *SROIQ*, а для мови OWL – описова логіка *SHOIN*. Іменування описових логік здійснюється шляхом використання абревіатур множин конструкторів (аксіом), які формують основу її граматики. Наприклад *SROIQ* складається з наступних елементів: *S* – позначає множину конструкторів найпростішої описової логіки *ALC*, розширеної транзитивними ролями (відношеннями); *R* – позначає розширену множину аксіом ролей, яку ще називають *RBox*; *O* – позначає номінали; *I* – позначає зворотні ролі; *Q* – позначає обмеження кардинальності ролі. Розмаїття екземплярів описових логік пояснюється комбінаторними можливостями вибору множини конструкторів, проте формування описової логіки з привабливими характеристиками виразності та числення є складною науковою задачею. Причина полягає у фундаментальному компромісі між виразністю та розв'язністю формальної мови – чим вона виразніша, тим вища ймовірність відсутності ефективного алгоритму для здійснення числення. Проєктування описових логік потребує формального доведення їхніх властивостей, а також ретельного підбору множини конструкторів.

Проведемо системний аналіз: 1) структури описових логік, на прикладі базової логіки *ALC* та її розширень; 2) основних задач, що подаються та розв'язуються формальним апаратом описових логік; 3) поняття бази знань в контексті описових логік; 4) фундаментальних властивостей описових логік; 5) теоретико-модельної семантики і методів числення логічного висновку. Системний аналіз зазначених аспектів створює підґрунтя для порівняльного аналізу описових логік та теорій типів як формальних моделей подання знання і дедуктивних методів міркування над ним. Результати дослідження опублікованої наукової літератури свідчать про відсутність порівняльного аналізу між вищезгаданими логічними формалізмами.

3.1. Описові логіки

Описові логіки – це сімейство логіко-орієнтованих мов подання знань, що бере свій початок з кінця 80-х років минулого століття. Назва «описова логіка» відповідає онтологічним зобов'язанням цього формалізму, адже основні аспекти предметної області подаються у вигляді «опису» її понять (концепти) та відношень (ролі), шляхом застосування конструкторів (аксіом) обраної описової логіки. Для зручності повторного використання знань, формальні описи класифікуються за ознакою загальності та розділяються на дві частини – набір термінологічних аксіом, *TBox*, та набір тверджень про індивідів, *ABox*, які в сукупності формують базу знань, *KB*, також відому як онтологія.

Таблиця 3.1

Співвідношення елементів логічних формалізмів та понять OWL

Логіка предикатів	***Описові логіки***	***OWL***
унарний предикат	поняття	клас
бінарний предикат	роль	властивість
константа	індивід	індивід

Фундаментальна властивість описових логік полягає в наявності формальної логічної семантики, яка дозволяє інтерпретувати коректно сформовані твердження, аксіоми та операції над ними, створюючи таким чином однозначне, спільне розуміння того, коли саме твердження є логічним наслідком з бази знань. Оскільки описові логіки є розв'язними фрагментами логіки предикатів, то для обчислення логічної вивідності твердження використовуються методи автоматизованого міркування, які ґрунтуються на принципах семантичної допустимості і дедуктивної обчислюваності твердження. Особливістю міркування в описових логіках є те, що числення завжди здійснюється над усією базою знань. Концептуальний компроміс між виразністю мови подання тверджень та обчислювальною складністю задач міркування, а також кількість тверджень в базі знань, породжують необхідність в різновиді описових логік. Зокрема, сімейство описових логік *EL* характеризується обмеженою виразністю мови подання тверджень, проте гарантує поліноміальний час обчислення деяких задач міркування у великих онтологіях.

Виразність описових логік є завжди обмеженою. Ця властивість обумовлена потребою забезпечення розв'язності задач міркування, і може бути посилена з метою забезпечення розв'язності задач за поліноміальний час стосовно розміру бази знань. Розв'язність, зокрема за поліноміальний час, є основним фактором використання описових логік для побудови сучасних баз знань, тому проєктування різновидів спеціалізованих описових логік потребує ретельного аналізу та формального доведення їх властивостей. Необхідність спричинена тим, що поєднання розширень описових логік, які поодинці характеризуються розв'язністю, може породжувати ситуації нерозв'язності. Якщо виразність описової логіки є недостатньою для подання предметної області, то замість використання нерозв'язних описових логік, її вбудовують у прикладні програми або альтернативні моделі подання знань. Визначення видів семантичних структур, що можуть бути подані в деякій описовій логіці, здійснюється в рамках формальної теорії моделей, яка зокрема визначає семантику логіки першого порядку.

Деякі види описових логік виходять за рамки логіки першого порядку, оскільки можуть використовувати оператори модальності, елементи нечіткої логіки чи теорії ймовірності. Синтаксично твердження описових логік є близькими за формою до виразів модальних логік, проте їхні значення істинності встановлюються згідно з відношеннями семантичної теорії істинності А. Тарського [62], яка є парадигмою означення істинності в теорії моделей. Незважаючи на відмінності, що присутні в різновидах описових логік, процес побудови онтологій є здебільшого однаковим. Спершу визначається словник предметної області, який потім формалізується у відповідний *TBox*. Деякі застосунки працюють лише з термінологічною онтологією, в той час як інші ще використовують онтологію для структурування та доступу до інформації в *ABox* чи навіть базах даних. Коректність процесу побудови онтологій, а також механізм опрацювання запитів користувачів до бази знань, забезпечуються модулем автоматичного або напівавтоматичного логічного міркування, який як правило є частиною середовища розробки онтологій.

3.1.1. Базова описова логіка *ALC*

Використання описових логік для концептуального моделювання предметної області передбачає опис деяких абстракцій, що їй властиві, а також індивідів, що є об’єктами цих абстракцій. Розрізняють чотири складові частини описових логік, які беруть участь у процесі створення формального подання предметної області [2]:

1) *Поняття* (концепти) – це абстракції, які будуються з *імен понять* та *імен ролей*, шляхом застосування конструкторів описової логіки. Інтерпретацією поняття в теорії моделей є множина елементів, а в логіці першого порядку йому відповідає унарний предикат.
2) *Ролі* (імена ролей) – це відношення між двома поняттями або індивідами понять. В теорії моделей інтерпретацією ролі є множина пар екземплярів, що належать до відповідних реалізацій понять, а в логіці першого порядку – бінарний предикат.

3) *Індивіди* (екземпляри) є втіленням абстракції поняття у формі об'єктів та покликані іменувати об'єкт певної абстракції. В теорії моделей індивід інтерпретується як елемент множини, яка відповідає поняттю, а в логіці першого порядку – константі.
4) *Понятійна (концептуальна) мова* – це формальна мова, яка за допомогою конструкторів описової логіки, створює описи понять та ролей з примітивів, зокрема імен понять та імені ролей. Інтерпретація конструкторів понятійної мови в теорії моделей ставить їм у відповідність операції над множинами, а в логіці першого порядку конструкторам співставляються логічні операції.

Атрибутивна мова з доповненнями, *ALC*, належить до сімейства описових логік *AL* – атрибутивних мов, що забезпечують операції атомарного заперечення, перетинів понять, квантори загальності та існування. На відміну від сімейства *AL*, де дозволені доповнення лише атомарних понять, мова *ALC* дозволяє доповнення будь-яких понять [63]. До атомарних понять належать імена понять, «верхнє» $\top$ і «нижнє» $\bot$ поняття; решта понять є складеними, адже формуються застосуванням операторів понятійної мови. Таким чином, множина *ALC* описів понять складається з імен понять, імен ролей, імен індивідів, $\top$ та $\bot$, перетину та об'єднання понять, заперечення чи доповнення поняття, а також обмежень значення та існування. Синтаксис формальної мови дозволяє розрізняти коректно сформовані вирази від довільних. Індуктивне означення синтаксису понять в *ALC* подано нижче:

- Кожне ім'я поняття є *ALC* описом поняття;
- $\top$ та $\bot$ є *ALC* описами понять;
- Якщо C та D є *ALC* описами понять, а r є іменем ролі, то *ALC* описами є:
  - $C \sqcap D$ (кон'юнкція або перетин понять);
  - $C \sqcup D$ (диз'юнкція або об'єднання понять);
  - $\neg C$ (заперечення або доповнення поняття);
  - $\exists r.C$ (обмеження існування);
  - $\forall r.C$ (обмеження значення).

Формальна семантика мови дозволяє однозначно встановити зміст її коректно сформованих виразів, а також є потрібною для введення предиката істинності, який присвоює значення логічним твердженням мови. Семантику можна розглядати як формальну метамову, структури якої описуються синтаксисом формальної мови; для семантики формальної мови теж існує метамова. Семантична теорія істинності А. Тарського [62] стверджує, що для уникнення в формальній мові парадоксу Рассела чи парадоксу брехуна, предикат істинності повинен означатися виключно у метамові. Ця парадигма дозволяє уникати несумісності логічної системи, що виникає через такі твердження як «Це речення є брехнею», адже його істинність визначається за межами мови, в якій воно сформульоване.

Семантикою описових логік є *інтерпретація* – структура, що характеризується наступними властивостями: 1) складається з непустої множини елементів, яка називається *областю інтерпретації*; 2) для кожного імені поняття визначає його реалізацію, тобто підмножину елементів з області інтерпретації; 3) для кожного імені ролі визначає множину пар елементів з області інтерпретації, які перебувають у відношенні цієї ролі. Формально, інтерпретація $I = (\Delta^I, \cdot^I)$ складається з непустої множини Δ^I, яка називається областю інтерпретації, та відображення $\cdot^I$, що відображає кожне ім'я поняття $A \in \boldsymbol{C}$, в множину $A^I \subseteq \Delta^I$, і кожне ім'я ролі $r \in \boldsymbol{R}$ в бінарне відношення $r^I \subseteq \Delta^I \times \Delta^I$. Інтерпретація інших описів понять здійснюється шляхом розширення відображення $\cdot^I$ такими правилами:

- $\top^I = \Delta^I$ (вся область інтерпретації),
- $\bot^I = \emptyset$ (пуста множина);
- $(C \sqcap D)^I = \mathrm{C}^I \cap D^I$ (перетин реалізацій понять);
- $(C \sqcup D)^I = \mathrm{C}^I \cup D^I$ (об'єднання реалізацій понять);
- $(\neg \mathrm{C})^I = \Delta^I \backslash \mathrm{C}^I$ (область інтерпретації без реалізації поняття);
- $(\exists r.C)^I = \{d \in \Delta^I \mid$ існує $e \in \Delta^I$, таке що $(d, e) \in r^I$ та $e \in \mathrm{C}^I\}$;
- $(\forall r.C)^I = \{d \in \Delta^I \mid$ для всіх $e \in \Delta^I$, якщо $(d, e) \in r^I$, то $e \in \mathrm{C}^I\}$.

Властивості інтерпретації обмежені виключно правилами, що подані явно в означенні, а саме: область інтерпретації повинна бути не пустою, але може бути довільної потужності, зокрема нескінченною; реалізація імені поняття може мати будь-яку кількість елементів, від нуля до всієї області інтерпретації; реалізація імені ролі може містити довільну кількість пар елементів, від нуля до усіх можливих пар. Зміна інтерпретації I, шляхом додавання чи видалення елементів з реалізації імен понять чи ролей, або з області інтерпретації, породжує альтернативні інтерпретації. Аналіз структури інтерпретації зручно здійснювати шляхом її подання у вигляді орієнтованого графа з мітками вершин та дуг. В цьому графі вершина відповідає елементу області інтерпретації, а мітки вершини позначають поняття, що містять зазначений елемент у свої реалізації; дуга графа позначає пару елементів, яка міститься в реалізації деякої ролі r, а мітка дуги відповідає імені цієї ролі.

Приклад 1. Змоделювати предметну область «Структура ІТ проєкту» шляхом використання описової логіки *ALC* та означити її інтерпретацію в контексті проєкту «Розумне місто» [64].

Розв'язання. Перший крок моделювання полягає у визначенні імен понять, імен ролей, які є властивими заданій предметній області. Нехай множина імен понять $\boldsymbol{C} =$ {Інженер, Технологія, Проєкт}, множина імен ролей $\boldsymbol{R} =$ {є_частиною, володіє}. Другий крок моделювання полягає в поданні інтерпретації для зазначених множин, елементи якої зазвичай відповідатимуть конкретній ситуації і прикладним цілям моделювання:

$\Delta^I =$ {Василь, Андрій, Javascript, Coq, Розумне_місто};

Інженер^I = {Василь, Андрій};

Технологія^I = {JavaScript, Coq, Розумне_місто};

Проєкт^I = {Розумне_місто};

є_частиною^I = {(Василь, Розумне_місто), (Coq, Розумне_місто)};

володіє^I = {(Василь, Coq), (Андрій, Javascript)}.

Означену інтерпретацію концептуальної моделі предметної області доцільно зобразити у вигляді графа, з метою аналізу структури зв’язків між екземплярами понять та виявлення кластерів:

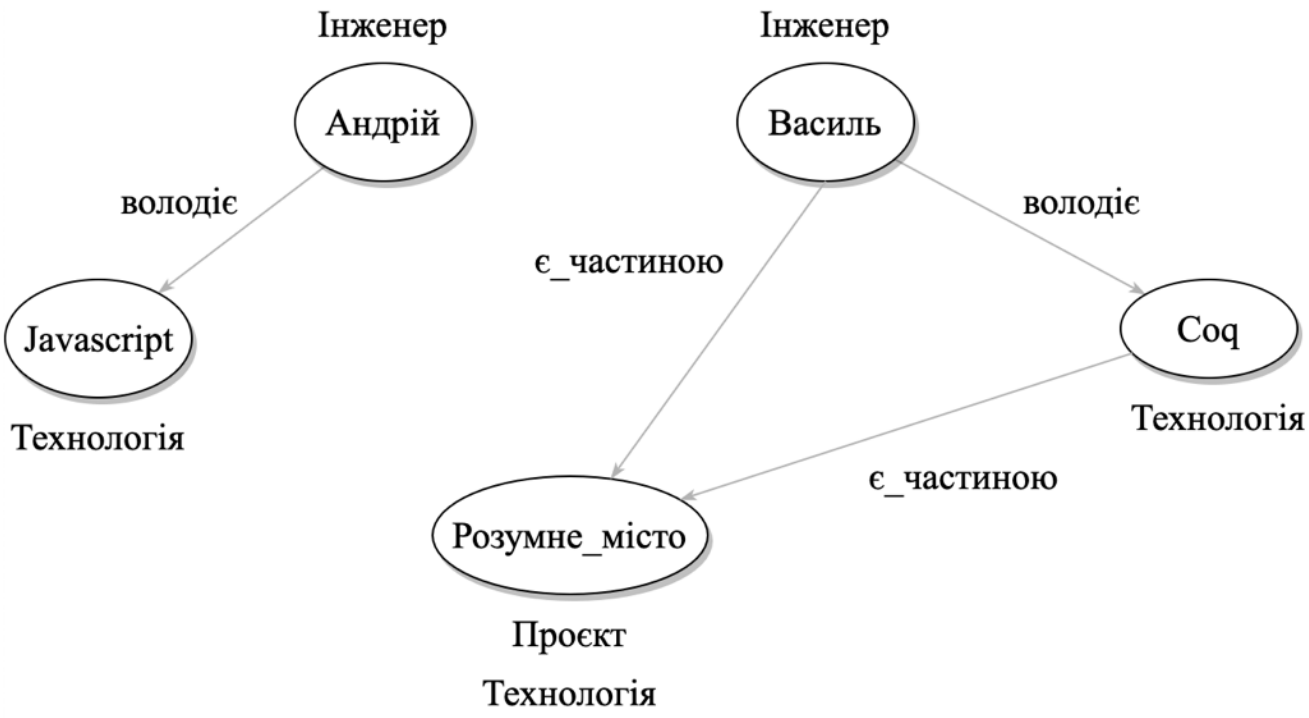

Рис. 3.1. Графічне подання інтерпретації у вигляді графа

Третій крок моделювання полягає у формалізації елементів інтерпретації за допомогою синтаксису описової логіки *ALC*. Варто зазначити, що способів подання елементів інтерпретації може бути кілька і всі вони будуть коректними, проте їхня ефективність для досягнення практичних цілей моделювання може відрізнятися. Наприклад, екземпляр «Розумне_місто» належить до реалізації атомарного поняття *Проєкт* та складеного поняття *Проєкт*⊓*Технологія*, а також реалізацій таких понять як ¬*Інженер* та ⊤. Екземпляр «Василь» належить до реалізацій понять *Інженер*, ∀*володіє.Технологія*, ∃*є_частиною.Проєкт* та багато інших. Усі елементи області інтерпретації завжди належать до реалізації верхнього поняття ⊤ та водночас жоден з них не належить до реалізації нижнього поняття ⊥, яке в системах логічного міркування відповідає стану абсурду.

Означення семантики описової логіки *ALC* містить правила, які дозволяють сформувати набір корисних лем для ефективного здійснення процесу міркування. Їхній синтаксис значною мірою нагадує дедуктивні форми аргументів логіки першого порядку, а саме закон подвійного заперечення, закони де Моргана, тощо. Нижче подано леми описової логіки *ALC* [2], доведення яких ґрунтується на властивостях інтерпретації I:

- $\top^I = (C \sqcup \neg \mathrm{C})^I$ (закон виключеного третього);
- $\bot^I = (C \sqcap \neg \mathrm{C})^I$ (доведення до абсурду);
- $(\neg\neg \mathrm{C})^I = \mathrm{C}^I$ (закон подвійного заперечення);
- $\neg(C \sqcap D)^I = (\neg C \sqcup \neg D)^I$ (закон де Моргана);
- $\neg(C \sqcup D)^I = (\neg C \sqcap \neg D)^I$ (закон де Моргана);
- $(\neg(\exists r.C))^I = (\forall r.\neg C)^I$ (різновид закону де Моргана);
- $(\neg(\forall r.C))^I = (\exists r.\neg C)^I$ (різновид закону де Моргана).

3.1.2. Бази знань в описовій логіці *ALC*

Можливість побудови формальних описів понять предметної області та їхнє поєднання логічними операторами понятійної мови є ґрунтовним досягненням, проте не достатнім. Послідовності синтаксичних перетворень та логічного синтезу формальних описів понять можуть прямувати до нескінченності, тому враховуючи обмеженість ресурсів пам’яті та часу на здійснення повторних обчислень, є зміст запам’ятовувати корисні (шляхом іменування) та суміжні до них описи (шляхом відношення). З цією метою в описовій логіці *ALC* вводяться наступні синтаксичні конструкції:

- $C \sqsubseteq D$ $= \mathrm{C}^I \subseteq D^I$ (включення поняття);
- $C \equiv D$ $= \mathrm{C}^I \subseteq D^I$ та $D^I \subseteq \mathrm{C}^I$ (еквівалентність поняття);
- $a\colon C$ $= a^I \in \mathrm{C}^I$ (твердження поняття);
- $(a, b)\colon r$ $= (a^I, b^I) \in r^I$ (твердження ролі).

Означені відношення дозволяють структурувати елементи бази знань згідно з класичними підходами до систематизації і повторного використання понять та їх властивостей. Наприклад C, оператор еквівалентності ≡ встановлює зміст поняття C через опис поняття D, що відповідає енциклопедичному формату означення змісту понять. Подання фонових знань зручно здійснювати з використання оператора включення поняття ⊑, який означає приналежність поняття C до більш загального поняття D, і таким чином неявно приписує йому деякі властивості та операційні обмеження. Подання конкретних ситуацій потребує наявності реалізації понять та ролей у формі припущень, тому оператор «:» встановлює відношення приналежності індивіда a до поняття C та пари індивідів (a, b) до ролі r відповідно. Нижче наведено приклади використання зазначених операторів:

$$\text{Технологія} \sqsubseteq \text{Методи} \sqcup \text{Засоби}$$

$$\text{Інженер} \equiv \text{Людина} \sqcap \exists \text{володіє.Технологія}$$

$$\text{Василь} : \text{Інженер};$$

$$\text{Coq} : \text{Технологія};$$

$$(\text{Василь}, \text{Coq}) : \text{володіє}$$

В описових логіках бази знань зазвичай поділяють на дві структурні частини: термінологічну, TBox, і декларативну, ABox. В базах знань загального призначення або термінологіях, декларативна частина може бути відсутньою. Термінологічна частина містить твердження, що описують поняття та відношення між ними (⊑, ≡), а декларативна – твердження, що описують індивіди понять та відношення між ними у вигляді ролей (:). Порівнюючи структуру бази знань та бази даних, можна встановити суттєву подібність TBox до схеми бази даних, оскільки обоє містять загальні обмеження: на форму концептуальної моделі предметної області та форму даних відповідно. ABox подібний до вмісту бази даних, тому що він містить вирази, що стосуються реалізації абстракцій, зокрема конкретні твердження понять та ролей.

Формально *TBox* складається зі скінченної множини тверджень форми $C \sqsubseteq D$, де C та D можуть бути описами як атомарних понять, так і складених. Якщо $C \sqsubseteq D$ та $D \sqsubseteq C$, то для зручності запису використовується позначення $C \equiv D$. Кожне з наведених тверджень має назву *загальне включення поняття* (GCI) або *аксіома* [2]. Інтерпретація I *задовольняє* GCI, якщо виконується $C^I \subseteq D^I$. Якщо інтерпретація I задовольняє кожне GCI, що містить *TBox* T, тоді вона називається *моделлю* T. Чим більше GCI містить *TBox*, тим менше моделей для нього існує, оскільки для двох $TBox\ T, T'$, якщо $T \subseteq T'$, то кожна модель T' є моделлю T. *TBox* T розділяє множину інтерпретацій на дві частини – множину моделей T та множину інтерпретації, які не задовольняють усі GCI з T. Таким чином, *TBox* зосереджує увагу моделювання виключно на інтерпретаціях, які відповідають інтуїтивним очікуванням інженера про структуру предметної області.

Декларативна частина бази знань, *ABox*, складається зі скінченної множини тверджень, яким властиві форми $a: C$, твердження поняття, чи $(a, b): r$, твердження ролі. Передбачається, що функція інтерпретації $\cdot^I$ ставить у відповідність кожному екземпляру a елемент $a^I \in \Delta^I$. Інтерпретація I *задовольняє* твердження поняття $a: C$, якщо $a^I \in C^I$, а також *задовольняє* твердження ролі $(a, b): r$, якщо $(a^I, b^I) \in r^I$. Якщо інтерпретація I задовольняє усі твердження понять та ролей, що містяться в *ABox* A, тоді вона називається *моделлю* A.

Поєднання *TBox* T та *Abox* A формує *ALC* базу знань $K = (T, A)$, в якій поняття та їх значення подаються в термінологічній частині, і згодом використовуються в декларативній частині. Інтерпретація I, яка водночас є моделлю T та моделлю A, називається *моделлю* K. База знань за структурою є подібною до бази даних, проте суттєва відмінність полягає у підході, щодо забезпечення узгодженості вмісту. Якщо дані відсутні в БД, то їхня істинність вважається хибою і може трактуватися як порушення цілісності даних. Для бази знань цілком прийнятною є ситуація, коли є опис поняття в *TBox* без опису екземпляра поняття в *ABox*. Цей підхід називається *гіпотезою відкритості світу* [65].

Синтаксис описової логіки ALC дозволяє формувати *циклічні* описи понять. Наприклад, поняття $OpenSource \equiv$ Технологія $\sqcap \forall$використовує$.OpenSource$ є циклічним, оскільки ім'я поняття $OpenSource$ з'являється в описі свого ж значення. Такі вирази можуть задовольнятися кількома інтерпретаціями, що може призвести до неоднозначності розв'язку деяких задач міркування. З метою уникнення вказаної проблеми, вводиться синтаксичне обмеження на формування циклічних описів, що збільшує *означальну потужність TBox* і породжує поняття *ациклічний TBox*. Варто зазначити, що форма циклічності поняття $OpenSource$ є простою та очевидною, проте існують приховані форми, які виникають через вкладеність описів понять.

Нехай вираз $A \equiv C$ є визначенням поняття A, де C є потенційно складеним поняттям, а T є скінченною множиною означень понять. Якщо $A \equiv C \in T$ та ім'я поняття B з'являється в C, тоді A *безпосередньо використовує* B. Поняття A *використовує* B, якщо A безпосередньо використовує B, або існує ім'я поняття B', таке що A використовує B', а B' безпосередньо використовує B. Скінченна множина визначень понять T називається *ациклічним TBox*, якщо: 1) жодне ім'я поняття в T не використовує само себе; 2) жодне ім'я поняття не з'являється більше ніж один раз в лівій частині визначення поняття в T. Якщо T є ациклічним *TBox* з $A \equiv C \in T$, то A називається *точно визначеним* в T, а C називається *визначенням* A в T.

Важливість ациклічних *TBox* полягає в тому, що їх можна розгортати в *ABox*, трактуючи визначення понять як макроси. Механізм процесу полягає у рекурсивній заміні в *ABox* усіх імен понять на їхні визначення з *TBox*. Нехай $K = (T, A)$ є базою знань з ациклічним *TBox*, який має форму $T = \{A_i \equiv C_i \mid 1 \leq i \leq m\}$. Нехай $A_0 = A$ та A_{j+1} є результатом застосування наступних замін: 1) знайти деякий $a: D \in A_j$, в якому D безпосередньо використовує A_i, $1 \leq i \leq m$; 2) замінити в D усі A_i на C_i. Якщо в A_k не можна здійснити більше замін, то A_k є *результатом розгортання* T *в* A. Розгортання є важливим етапом процесу розв'язання деяких задач міркування, проте воно може спричинити експоненціальний вибух розміру *Abox*. Тому важливо пам'ятати про стратегії *лінивого* та *жадібного* розкриття визначень понять.

3.2. Міркування в описових логіках

3.2.1. Проблеми та сервіси міркування в ALC базах знань

Структура бази знань в описових логіках дозволяє моделювати поняття, ролі, індивіди та відношення між ними. Кількість тверджень в базі знань, вкладеність означень понять, пошук моделей породжують необхідність в надійних і ефективних методах логічного міркування, які б дозволили отримати відповіді про властивості елементів та структури бази знань. Нехай $K = (T, A)$ є *ALC* базою знань, C та D є ймовірно складеними *ALC* поняттями, а b є іменем індивіда. Тоді, базові проблеми міркування, що властиві для *ALC* баз знань, мають наступне формулювання [2]:

- C *задовольняється* відносно T, якщо існує модель I для T та деякий елемент $d \in \Delta^I$, такий що $d \in C^I$;
- C *включене в* D відносно T, і записується як $T \vDash C \sqsubseteq D$, якщо $\mathrm{C}^I \subseteq D^I$ в усіх моделях I для T;
- C *еквівалентне* D відносно T, і записується як $T \vDash C \equiv D$, якщо $\mathrm{C}^I = D^I$ в усіх моделях I для T;
- K є *узгодженою* (сумісною), якщо існує модель для K;
- b є *екземпляром* C відносно K, і записується як $K \vDash b\colon C$, якщо $b^I \in \mathrm{C}^I$ в усіх моделях I для K.

Знак семантичного логічного наслідку $\vDash$, на відміну від знаку дедуктивного логічного наслідку $\vdash$, ґрунтується на включенні моделей виразів, яких він з'єднує. При фіксації T, включення і еквівалентність відносно T є бінарними відношеннями ймовірно складених понять, тому для підкреслення цього факту використовуються позначення $C \sqsubseteq_T D$ та $C \equiv_T D$ відповідно. Проблеми задоволення, включення та еквівалентності є означеними відносно лише *TBox*, проте проблеми узгодженості бази знань та екземпляру поняття є означеними відносно *TBox* та *ABox*. Властивість узгодженості лише *TBox* чи лише *ABox* розглядається як випадок узгодженості бази знань відповідної форми: $K = (T, \emptyset)$ для *TBox* і $K = (\emptyset, A)$ для *ABox*.

Відношення включення поняття $\sqsubseteq$ є транзитивним, а описова логіка ALC є *монотонною*, тобто чим більше тверджень містить база знань, тим більше логічних наслідків вона допускає. Еквівалентності, які є виводяться з усіх $TBox$ називаються *тавтологіями*, наприклад $T \vDash C \sqsubseteq D$ тоді і лише тоді $T \vDash \top \sqsubseteq (\neg C \sqcup D)$. Важливо, що між базовими проблемами міркування теж існують приховані взаємозв'язки, які є вельми корисним для їх розв'язання. Нехай $K = (T, A)$ є ALC базою знань, C та D є ймовірно складеними ALC поняттями, а b є іменем індивіда. Тоді виконуються наступні твердження:

- $C \equiv_T D$ тоді й лише тоді, коли $C \sqsubseteq_T D$ та $D \sqsubseteq_T C$;
- $C \sqsubseteq_T D$ тоді й лише тоді, коли $C \sqcap \neg D$ не задовольняється відносно T;
- C задовольняється відносно T тоді й лише тоді, коли $C \not\sqsubseteq \bot$;
- C задовольняється відносно T тоді й лише тоді, коли $(T, \{b: C\})$ є сумісною;
- $(T, A) \vDash b: C$ тоді й лише тоді, коли $(T, A \cup \{b: \neg C\})$ є неузгодженою;
- Якщо T ациклічний і A' є результатом розгортання T в A, то K є узгодженою тоді й лише тоді, коли $(\emptyset, A')$ є узгодженою.

Результатом описаних взаємозв'язків між проблемами міркування є той факт, що усі базові проблеми можна звести до спільної – узгодженості бази знань. Таким чином, наявність ефективного алгоритму для обчислення властивості узгодженості бази знань дозволяє вирішити інші базові проблеми міркування. Варто зауважити, що існують також проблеми міркування, що не зводяться до проблеми узгодженості бази знань або при зведенні спричиняють експоненціальний ріст розміру проблеми, наприклад відповідь на запит з використанням кон'юнкції.

Проектування бази знань зазвичай передбачає поступове наповнення $TBox$ та $ABox$ твердженнями про предметну область. Оскільки формальний синтаксис та вкладеність понять породжують певну неочевидність властивостей бази знань для інженера, то виникає потреба в допоміжних інструментах, які обчислюватимуть наслідки внесення, видалення та зміни тверджень. В базах знань цю роль виконують *сервіси міркування*, які ін'єктивно відповідають базовим проблемам міркування:

- Маючи *TBox* T та поняття C, визначити чи C *задовольняється* відносно T;
- Маючи *TBox* T та поняття C і D, визначити чи C *включене в* D відносно T;
- Маючи *TBox* T та поняття C і D, визначити чи C *еквівалентне* D відносно T;
- Маючи базу знань (T, A), визначити чи (T, A) є узгодженою;
- Маючи базу знань (T, A), поняття C та ім'я індивіда b, визначити чи b є *екземпляром* C відносно K.

Наведені формулювання є специфікацією сервісів міркування, для якої може існувати набір різноманітних методів її реалізації. Окрім цього, на основі базових сервісів міркування можна означити більш інтелектуальні сервіси, що розкривають потенціал автоматизованого перебору множини однорідних об'єктів і розв'язання для кожного з них деякої проблеми міркування. До таких «узагальнюючих» сервісів міркування можна віднести наступні [2]:

- *Класифікація TBox*: маючи *TBox* T, визначити ієрархію включень усіх імен понять, що присутні в T, відносно T. Тобто, для кожної пари імен понять A і B, що присутні в T, визначити чи $A \sqsubseteq_T B$ та чи $B \sqsubseteq_T A$. Результатом виконання сервісу є ієрархія включення, яку зручно подавати у вигляді графа чи діаграми Гессе для часткового порядку.
- Встановлення *задовільності понять* в T: маючи *TBox* T, для кожного імені поняття A, встановити чи A задовольняється відносно T. Незадовільність поняття є ознакою помилки моделювання.
- *Отримання екземплярів*: маючи поняття C та базу знань K, визначити усі імена індивідів b, таких що b є екземпляром C відносно K. Результатом виконання сервісу є множина екземплярів поняття C.
- *Втілення* імені індивіда: маючи ім'я індивіда b та базу знань K, визначити усі імена понять A, що присутні в T, для яких b є екземпляром A відносно K. Результатом виконання сервісу є множина імен понять, яка є втіленням імені індивіда.

3.2.2. Міркування з використанням табло-алгоритму

Міркування над твердженнями бази знань є процесом, що сприяє досягненню кількох цілей: 1) обґрунтуванню чи спростуванню існуючих знань; 2) породженню нових знань; 3) вирішенню задачі прийняття рішення; 4) проєктуванню узгоджених онтологій [66]. У логіко-математичному контексті міркування розглядається як «механізований» процес числення над твердженнями формальної системи, шляхом застосування дедуктивних правил виведення або конструктивних властивостей семантики. Щоб вважатися процедурою прийняття рішення у формальній системі, методи міркування мають володіти трьома якісними властивостями: розв'язністю, повнотою і обґрунтованістю. Розв'язність полягає в існуванні формальних гарантій зупинки роботи алгоритму для усіх можливих варіантів вхідних даних, та ймовірно існуванні «ефективного» алгоритму обчислення розв'язку, який характеризується прийнятною часовою складністю. Повнота методу міркування гарантує повернення коректного результату для усіх вхідних даних, що володіють шуканою властивістю, а обґрунтованість методу гарантує, що коректний результат повертається виключно для вхідних даних, що володіють шуканою властивістю.

В описовій логіці *ALC* існує розмаїття методів міркування, які ґрунтуються на різних підходах числення. До найпоширеніших методів міркування можна віднести метод резолюцій, числення послідовностей, числення на основі автоматів, табло-методу, переписування запитів [67]. У виразних описових логіках широкого поширення у застосуванні набув табло-метод, зокрема, його оптимізація гіпер-табло є основою модуля міркування HermiT в системі проєктування онтологій Protégé [68]. Оскільки базові проблеми міркування в описовій логіці *ALC* зводяться до проблеми перевірки узгодженості бази знань, то властивості табло-алгоритму як процедури прийняття рішення будуть формулюватися відносно вхідних даних проблеми узгодженості бази знань. Доведення властивостей табло-алгоритму ґрунтуються здебільшого на семантиці описової логіки *ALC*, а саме структурних особливостях тих моделей, що можуть бути виражені в понятійній мові *ALC*.

Згідно з означенням, база знань K є узгодженою, якщо для неї існує модель I. Таку модель називають *свідком* узгодженості K та позначають $I \vDash K$. Ідея табло-алгоритмів полягає в спробі доведення узгодженості бази знань $K = (T, A)$ шляхом побудови моделі I для K. Якщо під час побудови свідка узгодженості K виявлено явні суперечності, що спростовують можливість існування моделі I для K, тоді БЗ K вважається неузгодженою. Процес побудови моделі для K починається з A та при потребі розширює її обмеженнями, що випливають з семантики понять, а також аксіомами, що містяться в A і T. Ефективність і коректність табло-алгоритму забезпечується структурою моделі, яку вони будують, а саме деревовидну модель. Ця структура моделі значно обмежує кількість потенційних свідків (ефективність), дозволяє відстежувати конструктивні зациклення в гілках дерева (розв'язність), а також уможливлює доведення існування ймовірно нескінченних моделей завдяки методу структурної індукції (обґрунтованість). Також, відсутність можливості існування деревовидної моделі свідчить про неможливість існування будь-якої іншої моделі для бази знань K (повнота).

Принципи роботи алгоритму зручно аналізувати в контексті трьох ситуацій: при $K = (\emptyset, A)$, при $K = (T, A)$, де T ациклічний, та загальний випадок $K = (T, A)$. Оскільки усі поняття, що містяться в T або A, можна привести до заперечуваної нормальної форми, шляхом застосування законів де Моргана та дуальності між кванторами існування та загальності, то вираз $\dot{\neg}\, C$ відповідатиме запереченій нормальній формі $\neg C$. *ABox* A вважається *нормалізованим*, якщо виконуються наступні умови:

1) всі поняття, що містяться в T чи A, є у запереченій нормальній формі, тобто оператор заперечення застосовується тільки до імен понять і не стоїть перед дужками, кванторами чи операторами;
2) кожне ім'я індивіда, що міститься в A, присутнє хоча б в одному твердженні вигляду $a: C$;
3) A є непустим, тобто містить твердження.

Для ситуації $K = (\emptyset, A)$, тобто коли *TBox* є пустим, *правила розкриття* повинні лише застосувати семантику до понять, що містяться у твердженнях в A. Оскільки принцип роботи алгоритму полягає в розкладанні складних понять на підпоняття, то завжди існує ітерація, коли усі складні поняття є повністю розкладені на атомарні, і відповідно алгоритм завершується. Кожне задовільне поняття в *ALC* має деревовидну модель, проте екземпляри понять можуть бути з'єднані ролями, що породжує лісоподібну модель для *ALC* бази знань K. Якщо K є узгодженою, тоді вона має модель, що складається з непустої множини роз'єднаних дерев, в яких корінь дерева інтерпретує ім'я деякого індивіда в A та може бути з'єднаний ребрами з коренями інших дерев. Табло-алгоритм будує лісоподібну модель для *ABox*, шляхом застосування синтаксичних правил розкриття, які потребують задоволення двох умов задля розширення A новими твердженнями.

Таблиця 3.2

Синтаксичні правила розкриття для узгодженості ALC ABox

Назва	***Умова 1***	***Умова 2***	***Дія***
⊓-правило	$a: C \sqcap D$	$\{a: C, a: D\} \nsubseteq A$	$A \cup \{a: C, a: D\}$
⊔-правило	$a: C \sqcup D$	$\{a: C, a: D\} \cap A = \emptyset$	$A \cup \{a: X\}$, де $X \in \{C, D\}$
∃-правило	$a: \exists r. C \in A$	$\nexists b, \{(a, b): r, b: C\} \subseteq A$	$A \cup \{(a, d): r, d: C\}$, де d є новим в A
∀-правило	$\{a: \forall r. C, (a, b): r\} \subseteq A$	$b: C \notin A$	$A \cup \{b: C\}$

Алгоритм застосовує правила розкриття допоки A не стає *повним*, оскільки в повному *ABox* легко виявити логічні суперечності – *конфлікти*. *ABox* A містить конфлікт, якщо для деякого ім'я індивіда a і для деякого поняття C виконується $\{a: C, a: \neg C\} \subseteq A$; якщо A не містить конфліктів, то A називається *безконфліктним*. A називається повним, якщо він містить конфлікт, або не задовольняються умови усіх правил розкриття.

Оскільки будь-який *ALC ABox* можна привести до нормалізованої форми, то не зменшуючи загальності, алгоритм для визначення узгодженості *ABox* прийматиме як вхідний аргумент нормалізований *ABox*. Табло-алгоритм для узгодженості *ABox* можна подати наступним псевдо-кодом:

Алгоритм Узгодженість(A)*:*

Якщо Розкрити$(A) \neq \emptyset$

То повернути «узгоджений»

Інакше повернути «неузгоджений»

Процедура Розкрити(A)*:*

Якщо *A* не є повним

То вибрати правило розкриття R, умови якого задовольняються в A, та вибрати пару тверджень α з A, до яких застосовне правило R

Якщо існує $A' \in \exp(A, R, \alpha)$, така що Розкрити$(A') \neq \emptyset$

То повернути Розкрити(A')

Інакше повернути $\emptyset$

Інакше

Якщо *A* містить конфлікт

То повернути $\emptyset$

Інакше повернути *A*

Тут допоміжна функція $\exp(A, R, \alpha)$ повертає множину *ABox*, які утворюються шляхом застосування правила R до твердження α з A. Усі правила розкриття, окрім ⊔-правила, є детермінованими та повертають множину з єдиним *ABox*. Оскільки диз'юнктивне правило, в залежності від способу застосування, може породжувати два різні *ABox*, то необхідно рекурсивно розкривати кожен з них. Спосіб вибору послідовності застосування правили розкриття R та тверджень α не впливає на результат алгоритму, але може суттєво впливати на його швидкодію.

Табло-алгоритм для встановлення узгодженості бази знань $K = (T, A)$, де T є ациклічним *TBox*, значною мірою ґрунтується на правилах розкриття та процедурі розширення, що використовувалися для $K = (\emptyset, A)$. Ця особливість спричинена тим, що ациклічний *TBox* можна повністю розгорнути в *ABox* A', що власне зводить поточну задачу узгодженості до попередньої. Жадібна стратегія розгортання T в A може призвести до експоненціального зростання розміру бази знань, тому доцільно скористатися стратегією «лінивого» розгортання, яка розкриває означення понять в процесі роботи алгоритму, виключно при безпосередній потребі. Нижче подано правила лінивого розгортання означень понять, які разом з попередніми правилами розгортання, є основою табло-алгоритму для встановлення узгодженості бази знань з ациклічним *TBox*.

Таблиця 3.3

Синтаксичні правила розкриття аксіом ациклічного ALC TBox

Назва	***Умова 1***	***Умова 2***	***Дія***
$\sqsubseteq$-правило	$A \sqsubseteq C \in T, \quad a\colon A \in A$	$a\colon C \notin A$	$A \cup \{a\colon C\}$
$\equiv_1$-правило	$A \equiv C \in T, \quad a\colon A \in A$	$a\colon C \notin A$	$A \cup \{a\colon C\}$
$\equiv_2$-правило	$A \equiv C \in T, \quad a\colon \neg A \in A$	$a\colon \dot{\neg} C \notin A$	$A \cup \{a\colon \dot{\neg} C\}$

Встановлення узгодженості для загального випадку бази знань $K = (T, A)$, де T може містити циклічні означення понять, потребує додаткових конструкцій для забезпечення зупинки алгоритму. Не зменшуючи загальності, вхідним аргументом табло-алгоритму є *нормалізована* база знань $K = (T, A)$, яка згідно з означенням складається з нормалізованого *TBox* T та нормалізованого *ABox* A. *TBox* називається *нормалізованим*, якщо всі його аксіоми мають форму $\top \sqsubseteq E$, де E приведений до заперечуваної нормальної форми. Приведення аксіом до очікуваної форми можна здійснити виходячи з наступних властивостей:

- I задовольняє $C \sqsubseteq D \quad \Leftrightarrow \quad I$ задовольняє $\top \sqsubseteq D \sqcup \neg C$;
- I задовольняє $C \equiv D \quad \Leftrightarrow \quad I$ задовольняє $\top \sqsubseteq (D \sqcup \neg C) \sqcap (C \sqcup \neg D)$.

Набір правил розкриття для узгодженості бази знань містить п’ять елементів, три з яких повністю ідентичні правилам для узгодженості *ABox*, зокрема ⊓-правило, ⊔-правило та ∀-правило. Інші два правила теж значною мірою подібні до своїх аналогів, проте мають деякі зміни: ⊑-правило використовує нормалізовану форму аксіом *TBox*, а ∃-правило містить додаткову вимогу на «незаблокованість» індивіда, що є властивістю для гарантованого завершення роботи алгоритму.

Таблиця 3.4

Змінені правила розкриття аксіом ALC бази знань

Назва	***Умова 1***	***Умова 2***	***Дія***
⊑-правило	$\top \sqsubseteq D \sqcup \neg C \in T$, $a: C \in A$	$a: D \notin A$	$A \cup \{a: D\}$
∃-правило	$a: \exists r. C \in A$, a не заблокований	$\nexists b, \{(a, b): r, b: C\} \subseteq A$	$A \cup \{(a, d): r, d: C\}$, де d є новим в A

Проблема зупинки табло-алгоритму для загальної бази знань виникає власне через можливість комбінації ⊑-правила та ∃-правила, які без механізму блокування можуть породжувати нескінченні повторювані послідовності індивідів. Якщо *ABox* не містить конфліктів, то зазначені послідовності можна описати в моделі шляхом створення циклів. Під час роботи алгоритму, ∃-правило єдине додає нові індивіди-вершини та ролі-ребра до деревовидної моделі *ABox*. Якщо ∃-правило додає новий індивід b та екземпляр ролі $(a, b): r$, тоді b є *наступником* a та a є *попередником* b. Транзитивне замикання відношень наслідника і попередника іменується поняттями *нащадок* та *предок* відповідно. Нехай $con_A(a) = \{ C \mid a: C \in A \}$ є множиною імен понять, що присутні в твердженнях понять, які мають форму $a: C$. Ім’я індивіда b в *ALC ABox A є заблокованим іменем індивіда* a, якщо виконуються наступні умови: 1) a є предком b; 2) $con_A(a) \supseteq con_A(b)$. Тобто, якщо ім’я індивіда заблоковане, то всі його нащадки є також заблокованими. Оскільки в кореневих індивідів відсутні предки, то вони ніколи не блокуються [2].

Приклад 2. Перевірити узгодженість *ALC* загальної бази знань $K = (T, A)$ для предметної області «Мас-медіа» за допомогою табло-алгоритму. Структуру бази знань подано нижче:

T = {
Новина $\sqsubseteq$ ∀має. Автор;
Автор $\equiv$ Людина $\sqcup$ Організація $\sqcup$ Робот;
Робот $\sqsubseteq \neg$Людина $\sqcap$ ∀має. Автор;
Організація $\sqsubseteq$ ∃має. Людина;
};
A = {
a: Новина;
(a, b): має;
b: Робот;
(b, c): має;
c: Організація;
}.

Розв'язання. Першим кроком розв'язання поставленої задачі є нормалізація бази знань, яка є вхідним аргументом для табло-алгоритму. Зауважимо, що *ABox* вже є нормалізованим, проте твердження з *TBox* потребують синтаксичних змін, згідно з правилами приведення та можливо законами де Моргана:

T = {
$\top \sqsubseteq$ ∀має. Автор $\sqcup \neg$Новина;
$\top \sqsubseteq$ (Людина $\sqcup$ Організація $\sqcup$ Робот) $\sqcup \neg$Автор;
$\top \sqsubseteq (\neg$Людина $\sqcap$ ∀має. Автор) $\sqcup \neg$Робот;
$\top \sqsubseteq$ ∃має. Людина $\sqcup \neg$Організація;
};

Отримавши нормалізовану базу знань, табло-алгоритм розпочинає роботу. На кожній ітерації алгоритму вибираються правило розкриття та твердження (чи набір тверджень), які задовольняють умови вибраного правила. Виберемо $\sqsubseteq$-правило та застосуємо його до аксіоми $\top \sqsubseteq$ ∀має. Автор $\sqcup \neg$Новина. Зазначене твердження належить T, a: Новина $\in A$ та a: ∀має. Автор $\notin A$, тому до *ABox* A додається нове твердження поняття $A = A \cup \{a$: ∀має. Автор$\}$.

Спробуємо застосувати ⊑-правило до решти тверджень, що містяться в *TBox*. Твердження $\top \sqsubseteq (\text{Людина} \sqcup \text{Організація} \sqcup \text{Робот}) \sqcup \neg\text{Автор}$ не задовольняє усі умови ⊑-правила, зокрема $d\colon \text{Автор} \notin A$; інші твердження задовольняють усі умови, тому вони можуть розкриватися. Після застосування ⊑-правила до тверджень з T, *ABox* A матиме наступний вигляд:

A = {

$a\colon$ Новина;	$a\colon \forall$має. Автор;
$(a, b)\colon$ має;	$b\colon \neg$Людина $\sqcap \forall$має. Автор;
$b\colon$ Робот;	c: $\exists$має. Людина;
$(b, c)\colon$ має;	
$c\colon$ Організація;	

}.

Далі застосуємо ∃-правило до твердження c: ∃має. Людина. Оскільки індивід c початково присутній в A, то в моделі йому відповідає коренева вершина дерева, яка ніколи не блокується. Оскільки $\nexists d, \{(c, d)\colon \text{має}, d\colon \text{Людина}\} \subseteq A$, то твердження можна розкривати ∃-правилом. Таким чином в A додаються наступні твердження: $(c, d)\colon$ має та $d\colon$ Людина, причому d є новим ім'ям індивіда в A.

A = {

$a\colon$ Новина;	$a\colon \forall$має. Автор;
$(a, b)\colon$ має;	$b\colon \neg$Людина $\sqcap \forall$має. Автор;
$b\colon$ Робот;	c: $\exists$має. Людина;
$(b, c)\colon$ має;	$(c, d)\colon$ має;
$c\colon$ Організація;	$d\colon$ Людина;

}.

Для наступного кроку виберемо ⊓-правило та застосуємо його до твердження $b\colon \neg\text{Людина} \sqcap \forall\text{має. Автор}$. Умови правила задовольняються, $b\colon \neg\text{Людина} \notin A$ та $b\colon \forall\text{має. Автор} \notin A$, отже $A = A \cup \{b\colon \neg\text{Людина}, b\colon \forall\text{має. Автор}\}$. Далі застосуємо ∀-правило до тверджень $a\colon \forall\text{має. Автор}$ та $b\colon \forall\text{має. Автор}$. Оскільки кожне з них задовольняє умови, $(a, b)\colon \text{має} \in A$ і $b\colon \text{Автор} \notin A$ та $(b, c)\colon \text{має} \in A$ і $c\colon \text{Автор} \notin A$, то $A = A \cup \{b\colon \text{Автор}; c\colon \text{Автор}\}$.

Після застосування ⊓-правила та ∀-правила *ABox* A розширився до наступного вигляду:

A = {

a: Новина;	a: ∀має. Автор;	b: ¬Людина;
(a, b): має;	b: ¬Людина ⊓ ∀має. Автор;	b: ∀має. Автор;
b: Робот;	c: ∃має. Людина;	b: Автор;
(b, c): має;	(c, d): має;	c: Автор;
c: Організація;	d: Людина;	

}.

Тепер до твердження з T $\top \sqsubseteq$ (Людина ⊔ Організація ⊔ Робот) ⊔ ¬Автор можна двічі застосувати ⊑-правило, при b: Автор та при c: Автор. Таким чином отримаємо $A = A \cup \{b$: Людина ⊔ Організація ⊔ Робот; c: Людина ⊔ Організація ⊔ Робот$\}$. Умови ⊔-правила не задовольняються, оскільки b: Робот $\in A$ і c: Організація $\in A$. Більше правил до тверджень з A та T застосувати не можна, тому A є повним. Окрім цього в A відсутні явні конфлікти, які мають вигляд $\{x: D; x: \neg D\} \subseteq A$, тому A є безконфліктним.

Відповідь. *ALC* база знань $K = (T, A)$ для предметної області «Мас-медіа» є узгодженою. Модель I для K, що знайдена завдяки застосуванню табло-алгоритму, містить усі індивіди та атомарні поняття, до яких вони належать.

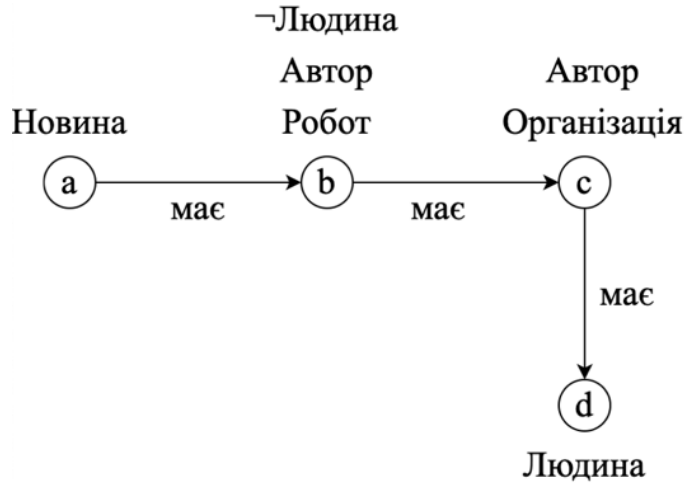

Рис. 3.2. Графічне подання знайденої моделі

3.3. Порівняльний аналіз описових логік та теорій типів

Логічні моделі подання знання, яким властива формальна семантика, сприяють однозначній інтерпретації синтаксичних виразів, а також міркуванню над знаннями методами, що використовують обґрунтовані форми логічного наслідку і володіють формально доведеними властивостями процедури прийняття рішення. Присутність зазначених рис дозволяють трактувати формальний логічний підхід до подання та міркування над знаннями, як надійну технологію, для якої властива прогностична, пояснювальна, обчислювальна та комунікаційна функції. Маючи спільний логічний підхід до управління знаннями, описові логіки і теорії типів суттєво різняться по множині властивостей, які є важливими з точки зору обчислюваності, виразності, істинності знання. Обґрунтований вибір формалізму для подання та міркування над знаннями в деякій предметній області, потребує визначення суттєвих ознак (потреб) та порівняльний аналіз доступних формалізмів за цими ознаками.

До найпоширеніших порівняльних ознак логічних формалізмів можна віднести наступні властивості:

- *Виразність* – характеризується можливістю подати знання різної природи та структури, синтаксичними виразами формальної мови;
- *Семантика* – формальні теорії, в яких означається предикат істинності та спосіб інтерпретації синтаксичних виразів формалізму;
- *Задачі міркування* – прикладні задачі про структуру бази знань, які можуть бути розв’язані методами міркування;
- *Методи міркування* – множина методів для розв’язання задач міркування, які ґрунтуються на властивостях семантики та дедуктивних систем;
- *Розв’язність* – характеризується наявністю ефективного алгоритму для обчислення коректності довільної теорії, що описана мовою формалізму;
- *Парадокси* – відкриті проблеми формалізму, які ставлять під сумнів його застосовність.

Розглянемо детальніше характеристику кожної з порівняльних властивостей в формалізмах описових логік і теорій типів. Виразність мови подання безпосередньо впливає на рівень абстрактності подання знання з одного боку та обчислювальні властивості алгоритмів міркування з іншого боку. Висока виразність мови подання дозволяє описувати знання деталізовано, з багатьма рівнями абстракції, складними зв'язками та різною природою походження, проте алгоритми міркування для такої складної структури є нерозв'язними, оскільки синтез деяких конструкторів знання породжує можливість нескінченного застосування правил міркування. Числення індуктивних конструкцій є формалізмом з високою виразністю, який дозволяє будувати нескінченну ієрархію типів, застосовувати квантори до залежних типів, подавати нескінченні структури індуктивно, проте міркування в ньому над деякими виразами є нерозв'язним, і тому потребує участі людини для керування напрямами пошуку доведення через формування гіпотез. Сучасна описова логіка $SROIQ^{(D)}$ є компромісом між виразністю формалізму та розв'язністю алгоритмів міркування, що автоматизує процес міркування над базою знань, проте унеможливлює подання деяких математичних теорії, виразів природньої мови, кванторування властивостей. Наприклад, наслідком з теореми компактності і теореми Ловенгейма-Сколема про підвищення потужності є неможливість формалізації властивостей зліченності та скінченності предметної області в логіці першого порядку, яка є надмножиною описових логік.

Наступною порівняльною ознакою логічних формалізмів є їхня семантика. Важливість семантики полягає в тому, що вона є метатеорією формалізму, тобто надає підґрунтя для інтерпретації синтаксичних виразів формалізму та присвоює значення істинності для логічних тверджень. Також, існує практика розширення формалізму структурами та закономірностями з його семантики, яка таким чином є джерелом обґрунтованого розвитку формалізму. Описовим логікам, як підмножині логіки першого порядку, властива стандартна семантика – теорія моделей, що була започаткована А. Тарським на основі його семантичної теорії істинності і теорії

множин. Згідно з теорією моделей значення синтаксичного виразу в описовій логіці задається інтерпретацією, яка, зокрема, до імен понять зіставляє множину елементів цього поняття. Недоліком зазначеної семантики є висока ймовірність попадання в інтерпретацію поняття елементів, спільних за формою проте відмінних за структурою або поведінкою, що в результаті породжує аргументи правильної форми, але безглузді за змістом. Для теорій типів властиве існування множини семантик, серед яких найпоширенішими є моделі Кріпке [69], теорія категорій, гомотопічна теорія типів [15]. Аналіз складних логічних виразів здійснюється згідно з інтерпретацією Брауера-Гейтінґа-Колмогорова [70], а істинність атомарних виразів ґрунтується на наявності доведення можливості створення синтаксичного виразу правилами виведення формальної системи. Описаний підхід породжує синтаксичні вирази з передбачуваною поведінкою і дозволяє будувати більш універсальні та надійні аргументи.

Для описових логік базовими задачами міркування є: 1) задовільність поняття; 2) включення поняття; 3) еквівалентність понять; 4) узгодженість бази знань; 5) задовільність індивіда. Зазначені задачі міркування можна звести до встановлення узгодженості бази знань. В теоріях типів основними задачами міркування є: 1) допустимість типізації; 2) присвоєння типу; 3) перевірка типу; 4) пошук терму; 5) доведення типу. Хоча семантики та структура двох формалізмів суттєво різняться, між їхніми задачами міркування можна провести деякі паралелі. Наприклад, задача задовільності поняття концептуально подібна на задачу пошуку терму для типу, оскільки розв’язки обох задач покликані дати відповідь на можливість існування екземплярів зазначених форм абстракції. Також, задача перевірки типу корелює із задачею перевірки узгодженості бази знань, за умови здійснення перевірки типів для усіх суджень в базі знань. Зауважимо, що задачі міркування в описових логіках здебільшого орієнтовані на пошук моделі бази знань, в той час як вирішення задач в теоріях типів забезпечує формальне підтвердження очікуваної поведінки термів, а також породжує екземпляри типів.

Міркування в описових логіках здійснюється методами, що використовують як семантичний, так і дедуктивний логічний наслідок. Згідно з модельно-теоретичним підходом, логічний наслідок приймає обґрунтованість арґументу як відсутність контрприкладу. Таким чином, арґумент вважається істинним, якщо в усіх моделях, де його засновки набувають значення істинності, висновок є також істинним. Зрозуміло, що в логіці першого порядку існують арґументи, для яких неможливо перевірити всі його моделі, оскільки його предметна область може бути безмежною. В такому випадку звертаються до дедуктивних методів міркування, що містять аксіоми та правила виведення, де кожне правило є ретельно перевіреним в контексті логіки як надійної форми міркування, а також формально доведено допустимість поєднання цих правил та аксіом в межах єдиної системи логічного числення. Для описових логік можливим є також застосування дедуктивних методів, зокрема методу резолюцій, проте завдяки їхній обмеженій виразності модельно-теоретичні методи дозволяють підтвердити чи спростувати будь-яке твердження, незалежно від його істинності, та роблять це ефективно.

В теоріях типів, де семантики є складними математичними абстракціями, міркування здійснюється зазвичай дедуктивними методами в контексті формальних дедуктивних систем. Виразність формальної мови зазначених систем впливає на властивість розв’язності деяких задач міркування, зокрема в численні індуктивних конструкцій задача пошуку терму є нерозв’язною, що узгоджується з теоремою про відсутність у виразних системах загального алгоритму доведення чи спростування довільного формального твердження. Згідно з ізоморфізмом Каррі-Говарда, в теорії типів перетворення синтаксичних виразів відповідають за структурою правилам логічного виведення системи доведення «природня дедукція» для інтуїціоністської логіки. Варто зазначити, що інтуїціоністська логіка є краще обґрунтованою для роботи з нескінченними об’єктами, оскільки вона була започаткована Л. Брауером після його спостереження за абстрагуванням «закону виключення третього» від скінченних випадків та його необґрунтованого поширення на нескінченні випадки.

Розв’язність задач міркування в описових логіках забезпечується обмеженням виразності понятійної мови, а також постійною оптимізацією методів міркування. Наприклад, для міркування над онтологіями OWL 2 часто використовується модуль HermiT [71], який базується на оптимізованій версії табло-алгоритму – гіпертабло. Крім цього, для широкого застосування методів міркування формально доводиться їхня повнота, коректність та розв’язність для кожної задачі міркування і різновидів вихідних параметрів. Поєднання цих трьох властивостей дозволяє класифікувати алгоритм, який ними володіє, як процедуру прийняття рішення в теорії числення. В сучасних теоріях типів основні задачі міркування є розв’язними, за винятком задачі пошуку терма типу, що рівнозначна задачі доведення істинності довільної теореми в формальній системі. Числення індуктивних конструкцій володіє доведеними властивостями збіжності та строгої нормалізації, а також є розв’язним для задач перевірки типу і допустимості типізації терму.

Останнім критерієм в порівняльному аналізі описових логік та теорій типів є наявність парадоксів – аргументів, що породжують абсурдні висновки з коректних засновків, шляхом коректного застосування правил формального виведення. До найвідоміших парадоксів описової логіки можна віднести поведінку матеріальної імплікації, яка згідно з семантикою цього оператора набуває значення хиби лише тоді, коли засновок є істинним, а висновок хибним. Наприклад, інтерпретація I виразу $C \sqsubseteq D$ є його моделлю у випадку $\mathrm{C}^I = \emptyset$, що породжує логічно коректні, проте інтуїтивно суперечливі аргументи. Застосування закону виключення третього чи його наслідку, закону подвійного заперечення, є теж дискусійним у формальній логіці, а його ігнорування породило різновиди некласичної інтуїціоністської логіки. В теоріях типів найвідомішим є парадокс Жирара, який є аналогом парадоксу Рассела, та полягає у формуванні виразу «тип всіх типів» $Type{:}Type$. З метою його уникнення, в численні індуктивних конструкцій передбачено кумулятивну ієрархію типів, де кожен тип завжди розташовується в ієрархії типів вище ніж терм, якого він типізує.

Порівняльний аналіз описових логік і теорій типів дозволяє зробити висновок, що кожен з формалізмів має свої переваги та недоліки, які необхідно враховувати в контексті цільової предметної області. Розв'язність описових логік для основних задач міркування забезпечує можливість його автоматизації, проте обмеженість виразності понятійної мови унеможливлює подання деяких речень природньої мови та певних математичних теорій, що потребують кванторування предикатів. Теорії типів характеризуються високою виразністю та використанням інтуїціоністської логіки для доведення істинності виразів, проте нерозв'язність задачі пошуку терма для типу перешкоджає можливості повної автоматизації процесу міркування. Тим не менш, напівавтоматичний пошук терму з формальною перевіркою коректності процесу, породжує під час доведення надійну реалізацію програмних специфікації, які згідно з ізоморфізмом Каррі-Говарда відповідають типам, а їхні програмні реалізації – термам.

3.4. Висновки до розділу 3

Системний аналіз базової описової логіки *ALC* як підґрунтя виразної описової логіки $SROIQ^{(D)}$ визначив основні засади подання знання, модельно-теоретичну семантику описових логік, підхід до структурування знання в бази знань, задачі міркування та принципи функціонування табло-алгоритму. Отримана інформація дозволила здійснити порівняльний аналіз описових логік та теорій типів, з метою формування методології їхнього застосування для побудови та розвитку баз знань цільових предметних областей. Висновками цього розділу є наступні положення:

- фундаментальною перевагою описових логік над виразними теоріями типів є розв'язність задач міркування, зокрема задачі пошуку моделі довільної бази знань, яка описана понятійною мовою цієї логіки;
- основним недоліком описових логік є обмеженість їхньої виразності, яка забезпечує розв'язність задач міркування, проте унеможливлює подання певних математичних теорій та деяких виразів природньої мови;

- в описових логіках доцільно застосовувати модельно-теоретичний підхід до міркування, який зокрема втілюється у табло-алгоритмі, оскільки він забезпечує коректний розв’язок для основних задач міркування і пропонує високо оптимізовані реалізації алгоритму;
- теорії типів, зокрема числення індуктивних конструкцій, характеризуються високою виразністю, що забезпечує можливість подання довільних виразів природньої мови, складних наукових теорій та програмних специфікацій, проте задача пошуку терму для типу є нерозв’язною;
- міркування над виразами теорії типів здійснюється методами некласичної інтуїціоністської логіки, яка не використовує логічні закони «виключення третього» та «подвійного заперечення» для доведень, через відсутність належного обґрунтування їхнього поширення на нескінченні структури;
- формалізація програмної специфікацій у вигляді типу та розв’язання задачі пошуку терму для такого типу, автоматично породжує коректне втілення програмної специфікації у вигляді терму.

РОЗДІЛ 4. ПРОГРАМНА СИСТЕМА УПРАВЛІННЯ ПЕРСОНАЛЬНОЮ ТА КОЛЕКТИВНОЮ БАЗОЮ ЗНАНЬ

4.1. Співпраця у віртуальних наукових колективах

Сучасні наукові дослідження виходять за межі класичних наукових установ і часто вимагають залучення експертів з різних інституцій, віддалених географічно. Віртуальний науковий колектив – це ситуативна група представників різних наукових установ, які, завдяки системному використанню інформаційних та комунікаційних технологій, об’єднуються для спільних досліджень та виконання інтеґрованих наукових проєктів. Ефективна співпраця у віртуальній дослідницькій групі є одним із ключових факторів успішного втілення проєкту. Як правило, вона здійснюється у цифровій формі, тому існує актуальна потреба у надійній та зручній програмній системі для забезпечення процедур їх комунікації [72].

Створення однорангових мереж, вдосконалення алгоритмів криптографії та гешування, активний розвиток Blockchain [73] та InterPlanetary File System (IPFS) [74] забезпечили перспективне технологічне підґрунтя для побудови і поширення децентралізованих систем. Співпраця над проєктами, зазвичай передбачає участь кількох сторін, які можуть бути розділені географічно чи інституційно, з різними ролями і обов’язками. В галузі інформаційних технологій поширеною практикою є збереження історії змін та поширення актуального стану проєкту за допомогою розподіленої системи контролю версій Git, але синхронізація все ще здійснюється з використанням централізованих сервісів, таких як Github, Bitbucket, GitLab, тощо. Технологія блокчейн пропонує вагомі переваги над централізованими сховищами даних, проте не пристосована для опрацювання даних великого розміру. Оскільки співпраця у віртуальних наукових спільнотах передбачає надсилання як коротких повідомлень, так і файлів даних великого обсягу, то постає потреба у використанні технології IPFS, яка забезпечує розподілене та контентно-адресоване зберігання файлів даних будь-якого розміру.

Аналіз ринку програмного забезпечення на предмет наявності готових рішень виявив відсутність інформаційних систем, які б поєднали пропоновані компоненти в межах інтеґрованої системи [75, 76]. В сьогоденні комунікація та співпраця у віртуальних наукових колективах здебільшого здійснюється через застосунки, що розроблені за принципом клієнт-серверної архітектури, зокрема Gmail, Drive, Skype, Viber, Github та інші веб-сервіси. Недоліком цього принципу є необхідність спрямовувати інформаційні потоки через серверну компоненту, що належить третім сторонам, і відповідно породжує ризики недоступності, цензури, порушення конфіденційності, низької пропускної здатності, тощо. Розподілені мережі дозволяють усунути або мінімізувати зазначені ризики і забезпечують нові можливості у сфері співпраці віртуальних наукових спільнот.

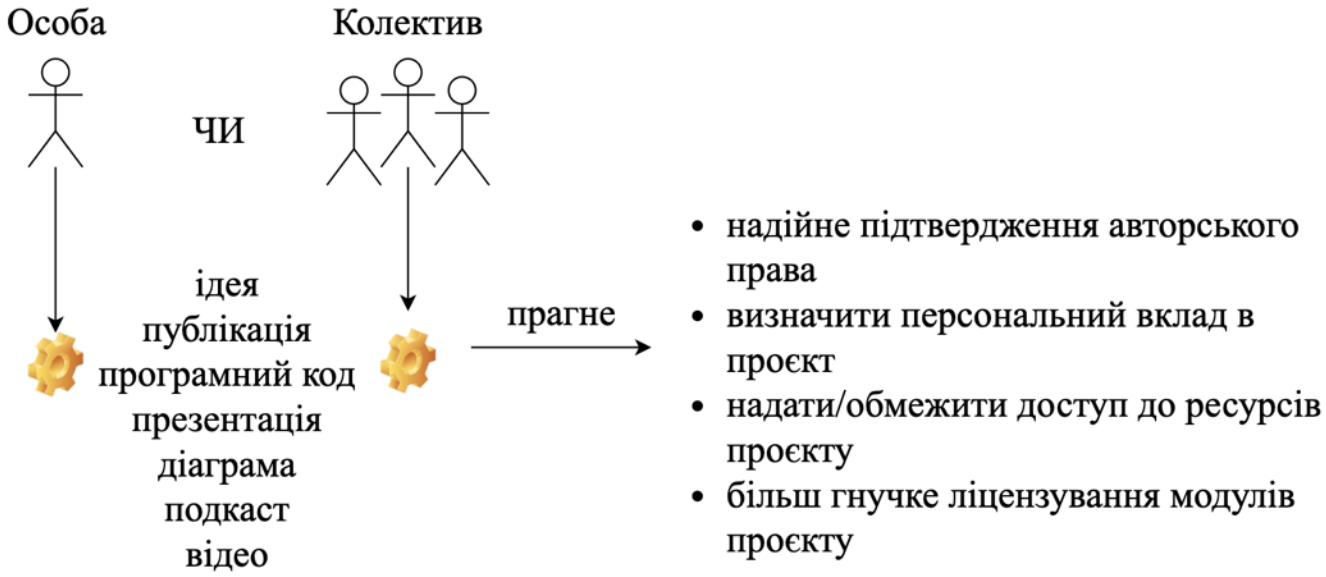

Рис. 4.1. Процеси, пов'язані з віртуальною співпрацею

Як правило, під час активної розробки проєкту створюються сотні артефактів різних версій, причому деякі з них можуть бути особливо цінними. Для учасників проєкту важливо мати технологічні можливості підтвердження авторства власних напрацювань, а також інструменти для гнучкого керування правами доступу до модулів проєкту та політикою їхнього ліцензування.

Розвиток однорангових контентних мереж та розподілених реєстрів заклав основу для нових можливостей в галузях цифрової безпеки та обміну інформацією. Наприклад, технологія блокчейн, що була вдосконалена запровадженням механізму розумних контрактів, забезпечила появу надійних електронних розподілених реєстрів. Одноранговість мережі та контентна адресація в IPFS, з використанням уніфікованого протоколу іменування IPLD [77], уможливлює побудову повністю децентралізованої, приватної інфраструктури для зберігання всіх інформаційних ресурсів. Концепція цифрової ідентичності існує вже декілька років, а нещодавно з'явилася її W3C специфікація [78]. Serto є одним з найпомітніших проєктів в галузі цифрової ідентичності [79], адже зробив суттєвий внесок у специфікацію концепції DID та реалізацію архітектури самокерованої ідентичності в середовищі Ethereum. Високоякісну реалізацію децентралізованого управління цифровою ідентичністю пропонує Hyperledger Indy [80], який є спеціалізованим програмних забезпеченням із відкритим вихідним кодом.

4.2. Технологія Blockchain

Якби існувала вимога описати технологію блокчейн єдиним словом, це було б слово «децентралізованість». Ця характеристика є важливою для мультиагентного середовища, оскільки вона забезпечує фізичну основу для надійності та довіри до системи. Асиметрична криптографія, гешування, протоколи консенсусу, розумні контракти також сприяють встановленню довіри до системи, але вже в контексті обміну інформацією. Варто зазначити, що «довіра до математично обґрунтованої поведінки системи» є конкурентною перевагою технології блокчейн, яка покликана змінити існуючі моделі комунікації і організації ресурсів проєктів в інформаційних системах. Сформувавши загальну характеристику технології блокчейн, настав час подати її загальноприйняте означення. Згідно з Hyperledger [81], блокчейн – це рівноправний розподілений реєстр, який забезпечений алгоритмом консенсусу та системою смарт-контрактів.

Рівноправні вузли. Множина усіх вузлів блокчейну утворює однорангову мережу (P2P) [82]. Зв'язок «клієнт-сервер» відсутній, оскільки до центрального серверу не підключаються, натомість, кожен вузол діє як «клієнт» (надсилає запити до мережі) та як «сервер» (опрацьовуючи запити від мережі). Щоб приєднатися до мережі, учасник повинен завантажити, встановити та запустити програму вузла. Під час початкового запуску вузол підключається до «стартових» вузлів, які вказані у вихідному коді, і отримує адреси інших вузлів однорангової мережі. При наступних запусках вузла виявлення партнерів здійснюється за допомогою розподіленої геш-таблиці (DHT) [83]. Залежно від призначення блокчейн-програми, однорангова мережа може бути публічною (загальнодоступною) або приватною (закритою). В публічних мережах, наприклад Bitcoin [84] або Ethereum [85], будь-якому агенту дозволено приєднуватися до мережі і використовувати блокчейн для власних потреб. Доступність однорангових мереж в Інтернеті, а також «хороша поведінка» агентів в блокчейні, економічно стимулюються винагородою у вигляді криптовалюти. В приватних мережах, таких як Hyperledger Fabric [86], присутня авторизаційна перевірка вузлів. Оскільки сторони, що використовують приватну мережу, як правило, знайомі між собою та зобов'язуються виконувати обов'язки у формі контракту, то криптовалюта в приватному блокчейні є необов'язковою.

Розподілений реєстр. Блокчейн – це тип розподіленого реєстру, де блоки записів зберігаються у формі лінійного зв'язаного ланцюга. Поняття «розподілений реєстр» означає консенсус агентів щодо дубльованих, спільних та синхронізованих цифрових даних, які є географічно розподілені між різними локаціями, країнами чи установами. Кожен вузол зберігає модель даних, яка фіксує поточний стан реєстру, а історія станів зберігається як ланцюг блоків в файлі бази даних. Після «видобування» нового блоку вузол-видобувач поширює його до мережі P2P, а інші вузли додають його до блоку з найвищим порядковим номером у локальній моделі даних – цей процес називається «відтворенням». Оскільки в мережі присутні тисячі вузлів-видобувачів, цілком можливим є майже одночасне створення двох блоків в

різних вузлах, де їм буде присвоєно однаковий порядковий номер. Поширення блоку в P2P-мережі потребує певного часу, тому різні вузли можуть отримати різні версії останнього блоку, що призводить до стану блокчейну, який називається «роздоріжжям». Проблема роздоріжжя вирішується за допомогою «правила вибору шляху на роздоріжжі». Ідея полягає в тому, щоб зачекати, поки блок отримає набір підтверджень, де кожне підтвердження – це новий блок, доданий до його ланцюга. Після цього, в залежності від алгоритму, за основний ланцюг береться «найважчий» або «найдовший» ланцюг. Блоки, що знаходяться в переможеному ланцюгу, не враховуються, а їх транзакції повертаються назад до буферу транзакцій. В Ethereum відкинутий блок все ще може бути включений в основний ланцюг блокчейну як блок «ommer» під час видобутку семи блоків-наступників [87].

Протокол консенсусу. Блокчейн можна змоделювати як скінченний автомат, де кожна транзакція є подією, що змінює стан. Роль протоколу консенсусу полягає у забезпеченні коректності нового стану за допомогою множини правил, які перевіряють правильність транзакції (підпис, залишок на рахунку, геш), форму блоку (підтвердження виконаної роботи, мітка часу, геш) та вибір роздоріжжя. Для запобігання зловживань блокчейн використовує різновиди алгоритмів консенсусу: доказ виконаної роботи (PoW) [88], доказ володіння часткою (PoS) [89], доказ затраченого часу (PoET) [90]. Алгоритми PoW споживають багато обчислювальних ресурсів та породжують проблему масштабованості P2P-мережі, тому все більше публічних блокчейнів переходять до PoS алгоритмів. Ідея PoS полягає у внесенні певної суми криптовалюти у вигляді депозиту на визначену адресу смарт-контракту та автоматичному виборі вузла для перевірки нового блоку на основі розміру його частки. Якщо вузол підозрюється у недоброчесній діяльності, то він може втратити свою частку. Цей підхід покращує безпеку, енергоефективність і масштабованість мережі, знижує ризик централізації. Алгоритми консенсусу постійно розвиваються та призначені захищати від усіх векторів атаки, зокрема: фальсифікації, подвійних витрат, DDoS-атак, підкупу, атаки Sybil, ASIC-схем, факторизації, тощо.

Розумні контракти. Технологія розподіленого реєстру (DLT) складається з розподіленого реєстру та системи смарт-контрактів [91]. Смарт-контракт – це комп'ютерна програма, яка виконується і розгортається в блокчейні за допомогою транзакції у формі облікового запису з унікальною адресою і власним станом. Коли розумний контракт розміщено в мережі, то його виконання можна ініціювати, надіславши транзакцію з даними очікуваної форми на адресу смарт-контракту. Роль блокчейну полягає у забезпеченні надійного середовища для виконання програми розумного контракту. Це дозволяє запрограмувати певну поведінку, яка залежить від платформи та мови програмування. Ethereum надає майже Тюрінг-повну мову Solidity, яка дозволяє втілювати будь-яку обчислювану функцію. Слово «майже» спричинене поняттям «газу», що використовується для уникнення проблеми зупинки алгоритму у віртуальній машині Ethereum. Смарт-контракту в Ethereum властиві режими виконання: ініціалізація та виклик через повідомлення. Перший активується під час розгортання контракту та створює середовище для його виконання, тоді як другий застосовується при надходженні транзакції на адресу контракту. В обох випадках контракт може оновити стан, створити нові смарт-контракти, надіслати транзакцію на інші адреси, тобто здійснити ті операції, що зазначені в його програмному коді. Відомим прикладом розумного контракту є «DAO» [55], який скомпроментували через рекурсивне виконання «резервної» функції смарт-контракту. Цей випадок демонструє актуальну потребу отримання доказів відповідності між специфікацією та реалізацією програми, зокрема за допомогою формальної перевірки коду.

Обмеження. Blockchain пропонує суттєві переваги над централізованими системами, проте йому також властиві певні обмеження, які доцільно враховувати. Оскільки співпраця у віртуальній науковій спільноті передбачає поширення як коротких повідомлень, так і файлів великого розміру, то існує потреба в технології, яка забезпечить надійну та ефективну інтеґрацію інформації довільного розміру в блокчейн. Для досягнення зазначеної мети варто скористатися технологією IPFS –

розподіленою файловою системою, в якій кожен файл та його блоки пов’язані з унікальним геш-кодом, що обчислюється на основі їх вмісту. Таке структурування запобігає появі копій файлу в мережі та дозволяє зберігати його версії. Інтеґрація IPFS з Blockchain полягає в записі геш-коду довільної версії файлу у спеціально створений смарт-контракт.

Приватний блокчейн. Публічні блокчейни, такі як Ethereum, є хорошою початковою платформою для створення розподілених застосунків – DApps, оскільки вони задовольняють принципи децентралізації, незмінності транзакцій і відкритості протоколу. Будь-хто, хто бажає взяти участь в консенсусі, може безперешкодно приєднатися до публічної мережі, запустити повний вузол, створювати транзакції, перевіряти блоки, покидати мережу. Ціною за такий ступінь свободи є низька пропускна здатність транзакцій (TPS), відсутність приватності даних та комісія за транзакції. Більшість вказаних проблем можна мінімізувати, змінивши протокол консенсусу, використанням геш-коду як посилання на чутливі дані, шифруванням відкритого ключа, мінімізацією видимості поза-ланцюгових операцій, проте коректність блоків, незмінність транзакцій, доступність записів в розподіленому реєстрі значною мірою залежить від мережі анонімних вузлів.

Приватні блокчейни, такі як Hyperledger Fabric, забезпечують розподілений консенсус по-іншому: кожен, хто прагне взяти участь в процесі роботи блокчейну, повинен бути ідентифікованим та авторизованим для бажаних операцій. Права та обов’язки довільного вузла легко визначити та перевірити, а ризики неналежної поведінки чи недоступності мережі суттєво зменшуються, усуваючи таким чином необхідність стимулювання учасників мережі та уникаючи комісії за транзакцію. Крім цього, група авторизованих вузлів, що беруть участь в певній діяльності, може досягти консенсусу щодо власних транзакцій самостійно, не розкриваючи будь-яких деталей іншим вузлам мережі. Ціною вищезазначеної переваги приватних блокчейнів є присутність квазіцентралізованої ієрархії вузлів та потреба у створенні і управлінні спеціалізованою операційною інфраструктурою.

4.3. Архітектура розподіленої бази знань

Специфікація та реалізація амбітної ідеї у вигляді проєкту – це безперервний процес, який зазвичай породжує сотні цифрових компонент різних версій від різних авторів. Проблема управління версіями цифрових активів є добре відомою задачею в сфері розробки програмного забезпечення, яка використовує розподілену систему контролю версій Git як галузевий стандарт для відстеження змін у файлах та поширення актуальної версії проєкту. Коли програмний код, опис ідеї чи наукова стаття (бажано у вигляді модульного LaTeX документу) оцифровані, то вони додаються та зберігаються командами системи Git в локальне сховище, а потім надсилаються до централізованого сховища, такого як Github або Bitbucket. Під час операції збереження змін до них додаються облікові дані автора, проте історію Git сховища можливо змінити за допомогою команди «git rebase -i», що власне створює ризик для надійного встановлення авторства цифрових артефактів проєкту [92].

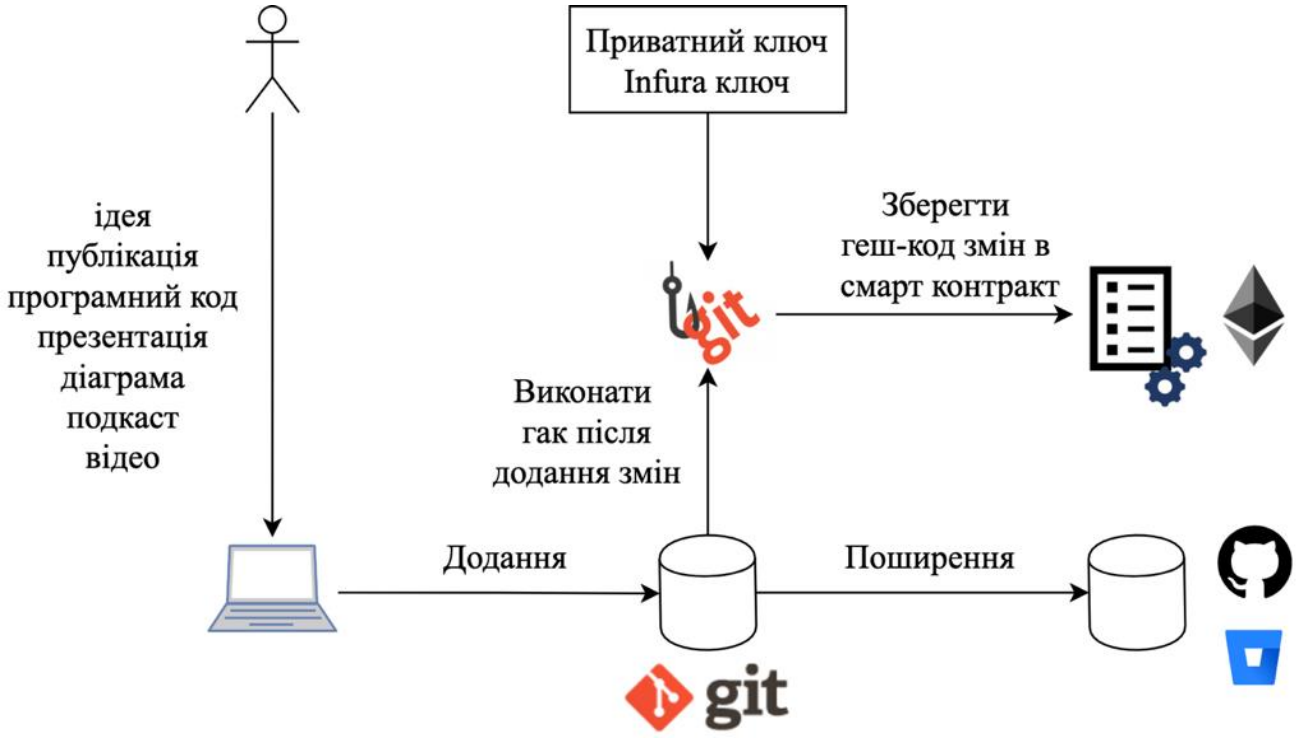

Рис. 4.2. Позначені часовими мітками зміни у централізованому сховищі

Не ігноруючи ризик фальсифікації авторства та враховуючи ймовірні вимоги проєкту щодо нерозголошення інформації, пропонується фіксувати геш-код Git зміни в реєстр спеціального смарт-контракту, за допомогою блокчейн транзакції. Оскільки Git використовує дерево Меркле [93] – структуру даних для гешування вмісту об’єкта зміни, а геш зміни однозначно ідентифікує її вміст, не розкриваючи конфіденційної інформації, то його безпечно розміщувати в транзакції публічного блокчейну. Для зручності використання рекомендовано створити «Git-гак», який виконується після збереження зміни в локальному сховищі, та потребує ключі для підписання (закритий ключ) та публікації (маркер Infura [94]) блокчейн транзакції на адресу спеціально розробленого Ethereum смарт-контракту. Оскільки кожна транзакція в основній мережі Ethereum є платною, то автор може вибрати, які зміни фіксувати в реєстрі з міткою часу, а які через малу значимість не фіксувати. Коли транзакція з геш-кодом Git зміни успішно «видобута» та підтверджена щонайменше дванадцятьма блоками, тоді її можна поширити до решти учасників проєкту. Щоб не скомпроментувати авторство власної Git зміни, необхідно надійно зберігати приватний ключ власного облікового запису в Ethereum.

Запропонована архітектура успішно усуває ризик відсутності підтвердження авторства, але не вирішує питання централізації. Поява протоколу InterPlanetary Linked Data (IPLD) і його активне впровадження в розподілену файлову систему IPFS, дозволяє по-справжньому децентралізувати зберігання та поширення Git сховища. Модуль інтеґрації IPLD для Git призначений для перетворення об’єктів Git в IPLD граф, обхід якого можна здійснити за допомогою команд IPFS. Кожен вузол в графі позначений самоописуваним ідентифікатором (CID) [95], який адресується вмістом та може бути використаний для посилання на відповідний Git об’єкт. Оновлена архітектура запропонованої системи передбачає зберігання CID об’єкта разом з міткою часу в смарт-контракті та розміщення IPLD графу Git зміни в мережі IPFS. Оскільки IPFS є публічною мережею вузлів, то для забезпечення конфіденційності інформації знадобиться створення приватного IPFS кластера.

Рис. 4.3. Децентралізація Git сховища в IPFS

Ще одне покращення системи стосується механізму авторизації та опирається на поняття самокерованої ідентичності [96]. Необхідність зберігання та поширення списку однорангових вузлів, яким надається доступ до приватного IPFS кластера, можливо замінити модулем розподіленої авторизації, що ґрунтується на понятті децентралізованих ідентифікаторів (DID). Запропонований підхід до авторизації вимагає наявності смарт-контракту для зіставлення DID та коректно налаштованого доступу до служби авторизації. Процес авторизації зазвичай розпочинається з DID автентифікації, де DID повинен підписати випадково згенерований ребус власним приватним ключем, для підтвердження права власності на ідентичність. Якщо рукостискання автентифікації відбулось успішно, то DID може надіслати запит на отримання документу Verifiable Claim [97] з підписом служби авторизації, який

надає авторизований доступ до визначених ресурсів проєкту, після його підписання приватним ключем одержувача. З технічної точки зору надання прав автентифікації та авторизації вимагає делегування управління службою авторизації керівнику DID проєкту.

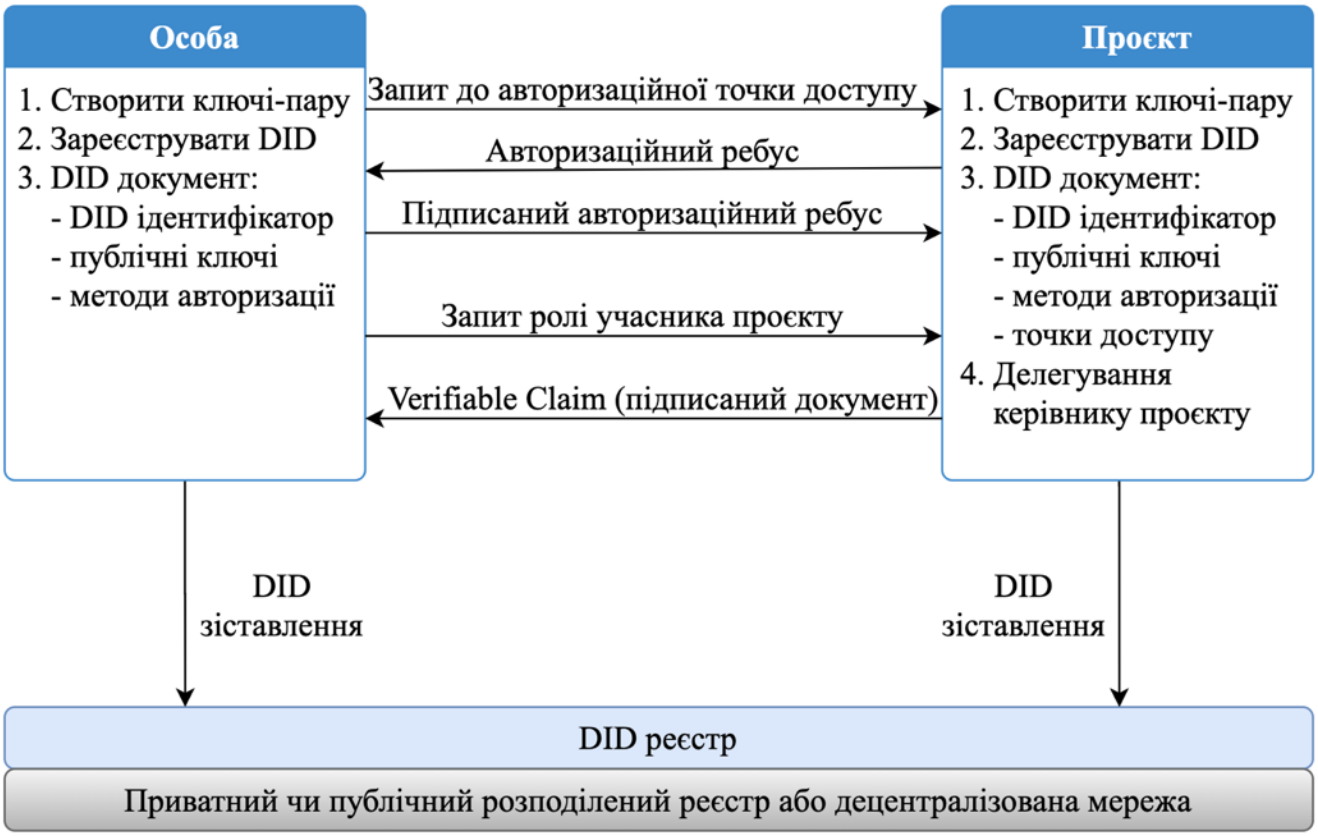

Рис. 4.4. Розподілена авторизація з використанням DID [79]

Дослідження IT ринку на предмет наявності платформ для децентралізованого управління ідентифікаторами виявило необхідність використання спеціалізованої платформи Hyperledger Indy. Використання в межах цієї платформи технології zero-knowledge proofs [98] дозволяє поширювати та перевіряти коректність лише бажаної інформації, в той час як конфіденційна інформація буде надійно захищена від зчитування та запису для неавторизованих користувачів. Механізм керованого доступу до ресурсів проєкту забезпечує високу безпеку конфіденційної інформації.

4.3.1. Програмна реалізація розподіленої бази знань

Побудова надійних розподілених баз знань для персонального користування чи співпраці в межах віртуального наукового колективу потребує використання сучасних та перевірених інструментів [99, 100]. Якщо вимагається максимальна безпека щодо поширення та зберігання даних, то першим кроком в розробці такої системи є забезпечення відповідної інфраструктури. Основою технологічного стеку для задоволення означених вимог є розподілена, версійна, контентно-адресована файлова система IPFS, яка використовується для пошуку та зберігання ресурсів бази знань. Програмний інтерфейс IPFS зручно використовувати з командної стрічки операційної системи, тому для його доступності необхідно встановити IPFS-клієнт, зокрема його програмну реалізацію мовою Go. Для цього встановимо Go інтерпретатор та «go-ipfs» модуль [101]. В операційній системі MacOS 10.15 встановлення зазначених модулів можна здійснити через застосунок «Термінал»:

```
curl -o golang.pkg https://dl.google.com/go/go1.15.darwin-amd64.pkg
sudo open golang.pkg
curl -o go-ipfs.tar.gz https://dist.ipfs.io/go-ipfs/v0.7.0/go-ipfs_v0.7.0_darwin-amd64.tar.gz
tar xvfz go-ipfs.tar.gz
mv go-ipfs/ipfs /usr/local/bin/ipfs
```

Перевірку коректності встановлення зазначених модулів здійснюється наступними командами:

```
go version
ipfs version
```

Початок роботи з файловою системою IPFS вимагає створення IPFS сховища, ключів-пари та інших налаштувань командою:

```
ipfs init
```

Перевірити коректність ініціалізації IPFS можна шляхом отримання тестового документу з локального IPFS вузла:

```
ipfs cat /ipfs/QmYwAPJzv5CZsnA625s3Xf2nemtYgPpHdWEz79ojWnPbdG/readme
```

Підключення локального IPFS вузла до глобальної мережі вузлів IPFS з метою публікації та отримання документів здійснюється командою:

ipfs daemon

Зазначена команда встановлює TCP і UDP зв'язки із завантажувальними серверами та інформує про готовність локального IPFS вузла до роботи в глобальній мережі IPFS. Динаміку комунікації з глобальною мережею можна відстежити через IPFS-клієнт, що доступний за замовчуванням у веб-оглядачі за локальною URL-адресою http://127.0.0.1:5001/webui.

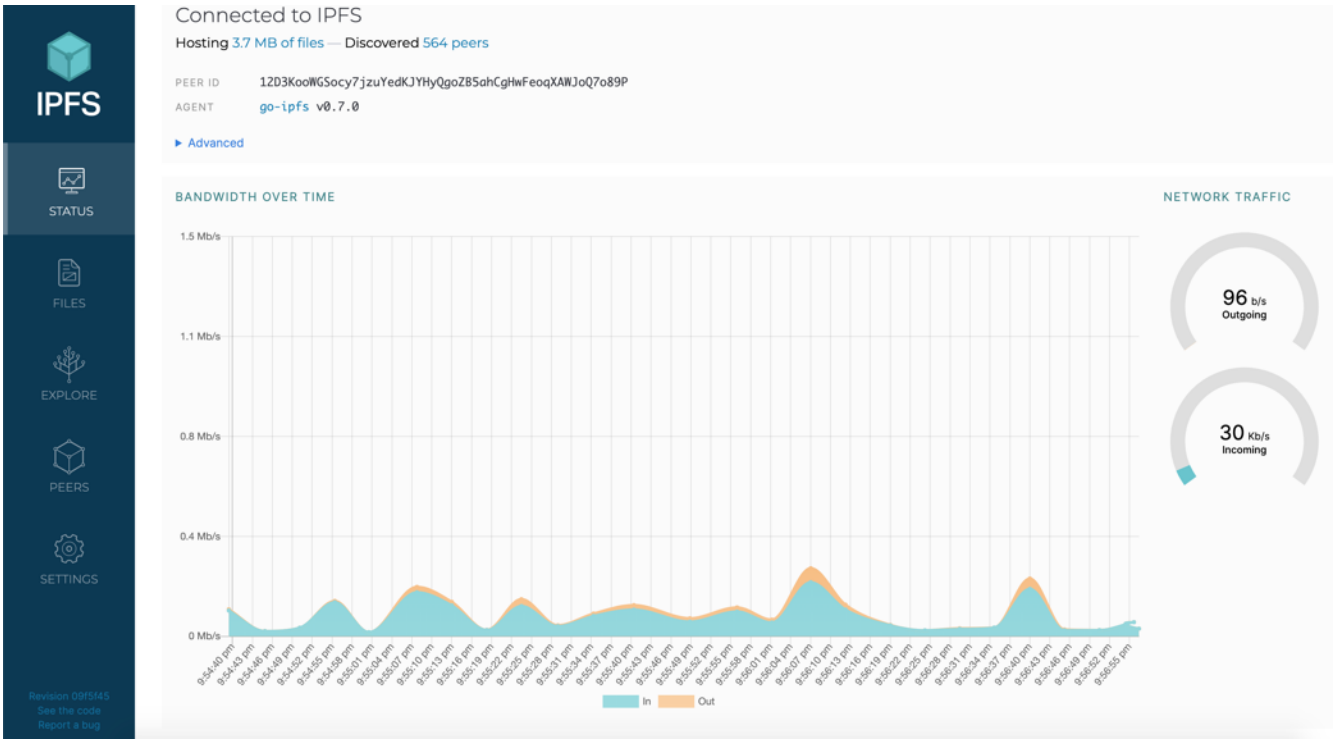

Рис. 4.5. Взаємодія локального вузла з глобальною мережею IPFS

Для створення IPFS кластеру з приватних серверів – згенеровано swarm-ключ, що використовується серверами кластеру для комунікації між собою, скопійовано цей ключ на кожен сервер, а також замінено усі сервери завантаження в кожному приватному IPFS вузлі на завантажувальний сервер з кластеру. Запуск IPFS вузлів з приватного кластеру здійснюється з використанням рекомендованої змінної середовища "export LIBP2P_FORCE_PNET=1" [102].

Після налаштування інфраструктури для розподіленого зберігання та доступу до проєктних ресурсів, забезпечується можливість їхньої версійності шляхом використання системи контролю версій Git та модуля її інтеграції з системою IPFS «git-remote-ipfs» [103], який для встановлення потребує середовище Node.js. Таким чином, потрібно виконати зазначені команди в застосунку «Термінал»:

```
brew install git
brew install node
npm install --global git-remote-ipfs
```

Наступним кроком є створення папки в локальній файловій системі для зберігання проєктних ресурсів, а також налаштування Git-репозиторію. Для кожного виду проєктного ресурсу доцільно створити окрему вкладену папку та зберігати їх відсортовано:

```
mkdir ~/Projects/StarKB2020
cd ~/Projects/StarKB2020
mkdir CoqKB  ProtegeKB  Publications  Presentations  Videos
git init
```

Файлова структура проєкту формується у вигляді папки, в якій розташована сукупність вкладених папок, кожна з яких містить ресурси певного виду, а прихована папка «.git» містить технічну інформацію про версійність ресурсів та властивості Git-репозиторію, зокрема налаштування «Git-гаків» в «.git/hooks» [104]:

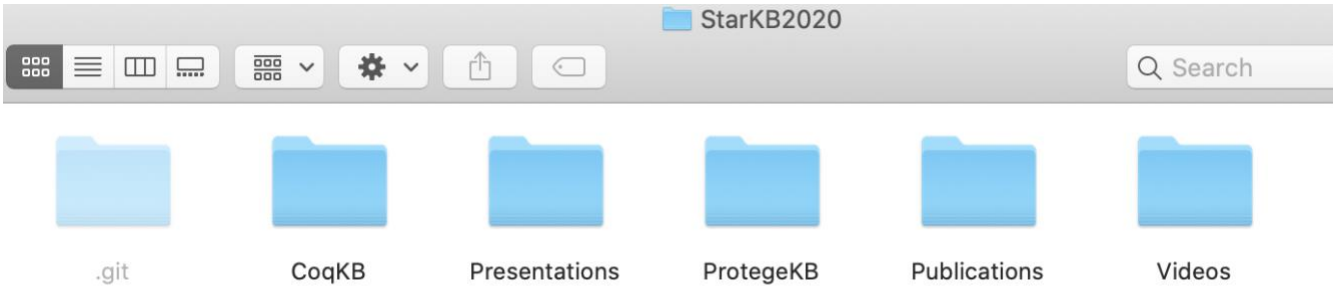

Рис. 4.6. Початкова структура проєкту у файловій системі MacOS

Розроблення онтологій для цільових предметних областей [105], що вимагають високої виразності формальної мови, здійснюється в інтерактивному асистенті доведення теорем Coq, а результати зберігаються у виглядів файлів у відповідній папці CoqKB:

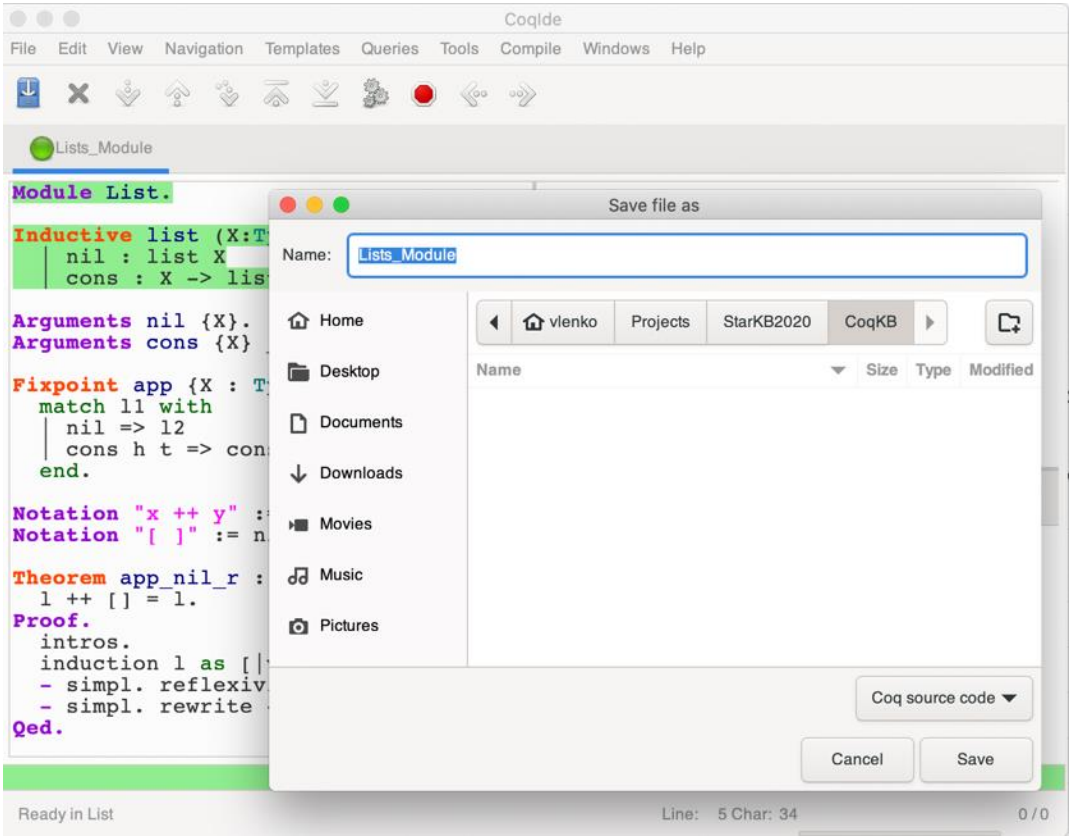

Рис. 4.7. Збереження змін до бази знань в локальний файловий ресурс

Коли новий файл додано, або внесено зміни в існуючий файл в межах папки проєкту, то Git дозволяє відстежити, вибрати та зберегти змінені файли в історію версій:

```
cd ~/Projects/StarKB2020
git status
git add -A
git commit -m "Lists module is added"
```

Остання команда записує додані зміни в локальне Git-сховище, а їх поширення до інших учасників проєкту виконується наступним чином:

```
vlenko@vlenko-mac StarKB2020 % git push ipfs://StarKB2020 --all
mit:230bbf8d6c17bcc9c1e56001586ae3a011af4dc2: tree:QmegVpptYENbqrpFK8Y1eZimmV3Q8Cd25CUZthLnjG93ro
ipfs::QmRcYndq9GBU8huHPuhtQr65eutHmHS3xrYkDCpBmz7Wvo
To ipfs://StarKB2020
 * [new branch]      master -> master
```

Рис. 4.8. Поширення проєкту з Git версійністю через IPFS

Варто зазначити, що виконання вказаної команди вимагає активного підключення до глобальної чи локальної мережі IPFS, яке зокрема можна встановити командою «ipfs daemon». Результатом виконання команди «git push» є адреса версії проєкту в мережі IPFS – “ipfs::QmRcYndq9GBU8huHPuhtQr65eutHmHS3xrYkDCpBmz7Wvo”, яка на рисунку позначена фіолетовим кольором. Перевірити IPFS ресурс за цією адресою можна через інтерфейс локального IPFS вузла, або ж через онлайн-сервіс IPLD Explorer [106]:

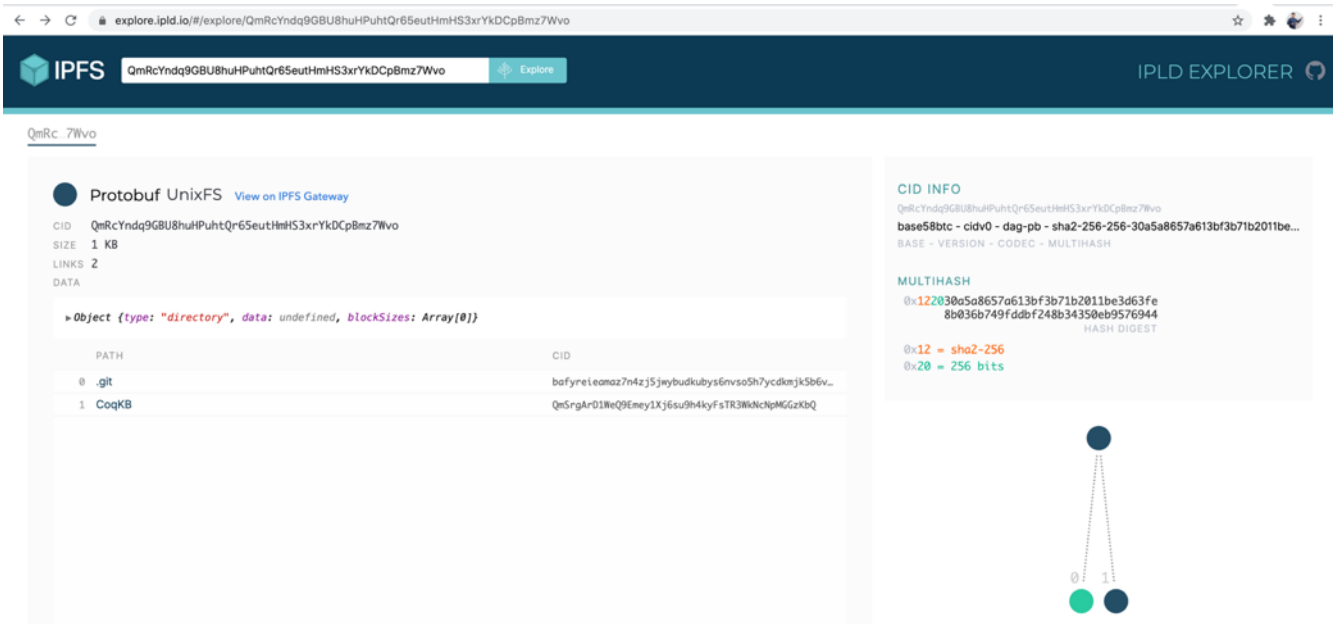

Рис. 4.9. Перегляд ресурсів проєкту за допомогою IPLD Explorer

Крім цього, вміст деяких файлів можна переглянути в онлайн-сервісі IPFS Gateway, зокрема файл «Lists_Module», що розташований в папці CoqKB доступний за URL https://ipfs.io/ipfs/QmQexyWa9M3t2WYVRS1sWtTHW5a9BcwfeNV7hLQFNEKzA9. Для завантаження Git проекту з IPFS достатньо виконати «git clone ipfs::<address>».

4.4. Висновки до розділу 4

У цьому розділі запропоновано сучасний підхід до проєктування платформ співпраці у віртуальних наукових спільнотах, заснованих на ідеях криптобезпеки та децентралізованого зберігання. Технологічний стек описаної програмної реалізації складається з системи контролю версій Git, що забезпечує збереження історії версій проєктних ресурсів, розподіленої файлової системи IPFS з контентною-адресацією, що гарантує децентралізоване зберігання та доступ до проєктних ресурсів, моделі даних IPLD для перетворення об’єктів Git на зв’язні структури в IPFS, а також інтерактивного асистента доведення теорем Coq для проектування онтологій предметних областей.

Архітектура запропонованої системи передбачає можливість її розширення елементами технології Blockchain, зокрема розумні контракти Ethereum можна використовувати як реєстр для підтвердження авторства конкретних змін до бази знань, а також як реєстр для розміщення інформації про поточну версію проєкту. Розглянуто можливість застосування децентралізованих ідентифікаторів та «підтверджених заяв», які забезпечують розподілений підхід до автентифікації та авторизації учасників проєкту завдяки сучасній інфраструктурі для управління децентралізованими ідентифікаторами Hyperledger Indy.

Представлені діаграми архітектури розподіленої системи показують способи інтеґрації вищезазначених технологічних компонент в інноваційну, розподілену систему для співпраці як окремих науковців, так і віртуальних наукових спільнот. Програмна реалізація підтверджує коректність запропонованої архітектури та надає покрокову інструкцію, щодо налаштування розподіленої інфраструктури, а також висвітлює спосіб децентралізованого зберігання та обміну проєктних ресурсів. Варто зазначити, що усі програмні компоненти системи є у відкритому доступі, та характеризуються або активною підтримкою віртуальних спільнот розробників, або статусом стандарту в своєму сегменті ринку, наприклад: Git, Ethereum, Hyperledger Indy, Coq.

ВИСНОВКИ

В дисертаційній роботі досліджено та розв’язано проблему управління персональними знання в інтелектуальних системах, а отримані результати мають вагоме значення для галузі інформаційних технологій. Кожен розділ дисертації містить системний аналіз задач дослідження, а також методів, моделей і засобів, що сприяють їхньому розв’язку. Висновки дисертації можна сформулювати у вигляді наукових положень, рекомендацій, фундаментальних та прикладних результатів досліджень:

- системний та порівняльний аналіз методів і моделей подання знання виявив якісну перевагу формальних логічних моделей, яка забезпечена наявністю формальної семантики, обґрунтованістю методів та правил міркування, а також абстрагованістю формальної мови від природніх асоціацій;
- системний аналіз теорій типів обґрунтував високий потенціал застосування числення індуктивних конструкцій та інтерактивного асистента доведення теорем Coq для подання і міркування над знаннями різної природи, зокрема математичними теоріями та реченнями природньої мови;
- системний аналіз описових логік, а також їх порівняльний аналіз з теоріями типів, встановив переваги та недоліки використання кожного з формалізмів, де основний компроміс полягає в балансі між виразністю формальної мови та розв’язністю задач міркування;
- розроблено онтологічно-орієнтований метод подання бази знань в середовищі асистента доведення теорем Coq, що сприяє формалізації нових та існуючих онтологій, а також уможливлює застосування логік вищих порядків як надійного інструменту підтримки прийняття рішення;
- розроблено архітектуру та програмну реалізацію інтелектуальної системи для управління персональними та колективними базами знань, яка володіє

властивостями версійності, розподіленості, автономності та передбачає можливість надійного підтвердження авторства;

- розроблено приклади формального доведення істинності описових знань, зокрема для задачі управління правами доступу до програмних проектів ІТ-підприємства, матеріалів мас-медіа і моделей інфраструктурних мереж, що полегшує та пришвидшує процес перевірки гіпотез в напівавтоматичному режимі;
- здійснено системний аналіз понять «знання» та «істинність знання», і аргументовано використання конструктивної логіки з метою усунення проблем Ґетьє, які компрометують традиційні методи обґрунтування істинності знання;
- результати дисертаційного дослідження впроваджено та використано в установах, ТОВ «Ботскрю», ТОВ «Сомбра Софтвер», ТОВ «Перфектіал», які спеціалізуються на розробці високоякісного програмного забезпечення та послугах консалтингу в галузі інформаційних технологій, що підтверджено відповідними актами.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Liew, A. (2007). Understanding Data, Information, Knowledge and Their Inter-Relationships. *Journal of Knowledge Management Practice, 8(2)*, 1-10.
2. Baader, F., Horrocks, I., Lutz, C., & Sattler, U. (2017). An introduction to description logic. *Cambridge University Press*. https://doi.org/10.1017/9781139025355.
3. Басюк, Т., Досин, Д., & Литвин, В. (2017). Онтологічний інжиніринг. *Львів: Видавництво Львівської політехніки*.
4. Chomsky, N. (1956). Three models for the description of language. *IRE Transactions on Information Theory (2)*, 113-124 https://doi.org/10.1109/TIT.1956.1056813.
5. Smith, B. (1982). Procedural reflection in programming languages, *Ph.D. Thesis*. Cambridge: MIT.
6. Levesque, H., & Brachman, R. (1987). Expressiveness and tractability in knowledge representation and reasoning. *Computational Intelligence*, 3(1), 78-93. https://doi.org/10.1111/j.1467-8640.1987.tb00176.x.
7. Stanford Encyclopedia of Philosophy. Epistemology [Електронний ресурс]. – Режим доступу: *https://plato.stanford.edu/entries/epistemology*.
8. Stanford Encyclopedia of Philosophy. The Analysis of Knowledge [Електронний ресурс]. – Режим доступу: *https://plato.stanford.edu/entries/knowledge-analysis.*
9. Artemov, S., & Protopopescu, T. (2016). Intuitionistic epistemic logic. *The review of symbolic logic*, 9(2), 266-298. https://doi.org/10.1017/s1755020315000374.
10. Brachman, R. & Levesque, H. (2004). Knowledge Representation and Reasoning. *San Francisco: Morgan Kaufmann Publishers Inc.*
11. Lenko, V., Pasichnyk, V., & Shcherbyna, Y. (2017). Knowledge Representation Models. *Вісник НУ «Львівська політехніка». Серія: Комп'ютерні науки та інформаційні технології*, 864, 157-168.
12. Sowa, J. Semantic networks. [Електронний ресурс]. – Режим доступу: *http://www.jfsowa.com/pubs/semnet.htm.*

13. Schank, R., & Abelson., R. (1977). Scripts, Plans, Goals, and Understanding. *Hillsdale: Lawrence Erlbaum*.
14. Литвин, В. (2011). Задачі оптимізації структури та змісту онтології та методи їх розв'язування. *Вісник НУ «Львівська політехніка». Серія: Інформаційні системи та мережі*, 715, 203-214.
15. The Univalent Foundations Program. (2013). Homotopy Type Theory: Univalent Foundations of Mathematics. *Princeton: Institute for Advanced Study* https://homotopytypetheory.org/book.
16. Lenko, V., Pasichnyk, V. & Shcherbyna, Y. (2016). Knowledge Representation Models. *Теоретичні та прикладні аспекти побудови програмних систем: матеріали XIII міжнар. наук. конф. TAAPSD'2016*, 143-152.
17. Church, A. (1936). A note on the Entscheidungsproblem. *Journal of Symbolic Logic*, 1, 40-41.
18. Turing, A. (1938). On Computable Numbers, with an Application to the Entscheidungsproblem. A Correction. *Proceedings of the London Mathematical Society*, s2-43(6), 544-546. https://doi.org/10.1112/PLMS/S2-43.6.544.
19. Lindström, P. (1969). On Extensions of Elementary Logic. *Theoria,* 35, 1-11. https://doi.org/10.1111/j.1755-2567.1969.tb00356.x.
20. Whitehead, A., & Russell, B. (1910-1913). Principia mathematica. *Cambridge: The University Press.*
21. Stanford Encyclopedia of Philosophy. Russell's Paradox [Електронний ресурс]. – Режим доступу: *https://plato.stanford.edu/entries/russell-paradox.*
22. Stanford Encyclopedia of Philosophy. Type Theory [Електронний ресурс]. – Режим доступу: *https://plato.stanford.edu/entries/type-theory.*
23. Howard, W. (1980). The formulae-as-types notion of construction. *To H.B. Curry: Essays on Combinatory Logic, Lambda Calculus and Formalism*. Boston: Academic Press, 1980, 479-490.

24. Wadler, P. (2015). Propositions as Types. *Communications of the ACM*, 58(2), 75-84. https://doi.org/10.1145/2699407.
25. Ленько, В. & Щербина, Ю. (2016). Відновлення графічних образів за допомогою карт Кохонена. *Сучасні проблеми прикладної математики та інформатики: збірник наукових праць конф. APAMCS-20*, 108-111.
26. Ленько В. (2011). Застосування методів штучного інтелекту до сегментації графічних образів. *Чотирнадцята всеукраїнська (дев'ята міжнародна) студентська наукова конференція з прикладної математики та інформатики СНКПМІ-2011: тези доповідей*, 60-61.
27. Minski, M. (1974). A Framework for Representing Knowledge. *Technical Report.* Cambridge: MIT.
28. Ленько, В., Кунанець, Н., Пасічник, В., & Щербина, Ю. (2017). Побудова моделі «Розумного соціополісу» на основі знань експертів. *Інформаційні технології, економіка та право: стан та перспективи розвитку. (ІТЕП-2017): матеріали міжнародної науково-практичної конференції*, 141-143.
29. Matsiuk, O., Kunanets, N., Pasichnyk, V., Lenko, V., Shcherbyna, Y. & Rzheuskyi, A. (2019). The procedures of processing of geolocation data on urban underground spaces. *Proceedings of the 2019 9th International Conference on Advanced Computer Information Technologies (ACIT)*, 500-503.
30. Ленько, В. & Щербина, Ю. (2011). Застосування методів штучного інтелекту до сегментації графічного образу. *Вісник НУ «Львівська політехніка». Серія: Інформаційні системи та мережі*, 715, 194-203.
31. Forgy, C. (1982). Rete: A Fast Algorithm for the Many Pattern/Many Object Pattern Match Problem. *Artificial Intelligence*, 19 (1), 17-37.
32. Floridi, L. (2003). Philosophy of Computing and Information. *Blackwell Publishing*.
33. Noy, N., McGuinness, D. Ontology Development 101: A Guide to Creating Your First Ontology. – [Електронний ресурс]: *http://protege.stanford.edu/publications/ontology_development/ontology101-noy-mcguinness.html*.

34. Ленько, В., Пасічник, В. & Щербина, Ю. (2017). Подання знань та логічні міркування у формальних онтологіях. *Комп'ютерне моделювання та програмне забезпечення інформаційних систем і технологій. (КМПЗ-2017): збірник наукових праць III Всеукраїнської між. наук.-практ. конф.*, 94-95.
35. Farmer, W. (2008). The seven virtues of simple type theory. *Journal of Applied Logic*, 6(3), 267-286.
36. Ramsey, F. (1926). The foundations of mathematics. *Proceedings of the London Mathematical Society*, 2, (25), 338-384.
37. Church, A. (1940). A formulation of the simple theory of types. *Menasha: WI*.
38. Martin-Löf, P. (1984). Intuitionistic type theory. *Napoli: Bibliopolis.*
39. Sørensen. M., & Urzyczyn, P. (2007). Lectures on the Curry-Howard isomorphism. *Boston: Elsevier*.
40. Constable, R. (1998). Types in Logic, Mathematics and Programming. *Studies in Logic and the Foundations of Mathematics Handbook of Proof Theory*, 683-786.
41. Rojas, R. A Tutorial Introduction to the Lambda Calculus. [Електронний ресурс]. – Режим доступу*: https://arxiv.org/pdf/1503.09060.pdf.*
42. Nederpelt, R., & Geuvers, H. (2014). Type theory and formal proof: an introduction. Cambridge: *Cambridge University Press*.
43. Ленько, В., Пасічник, В., Кунанець, Н. & Щербина, Ю. (2018). Теоретичні аспекти логічного міркування у середовищі Coq. *Інтелектуальні системи прийняття рішень і проблеми обчислювального інтелекту: матеріали міжнародної наукової конференції ISDMCI-2018*, 174-176.
44. Coquand, T., & Gérard, H. (1986). The Calculus of Constructions. *Rocquencourt, le Chesnay: Institut National de Recherche en, Informatique et en Automatique*.
45. Jutting, L. (1993). Typing in Pure Type Systems. *Information and Computation*, 105(1), 30-41.
46. Barendregt, H. (1991). Introduction to generalized type systems. *Journal of Functional Programming*, 1(2), 125-154.

47. The Coq Proof Assistant Reference Manual (v.8.10.2) [Електронний ресурс]. – Режим доступу: *https://coq.inria.fr/distrib/ current/refman*.
48. Paulin-Mohring, C. (2015). Introduction to the calculus of inductive constructions. *London: College Publications.*
49. Chlipala, A. (2013). Certified programming with dependent types: a pragmatic introduction to the Coq proof assistant. *Cambridge: MIT Press.*
50. Paulin-Mohring, C. (2012). Introduction to the Coq Proof-Assistant for Practical Software Verification. *Lecture Notes in Computer Science Tools for Practical Software Verification*, 45-95.
51. Gonthier, G. (2008). Formal Proof – The Four-Color Theorem. *Notices of the American Mathematical Society, 55(11)*, 1382-1393.
52. Lenko, V., Pasichnyk, V., Kunanets, N. & Shcherbyna, Y. (2017). Type-Theoretical Foundations of the Derivation System in Coq. *Proceedings of the 2018 IEEE First International Conference on System Analysis & Intelligent Computing (SAIC)*, 220-225. https://doi.org/10.1109/SAIC.2018.8516885.
53. Ленько, В., Пасічник, В., Щербина, Ю. & Кунанець, Н. (2017). Міркування в онтологіях з використанням теорії типів. *Теоретичні та прикладні аспекти побудови програмних систем: матеріали XIV міжнар. наук. конф. TAAPSD'2017*, 138-141.
54. Casset, L., Burdy, L., & Requet, A. (2002). Formal development of an embedded verifier for Java Card byte code. *Proceedings International Conference on Dependable Systems and Networks (DSN).*
55. Siegel, D. (2016). Understanding The DAO Attack [Електронний ресурс]. – Режим доступу: *https://coindesk.com/understanding-dao-hack-journalists.*
56. The DeepSpec consortium. (2016). The Science of Deep Specification [Електронний ресурс]. – Режим доступу: *https://deepspec.org*.
57. Dowty, D., Wall, R., & Peters, S. (1981). Introduction to Montague Semantics. *Kluwer Academic Publishers*.

58. Chatzikyriakidis, S., & Luo, Z. (2014). Natural Language Inference in Coq. *Journal of Logic, Language and Information, 23(4),* 441-480.
59. Lenko, V., Pasichnyk, V., Kunanets, N. & Shcherbyna, Y. (2018). Knowledge representation and formal reasoning in ontologies with Coq. *Advances in Computer Science for Engineering and Education,* 754, 759-770. https://doi.org/10.1007/978-3-319-91008-6.
60. Кунанець, Н., Ленько, В., Пасічник, В. & Щербина, Ю. (2017). Подання онтологій з використанням теорії типів. *Системи та засоби штучного інтелекту: тези доповідей Міжнародної наукової молодіжної школи AIIS'2017*, 114-117.
61. OWL 2 Web Ontology Language Direct Semantics (Second Edition) [Електронний ресурс]. – Режим доступу: *https://www.w3.org/TR/owl2-direct-semantics.*
62. Tarski, A. (1933). Pojęcie prawdy w językach nauk dedukcyjnych. *Prace Towarzystwa Naukowego Warszawskiego, Wydzial III Nauk Matematyczno-Fizycznych 34*.
63. Baader, F., Calvanese D., McGuinness, D., Nardi, D., & Patel-Schneider, P. (2007). The description logic handbook: theory, implementation, and applications. Second edition. *Cambridge University Press.* https://doi.org/10.1017/CBO9780511711787.
64. Табачишин, Д., Ленько, В., Кунанець, Н., Пасічник, В. & Щербина, Ю. (2017). Експертне оцінювання «розумності міста» із застосуванням нечіткої логіки. *Штучний інтелект,* 1(75), 102-110.
65. Keet, C. (2013). Open World Assumption. *In: Dubitzky W., Wolkenhauer O., Cho KH., Yokota H. (eds) Encyclopedia of Systems Biology*. https://doi.org/10.1007/978-1-4419-9863-7_734.
66. Lenko, V., Pasichnyk, V. & Shcherbyna, Y. On personal knowledge management systems. *Математика. Інформаційні технології. Освіта: тези доповідей VI міжнар. наук.-практ. конф. МІТО-2017,* 57-59.

67. Motik, B. (2006). Reasoning in Description Logics using Resolution and Deductive Databases. *PhD Thesis.* [Електронний ресурс]. – Режим доступу: *https://www.cs.ox.ac.uk/people/boris.motik/pubs/motik06PhD.pdf*.
68. Protégé: a free, open-source ontology editor & framework for building intelligent systems. [Електронний ресурс]. – Режим доступу: *https://protege.stanford.edu*.
69. Awodey, S. & Rabe, F. (2011). Kripke Semantics for Martin-Löf's Extensional Type Theory. *Logical Methods in Computer Science*, 7 (3:18), 1-25.
70. Sato M. (1997) Classical Brouwer-Heyting-Kolmogorov interpretation. *In: Li M., Maruoka A. (eds) Algorithmic Learning Theory. ALT 1997. Lecture Notes in Artificial Intelligence*, 1316, 176-196. https://doi.org/10.1007/3-540-63577-7_43.
71. HermiT OWL Reasoner. [Електронний ресурс]. – Режим доступу: *http://www.hermit-reasoner.com.*
72. Кунанець, Н., Ленько, В., Пасічник, В. & Щербина, Ю. (2017). Персональні бази даних та знань віртуальних дослідницьких спільнот. *Науковий вісник НЛТУ України*, 27(6),185-191. https://doi.org/10.15421/40270638.
73. Werbach, K. (2018). The Blockchain and the New Architecture of Trust. *The MIT Press*.
74. Benet J. (2014). InterPlanetary File System (IPFS) – Content Addressed, Versioned, P2P File System. [Електронний ресурс]. – Режим доступу: *https://github.com/ipfs /papers/raw/master/ipfs-cap2pfs/ipfs-p2p-file-system.pdf*.
75. Ленько, В., Пасічник, В., & Щербина, Ю. (2017). Проект системи управління персональними знаннями. *Управління проектами: стан та перспективи. (ІТПРР-2017): матеріали XIII міжнародної наук.-практ. конференції*, 61-62.
76. Ленько, В., Пасічник, В., Кунанець, Н. & Щербина, Ю. (2018). Проектування відкритих децентралізованих реєстрів з використанням технології Blockchain. *Математика. Інформаційні технології. Освіта: тези доповідей VII міжнар. наук.-практ. конф. МІТО-2018*, 72-74.

77. Protocol Labs. (2016). InterPlanetary Linked Data (IPLD) – The data model of the content-addressable web [Електронний ресурс]. – Режим доступу: *https://ipld.io*.
78. W3C. Decentralized Identifiers (DIDs) v1.0 [Електронний ресурс]. – Режим доступу: *https://www.w3.org/TR/did-core.*
79. ConsenSys. (2016). Serto – Data meets identity. [Електронний ресурс]. – Режим доступу: *https://www.serto.id.*
80. Hyperledger Indy: distributed ledger for decentralized identity [Електронний ресурс]. – Режим доступу: *https://www.hyperledger.org/use/hyperledger-indy.*
81. Hyperledger: Open Source Blockchain Technologies [Електронний ресурс]. – Режим доступу: *https://www.hyperledger.org*.
82. Leibnitz, K., Hoßfeld, T., Wakamiya, N. & Murata M. (2007). Peer-to-Peer vs. Client/Server: Reliability and Efficiency of a Content Distribution Service. *In: Mason L., Drwiega T., Yan J. (ed.) Managing Traffic Performance in Converged Networks. ITC 2007. Lecture Notes in Computer Science*, 4516, 1161-1172. https://doi.org/10.1007/978-3-540-72990-7_99
83. Maymounkov, P. & Mazières, D. (2002). Kademlia: A Peer-to-Peer Information System Based on the XOR Metric. *In: Druschel P., Kaashoek F., Rowstron A. (ed.) Peer-to-Peer Systems. IPTPS 2002. Lecture Notes in Computer Science*, 2429, 53-65. https://doi.org/10.1007/3-540-45748-8_5.
84. Nakamoto, S. (2008). Bitcoin: A Peer-to-Peer Electronic Cash System [Електронний ресурс]. – Режим доступу: *https://bitcoin.org/bitcoin.pdf.*
85. Wood, G. (2014). Ethereum: A Secure Decentralised Generalised Transaction Ledger [Електронний ресурс]. – Режим доступу: *https://ethereum.github.io/yellowpaper/paper.pdf.*
86. Hyperledger Fabric – enterprise-grade permissioned distributed ledger framework. [Електронний ресурс]. – Режим доступу: *https://www.hyperledger.org/use/fabric*.

87. Ritz, F. & Zugenmaier, A. (2018). The Impact of Uncle Rewards on Selfish Mining in Ethereum. *2018 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW)*, 50-57.
88. Jakobsson, M. & Juels, A. (1999). Proofs of Work and Bread Pudding Protocols (Extended Abstract). *In: Preneel B. (eds) Secure Information Networks. IFIP — The International Federation for Information Processing*, (23), 258-272.
89. Kiayias, A., Russell, A., David, B. & Oliynykov R. (2017). Ouroboros: A Provably Secure Proof-of-Stake Blockchain Protocol. *In: Katz J., Shacham H. (eds) Advances in Cryptology. Lecture Notes in Computer Science*, 10401, 357-388.
90. Chen, L., Xu, L., Shah, N., Gao, Z., Lu, Y. & Shi W. (2017). On Security Analysis of Proof-of-Elapsed-Time (PoET). *Stabilization, Safety, and Security of Distributed Systems. SSS 2017. Lecture Notes in Computer Science*, 10616, 282-297.
91. UK Government Office for Science. (2016). Distributed ledger technology: Blackett review [Електронний ресурс]. – Режим доступу: *https://www.gov.uk/government /publications/distributed-ledger-technology-blackett-review.*
92. Lenko, V., Pasichnyk, V., Kunanets, N. & Shcherbyna, Y. (2019). Decentralized Blockchain-based platform for collaboration in virtual scientific communities. *ECONTECHMOD*, 08 (1), 21-26.
93. Merkle, R. (1988). A Digital Signature Based on a Conventional Encryption Function. *Advances in Cryptology – CRYPTO'87. Lecture Notes in Computer Science*, 293, 369-378. https://doi.org/10.1007/3-540-48184-2_32.
94. ConsenSys. (2016). Infura: Scalable Blockchain Infrastructure. [Електронний ресурс]. – Режим доступу: *https://infura.io.*
95. CID (Content IDentifier) Specification [Електронний ресурс]. – Режим доступу: *https://github.com/multiformats/cid.*
96. Der, U., Jähnichen, S. & Sürmeli J. (2017). Self-sovereign Identity − Opportunities and Challenges for the Digital Revolution [Електронний ресурс]. – Режим доступу: *https://arxiv.org/abs/1712.01767.*

97. W3C. (2019). Verifiable Credentials Use Cases. [Електронний ресурс]. – Режим доступу: *https://www.w3.org/TR/vc-use-cases.*

98. Feige, U., Fiat, A. & Shamir, A. (1988). Zero-knowledge proofs of identity. *Journal of Cryptology*, 1, 77-94. https://doi.org/10.1007/BF02351717.

99. Ленько, В., Кунанець, Н., Пасічник, В., & Щербина, Ю. (2017). Децентралізована комунікація у віртуальних наукових колективах. *Практичне застосування нелінійних динамічних систем в інфокомунікаціях: матеріали VI міжнар. наук.-практ. конф.*, 118-119.

100. Ленько, В., Пасічник, В., & Щербина, Ю. (2016). Методи та засоби управління персональними знаннями. *Інформаційно-обчислювальні технології, автоматика та електротехніка: матеріали Міжнар. наук.-практ. конф. ITAE-2016,* 143-144.

101. IPFS implementation in Go [Електронний ресурс]. – Режим доступу: *https://github.com/ipfs/go-ipfs.*

102. Laar, S. Deploy a private IPFS network in 5 steps [Електронний ресурс]. – Режим доступу: *https://medium.com/@s_van_laar/deploy-a-private-ipfs-network-on-ubuntu-in-5-steps-5aad95f7261b.*

103. Interplanetary Filesystem (IPFS) Git Remote Helper. [Електронний ресурс]. – Режим доступу: *https://github.com/dhappy/git-remote-ipfs.*

104. Kemper, C. & Oxley I. (2012). Hooks, and Why They Can Be Useful. *In: Foundation Version Control for Web Developers*, 341-358. https://doi.org/10.1007/978-1-4302-3973-4_13.

105. Кунанець, Н., Ленько, В., Пасічник, В. & Щербина, Ю. (2017). Подання онтологічних моделей знань в системі інтерактивного доведення теорем Coq. *Інформаційні технології та взаємодії: матеріали доповідей IV міжнар. наук.-практ. конф. IT&I-2017,* 169-170.

106. IPLD Explorer [Електронний ресурс]. – Режим доступу: *https://explore.ipld.io.*

Додаток А

Список публікацій здобувача за темою дисертації та відомості про апробацію результатів дисертації

Наукові праці, в яких опубліковані основні наукові результати дисертації:

1. Кунанець Н. Е., **Ленько В. С.**, Пасічник В. В., Щербина Ю. М. Персональні бази даних та знань віртуальних дослідницьких спільнот. *Науковий вісник НЛТУ України*. 2017. Вип. 27(6). С. 185–191. *Особистий внесок здобувача: запропоновано принципово новий підхід до проектування платформ комунікації, що ґрунтується на ідеях децентралізації та криптобезпеки.*
2. **Lenko V.**, Kunanets N., Pasichnyk V., Shcherbyna Yu. Decentralized Blockchain-based platform for collaboration in virtual scientific communities. *Econtechmod*. 2019. Vol. 8 (1). P. 21–26. *Особистий внесок здобувача: розроблено структуру розподіленої системи управління персональними знаннями, з можливістю їх поширення та встановлення авторства*.
3. **Lenko V. S.**, Pasichnyk V. V., Shcherbyna Y. M. Knowledge representation models. *Вісник Національного університету «Львівська політехніка»*. Серія: Комп'ютерні науки та інформаційні технології. 2017. № 864. С. 157–168. *Особистий внесок здобувача: здійснено порівняльний аналіз найпоширеніших моделей подання знання, з метою удосконалення методології вибору технології для подання та міркування над знанням різної природи.*
4. Табачишин Д. Р., **Ленько В. С.**, Кунанець Н. Е., Пасічник В. В., Щербина Ю. М. Експертне оцінювання «розумності міста» із застосуванням нечіткої логіки. *Штучний інтелект*. 2017. № 1 (75). С. 102–110. *Особистий внесок здобувача: розроблено програмне забезпечення для обчислення «розумності міста» з використання нечіткої логіки та експертної оцінки.*
5. **Ленько В. С.**, Щербина Ю. М. Застосування методів штучного інтелекту до сегментації графічного образу. *Вісник Національного університету «Львівська політехніка»*. Серія: Інформаційні системи та мережі. 2011. № 715. С. 194–203. *Особистий внесок здобувача: удосконалено ефективний графовий алгоритм сегментації зображення та розроблено програмне забезпечення для експерименту.*
6. **Lenko V.**, Pasichnyk V., Kunanets N., Shcherbyna Y. Knowledge representation and formal reasoning in ontologies with Coq. *Advances in Computer Science for Engineering and Education*. 2018. Vol. 756. P. 759–770. *Особистий внесок*

здобувача: запропоновано онтологічно-орієнтований метод подання бази знань в середовищі асистента доведення теорем Coq, що сприяє формалізації нових та існуючих онтологій.

7. **Lenko V**. Type-theoretical foundations of the derivation system in Coq / V. Lenko, V. Pasichnyk, N. Kunanets, Y. Shcherbyna // *Proceedings of the 2018 IEEE First International Conference on System Analysis & Intelligent Computing (SAIC)*. – IEEE, 2018. – С. 220-225. *Особистий внесок здобувача: здійснено порівняльний аналіз найпоширеніших систем типізованого лямбда-числення і встановлено відповідність між формальними виразами та конструкціями мови Gallina.*

Наукові праці, які додатково відображають наукові результати дисертації:

8. Matsiuk O. The procedures of processing of geolocation data on urban underground spaces / O. Matsiuk, N. Kunanets, V. Pasichnyk, **V. Lenko**, Y. Shcherbyna, A. Rzheuskyi // *Proceedings of the 2019 9th International Conference on Advanced Computer Information Technologies (ACIT).* – IEEE, 2019. – С. 500-503. *Особистий внесок здобувача: розроблено програмне забезпечення для здійснення сегментації радарограм*.

Наукові праці, які засвідчують апробацію матеріалів дисертації:

9. **Ленько В. С.** Теоретичні аспекти логічного міркування у середовищі Coq / В. С. Ленько, В. В. Пасічник, Н. Е. Кунанець, Ю. М. Щербина // *Інтелектуальні системи прийняття рішень і проблеми обчислювального інтелекту: матеріали міжнародної наукової конференції ISDMCI-2018*, 21-27 трав. 2018 р., Залізний Порт, Україна / Херсонський нац. технічний ун-т. – Херсон, 2018. – С. 174-176. *Особистий внесок здобувача: досліджено структуру дедуктивної системи числення конструкцій.*
10. **Ленько В. С.** Проектування відкритих децентралізованих реєстрів з використанням технології Blockchain / В. С. Ленько, В. В. Пасічник, Н. Е. Кунанець, Ю. М. Щербина // *Математика. Інформаційні технології. Освіта: тези доповідей VII міжнар. наук.-практ. конф. МІТО-2018*, 3-5 черв. 2018 р., Луцьк-Світязь, Україна / Східноєвропейський нац. ун-т ім. Л. Українки. – Луцьк-Світязь, 2018. – С. 72-74. *Особистий внесок здобувача: запропоновано основні елементи системи розподілених реєстрів, що ґрунтуються на технології Blockchain.*

11. **Ленько В. С.** Міркування в онтологіях з використанням теорії типів / В. С. Ленько, В. В. Пасічник, Ю. М. Щербина, Н. Е. Кунанець // *Теоретичні та прикладні аспекти побудови програмних систем: матеріали XIV міжнар. наук. конф. TAAPSD'2017*, 4-8 груд. 2017 р., Київ, Україна / Київський нац. ун-т ім. Т. Г. Шевченка. – Київ, 2017. – С. 138-141. *Особистий внесок здобувача: запропоновано метод подання елементів онтології функціональною мовою програмування в середовищі Coq.*
12. **Ленько В. С.** Децентралізована комунікація у віртуальних наукових колективах / В. С. Ленько, Н. Е. Кунанець, В. В. Пасічник, Ю. М. Щербина // *Практичне застосування нелінійних динамічних систем в інфокомунікаціях: матеріали VI міжнар. наук.-практ. конф.*, 9-11 лист. 2017 р., Чернівці, Україна / Чернівецький нац. ун-т ім. Ю. Федьковича. – Чернівці, 2017. – С. 118-119. *Особистий внесок здобувача: запропоновано сучасний підхід до побудови платформ комунікації у віртуальних наукових колективах, який ґрунтується на ідеях децентралізації та криптобезпеки.*
13. Кунанець Н. Е. Подання онтологічних моделей знань в системі інтерактивного доведення теорем Coq / Н. Е. Кунанець, **В. С. Ленько**, В. В. Пасічник, Ю. М. Щербина // *Інформаційні технології та взаємодії: матеріали доповідей IV міжнар. наук.-практ. конф. IT&I-2017*, 8-10 лист. 2017 р., Київ, Україна / Київський нац. ун-т ім. Т. Шевченка. – Київ, 2017. – С. 169-170. *Особистий внесок здобувача: запропоновано правила для формалізації онтологій в середовищі Coq.*
14. Кунанець Н. Е. Подання онтологій з використанням теорії типів / Н. Е. Кунанець, **В. С. Ленько**, В. В. Пасічник, Ю. М. Щербина // *Системи та засоби штучного інтелекту: тези доповідей Міжнародної наукової молодіжної школи AIIS'2017*, 18 жовт. 2017 р., Київ, Україна / Київський нац. ун-т ім. Т. Шевченка. – Київ, 2017. – С. 114-117. *Особистий внесок здобувача: розроблено приклад подання онтології та логічного міркування над її елементами в асистенті доведення теорем Coq.*
15. **Ленько В. С.** Подання знань та логічні міркування у формальних онтологіях/ В. С. Ленько, В. В. Пасічник, Ю. М. Щербина // *Комп'ютерне моделювання та програмне забезпечення інформаційних систем і технологій. (КМПЗ-2017) : збірник наукових праць (тези доповідей і вибрані статті) III Всеукраїнської міжнародної науково-практичної конференції*, 28-30 вересня 2017 р., Рівне, Україна / Національний університет водного господарства та природокористування. – Рівне, 2017.

– С. 94-95. *Особистий внесок здобувача: запропоновано ідею використання формальних логічних систем для подання та міркування над онтологіями.*

16. **Ленько В.С.** Проект системи управління персональними знаннями / В. С. Ленько, В. В. Пасічник, Ю. М. Щербина // *Управління проектами: стан та перспективи. (ІТПРР-2017) : матеріали XIII міжнародної науково-практичної конференції*, 12-15 вересня 2017 р., Миколаїв, Україна / Національний університет кораблебудування імені адмірала Макарова. – Миколаїв, 2017. – С. 61-62. *Особистий внесок здобувача: запропоновано проект системи управління персональними знаннями, що ґрунтується на поєднані традиційного, онтологічного та формального поданні знань.*
17. **Lenko V. S.** On personal knowledge management systems / V. S. Lenko, V. V. Pasichnyk, Y. M. Shcherbyna // *Математика. Інформаційні технології. Освіта: тези доповідей VI міжнар. наук.-практ. конф. МІТО-2017*, 5-7 черв. 2017 р., Луцьк-Світязь, Україна / Східноєвропейський нац. ун-т ім. Л. Українки. – Луцьк-Світязь, 2017. – С. 57-59. *Особистий внесок здобувача: запропоновано гібридний підхід до подання знання, з використанням формальних та концептуальних моделей.*
18. **Ленько В. С.** Побудова моделі «Розумного соціополісу» на основі знань експертів / В. С. Ленько, Н. Е. Кунанець, В. В. Пасічник, Ю. М. Щербина // *Інформаційні технології, економіка та право: стан та перспективи розвитку. (ІТЕП-2017) : матеріали міжнародної науково-практичної конференц*ії, 27-28 квітня 2017 р., Чернівці, Україна / Буковинський університет. – Чернівці, 2017. – С. 141-143. *Особистий внесок здобувача: розроблено програмне забезпечення для оцінки «розумності» соціополісів.*
19. **Lenko V. S.** Knowledge Representation Models / V. S. Lenko, V. V. Pasichnyk, Y. M. Shcherbyna // *Теоретичні та прикладні аспекти побудови програмних систем: матеріали XIII міжнар. наук. конф. TAAPSD'2016*, 5-9 груд. 2016 р., Київ, Україна / Київський нац. ун-т ім. Т. Шевченка, Ін-т післядипломної освіти. – Київ, 2016. – С. 143-152. *Особистий внесок здобувача: здійснено порівняльний аналіз властивостей найпоширеніших моделей подання знання.*
20. **Ленько В. С.** Методи та засоби управління персональними знаннями / В. С. Ленько, В. В. Пасічник, Ю. М. Щербина // *Інформаційно-обчислювальні технології, автоматика та електротехніка: матеріали Міжнар. наук.-практ. конф. ІТАЕ-2016,* 10-11 листоп. 2016 р., Рівне, Україна / Нац. ун-т водного господарства та природокористування. – Рівне, 2016. – С. 143-144.

Особистий внесок здобувача: проаналізовано основні підходи та приклади проектування персональних баз знань.

21. **Ленько В. С.** Відновлення графічних образів за допомогою карт Кохонена / В. С. Ленько, Ю. М. Щербина // *Сучасні проблеми прикладної математики та інформатики: збірник наукових праць конф. APAMCS-2016,* 5-7 жовт. 2016 р., Львів, Україна / Львівський нац. ун-т ім. І. Франка. – Львів, 2016. – С. 108-111. *Особистий внесок здобувача: покращено метод відновлення графічних образів на основі карт Кохонена*.
22. **Ленько В. С.** Застосування методів штучного інтелекту до сегментації графічних образів / В. С. Ленько // *Чотирнадцята всеукраїнська (дев'ята міжнародна) студентська наукова конференція з прикладної математики та інформатики СНКПМІ-2011: тези доповідей*, 5-6 трав. 2011 р., Львів, Україна / Львівський нац. ун-т ім. І. Франка. – Львів, 2011. – С. 60-61.

Додаток Б

Акти впровадження результатів дисертації

«ЗАТВЕРДЖУЮ»
Проректор
з наукової роботи
Національного університету
«Львівська політехніка»
_____ Демидов І.В.
«__» ____12____ 2020 р.

АКТ

про використання результатів дисертації Ленька Василя Степановича
«Методи та засоби управління персональними знаннями в інтелектуальних системах»
представленої на здобуття наукового ступеня доктора філософії при виконанні науково-дослідної роботи кафедри інформаційних систем та мереж Національного університету
«Львівська політехніка»

Комісія у складі начальника науково-дослідної частини д.т.н., ст.досл. Небесного Р.В., завідувача кафедри інформаційних систем та мереж д.т.н., проф. Литвина В.В., завідувачки відділу науково-організаційного супроводу наукових досліджень к.т.н. Лазько Г.В. та начальниці планово-фінансової групи Чулой Т.М. підтверджують цим актом, що результати дисертації Ленька В.С. «Методи та засоби управління персональними знаннями в інтелектуальних системах» використано при виконанні державної науково-дослідної роботи «Методи та засоби функціонування систем підтримки прийняття рішень на основі онтологій», номер державної реєстрації 0118U000269.

В результаті досліджень, виконаних Ленько В.С.

- здійснено порівняльний аналіз моделей подання знання, описової логіки та формальних систем теорії типів, а також особливостей їх застосування для подання знань різної природи, зокрема описових речень природньої мови, логічних і математичних теорій та виразів;
- удосконалено методологію вибору формальних систем для подання та міркування над знанням, яка ґрунтується на системному аналізі ознак виразності, розв'язності, семантики та задач міркування;
- розроблено онтологічно-орієнтований метод подання бази знань в середовищі асистента доведення теорем Coq, що сприяє формалізації нових та існуючих онтологій.

Голова комісії
Начальник НДЧ, д.т.н., ст.досл. Небесний Р.В.

Члени комісії:
Завідувач кафедри інформаційних
систем та мереж, д.т.н., проф. Литвин В.В.

Завідувачка відділу науково-організаційного
супроводу наукових досліджень, к.т.н. Лазько Г.В.

Заступник начальника планово-фінансового відділу Чулой Т.М.

«ЗАТВЕРДЖУЮ»
Проректор
з наукової роботи
Національного університету
«Львівська політехніка»
_____ Демидов І.В.
12 2020 р.

АКТ

про використання результатів дисертації Ленька Василя Степановича
«Методи та засоби управління персональними знаннями в інтелектуальних системах»
представленої на здобуття наукового ступеня доктора філософії при виконанні науково-дослідної роботи кафедри інформаційних систем та мереж Національного університету «Львівська політехніка»

Комісія у складі начальника науково-дослідної частини д.т.н., ст.досл. Небесного Р.В., завідувача кафедри інформаційних систем та мереж д.т.н., проф. Литвина В.В., завідувачки відділу науково-організаційного супроводу наукових досліджень к.т.н. Лазько Г.В. та начальниці планово-фінансової групи Чулой Т.М. підтверджують цим актом, що результати дисертації Ленька В.С. «Методи та засоби управління персональними знаннями в інтелектуальних системах» використано при виконанні державної науково-дослідної роботи «Система підтримки прийняття рішень розпізнавання мультиспектральних образів на основі технологій машинного навчання та онтологічного підходу», номер державної реєстрації 0120U102203.

В результаті досліджень, виконаних Ленько В.С.

- розвинуто практику використання теорії типів та конструктивної логіки для формального подання описового знання та обчислення логічного висновку в середовищі асистента доведення теорем Coq;
- удосконалено методи та засоби подання, управління, зберігання та поширення знання в межах систем персональних та колективних баз знань, з використанням розподілених систем;
- розроблено приклади формального доведення істинності описових знань, зокрема для задач проектного управління в ІТ-підприємстві, узгодженості новинних матеріалів в мас-медіа та структурних моделей шляхопроводів, що полегшує та пришвидшує перевірку гіпотез в напівавтоматичному режимі.

Голова комісії
Начальник НДЧ, д.т.н., ст.досл. Небесний Р.В.

Члени комісії:
Завідувач кафедри інформаційних
систем та мереж, д.т.н., проф. Литвин В.В.

Завідувачка відділу науково-організаційного
супроводу наукових досліджень, к.т.н. Лазько Г.В.

Заступник начальника планово-фінансового відділу Чулой Т.М.

ТОВАРИСТВО З ОБМЕЖЕНОЮ ВІДПОВІДАЛЬНІСТЮ «БОТСКРЮ»

ЄДРПОУ 41223363

79024, Львівська обл., м. Львів, вул. Богдана Хмельницького, 223В, оф. 301/02

тел. +380937460992, email: info@botscrew.com

Вих. №5/05/20

«20» травня 2020р.

АКТ

про впровадження результатів дисертації
Ленька Василя Степановича «Методи та засоби управління
персональними знаннями в інтелектуальних системах» представленої
на здобуття наукового ступеня доктора філософії
за спеціальністю 124 «Системний аналіз»

Цим актом засвідчуємо, що результати дисертації Ленька Василя Степановича «Методи та засоби управління персональними знаннями в інтелектуальних системах» апробовано та впроваджено в ІТ-компанії ТзОВ «Ботскрю» для розробки інтелектуальних автоматизованих систем підтримки діалогу з користувачами різних сервісів. Корисність впроваджених результатів полягає в покращенні якості аналізу тверджень користувача, яка зумовлена наступними факторами:

- виокремлення компоненти, що використовує задані і накопичені знання, та її реорганізація в загальну і спеціалізовану онтології, забезпечили модульність знання і прогнозованість результату ведення діалогу з користувачами обраної предметної області;
- універсальність загальної онтології передбачає її широке застосування як формальної основи для спеціалізованих онтологій, тому її логічна угодженість є важливою характеристикою при забезпеченні якості ведення діалогу. Для подання загальної онтології використано формалізм описових логік, якому властиві механізми автоматизованого підтвердження узгодженості бази знань;
- подання елементів речень природньої здійснено за допомогою формалізму числення індуктивних конструкцій, який попри свою нерозв'язність володіє високою виразністю та можливістю міркування над власними твердженнями в напівавтоматичному режимі;
- системний аналіз різновидів теорій типів і їх порівняльний аналіз з описовими логіками дозволив обгрунтовано вибрати технології для подання загальної та спеціалізованої онтологій.

Директор ТЗОВ «Ботскрю» Пилипчак О.Я.

Perfectial
EMPOWER YOUR IDEAS

Товариство з обмеженою
відповідальністю "Перфектіал"
КОД ЄДРПОУ 37693442

Україна, м. Львів,
вул. Яворницького,
3-Б/55, 79054
finance@perfectial.com
+38 032 270 00 10

АКТ

про використання результатів дисертації
Ленька Василя Степановича «Методи та засоби управління
персональними знаннями в інтелектуальних системах» представленої
на здобуття наукового ступеня доктора філософії

Цим актом підтверджую використання результатів дисертації Ленька Василя Степановича «Методи та засоби управління персональними знаннями в інтелектуальних системах», зокрема:

- онтологічно-орієнтований метод подання знання використано для побудови бази знань предметної області «Проектне управління», існування якої сприяє виявленню логічних неузгодженостей в процесах та об'єктах проектного управління;
- порівняльний аналіз описових логік та теорій типів використано при виборі формальної мови та інструменту для подання і дедуктивного міркування над знанням предметної області «Технологічна структура проекту»;
- елементи архітектури системи децентралізованого управління загальними та персональними базами знань використано при розробці програмних модулів безпеки, що ґрунтуються на технології Blockchain;
- системний аналіз методів та засобів логічно-обґрунтованого програмування доречно розширила профіль експертизи установи і сприяє її позиціонуванню як високотехнологічного розробника програмного забезпечення.

Директор ТОВ «Перфектіал» _____ Скоропад О.Б.

sombra
software development

Sombra Inc.
Stryiska St, 108,
Lviv, 79000, Ukraine

phone: +38 097 275 00 55
info@sombrainc.com
www.sombrainc.com

АКТ

про впровадження результатів дисертації
Ленька Василя Степановича «Методи та засоби управління
персональними знаннями в інтелектуальних системах» представленої
на здобуття наукового ступеня доктора філософії
за спеціальністю 124 «Системний аналіз»

Основні результати дисертації Ленька Василя Степановича «Методи та засоби управління персональними знаннями в інтелектуальних системах» впроваджено в технологічну методологію розробки програмного забезпечення та інформаційних систем, що засновані на знаннях, ТзОВ «Сомбра Софтвер». Логічні моделі подання та міркування над знанням зарекомендували себе найкращим чином в системах, для яких ціна помилки є вельми високою, наприклад для побудови програмних модулів безпеки. Функціональне, формально строге програмування глибоких специфікацій забезпечило концептуально новий рівень коректності реалізації складних проектів і створило можливостві надійно перевіряти гіпотези про властивості інформаційних систем.

Порівняльний аналіз логічних систем міркування, зокрема описових логік та теорій типів, пропонує перелік важливих ознак, які використовуються в компанії при виборі технологій для створення баз знань. Онтологічно-орієнтований метод подання знання застосовано для проектування баз знань різних предметних областей. Результати системного аналізу, що здійснений в межах дисертаційного дослідження Ленька В.С., створюють надійне підгрунтя для прийняття важливих технологічних рішень при побудові сучасних інформаційних систем.

Директор
ТзОВ «Сомбра Софтвер» Мякишинов С.Ю.