НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ

КАФЕДРА СИСТЕМНОГО ПРОГРАМУВАННЯ І
СПЕЦІАЛІЗОВАНИХ КОМП'ЮТЕРНИХ СИСТЕМ

«На правах рукопису»
УДК 004.62

«До захисту допущено»

Завідувач кафедри СПСКС

_____ В.П.Тарасенко_
(підпис)

"___"_____2018р.

Магістерська дисертація

на здобуття ступеня магістра

зі спеціальності 123 Комп'ютерна інженерія
(Комп'ютерні системи та компоненти)

на тему: Способи обчислення міри неконсистентностей OWL онтологій

Виконала: студентка VI курсу, групи _КВ-61м

Дзицюк Євгенія Володимирівна _____ _____
(підпис)

Науковий керівник доцент каф. СПіСКС, к.т.н., доцент Щербина О.А. _____
(підпис)

Рецензент професор каф. ОТ, д.т.н., проф. _____ _____
(підпис)

Засвідчую, що у цій магістерській дисертації немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ – 2018 року

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Факультет прикладної математики

Кафедра системного програмування і спеціалізованих комп'ютерних систем

Рівень вищої освіти – другий (магістерський)
Спеціальність 123 Комп'ютерна інженерія
(Комп'ютерні системи та компоненти)

ЗАТВЕРДЖУЮ
Завідувач кафедри СПСКС
_____ _В.П.Тарасенко__
(підпис)
«___»_____2018р.

ЗАВДАННЯ
на магістерську дисертацію студенту
Дзицюк Євгенії Володимирівні

1. Тема дисертації Способи обчислення міри неконсистентностей OWL онтологій, науковий керівник дисертації доцент кафедри СПіСКС, к.т.н., доцент Щербина О.А.,
затверджені наказом по університету від «22» березня 2018 р. №986-с
2. Термін подання студентом дисертації 11 травня 2018 р._____
3. Об'єкт дослідження онтологічні системи, некосистентність при побудові онтологій
4. Предмет дослідження способи обчислення міри неконсистентності OWL онтологій
5. Перелік завдань, які потрібно розробити
   - проаналізувати способи обчислення міри неконсистентностей OWL онтологій та визначити недоліки існуючих способів;
   - адаптувати способи обчислення міри неконсистентності онтологій в описовій логіці до OWL онтологій;
   - запропонувати модифікований спосіб обчислення міри неконсистентностей онтологій, який призводив би до зменшення часу обробки OWL онтологій.
6. Перелік ілюстративного матеріалу
   - Класифікація методів обчислення міри невідповідності онтологій.

- Блок схема розробленого способу обчислення міри невідповідності онтології.
- Порівняння часу роботи існуючих та розробленого способів обчислення міри невідповідності та розміру онтологій.
- Порівняння часу роботи та значень MI-Шеплі та D_f-міри невідповідності.
- Описова статистика тестових випадків для способів вимірювання невідповідності.
- Значення MI-Шеплі та D_f-міри неконсистентності для тестових випадків.

7. Перелік публікацій
   - X наукова конференція магістрантів та аспірантів «Прикладна математика та комп'ютинг» ПМК-2018 (Київ, 21-23 березня 2018 р.);
   - VIII наукова конференція магістрантів та аспірантів «Прикладна математика та комп'ютинг» ПМК-2016 (Київ, 20-22 квітня 2018 р.);
   - Міжнародний науковий журнал "Інтернаука". – 2018. – №3.
8. Дата видачі завдання 5 вересня 2016 р.

Календарний план

№ з/п	Назва етапів виконання магістерської дисертації	Термін виконання етапів магістерської дисертації	Примітка
	Вивчення літератури за тематикою проекту	12.10.2016	
	Розроблення та узгодження технічного завдання	12.12.2016	
	Аналіз існуючих рішень	10.01.2017	
	Підготовка матеріалів першого розділу магістерської дисертації	12.02.2017	
	Підготовка матеріалів другого розділу магістерської дисертації	17.04.2017	
	Підготовка матеріалів третього розділу магістерської дисертації	23.09.2017	
	Підготовка матеріалів четвертого розділу магістерської дисертації	12.12.2018	
	Підготовка графічної частини дипломного проекту	17.02.2018	
	Оформлення документації дипломного проекту	20.04.2018	
	Попередній розгляд магістерської дисертації на кафедрі	26.04.2018	

Студент _____ Є. В. Дзицюк
(підпис)

Науковий керівник дисертації _____ О. А. Щербина
(підпис)

РЕФЕРАТ

Актуальність теми. Розвиток інформаційно-телекомунікаційних технологій сприяє збільшенню обсягів інформації, необхідної для роботи корпоративних систем. Тому на сьогодні існує проблема ефективної обробки даних. Одним із варіантів рішення цієї задачі є обробка даних в системах з використанням онтологій. Онтологія — формалізоване представлення знань про певну предметну область, придатне для автоматизованої обробки. Таким чином дані охоплюють менший об'єм пам'яті, а інформації з них можна отримати більше. Розмір онтологій невпинно зростає, тому неконсистентність або внутрішнє протиріччя онтології в таких випадках є звичним явищем. Для обробки та аналізу таких онтологій необхідно застосовувати способи обчислення міри неконсистентності, які і будуть розглянуті в даній дисертаційній роботі.

Об’єктом дослідження є онтологічні системи, некосистентність при побудові онтологій.

Предметом дослідження є способи обчислення міри неконсистентності OWL онтологій.

Методи дослідження – методи математичної статистики для аналізу обчислення міри некосистентності OWL онтологій.

Мета роботи: підвищення ефективності обробки неконсистентних онтологій шляхом застосування обчислення міри невідновідності; адаптація підходів до обчислення міри неконсистентності онтологій в описовій логіці до OWL онтологій; оптимізація способів обчислення міри неконсистентності онтологій задля зменшення часу їх виконання.

Наукова новизна одержаних результатів полягає в наступному:

1. Проаналізовано способи обчислення міри неконсистентностей OWL онтологій та показано, що недоліком існуючих способів є невелика кількість результуючих унікальних значень міри неконсистентності, що не

дозволяє зробити висновок щодо причин виникнення невідповідності. Показано, що існуючі способи обчислення міри неконсистентності онтологій в описовій логіці ефективно застосовувати і для широковикористовуваних OWL онтологій.

2. Запропоновано модифікований спосіб обчислення міри неконсистентностей онтологій, який відрізняється від існуючих наявністю обчислення значення Шеплі тільки для мінімально неконсистентних підмножин онтології, що призводить до зменшення часу обробки OWL онтологій.

Практична цінність одержаних в роботі результатів полягає в тому, що запропонований спосіб дозволяє ефективно, з точки зору часу виконання, отримати дані стосовно консистентності онтології, що допомагає розробнику онтології знайти невідповідність та виправити базу знань.

Апробація роботи. Основні положення і результати роботи були представлені та обговорювались на:

- X науковій конференції магістрантів та аспірантів «Прикладна математика та комп'ютинг» ПМК-2018 (Київ, 21-23 березня 2018 р.);

- VIII науковій конференції магістрантів та аспірантів «Прикладна математика та комп'ютинг» ПМК-2016 (Київ, 20-22 квітня 2018 р.).

Публікації. За результатами дослідження опубліковано 3 наукові праці, з них 1 стаття та 2 тези конференції.

Структура та обсяг роботи. Магістерська дисертація складається з вступу, чотирьох розділів та висновків.

У вступі подано загальну характеристику роботи, зроблено оцінку сучасного стану проблеми, обґрунтовано актуальність напрямку досліджень, сформульовано мету і задачі досліджень, показано наукову новизну отриманих результатів і практичну цінність роботи, наведено

відомості про апробацію результатів.

У першому розділі розглянуто теоретичні відомості по заданій темі, а також проведений аналіз, який дає змогу визначити основні переваги та недоліки існуючих способів обчислення міри неконсистентності онтологій.

У другому розділі наведено результати адаптації способів обчислення міри неконсистентності онтологій в описовій логіці до OWL онтологій.

У третьому розділі описана реалізація даних способів з використанням існуючих обробників онтологій.

У четвертому розділі надані експериментальні результати з обчислення міри неконсистентності онтологій.

У висновках представлені результати проведеної роботи.

Робота представлена на 90 аркушах, містить 30 рисунків, 2 таблиці та список використаних літературних джерел з 25 найменувань.

Ключові слова: описова логіка, OWL онтологія, неконсистентність, мінімально неконсистентні підмножини, метод Шеплі.

ABSTRACT

Actuality of subject. The development of information and telecommunication technologies contributes to the increase of the amount of information necessary for the work of corporate systems. Therefore, today there is a problem of efficient data processing. One of the solutions to this problem is the processing of data in systems using ontologies. Ontology is a formalized representation of knowledge about a particular subject area, suitable for automated processing. This way, the data covers a smaller amount of memory, and more information can be obtained from it. The size of the ontologies is constantly increasing, so the inconsistency or internal contradiction of ontology in such cases is a common occurrence. For the processing and analysis of such ontologies, it is necessary to use methods for calculating the degree of inconsistency, which will be considered in this thesis.

The object of the study is ontological systems, non-consistency in the construction of ontologies.

The subject of the study is how to calculate the degree of non-consistency of OWL ontologies.

Methods of research - methods of mathematical statistics for the analysis of the calculation of the degree of non-consistency of OWL ontologies.

The purpose of the work: to increase the efficiency of processing inconsistent ontologies by applying the calculation of the degree of non-compliance; adaptation of approaches to calculating the degree of inconsistency of ontologies in descriptive logic to OWL ontologies; optimization of methods for calculating the degree of inconsistency of ontologies to reduce the time of their implementation.

The scientific novelty of the results obtained is as follows:

1. The methods of calculating the degree of inconsistency of OWL ontologies are analyzed, and it is shown that the lack of existing methods is a small amount of resulting unique values of the degree of non-consistency, which does not allow us to conclude on the causes of the inconsistency. It is shown that

existing methods for calculating the degree of inconsistency of ontologies in descriptive logic can be effectively applied to widely used OWL ontologies.

2. A modified method for calculating the degree of non-consistency of ontologies is proposed, which differs from the existing with availability of the Shapley value calculation only for minimally incomplete subsets of ontology, which leads to a reduction in the processing time of OWL ontologies.

The practical value of the results obtained in the work is that the proposed method allows efficiently, in terms of the time of execution, to obtain data on the consistency of ontology, which helps the developer of ontology to find inconsistency and correct the knowledge base.

Work approbation. The main provisions and results of work were presented and discussed at:

- X scientific conference of masters and postgraduates "Applied Mathematics and Computer" PMK-2018 (Kiev, March 21-23, 2018);

- VII International Conference of Masters and Postgraduate Students "Applied Mathematics and Computing" PMK-2016 (Kyiv, April 20-22, 2018).

Publications According to the results of the study, 3 scientific works were published, including 1 article and 2 theses of the conference.

Structure and scope of work. The master's thesis consists of an introduction, four chapters and conclusions.

The introduction gives a general description of the work, assesses the current state of the problem, substantiates the relevance of the research direction, formulates the purpose and objectives of the research, shows the scientific novelty of the results obtained and the practical value of the work, provides information on the approbation of the results.

In the first section the theoretical information on the given topic is considered, as well as the analysis, which allows to determine the main advantages and disadvantages of the existing methods of calculating the degree of inconsistency of ontologies.

The second section presents the results of the adaptation of methods for

calculating the degree of inconsistency of ontologies in descriptive logic to OWL ontologies.

The third section describes the implementation of these methods using existing ontology handlers.

The fourth section provides experimental results for calculating the degree of non-consistency of ontologies.

The conclusions are the results of the work.

The work is presented on 90 Sheets, contains 30 pictures, 2 tables and list of used literary sources with 25 names.

Keywords: descriptive logic, OWL ontology, inconsistency, minimally incomplete subsets, Shapley's method.

РЕФЕРАТ

Актуальность темы. Развитие информационно-телекоммуникационных технологий способствует увеличению объемов информации, необходимой для работы корпоративных систем. Поэтому на сегодняшний день существует проблема эффективной обработки данных. Одним из вариантов решения этой задачи является обработка данных в системах с использованием онтологий. Онтология - формализованное представление знаний об определенной предметной области, пригодное для автоматизированной обработки. Таким образом данные охватывают меньший объем памяти, а информации по ним можно больше. Размер онтологий постоянно растет, поэтому неконсистентнисть или внутреннее противоречие онтологии в таких случаях является обычным явлением. Для обработки и анализа таких онтологий необходимо применять способы вычисления степени неконсистентности, которые и будут рассмотрены в данной диссертационной работе.

Объектом исследования является онтологические системы, некосистентнисть при построении онтологий.

Предметом исследования являются способы вычисления степени неконсистентности OWL онтологий.

Методы исследования - методы математической статистики для анализа вычисления меры некосистентности OWL онтологий.

Цель работы: повышение эффективности обработки неконсистентних онтологий путем применения вычисления меры несоответствия; адаптация подходов к вычислению степени неконсистентности онтологий в описательной логике в OWL онтологии; оптимизация способов вычисления меры неконсистентности онтологий для уменьшения времени их выполнения.

Научная новизна исследования заключается в следующем:

1. Проанализированы способы вычисления степени неконсистентностей OWL онтологий и показано, что недостатком

существующих способов является небольшое количество результирующих уникальных значений меры неконсистентности, что не позволяет сделать вывод о причинах возникновения несоответствия. Показано, что существующие способы вычисления степени неконсистентности онтологий в описательной логике эффективно применять и для широко используемых OWL онтологий.

2. Предложен модифицированный способ вычисления степени неконсистентностей онтологий, который отличается от существующих наличием вычисления значения Шепли только для минимально неконсистентних подмножеств онтологии, что приводит к уменьшению времени обработки OWL онтологий.

Практическая ценность полученных в работе результатов заключается в том, что предложенный способ позволяет эффективно, с точки зрения времени выполнения, получить данные о консистентности онтологии, помогает разработчику онтологии найти несоответствие и исправить базу знаь.

Апробация работы. Основные положения и результаты работы были представлены и обсуждались на:

- X научной конференции магистрантов и аспирантов «Прикладная математика и компьютинг» ПМК-2018 (Киев, 21-23 марта 2018)

- VIII научной конференции магистрантов и аспирантов «Прикладная математика и компьютинг» ПМК-2016 (Киев, 20-22 апреля 2016).

Публикации. По результатам исследования опубликовано 3 научные работы, из них 1 статья и 2 тезисы конференции.

Структура и объем работы. Магистерская диссертация состоит из введения, четырех глав и выводов.

Во введении представлена общая характеристика работы, произведена оценка современного состояния проблемы, обоснована актуальность направления исследований, сформулированы цели и задачи исследований, показано научную новизну полученных результатов и практическую

ценность работы, приведены сведения об апробации результатов.

В первом разделе рассмотрены теоретические сведения по заданной теме, а также проведен анализ, который позволяет определить основные преимущества и недостатки существующих способов вычисления меры неконсистентности онтологий.

Во втором разделе приведены результаты адаптации способов вычисления меры неконсистентности онтологий в описательной логике в OWL онтологий.

В третьем разделе описана реализация данных способов с использованием существующих обработчиков онтологий.

В четвертом разделе предоставлены экспериментальные результаты по исчислению меры неконсистентности онтологий.

В выводах представлены результаты проведенной работы.

Работа представлена на 90 листах, содержит 30 рисунков, 2 таблицы и список использованных литературных источников из 25 наименований.

Ключевые слова: описательная логика, OWL онтология, неконсистентнисть, минимально неконсистентные подмножества, метод Шепли.

Зміст

ВСТУП

Розвиток інформаційно-телекомунікаційних технологій сприяє збільшенню обсягів інформації, необхідної для роботи корпоративних систем. Тому проблема ефективної обробки та зберігання даних на сьогодні є актуальною. Одним із рішень даної задачі є використання онтологій. Онтологія — формалізоване представлення знань про певну предметну область, придатне для автоматизованої обробки. Онтології використовуються в процесі програмування як форма подання знань про реальний світ або його частини. Основні сфери застосування - моделювання бізнес-процесів, семантичний Web (Semantic Web), штучний інтелект. Онтології відіграють важливу роль у Web просторі, оскільки вони представляють загальну розподілену модель, яка являє собою домен та зв'язки об'єктів в домені. Оскільки розмір онтологій зростає, а додатки, які їх використовують, стають більш складними, неконсистентність є неминучим явищем в розробці онтологій. Неконсистентність онтології – її внутрішнє протиріччя, невідповідність аксіом, з яких складається онтологія. Для обробки та аналізу таких онтологій необхідно застосовувати методи обчислення міри неконсистентності, які і будуть розглянуті в даній дисертаційній роботі.

Питаннями обробки неконсистентних онтологій займалися вчені з Франції Meghyn Bienvenu, Camille Bourgaux, Francois Goasdoue; Італії - Diego Calvanese1, Giuseppe De Giacomo, Domenico Lembo, Maurizio Lenzerini, Riccardo Rosati.

Проте, у роботах всіх перелічених авторів відсутній комплексний підхід до аналізу неконсистентностей в OWL онтологіях. До того ж недоліком цих підходів є те, що в них вважається, що всі невідповідності є однаково незадовільними. Однак, як показано в реальних додатках, онтологія може містити різні за значущістю джерела неконсистентностей, що можна визначити за допомогою кількісної міри невідповідності.

РОЗДІЛ 1. АНАЛІЗ ІСНУЮЧИХ РІШЕНЬ ТА ОБҐРУНТУВАННЯ ТЕМИ МАГІСТЕРСЬКОЇ ДИСЕРТАЦІЇ

1.1 Огляд описових логік

Описові логіки (DL) як сімейства формального представлення знань зазвичай використовуються в штучному інтелекті та в багатьох інших напрямках, наприклад біомедичній інформатиці. Зібрана інформація для представлення знань про домен називається онтологією. Онтологія часто грає важливу роль у семантичній мережі як розширення мережі для полегшення обміну даними через стандарти, визначені консорціумом World Wide Web (W3C). Оскільки онтологія в реальних додатках зазвичай складається більше ніж з одної аксіоми, її невідповідність можлива. Неконсистентність онтології може спотворити інформацію, отриману для певних цілей. Крім того, висока ступінь невідповідність може бути серйозною проблемою в певній області, яка потребує вирішення.

На сьогодні часто для онтологій невідомо кількість неконсистентостей, які вона містить. Ручна перевірка всіх аксіом в онтології для визначення ступеню невідповідності є задачею, що потребує багато часу та не виключає можливість людської помилки. Тому розробляються системи для виміру ступеню неконсистентності та надання рекомендацій щодо усунення виявлених невідповідностей.

Обчислення невідповідності було здебільшого побудоване для пропозиційної логіки [1, 2, 3], яке складається з деяких різних вимірів. Є деякі обчислення були розроблені для параконсистентної логіки [4], яка визначає семантику (описує зв'язок між символічним уявленням та сутностями реального світу) логічних систем для обробки протирічь.

Один з їх заходів представлений шляхом розгляду Квазікласичної семантики (Q-семантика [7]) як форми параконсистентної логіки, яка має більш виразну семантику, ніж звичайні чотиризначні логіки [8], які визнають чотири значення правди, що вказують на правду, неправдивість, суперечність

та неповноту. Після того Matthias Thimm[9] провів дослідження щодо підходів до вимірювання невідповідності, в якому містяться деякі інші різновиди обчислень для пропозиційної логіки.

У відповідних літературних джерелах можна знайти існуючі методи обчислення непослідовностей для описових логік. Yue Ma [12] пропонує міру несумісності у параконсистентній семантиці. Вона визначена для описових логік, а саме ALC, з використанням чотиризначної семантики. Yue Ma та Pascal Hitzler [13] також пропонують інший метод обчислення, що базується на відстані, який можна використовувати для будь-якої описової логіки. Обчислюється певна дистанція, яка показує, наскільки аксіома далеко знаходиться від того, щою бути консистентною. Xi Deng [14] пропонує міри непослідовності на основі значення Шеплі, де кожен розрахунок суперечливості набору вхідних даних базується на радикальних значеннях невідповідності: значення 0 для послідовного набору, а 1 – для непослідовного. Цей підхід є невизначеним для певного фрагмента описової логіки. Між тим, Liping Zhou [15] пропонує міру неконсистентності для сімейства описових логік DL-Lite. Ця міра використовує тризначну семантику [16], яка визначає значення істини як і чотирьохзначна семантика, не беручи до уваги неповноту.

Обробка невідповідності відіграє важливу роль у OWL (мова веб-онтології)[17]. Предметом даної роботи є розгляд наступних питань:

1. Які існують методи обчислення неконсистентності для описової логіки, які можна використати і для онтологій OWL?

2. Опис та аналіз прийнятих методів обчислення невідповідностей.

3. Розробка та тестування методів обчислення неконсистентності для описової логіки.

В даній роботі представлено вирішення вищезазначених проблем, використовуючи огляд літератури та експериментальні методи. Були зібрані та проаналізовані існуючі методи обчислення невідповідностей для описової логіки, які можна перенести на онтології OWL. Для того, щоб описати відомі методи обчислення неконсистентності, представлені приклади онтологій. Для

отримання експериментальних результатів було розроблене програмне забезпечення для обчислення неконсистентностей. Для тестування використовуємо онтології OWL в якості вхідних параметрів і визначаємо міру невідповідності.

1.2 Описова логіка SROIQ

У даному дослідженні використовується описова логіка (DL) SROIQ та OWL 2. Розділ 2 починається з основних визначень та понять DL , а саме SROIQ. Далі окреслюється OWL 2 та вводиться поняття міри невідповідності.

Визначення 1. Нехай *Sign* = (*NI*, *NC*, *NR*) – деяке фіксоване визначення описової логіки. Воно складається з *NI* як сукупності окремих імен, *NC* як сукупності імен концепцій, і *NR* як набору імен ролей.

Кожна DL надає набір описів понять для представлення бази знань. Опис концепції рекурсивно отримується з множини концептів (*NC*) і множини ролей (*NR*) з конструкторами.

C та *D* позначають описи концептів (доменів), *A* та *B*, як правило, використовуються для імен концептів, ролі позначають як *r* та *s*. Якщо ці визначення (Sign) об'єднані, їх називають складеними.

Для формалізації описів концепів в DL можна використовувати визначений синтаксис. Це передбачає явне символічне зображення бази знань. Є багато різновидів описових логік, кожна з яких представляє окремий синтаксиси. Наприклад, DL ALC, найчастіше використовувана складається з домену ($\top$), пустої множини ($\bot$), заперечення рівня концепту ($\neg C$), атомарного заперечення ($\neg A$), диз'юнкції ($C \sqcup D$), кон'юнкції ($C \sqcap D$), індивідів, у яких всі послідовності належать інтерпретації домену ($\exists r.C$) та індивідів, у яких хоча б одна послідовності належить інтерпретації домену ($\forall r.C$).

В даній роботі всі визначення та поняття обмежені для DL SROIQ, яка є основною описовою логікою, яка використовується в роботі. На сьогоднішній день більшість найбільш використовуваних логік є, по суті, діалектами DL SROIQ[18]. Щоб мати змогу сформувати складені ролі, DL SROIQ допускає

деякі різновиди конструкторів ролей. Найважливішим є зворотна роль. Наприклад, у відносинах між вчителем та учнем: якщо A є вчителем D, то D є студентом A, і навпаки. Інверсія *studentOf* позначається $studentOf^{-}$. Роль може бути виражена еквівалентною формою: $teacherOf \equiv studentOf^{-}$

В DL SROIQ також є універсальна роль (U). Цей тип ролі завжди стосується всіх пар окремих індивідів. Інтуїтивно, аналогією для цієї ролі є функції є ⊤ – весь домен, де універсальна роль може означати "все". Фактично ця функція рідко використовується на практиці.

Визначення 2. Ланцюжок ролей – послідовність ролей $\rho = r_1, \ldots, r_n$, $n \geq 0$, де $r_i \neq U$, $1 \leq i \leq n$.

Ще одна особливість DL SROIQ – ланцюжок ролей. Іноді позначається як $r \circ s$, де $r, s \in N_R$ і $\circ$ – це композиція бінарного співвідношення (наприклад, кон'юнкція або диз'юнкція). На основі цих функцій, множина ролей в SROIQ r визначається наступною граматикою:

$$r ::= N_R | N_R^{-} | U$$

Визначення 3. Нехай N_C представляє собою множину імен концепцій, N_R – множина імен ролей, а N_I – множина індивідів. Множину SROIQ аксіом можна побудувати, використовуючи наступні структури:

$$C,D ::= N_C | \top | \bot | \neg C | C \sqcup D | C \sqcap D | \exists r.C | \forall r.C | \leq nr.C | \geq nr.C | \exists r.Self | \{N_I\}$$

DL представляє базу знань через TBox (термінологічне сховище), ABox (сховище тверджень) і RBox (база ролей).

Визначення 4. Нехай C та D – концепти; r,s – ролі; $a,b \in N_I$; ρ – ланцюжок ролей; *Disj* – диз'юнктність в DL SROIQ. Аксіома в концепті може бути аксіомою рівності $C \equiv D$ або аксіомою включення $C \sqsubseteq D$. Аксіома рівності $C \equiv D$ еквівалентна двом аксіомам в домені $C \sqsubseteq D$ та $D \sqsubseteq C$. Крім того існує стверждувальна аксіома $C(a)$, $r(a,b)$, $a \approx b$, a 2249 b. Аксіома включення ролі (Role Inclusion Axiom, RIA) позначається як $r \sqsubseteq s$, а аксіома еквівалентності ролі – $r \equiv s$. Аксіома ланцюжка ролей (складена RIA) – $\rho \sqsubseteq r$; аксіома диз'юнкції – $Disj(r,s)$.

TBox побудований за допомогою множини концептуальних аксіом, які описують логічне твердження через поняття, ролі та їх зв'язки (рисунок 1.1). ABox побудований за допомогою множини стверджувальних аксіом. В RBox використовується множина аксіом ролей (RIAs, аксіоми еквівалентності, складені RIA та аксіоми диз'юнкції).

Визначення 5. Нехай $K = (T,A,R)$ – база знань. База знань K – це триплет T,A,R, де T – Tbox, A – Abox, R – Rbox. Нехай K – множина всіх баз знань.

Як зазначено в Визначенні 4, DL SROIQ забезпечує незмінність аксіоми ролей, наприклад *Disj*(*teacherOf*; *studentOf*) означає, що ніхто не може бути як вчителем, так і учнем. До того ж аксіоми RBox включають такі характеристики ролей:

1. Транзитивність: $teacherOf \circ teacherOf \sqsubseteq teacherOf$;
2. Симетричність: $teacherOf \equiv teacherOf^{-}$;
3. Асиметричність: *Disj*(*teacherOf*, $teacherOf^{-}$);
4. Рефлексивність: 22A4 $\sqsubseteq \exists$ *teach*:*Self*;
5. Нерефлексивність: 22A4 $\sqsubseteq \exists \neg$*teach*:*Self*;

Приклад 1. Tbox зображений на рисунку 1.1.

Woman ≡ Person ⊓ Female

Man ≡ Person ⊓ ¬Woman

Teacher ≡ (Woman ⊔ Man) ⊓ ∃ teaches.Student

Student ≡ (Woman ⊔ Man) ⊓ ∃ studies.Material

GoodStudent ≡ Student ⊓ ∃ passes.Exam

BadStudent ≡ ¬GoodStudent

Рисунок 1.1 - Tbox концептів та зв'зків між ними в базі знань університету

Можно помітити, що DL SROIQ – це DL ALC розширена зворотними ролями, транзитивнимиролями, Rbox.

Визначення 6. Інтерпретація $I = (\Delta^I, \cdot^I)$ складається з непустого домену Δ^I та відображення $\cdot^I$:

$A^I \subseteq \Delta^I$ для всіх $A \in N_C$;

$r^I \subseteq \Delta^I$ для всіх $r \in N_R$;

$a^I \subseteq \Delta^I$ для всіх $a \in N_I$.

Це означає, що концепти інтерпретуються як множини, ролі інтерпретуються як бінарні відношення, а індивіди трактуються як елементи. Інтерпретація I відповідає α якщо $I \vDash \alpha$. Інтерпретація I називається моделлю T,A,R ($I \vDash T, I \vDash A, I \vDash R$), якщо I задовільняє всім аксіомам T,A,R. Інтерпретація I є моделлю бази знань $K = (T,A,R)$, якщо I є моделлю T, I є моделлю A, I є моделлю R.

У DL інтерпретація важлива для опису значень для різних концептів. Сенмантика DL виражається за допомогою інтерпретації, яка складається з домену та відображення. Кожен конструктор має власну семантику в DL. Домен ($\top$) є одним з найпростіший з усіх конструкторів, оскільки він розглядає лише область інтерпретації (Δ^I) як семантику. Інший простий конструктор – пуста множина ($\bot$), яка семантично означає пустоту ($\emptyset$)

Відображення $\cdot^I$ інтерпретації поширюється на SROIQ, що визначено у Таблиці 1.1, – це зразок інтерпретацій, що ілюструє відображення деяких синтаксисів з області тварин і відносин між спорідненими тваринами.

1.3 База знань

База знань, БЗ (Knowledge base, KB) – це особливого роду база даних, розроблена для управління знаннями (метаданими), тобто збором, зберіганням, пошуком і видачею знань.

Залежно від рівня складності систем, в яких застосовуються бази знань, розрізняють:

- БЗ всесвітнього масштабу, наприклад: Інтернет;
- БЗ національні – наприклад, електронна енциклопедія;

- БЗ галузеві – на кшталт автомобільної енциклопедії;
- БЗ експертних систем;
- БЗ спеціалістів – наприклад, Гідрогеологічна.

Прості бази знань можуть використовуватися для зберігання даних про організації: документації, інструкцій, статей технічного забезпечення. Головна мета створення таких баз – допомогти менш досвідченішим користувачам знайти існуючий опис способу вирішення будь-якої проблеми предметної області.

Онтологія може служити для представлення в базі знань ієрархії понять і відношень між ними. Онтологія, яка містить і екземпляри об'єктів є базою знань.

База знань – важливий компонент інтелектуальної системи. Найвідоміший клас таких програм – експертні системи. Вони призначені для знаходження способу вирішення специфічних проблем, базуючись на записах БЗ і на користувацькому описі ситуації.

Нижче перераховані особливості, які можуть бути в інтелектуальній системі, і які стосуються баз знань.

- Машинне навчання: Це модифікація БЗ в процесі роботи інтелектуальної системи, адаптація до проблемної області. Аналогічна можливості людини «набиратися досвіду»;
- Автоматичне доведення (висновки): Здатність системи виводити нові знання із старих, знаходити закономірності в БЗ. Деякі автори вважають, що БЗ відрізняється від бази даних наявністю механізму висновків;
- Інтроспекція: Знаходження протиріч, нестиковок в БЗ, відслідковування правильної організації і коректності роботи БЗ;
- Доведення висновку: Здатність системи «пояснювати» хід її думок при знаходження вирішення задачі.

1.4 База знань в описовій логіці

Концепти описових логік цікаві не стільки самі по собі, скільки як інструмент для запису знань про описувану предметну області. Ці знання поділяються на загальні знання про поняття та їх взаємозв'язки (інтенсивні знання) і знання про індивідуальні об'єкти, їх властивості та зв'язки з іншими об'єктами (екстенсивні знання). Перші більш стабільні і постійні, тоді як другі більш схильні до модифікацій[25].

Відповідно до цього поділу, записувані за допомогою мови описових логік знання поділяються на

- Набір термінологічних аксіом або Tbox T;
- Набір тверджень про індивідів або Abox A.

Сукупність аксіом і тверджень разом складають так звану базу знань $K = T \vee A$. Далі окремо розглянемо види аксіом і тверджень, з яких може складатися TBox і Abox.

1.4.1 Аксіоми і TBox

Аксіомою вкладеності концептів називається вираз виду $C \subseteq \subset D$, а аксіомою еквівалентності концептів – вираз виду $C \equiv D$, де C і D – довільні концепти. Аналогічно, аксіомою вкладеності ролей називається вираз виду $R \subset \subseteq S$, а аксіомою еквівалентності ролей – вираз виду $R \equiv S$, де R і S – довільні ролі.

Термінологією або набором термінологічних аксіом або TBox (Terminological box) називається кінцевий набір аксіом перерахованих видів. Іноді аксіоми для ролей виділяються в окремий набір і називають його ієрархією ролей або RBox. Крім перерахованих видів аксіом, в термінології можуть допускатися й інші аксіоми (наприклад, транзитивність ролей).

Семантика термінології визначається природним чином. Нехай дана інтерпретація I. Аксіома $C \subseteq D$ виконується в інтерпретації I , якщо $C^I \subseteq D^I$. В цьому випадку також кажуть, що I є моделлю аксіоми $C \subseteq D$.

Аналогічно для інших видів аксіом. Термінологія T виконується в інтерпретації I, а інтерпретація I називається моделлю термінології T , якщо I

є моделлю всіх вхідних в *T* аксіом.

Приклад. Наступна сукупність є термінологією (або TBox) в мові логіки ALC:

Woman ≡ Person ∩ Female

Mother ≡ Woman ∩ ∃HasChild

Person ⊂ ∀HasChild.Person

Doctor ⊂ Person

Інтуїтивно ці аксіоми кажуть, що бути жінкою означає точно бути людиною і бути жіночої статі; бути матір'ю означає точно бути жінкою і мати дітей; у будь-якої людини будь-яка дитина є теж людина, кожен доктор є людиною. Перші дві аксіоми разом являють собою приклад так званої ацикличної термінології.

1.4.2 Твердження і ABox

Термінології дозволяють записувати загальні знання про концепції і ролі. Однак крім цього зазвичай потрібно також записати знання про конкретні індивіди: до якого класу (концепції) вони належать, якими відносинами (ролями) вони пов'язані один з одним. Це робиться в тій частині бази знань описової логіки, яка називається Abox (або набір тверджень про індивідів).

З цією метою, крім атомарних концептів і атомарних ролей, тобто імен для класів і відносин, вводиться також скінченна множина імен для індивідів. Твердження про індивідів бувають двох видів:

- Твердження про належність індивіда *a* концепту *C* записується як $C(a)$ або $a:C$;

- Твердження про зв'язки двох індивідів *a* і *b* роллю *R* записується як $R(a,b)$ або $(a,b):R$ або aRb.

Набором тверджень про індивідів або Abox (Assertional box) називається кінцевий набір тверджень цих двох видів.

Щоб задати семантику ABox, необхідно розширити інтерпретацію *I* а саме кожному імені індивіда a зіставити певний елемент домену $a^I \in \Delta^I$

Тоді кажуть, що твердження $C(a)$ або $R(a,b)$ виконуються в інтерпретації I якщо має місце $a^I \in C^I$ або $(a^I, b^I) \in R^I$, відповідно. Кажуть, що Abox виконується в інтерпретації I, а інтерпретація I є моделлю даного ABox, якщо всі його твердження виконуються в цій інтерпретації.

Приклад. Наступна сукупність є набором тверджень про індивідів (або ABox) в мові логіки ALC:

Mary: Woman ∩¬Doctor

Mary: ∃HasChild.Female

Mary hasChild Peter

Peter: Doctor ∩ ∀HasChild.⊥

Тут Mary і Peter є імена індивідів. Інтуїтивно ці твердження означають, що Mary є жінкою, але не доктором, у неї є дитина жіночої статі, Peter також є дитиною Mary, причому Peter є доктором і не має дітей.

Відмінність бази знань від бази даних полягає в наступному. Крім того, що бази знань формуються іншою мовою, ніж бази даних, їх головна відмінність полягає у використанні описової логіки при логічному виводі так званого припущення про відкритість світу, тоді як в базах даних приймається припущення про замкнутість світу. Останнє означає, що якщо деяке твердження не є істинним, то воно приймається хибним. Припущення ж про відкритість світу в цьому випадку вважає таке твердження ні істинним, ні хибним. Це кардинальним чином впливає на те, які факти вважаються логічно наступними із заданої бази знань, а значить, і на саме поняття логічного слідування в описовій логіці.

1.5 Мова OWL

OWL (Web Ontology Language) – мова опису онтологій для Web-простору. Мова OWL дозволяє описувати класи і відносини між ними, властиві для веб-документів і застосунків (Рисунок 1.2).

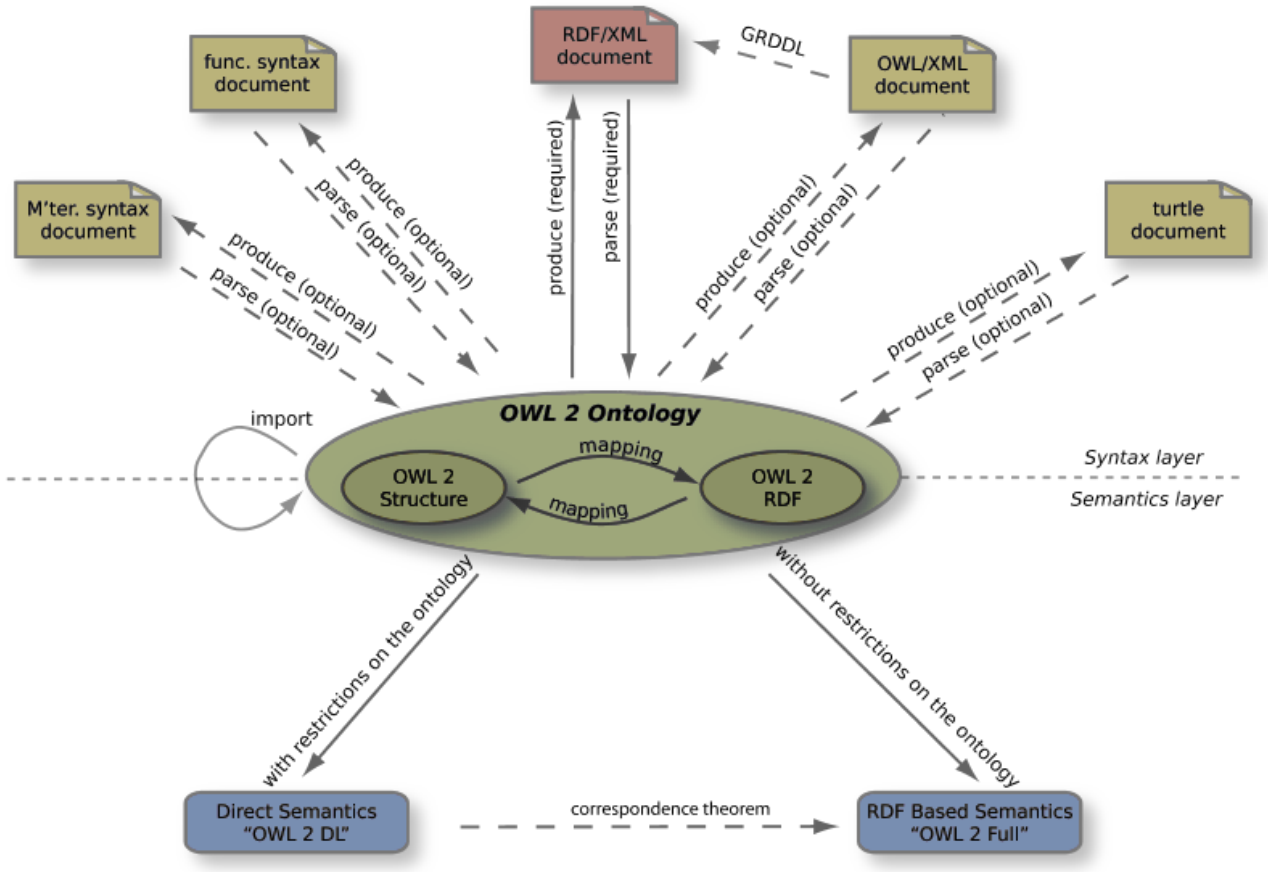


Рисунок 1.2 - Структура OWL 2

В основі мови – уявлення дійсності в моделі даних «об'єкт – властивість». OWL придатна для опису не тільки web-сторінок, але і будь-яких об'єктів дійсності. Кожному елементу опису в цій мові (в тому числі властивостями, що зв'язує об'єкти) ставиться у відповідність URI.

OWL має три діалекти (у порядку зростання виразності):

- OWL Lite призначена для користувачів, які потребують передусім класифікаційної ієрархії і простих обмежень. Наприклад, при тому, що вона підтримує обмеження кардинальності (кількості елементів), допускаються

значення кардинальності тільки 0 або 1. Для розробників повинно бути простіше в своїх продуктах забезпечити підтримку OWL Lite, чим виразніших варіантів OWL. Зокрема, OWL Lite дозволяє швидко перенести існуючі тезауруси і інші таксономії. OWL Lite також має нижчу формальну складність, ніж OWL DL.

- OWL DL призначена для користувачів, яким потрібна максимальна виразність при збереженні повноти обчислень (всі логічні висновки, що припускаються тією чи іншою онтологією, будуть гарантовано обчислюваними) і розв'язуваності (всі обчислення завершаться за певний час). OWL DL включає всі мовні конструкції OWL, але вони можуть використовуватися лише згідно з певним обмеженням (наприклад, клас може бути підкласом багатьох класів, але не може сам бути представником іншого класу). OWL DL так названий з-за його відповідності дескриптивній логіці – дисципліні, в якій розроблені логіки, що складають формальну основу OWL.
- OWL Full призначена для користувачів, яким потрібна максимальна виразність і синтаксична свобода RDF без гарантій обчислення. Наприклад, у OWL Full клас може розглядатися одночасно як сукупність індивідів і як один індивід у своєму власному значенні. OWL Full дозволяє будувати такі онтології, які розширюють склад зумовленого (RDF або OWL) словника. Малоймовірно, що будь-яке програмне забезпечення зможе здійснювати повну підтримку кожної особливості OWL Full.

1.6 Зв'язок описової логіки з мовою OWL

Мова web-онтологій OWL розробляється як мова, на якій можна формулювати і публікувати в web так звані мережеві онтології – формально записані твердження про поняття та об'єкти деякої предметної області. Одна з вимог до таких онтологій полягає в тому, щоб наявні в них знання були «доступні» для машинної обробки, зокрема, для автоматизованого логічного виведення нових знань із уже наявних. Для цього потрібно, щоб мова, якою формулюються онтології, мала точну семантику, а відповідні логічні проблеми

були розв'язані (і мали практично допустиму обчислювальну складність). Крім того, бажано, щоб така мова мала досить велику виражальну силу, придатну для формулювання на ній практично значущих фактів.

Описові логіки мають такі властивостями, і з цієї причини вони були обрані як логічні основи для мови web-онтологій OWL. Остання є мовою, що має XML-формат, тому можна сказати, що OWL є переформулюванням деяких описових логік з використанням синтаксису XML. Оскільки існує багато описових логік, що розрізняються як силою виражальності, так і обчислювальною складністю, це призвело до того, що в мови OWL є кілька варіантів.

Відповідність термінів: наявні в описовій логіці поняття концепт, роль, індивід і база знань в OWL відповідають поняттям клас, властивість, об'єкт і онтологія, відповідно.

Офіційною рекомендацією W3C від 10 лютого 2012 року а є версія мови OWL 2.0. Дана специфікація мови OWL підрозділяється на наступні варіанти:

- OWL-Lite – відповідає описовій логіці SHIF(D);

- OWL-DL – відповідає описовій логіці SHOIN(D);

- OWL-Full – не відповідає будь-якій описовій логіці, більш того, є нерозв'язною.

OWL 2.0 дає можливість формулювати онтології у мові, представленій в описовій логіці EL (перевага якої в тому, що вона має поліноміальну обчислювальну складність); має синтаксичні поліпшення порівняно з OWL 1.0, що дозволяє легше складати запити до баз знань і видавати відповіді на них; а також містить механізми для формулювання правил логічного висновку.

1.6.1 Мова OWL2

Сімейство мов представлення знань для онтологій відоме як OWL (мова веб-онтології). Онтологія сьогодні часто називається термінологічною базою знань. Віна представляє дані у формі аксіом та фактів для визначення ієрархії класів і властивостей. DL відіграють важливу роль для забезпечення логічного

формалізму для онтологій. Мова онтології OWL базується на DL SHOIN, яка складається з простих ієрархій ролей.

Онтологічна мова OWL2 базується на DL SROIQ, яка є однією з найбільш виразних DL, які на сьогодні широко вивчаються. Тому в даній роботі будемо використовувати DL SROIQ.

Концепції та ролі в DL, відповідно, називаються класами та властивостями в OWL2. Синтаксис OWL2 (таблиця 1.1) обробляється автоматично, тому не будемо розглядати його в контексті даної роботи.

Таблиця 1.1 - Синтаксис та семантика DL SROIQ

Визначення	Синтаксис	Семантика
Концепти		
Домен	$\top$	Δ^I
Пуста множина	$\bot$	$\emptyset$
Атомарне заперечення	$\neg A$	A^I
Складене заперечення	$\neg C$	$\Delta^I \backslash C^I$
Кон'юнкція	$C \sqcap D$	$C^I \cap D^I$
Диз'юнкція	$C \sqcup D$	$C^I \cup D^I$
Екзистенціальне обмеження	$\exists r.C$	$\{x \in \Delta^I \mid$ існує $y \in \Delta^I$: $(x,y) \in r^I$ та $y \in C^I\}$
Обмеження значення	$\forall r.C$	$\{x \in \Delta^I \mid$ для всіх $y \in \Delta^I$: $(x,y) \in r^I$, з чого випливає $y \in C^I\}$
Більше-або-дорівнює значенню	$\geq nr.C$	$\{x \in \Delta I \mid \mid (\, y \in \Delta I$: $(x,y) \in rI \wedge y \in CI\,) \mid \geq n\}$

Продовження таблиці 1.1		
Менше-або-дорівнює значенню	≤nr.C	$\{x \in \Delta^I \mid$ кардинальність $(y \in \Delta^I : (x,y) \in r^I \vee y \in C^I) \leq n \}$
Само-екзистенціальне обмеження	$\forall r.Self$	$\{x \in \Delta^I \mid$ існує $x \in \Delta^I : (x,y) \in r^I$ та $x \in C^I\}$
Номінал	$\{a\}$	$\{a^I\}$
Індивіди		
Ім'я індивіда	a	a^I
Ролі		
Роль	r	r^I
Зворотня роль	r^-	$\{<x,y> \in \Delta^I \square \Delta^I \mid <y,x> \in r^I \}$
Універсальна роль	U	$\Delta^I \square \Delta^I$
Аксіоми		
Включення концепту	$C \sqsubseteq D$	$C^I \subseteq D^I$
Еквівалентність концепту	$C \equiv D$	$C^I = D^I$
Призначення концепту	$C(a)$	$a^I \in C^I$
Призначення ролі	$r(a,b)$	$<a,b> \in r^I$
Призначення рівності індивіда	$a \approx b$	$a^I = b^I$
Призначення нерівності індивіда	$a \not\approx b$	$a^I \neq b^I$

Продовження таблиці 1.1		
Включення ролі	$r \sqsubseteq s$	$r^I \subseteq s^I$
Еквівалентність ролі	$r \equiv s$	$r^I = s^I$
Складене включення ролі	$\rho \sqsubseteq r$	$\rho^I \subseteq r^I$
Диз'юнкція ролі	$Disj(r,s)$	$r^I \cap s^I = \emptyset$

1.7 Обчислення неконсистентностей

Невідповідність бази знань може бути проаналізована за допомогою міри неконсистентності. Така міра необхідна для аналізу наскільки суперечливою є неконсистентність в онтології. Використовуючи ступінь неконсистентності, можно визначити спосіб виправлення невідповідності онтології. В даній роботі аналізуються методи обчислення міри невідповідностей для OWL 2 .

Визначення 7. Нехай *K* – база знань. *K* є несумісною, якщо вона не має моделей. В іншому випадку *K* – консистентна [3].

База знань:

Δ = {lion1;horse1;horse2; tiger1; tiger2; jaguar1}

$Tiger^I$ = {tiger1; tiger2}

$Horse^I$ = {horse1;horse2}

$eats^I$ = {(tiger1;horse1); (tiger1;horse2); (tiger2;horse1); (jaguar1;horse2); (lion1;horse2);

(jaguar1; tiger1)}

Дані:

$(\neg Tiger)^I = \Delta^I \backslash Tiger^I$ = {lion1;horse1;horse2; tiger1; tiger2; jaguar1} \ {tiger1; tiger2} =

{lion1;horse1;horse2; jaguar1};

$(Tiger \sqcap Horse)^I = Tiger^I \cap Horse^I = \emptyset = \bot$;

$(Tiger \sqcup Horse)^I = Tiger^I \cup Horse^I$ = {tiger1; tiger2;horse1;horse2};

$(\exists$ eats.Horse$)^I = \{a \in \Delta^I \mid \exists\ b \in \Delta^I : (a,b) \in eats^I \wedge b \in Horse^I\} = \{$tiger1; tiger2; jaguar1; lion1$\}$;

$(\forall$ eats.Horse$)^I = \{a \in \Delta^I \mid \forall\ b \in \Delta^I : (a;b) \in eats^I \rightarrow b \in Horse^I\} = \{$tiger1; tiger2; horse1; horse2; lion1$\}$.

Онтологія, яка не може мати будь-яких моделей, називається непослідовною онтологією. Будемо використовувати термін "неконсистентність", щоб послатися на проблеми невідповідності як в TBox, так і в Abox.

В даній роботі використовуємо базу знань для представлення онтології, яка містить множину елементів TBox і ABox без RBox. Наприклад, вона може бути описаний наступним чином:

$K1 = \{A1 \sqsubseteq A2 \sqcap \neg A, A2 \sqsubseteq A \sqcap A4, A1(a)\}$

Описані в даній роботі методи обчислення міри невідповідності для OWL 2 поділені на дві групи. Одна група аналізує неконсистентність всієї бази онтології, а інший аналізує неконсистентність кожної аксіоми бази.

Визначення 8. Нехай $R_{\geq 0}^{\infty}$ – множина невід'ємних раціональних значень включно з ∞. Міра неконсистентності на основі онтології виражається функцією $I : K \rightarrow R_{\geq 0}^{\infty}$.

Деякі методи для аналізу невідповідність всієї бази онтології використовуються для визначення міри невідповідності на основі онтології.

Визначення 9. Нехай $R_{\geq 0}^{\infty}$- множина невід'ємних раціональних значень включно з ∞. Міра неконсистентності на основі аксіоми ставить у відповідність пари (K, α) до елементів $R_{\geq 0}^{\infty}$, де K – база знань, $\alpha \in K$.

На відміну від міри несумісності на основі онтологій, яка аналізує невідповідність всієї бази, інша група методів обчислює міру невідповідності на основі аксіоми та визначає ступінь невідповідності кожної аксіоми бази. Ці методи будуть розглянуті в наступному розділі.

Висновки до розділу 1

1. Проведено аналітичний огляд літератури за темою некосистентності онтологій та проаналізовано існуючі підходи до обчислення міри невідповідності онтології з точки зору часу обчислення, кількості результуючих унікальних значень міри неконсистентності.

2. Показано, що недоліком цих підходів є:

а) те, що вони виконані в описовій логіці і лише деякі з них можна використати для онтологій OWL;

б) більшість методів має невеликий діапазон унікальних значень, відповідно за ними не можна надати точні рекомендації щодо виправлення невідповідності онтології;

б) відсутній комплексний підхід до аналізу методів обчислення міри невідповідності онтології.

РОЗДІЛ 2. ОБЧИСЛЕННЯ МІРИ НЕВІДПОВІДНОСТІ ДЛЯ OWL-ОНТОЛОГІЙ

У цьому розділі будуть визначені міри невідповідностей OWL 2 онтологій на основі невідповідностей для пропозиційної логіки.

Методи для обчислення міри неконсистентності на основі онтологій зазначені в першій чатині, на основі аксіом – в другій частині розділу. Для ілюстрації методів будемо використовувати бази знань K1, K2 та K3 з наступного прикладу.

Приклад 2. Нехай N_C := {ProductOwner, Employee, Developer, Person}; N_I := {tom, tim}; N_R := {manage}. Бази знань K1, K2, K3 визначаються наступним чином.

K1 = {ProductOwner ⊑ Employee ⊓ ¬Developer, Employee ⊑ Developer ⊓ Person,

ProductOwner(tom)}

K2 = {ProductOwner ⊑ Employee ⊓ ¬Developer, Employee ⊑ Developer ⊓ Person,

ProductOwner(tom), ProductOwner(tim)};

K3 = {ProductOwner ⊑ Employee ⊓ ¬∃ manage.Developer, Employee ⊑ Developer ⊓ Person, ProductOwner(tom)}.

K1 і K2 – неконсистентні бази знань з різною кількістю твердження, які передбачають різну ступінь невідповідності. K3 – послідовна база знань. Чим більше значення міри невідповідності, тим більше суперечлива база знань. З іншого боку, консистентна база знань матиме 0 як міру невідповідності. Для того, щоб передати міру невідповідності для OWL онтологій, формула в пропозиційній логіці представляється як аксіома в онтологіях.

2.1. Міра неконсистентності на основі онтології

Маттіас Тімм [9] досліджує декілька методів обчислення міри непослідовності для пропозиційної логіки. Ми переносимо деякі з них на OWL2.

Найпростіша з точки зору реалізації метод критичної міри невідповідності, а інші методи базуються на мінімальних/максимальних непослідовних множинах або на змінних параметрах. Ці методи будуть розглянуті в наступних підрозділах.

2.1.1. Критична міра невідповідності

Невідповідність бази знань можна виміряти, використовуючи простий спосіб, який дає лише два значення параметра для непослідовності. Проте він не оцінює ступінь впливу непослідовності на онтологію.

Визначення 10. Критичне значення міри неконсистентності I_d : $K \rightarrow R_{\geq 0}^{\infty}$ визначається як :

$I_d(K)$ = {1, якщо К – неконсистентна, 0 – в іншому випадку}

Обчислення критичної міри невідповідності є одним з найпростіших способів визначення непослідовності, яких лише присвоює значення 1 або 0 для непослідовної або послідовної бази знань. Міра дорівнює 1, якщо база знань є непослідовною та 0, якщо вона консистентна.

Приклад 3. Використовуємо бази знань К1, К2 та К3. Враховуючи те, що К1 та К2 є неконсистентними $I_d(K1) = I_d(K2) = 1$. База знань К3 є консистентною, тому $I_d(K3) = 0$.

2.1.2. Міра невідповідності на основі мінімальної неконсистентності

Невідповідності в базі знань можна отримати, знайшовши найменші за кількістю елементів підмножини бази знань, що викликають неконсистентність і об'єднати їх у множину.

Визначення 11. М – мінімально неконсистентна множина бази знань К, якщо виконуються наступні умови:

- М є неконсистентною;

- для всіх $M` \subset M$, M` – консистентна.

Використовується позначення MI(K) для визначення множини всіх мінімально неконсистентних підмножин.

Група методів обчислення міри невідновідності, в яких використовується значення MI(K), складається з п'яти методів. Для деяких з

них необхідне обчислення розміру MI(K), для інших необхідне використання підмножин MI(K). Кожен з цих методів описаний нижче.

2.1.2.1 MI-міра невідповідності

MI -міра невідповідності, яка базується на множині MI(K), лише розраховує розмір MI(K) у пов'язаній з ній базі знань. Це означає, що для обчислення цієї міри потрібно підрахувати кількість підмножин в MI (K).

Визначення 12. MI-міра невідповідності $I_{MI} : K \rightarrow R_{\geq 0}^{\infty}$ визначається як $I_{MI}(K) = |MI(K)|$ для бази знань K.

Оскільки MI-міра невідповідності визначає лише розмір MI(K), такий метод вважають простим. Наступний приклад відображає роботу метода на базі знань з прикладу 2.

Приклад 4.

MI(K1) := {{ProductOwner ⊑ Employee ⊓ ¬Developer, Employee ⊑ Developer ⊓ Person, ProductOwner(tom)}};

MI(K2) := {{ProductOwner ⊑ Employee ⊓ ¬Developer, Employee ⊑ Developer ⊓ Person, ProductOwner(tom)}, {ProductOwner ⊑ Employee ⊓ ¬Developer, Employee ⊑ Developer ⊓ Person, ProductOwner(tim)}};

MI(K3) := ∅ ;

Таким чином, $I_{MI}(K1) = 1$, $I_{MI}(K2) = 2$, $I_{MI}(K3) = 0$.

2.1.2.2 MI^C- міра невідповідності

В наступній мірі некосистентності, яка використовує множину MI(K), береться до уваги до уваги кожен елеимент MI(K). Розмір елемента MI(K) визначається шляхом підрахунку кількості його аксіом. Це значення використовується в якості знаменника, в формулі визначення $I_{MIC}(K)$. Загальне значення невідповідності отримується шляхом додавання всіх результатів ділення для кожної з підмножин MI(K).

Визначення 13. MI^C-міра невідповідності $I_{MIC}(K) : K \rightarrow R_{\geq 0}^{\infty}$ визначається як

$$I_{MI^C} = \sum_{M \in MI(K)} \frac{1}{|M|}$$

для бази знань К.

Для того щоб показати застосування даної міри неконсистентності в наступному прикладі використаємо базу знань з прикладу 2.

Приклад 5.

MI(K1) := {{ProductOwner ⊑ Employee ⊓ ¬Developer, Employee ⊑ Developer ⊓ Person, ProductOwner(tom)}};

MI(K2) := {{ProductOwner ⊑ Employee ⊓ ¬Developer, Employee ⊑ Developer ⊓ Person, ProductOwner(tom)}, {ProductOwner ⊑ Employee ⊓ ¬Developer, Employee ⊑ Developer ⊓ Person, ProductOwner(tim)}};

MI(K3) := ∅ ;

Таким чином,

I_{MIC}(K1) = 1/|{ProductOwner ⊑ Employee ⊓ ¬Developer, Employee ⊑ Developer ⊓ Person, ProductOwner(tom)}| = 1/3

I_{MI}(K2) = 1/|{ProductOwner ⊑ Employee ⊓ ¬Developer, Employee ⊑ Developer ⊓ Person, ProductOwner(tom)}| + 1/|{ProductOwner ⊑ Employee ⊓ ¬Developer, Employee ⊑ Developer ⊓ Person, ProductOwner(tim)}| = 1/3 + 1/3 = 2/3

Враховуючи те, що MI(K3) – пуста множина, ділення не виконується, адже ми б отримали нескінченість як значення неконсистентності. Таким чином, приймаємо, що I_{MIC}(K3) = 0.

2.1.2.3 D_f-міра неконсистентності

В [19] пропонується більш ефективна з точки зору точності обчислення міра подібності. В цьому методі використовуються не тільки мінімальні непослідовні підмножини, але й послідовні підмножини бази знань. Цей метод можна застосувати і для обчислення міри неконсистентності.

Визначення 14. Нехай K – база знань, i = 1, ..., |K|, $MI^I(K)$ – множина мінімальних неконсистентних підмножин K розміру i, $CN^I(K)$ – множина консистентних підмножин K розміру i, $R^I(K)$ – відношення розміру мінімально неконсистентної множини до розміру мінімально неконсистентної і консистентної множин. D_f-міра неконсистентності $I_{Df} : K \rightarrow R_{\geq 0}^{\infty}$ визначається

як

$$I_{D_f} = 1 - \prod_{i=1}^{|K|}(1 - R_i(K)/i),$$

де $MI^{(i)}(K) = \{M \in MI(K) \mid |M| = i\}$

$CN^{(i)}(K) = \{C \subseteq K \mid |C| = i \wedge C$ – консистентна$\}$

та $\begin{cases} 0, \text{якщо } \left|MI^{(i)}(K)\right| + \left|CN^{(i)}(K)\right| = 0; \\ \frac{|MI^{(i)}|}{|MI^{(i)}|+|CN^{(i)}(K)|}, \text{в іншому випадку} \end{cases}$

Міру неконсистентності називають D_f . Цей метод об'єднує вплив як консистентних, так і мінімально непослідовних підмножин. Виходячи з вищевикладеного обчислення, це означає, що чим більше значення $R_i(K)$, тим більше несумісна база знань K. Розглянемо це на наступному прикладі (використовуємо базу знань з прикладу 2).

Приклад 6.

$MI^{(3)}$(K1) = {{ProductOwner ⊑ Employee ⊓ ¬Developer, Employee ⊑ Developer ⊓ Person, ProductOwner(tom)}}, $MI^{(i)}$(K1) = ∅ для i ≠ 3.

$MI^{(3)}$(K2) = {{ProductOwner ⊑ Employee ⊓ ¬Developer, Employee ⊑ Developer ⊓ Person, ProductOwner(tom)}, {ProductOwner ⊑ Employee ⊓ ¬Developer, Employee ⊑ Developer ⊓ Person, ProductOwner(tim)}}, $MI^{(i)}$(K2) = ∅ для i ≠ 3.

$MI^{(3)}$(K3) = ∅ для i = 1,2,3.

$CN^{(1)}$(K1) = {{ProductOwner ⊑ Employee ⊓ ¬Developer}, {Employee ⊑ Developer ⊓ Person}, {ProductOwner(tom)}};

$CN^{(2)}$(K1) = {{ProductOwner ⊑ Employee ⊓ ¬Developer, Employee ⊑ Developer ⊓ Person}, {ProductOwner ⊑ Employee ⊓ ¬Developer, ProductOwner(tom)}, {Employee ⊑ Developer ⊓ Person , ProductOwner(tom)}};

$CN^{(3)}$(K1) = ∅.

$CN^{(1)}$(K2) = {{ProductOwner ⊑ Employee ⊓ ¬Developer}, {Employee ⊑ Developer ⊓ Person}, {ProductOwner(tom)}, {ProductOwner(tim)}};

$CN^{(2)}$(K2) = {{ProductOwner ⊑ Employee ⊓ ¬Developer, Employee ⊑ Developer ⊓ Person}, {ProductOwner ⊑ Employee ⊓ ¬Developer,

ProductOwner(tom)}, {Employee ⊑ Developer ⊓ Person, ProductOwner(tom)}, {ProductOwner ⊑ Employee ⊓ ¬Developer, ProductOwner(tim)}, {Employee ⊑ Developer ⊓ Person, ProductOwner(tim)}, {ProductOwner(tom), ProductOwner(tim)}};

$CN^{(3)}$(K2) = {{ProductOwner ⊑ Employee ⊓ ¬Developer, ProductOwner(tom), ProductOwner(tim)}, {Employee ⊑ Developer ⊓ Person, ProductOwner(tom), ProductOwner(tim)}}

$CN^{(4)}$(K2) = ∅.

$CN^{(1)}$(K3) = {{ProductOwner ⊑ Employee ⊓ ¬∃ manage.Developer}, {Employee ⊑ Developer ⊓ Person}, {ProductOwner(tom)}};

$CN^{(2)}$(K3) = {{ProductOwner ⊑ Employee ⊓ ¬∃ manage.Developer, Employee ⊑ Developer ⊓ Person}, {ProductOwner ⊑ Employee ⊓ ¬∃ manage.Developer, ProductOwner(tom)}, {Employee ⊑ Developer ⊓ Person , ProductOwner(tom)}};

$CN^{(3)}$(K3) = {{ProductOwner ⊑ Employee ⊓ ¬∃ manage.Developer, Employee ⊑ Developer ⊓ Person, ProductOwner(tom)}}

Відтак:

R1(K1) = 0; R2(K1) = 0; R3(K1) = 1;

R1(K2) = 0; R2(K2) = 0; R3(K2) =1/2;

R4(K2) = 0; R1(K3) = 0; R2(K3) = 0;

R3(K3) = 0.

Таким чином отримуємо:

I_{Df}(K1) = 1 – (1 – R_1(K1))(1 – R_2(K1)/2)(1 – R_3(K1)/3) = 1 – (1 – 0)(1 – 0/2)(1 – 1/3) = 1 – 2/3 = 1/3.

I_{Df}(K2) = 1 – (1 – R_1(K2))(1 – R_2(K2)/2)(1 – R_3(K2)/3)(1 – R_3(K2)/4) = 1 – (1 – 0)(1 – 0/2)(1 – 1/2/3)(1 – 0/4) = 1 – 5/6 = 1/6.

I_{Df}(K3) = 1 – (1 – R_1(K3))(1 – R_2(K3)/2)(1 – R_3(K3)/3) = 1 – (1 – 0)(1 – 0/2)(1 – 0/3) = 1 – 1 = 0.

2.1.3.4 Проблемна міра невідповідності

Визначення 15. Проблемна міра невідповідності $I_P : K \rightarrow R_{\geq 0}^{\infty}$

визначається як

$$I_p(K) = | \bigcup_{M \in MI(K)} M |$$

для бази знань К.

Проблемна міра невідповідності була запропонована Джоном Грантом та Ентони Хантером [20]. В цій мірі враховується ступінь неконсистентності на основі розміру множини *MI*(*K*). така ідея об'єднує всі аксіоми, що є членами множини *MI*(*K*) та кількість унікальних аксіом в *MI*(*K*). Розглянемо обчислення проблемної міри невідповідності для бази знань з прикладу 2.

Приклад 7. Для бази знань з прикладу 3 отримуємо $I_P(K1) = 3$, $I_P(K2) = 4$, $I_P(K3) = 0$.

2.1.2.5 Коефіцієнт несумісності для міри невідповідності

Інша міра невідповідності, запропонована Ентони Хантером [3], яка використовує *MI*(*K*) визначає коефіцієнт несумісності. В обчисленні також враховується розмір *K*.

Визначення 16. Міра невідповідності з коефіцієнтом несумісності $I_{IR} : K \rightarrow R_{\geq 0}^{\infty}$ визначається як

$I_{IR}(K) = |MI(K)|/|K|$

для бази знань *K*.

Таким чином, коефіцієнт несумісності визначається як відношення розміру множини *MI*(*K*) до розміру бази знань *K*. Розглянемо наступний приклад.

Приклад 8. Використовуємо базу знань з прикладу 2.

Отримуємо $|K1| = 3$, $|K2| = 4$, $|K3| = 3$, таким чином,

$I_{IR}(K) = 1/3$;

$I_{IR}(K) = 2/4 = ½$;

$I_{IR}(K) = 0/3 = 0$.

2.1.3 Міра невідповідності на основі максимальної консистентності

Невідповідність бази знань може бути отримана шляхом використання

консистентності бази знань. У цьому підрозділі описаний інший підхід до обчислення невідповідності на основі онтології, який базується на максимально консистентних підмножинах бази знань. Методи, в яких використовується цей підхід, MC-невідповідність та nc-непослідовність, адаптовані для онтологій OWL2 з оригінальної версії.

2.1.3.1 MC-міра невідповідності

MC-міра невідповідності була запропонована Джоном Грантом та Ентоні Хантером [20]. Для обчислення цієї міри використовується розмір максимально консистентної множини бази знань і кількість аксіоми, які в яких присутня внутрішня суперечливість.

Визначення 17. Нехай $MC(K)$ – множина максимально консистентних підмножин бази знань K, а $SC(K)$ – множина аксіом із внутрішньою суперечливістю в K. MC-міра невідповідності $I_{MC} : K \rightarrow R_{\geq 0}^{\infty}$ визначається як $I_{MC} = |MC(K)| + |SC(K)| - 1$,

Де $MC(K) = \{K` \subseteq K \mid$ K` консистентна $\wedge \forall K'' \supsetneq K` : K``$ – неконситентна$\}$

та

$SC(K) = \{\emptyset \in K \mid \emptyset$ – незадовільна$\}$.

Аксіома в множині $SC(K)$ вважається незадовільною, якщо вона є неконсистентною[21]. Розглянемо наступний приклад на основі бази знань з прикладу 2.

Приклад 9.

$MC(K1)$ = {{ProductOwner ⊑ Employee ⊓ ¬Developer, Employee ⊑ Developer ⊓ Person}, {ProductOwner ⊑ Employee ⊓ ¬Developer, ProductOwner(tom)}, {Employee ⊑ Developer ⊓ Person , ProductOwner(tom)}};

$MC(K2)$ = {{ProductOwner(tim), Employee ⊑ Developer ⊓ Person, ProductOwner(tom)}, {ProductOwner ⊑ Employee ⊓ ¬Developer, ProductOwner(tim),ProductOwner(tom)}, {ProductOwner ⊑ Employee ⊓ ¬Developer, Employee ⊑ Developer ⊓ Person}};

$MC(K3)$ = {{ProductOwner ⊑ Employee ⊓ ¬∃ manage.Developer, Employee ⊑ Developer ⊓ Person, ProductOwner(tom)}}.

$SC(K1) = SC(K2) = SC(K3) = \emptyset$;

Таким чином, отримуємо $I_{MC}(K1) = 3 + 0 – 1 = 2$, $I_{MC}(K2) = 3 + 0 – 1 = 2$, $I_{MC}(K3) = 1 + 0 – 1 = 0$.

2.1.3.2 NC-міра невідповідності

NC-міра невідповідності була запропонована в [21] та базується на максимально консистентній множині. Розмір бази знань також використовується в обчисленні.

Визначення 18. NC-міра невідповідності $I_{NC} : K \rightarrow R_{\geq 0}^{\infty}$ визначається як $I_{NC}(K) = |K| – \max\{n \mid \forall K` \subseteq K : |K`| = n \Rightarrow K` – \text{консистентна}\}$

для бази знань K.

NC-міра невідповідності розраховується як різниця між розміром бази знань та максимальним розміром консистнетної підмножини.

Приклад 10. Використовуємо базу знань із прикладу 2.

Консистентні підмножини $K1$ =

{{ProductOwner ⊑ Employee ⊓ ¬Developer} {Employee ⊑ Developer ⊓ Person}; {ProductOwner(tom)};

{ProductOwner ⊑ Employee ⊓ ¬Developer; Employee ⊑ Developer ⊓ Person}; {ProductOwner ⊑ Employee ⊓ ¬Developer; ProductOwner(tom)}; {Employee ⊑ Developer ⊓ Person; ProductOwner(tom)}};

$K2$ = {{ProductOwner ⊑ Employee ⊓ ¬Developer}, {Employee ⊑ Developer ⊓ Person}, {ProductOwner(tom)}, {ProductOwner(tim)},

{ProductOwner ⊑ Employee ⊓ ¬Developer, Employee ⊑ Developer ⊓ Person}, {ProductOwner ⊑ Employee ⊓ ¬Developer, ProductOwner(tom)}, {Employee ⊑ Developer ⊓ Person, ProductOwner(tom)}, {ProductOwner ⊑ Employee ⊓ ¬Developer, ProductOwner(tim)}, {Employee ⊑ Developer ⊓ Person, ProductOwner(tim)}, {ProductOwner(tom), ProductOwner(tim)},

{ProductOwner ⊑ Employee ⊓ ¬Developer, ProductOwner(tom), ProductOwner(tim)}, {Employee ⊑ Developer ⊓ Person, ProductOwner(tom), ProductOwner(tim)}}.

$K3$ = {{ProductOwner ⊑ Employee ⊓ ¬∃ manage.Developer}, {Employee

⊑ Developer ⊓ Person}, {ProductOwner(tom)},

{ProductOwner ⊑ Employee ⊓ ¬∃ manage.Developer, Employee ⊑ Developer ⊓ Person}, {ProductOwner ⊑ Employee ⊓ ¬∃ manage.Developer, ProductOwner(tom)}, {Employee ⊑ Developer ⊓ Person , ProductOwner(tom)}, {ProductOwner ⊑ Employee ⊓ ¬∃ manage.Developer, Employee ⊑ Developer ⊓ Person, ProductOwner(tom)}}.

Оскільки максимальний розмір консистентних підмножин $K1$, $K2$ та $K3$ – 2, 3, 3, отримуємо

$I_{NC}(K1) = 3 - 2 = 1,$

$I_{NC}(K2) = 4 - 3 = 1,$

$I_{NC}(K3) = 3 - 3 = 0.$

2.1.4 Міра невідповідності на основі змінної

Методи обчислення невідповідності онтологій основі змінних для описової логіки були перенесені на OWL 2 з використанням функції sign в базі знань, які позначаються як $Sign(K)$. У цій групі існують два методи обчислення: mv-непослідовність та невідповідність ID_{MCS}. Обидва методи були вперше описані в [5].

2.1.4.1 MV-міра невідповідності

MV-міра невідповідності входить в групу методів на основі змінної, оскільки для її обчислення використовують функцію sign бази знань та мінімально неконсистентностні підмножини.

Визначення 19. Нехай $Sign(K)$ та $Sign(M)$ – функції sign бази знань K та $M \in MI(K)$. MV-міра невідповідності $I_{MV} : K \to R_{\geq 0}^{\infty}$ визначається як

$$I_{mv}(K) = \frac{|\bigcup_{M \in MI(K)}(Sign(M) \cup \{\bot, \top\})|}{|Sign(K) \cup \{\bot, \top\}|}$$

Дана міра невідповідності використовує множину мінімально неконсистентних підмножин, хоча і не входить до групи таких методів. На основі обчислень розмір $Sign(M)$ завжди буде менше або дорівнювати розміру $Sign(K)$, відтак значення міри неконсистентності завжди буде в рамках від 0 до 1[22].

Приклад 11. Використовуємо бази знань прикладу 2 та додаємо нову базу знань $K4$.

$K4$ = {ProductOwner ⊑ Employee ⊓ ¬Developer, Employee ⊑ Developer ⊓ Person, ProductOwner ⊑ ⊥, ⊤ ⊑ ⊥, ProductOwner(tom), Employee(tim)}.

Отримуємо:

MI(K1) = {{ProductOwner ⊑ Employee ⊓ ¬Developer, Employee ⊑ Developer ⊓ Person, ProductOwner(tom)}};

MI(K2) = {{ProductOwner ⊑ Employee ⊓ ¬Developer, Employee ⊑ Developer ⊓ Person, ProductOwner(tom)}, {ProductOwner ⊑ Employee ⊓ ¬Developer, Employee ⊑ Developer ⊓ Person, ProductOwner(tim)}};

MI(K3) = ∅ ;

MI(K4) = {{ProductOwner ⊑ Employee ⊓ ¬Developer, Employee ⊑ Developer ⊓ Person, ProductOwner(tom)} , {ProductOwner(tom), ProductOwner ⊑ ⊥}, {⊤ ⊑ ⊥}}.

Таким чином MV-міра невідповідності:

$I_{MV}(K1) = 5/5 = 1$;

$I_{MV}(K2) = 6/6 = 1$;

$I_{MV}(K3) = 0/6 = 0$;

$I_{MV}(K4) = 7/8 = 0{,}875$.

2.1.4.2 ID_{MCS}-міра невідповідності

Визначення 20. Підмножина $M \subseteq K$ – підмножина мінімальної корекції (*MCS*), якщо $K \setminus M$ – консистентна і $\forall\ C_i \subseteq M$, $K \setminus (M \setminus \{C_i\})$ – неконсистентна. Множину *MCS* в базі знань K позначають як $MCSes(K)$.

Дана міра неконсистентності була запропонована в [5] та використовує для обчислення підмножини мінімальної корекції.

Визначення 21. Нехай $Sign(K)$ та $Sign(MCSes(K))$ – функції sign в базі знань K та *MCSes* з K. ID_{MCS}-міра невідповідності $ID_{MCS} : K \rightarrow R_{\geq 0}^{\infty}$ визначається як

$$I_{mv}(K) = \frac{|Sign(MCSes(K)) \cup \{\bot, \top\}|}{|Sign(K) \cup \{\bot, \top\}|}$$

для бази знань *K*.

Міра невідповідності ID_{MCS} використовує ступінь невідповідності бази знань за допомогою функції sign *MCS* бази знань. Оскільки база знань може мати більше однієї *MCS*, *MCSes* як множина MCS використовується для обчислення ступеню невідповідності. Обчислення даної міри невідповідності полягає в обчисленні функції sign *MCS* та загальних функцій sign бази знань, де *Sign*(*MCSes*(*K*)) ≤ *Sign*(*K*).

Приклад 12. Використовуємо базі знань *K*1, *K*2, *K*3 з прикладу 2 та базу знань *K*4 з прикладу 11.

MCSes(*K*1) = {{ProductOwner ⊑ Employee ⊓ ¬Developer}, {Employee ⊑ Developer ⊓ Person}, {ProductOwner(tom)}};

MCSes(*K*2) = {{ProductOwner ⊑ Employee ⊓ ¬Developer}, {Employee ⊑ Developer ⊓ Person}, {ProductOwner(tom), ProductOwner(tim)}};

MCSes(*K*3) = ∅;

MCSes(*K*4) = {{Employee ⊑ Developer ⊓ Person, ProductOwner ⊑ ⊥, ⊤ ⊑ ⊥}, {ProductOwner(tom), ⊤ ⊑ ⊥}, {ProductOwner ⊑ Employee ⊓ ¬Developer, ProductOwner ⊑ ⊥, ⊤ ⊑ ⊥}}.

Отримуємо:

ID_{MCS}(*K*1) = |Sign(MCSes(K1))| / |Sign(K1)| = 5/5 = 1;

ID_{MCS}(*K*2) = |Sign(MCSes(K2))| / |Sign(K2)| = 6/6 =1;

ID_{MCS}(*K*3) = |Sign(MCSes(K3))| / |Sign(K3)| = 0/6 = 0;

ID_{MCS}(*K*4) = |Sign(MCSes(K4))| / |Sign(K4)| = 7/8 = 0,875.

2.2 Міра неконсистентності на основі аксіоми

Група методів обчислення міри неконсистентності онтології OWL2 на основі аксіоми складається з п'яти алгоритмів. В деяких з них використовується мінімально неконсистентні множини, описані в пункті 2.1.2.

Визначення 22. Мінімальне значення неконсистентності – функція

MIV(*K, α*), яка співставляє пари (*K, α*) з елементами $R_{\geq 0}^{\infty}$ для бази знаь *K*, де $\alpha \in K$ – аксіома.

В [1] була запропонована група методів обчислення міри неконсистентності на основі мінімального значення неконсистентності, яка спочатку була розроблена для описової логіки. Ці підходи допомагають визначити значення невідповідності для кожної аксіоми бази знань. Такий підхід відрізняється від тих, що містяться в розділі 2.1.2, в яких визначається міра невідповідності всієї бази онтології. В даному розділі методи обчислення міри неконсистентності онтології на основі аксіоми для описової логіки будуть перенесені на онтології OWL2.

2.2.1 MIV_D – міра невідповідності

Визначення 23. Нехай *MI*(*K*) – множина мінімально неконсистентних підмножин. Вільна аксіома бази знань *K* – аксіома, яка не належить до жодної підмножини множини *MI*(*K*).

MIV_D міра невідповідності дорівнює 1, якщо аксіома належить будь-якій підмножині множини *MI*(*K*), і 0 в іншому випадку. Таким чином вільні аксіоми відрізняються від інших.

Визначення 24. Нехай $\alpha \in K$ – аксіома бази знаь *K*. MIV_D міра невідповідності: $(K, \alpha) \rightarrow R_{\geq 0}^{\infty}$ визначається як

$\text{MIV}_\text{D}(K) = 1$, якщо $\exists M \in MI(K), \alpha \in M$

0, в іншому випадку

Приклад 13. Використовуємо бази знань з прикладу 2.

MI(K1) := {{ProductOwner ⊑ Employee ⊓ ¬Developer, Employee ⊑ Developer ⊓ Person, ProductOwner(tom)}};

MI(K2) := {{ProductOwner ⊑ Employee ⊓ ¬Developer, Employee ⊑ Developer ⊓ Person, ProductOwner(tom)}, {ProductOwner ⊑ Employee ⊓ ¬Developer, Employee ⊑ Developer ⊓ Person, ProductOwner(tim)}};

MI(K3) := ∅ ;

Відповідно отримуємо

MIV_D(*K*1, ProductOwner ⊑ Employee ⊓ ¬Developer) = 1;

$MIV_D(K1,$ Employee ⊑ Developer ⊓ Person) = 1;

$MIV_D(K1,$ ProductOwner(tom)) = 1;

$MIV_D(K2,$ ProductOwner ⊑ Employee ⊓ ¬Developer) = 1;

$MIV_D(K2,$ Employee ⊑ Developer ⊓ Person) = 1;

$MIV_D(K2,$ ProductOwner(tom)) = 1;

$MIV_D(K2,$ ProductOwner(tim)) = 1.

Оскільки MI(K3) – пуста множина, отримуємо:

$MIV_D(K3,$ ProductOwner ⊑ Employee ⊓ ¬∃manage.Developer) = 0;

$MIV_D(K3,$ Employee ⊑ Developer ⊓ Person) = 0;

$MIV_D(K3,$ ProductOwner(tom)) = 0.

2.2.2 $MIV_{\#}$- міра невідповідності

Друга міра *MIV* розраховується як кількість мінімально неконсистентних підмножин, яким належить аксіома.

Визначення 25. Нехай $\alpha \in K$ – аксіома бази знань *K*. $MIV_{\#}$-міра невідповідності: $(K, \alpha) \rightarrow R_{\geq 0}^{\infty}$ визначається як

$MIV_{\#}(K, \alpha) = |\{M \in MI(K) \mid \alpha \in M\}|$.

Приклад використання $MIV_{\#}$-міри невідповідності наступний.

Приклад 14. Використовуємо бази знань з прикладу 2.

MI(K1) := {{ProductOwner ⊑ Employee ⊓ ¬Developer, Employee ⊑ Developer ⊓ Person, ProductOwner(tom)}};

MI(K2) := {{ProductOwner ⊑ Employee ⊓ ¬Developer, Employee ⊑ Developer ⊓ Person, ProductOwner(tom)}, {ProductOwner ⊑ Employee ⊓ ¬Developer, Employee ⊑ Developer ⊓ Person, ProductOwner(tim)}};

MI(K3) := ∅ ;

Таким чином, отримуємо

$MIV_{\#}(K1,$ ProductOwner ⊑ Employee ⊓ ¬Developer) = 1;

$MIV_{\#}(K1,$ Employee ⊑ Developer ⊓ Person) = 1;

$MIV_{\#}(K1,$ ProductOwner(tom)) = 1;

$MIV_{\#}(K2,$ ProductOwner ⊑ Employee ⊓ ¬Developer) = 2;

$MIV_{\#}(K2,$ Employee ⊑ Developer ⊓ Person) = 2;

$MIV_{\#}(K2,$ ProductOwner(tom)) = 1;

$MIV_{\#}(K2,$ ProductOwner(tim)) = 1.

$MIV_{\#}(K3,$ ProductOwner ⊑ Employee ⊓ ¬∃manage.Developer) = 0;

$MIV_{\#}(K3,$ Employee ⊑ Developer ⊓ Person) = 0;

$MIV_{\#}(K3,$ ProductOwner(tom)) = 0.

2.2.3 MIV_C- міра невідповідності

Третя міра невідповідності в групі MIV враховує кардинальність мінімально неконсистентних підмножин, яким належить аксіома.

Визначення 26. Нехай $\alpha \in K$ – аксіома бази знань K. $MIV_{\#}$-міра невідповідності: $(K, \alpha) \rightarrow R_{\geq 0}^{\infty}$ визначається як

$$MIV_C(K, \alpha) = \sum_{M \in MI(K), \alpha \in M} \frac{1}{|M|}$$

В MIV_C- мірі невідповідності додаються всі результати ділення одиниці на кардинальність кожної мінімально неконсистентної підмножини.

Приклад 15. Використовуємо бази знань з прикладу 2.

MI(K1) := {{ProductOwner ⊑ Employee ⊓ ¬Developer, Employee ⊑ Developer ⊓ Person, ProductOwner(tom)}};

MI(K2) := {{ProductOwner ⊑ Employee ⊓ ¬Developer, Employee ⊑ Developer ⊓ Person, ProductOwner(tom)}, {ProductOwner ⊑ Employee ⊓ ¬Developer, Employee ⊑ Developer ⊓ Person, ProductOwner(tim)}};

MI(K3) := ∅ ;

Таким чином отримуємо:

$MIV_c(K1,$ ProductOwner ⊑ Employee ⊓ ¬Developer) = 1 / |{ProductOwner ⊑ Employee ⊓ ¬Developer, Employee ⊑ Developer ⊓ Person, ProductOwner(tom)}| = 1/3;

$MIV_c(K1,$ Employee ⊑ Developer ⊓ Person) = 1 / |{ProductOwner ⊑ Employee ⊓ ¬Developer, Employee ⊑ Developer ⊓ Person, ProductOwner(tom)}| = 1/3;

$MIV_c(K1,$ ProductOwner(tom)) = 1 / |{ProductOwner ⊑ Employee ⊓ ¬Developer, Employee ⊑ Developer ⊓ Person, ProductOwner(tom)}| = 1/3;

MIV_c(*K*2, ProductOwner ⊑ Employee ⊓ ¬Developer) = 1 / |{ProductOwner ⊑ Employee ⊓ ¬Developer, Employee ⊑ Developer ⊓ Person, ProductOwner(tom)} + 1/ |{ProductOwner ⊑ Employee ⊓ ¬Developer, Employee ⊑ Developer ⊓ Person, ProductOwner(tim)}| = 1/3 + 1/3 = 2/3;

MIV_c(*K*2, Employee ⊑ Developer ⊓ Person) = 1 / |{ProductOwner ⊑ Employee ⊓ ¬Developer, Employee ⊑ Developer ⊓ Person, ProductOwner(tom)}| + 1/ |{ProductOwner ⊑ Employee ⊓ ¬Developer, Employee ⊑ Developer ⊓ Person, ProductOwner(tim)}| = 1/3+ 1/3 = 2/3;

MIV_c(*K*2, ProductOwner(tom)) = 1 / |{ProductOwner ⊑ Employee ⊓ ¬Developer, Employee ⊑ Developer ⊓ Person, ProductOwner(tom)}| = 1/3;

MIV_c(*K*2, ProductOwner(tim)) = 1 / |{ProductOwner ⊑ Employee ⊓ ¬Developer, Employee ⊑ Developer ⊓ Person, ProductOwner(tim)}| = 1/3.

MIV_c(*K*3, ProductOwner ⊑ Employee ⊓ ¬∃manage.Developer) = 0;

MIV_c(*K*3, Employee ⊑ Developer ⊓ Person) = 0;

MIV_c(*K*3, ProductOwner(tom)) = 0.

2.2.4 MI-Шеплі міра невідповідності

В [23] запропонована міра некосистентності, в якій модифікуються алгоритми MIV та Шеплі.

Визначення 27. Нехай $\alpha \in K$ – аксіома бази знань *K*, *I* – базова міра неконсистентності [24]. $S^I{}_\alpha(K)$ – міра невідповідності Шеплі визначається як

$$S_\alpha^I(K) = \sum_{C \subseteq K} \frac{(c-1)!(n-c)!}{n!}(I(C) - I(C\{\alpha\})),$$

де *n* – розмір *K*, *c* – розмір *C*.

Міра невідповідності Шеплі, яку зазвичай позначають SIV, визначає міру невідповідності для кожної аксіоми. Обчислення цього показника базується на теорії ігор. MI-Шеплі міра невідповідності для OWL 2 онтологій описана в наступному визначенні.

Визначення 28. Нехай $\alpha \in K$ – аксіома бази знань *K*, I_{MI} – мінімальна міра невідповідності. $S^{\text{iMI}}{}_\alpha(K)$ – міра невідповідності визначається як

$$S_{\alpha}^{I_{MI}}(K) = \sum_{C \subseteq K} \frac{(c-1)!\,(n-c)!}{n!}\left(I_{MI}(C) - I_{MI}(C\{\alpha\})\right)$$

, де n – розмір K, c – розмір C.

Приклад 16. Використовуємо бази знань з прикладу 2.

MI(K1) := {{ProductOwner ⊑ Employee ⊓ ¬Developer, Employee ⊑ Developer ⊓ Person, ProductOwner(tom)}};

MI(K2) := {{ProductOwner ⊑ Employee ⊓ ¬Developer, Employee ⊑ Developer ⊓ Person, ProductOwner(tom)}, {ProductOwner ⊑ Employee ⊓ ¬Developer, Employee ⊑ Developer ⊓ Person, ProductOwner(tim)}};

MI(K3) := ∅ ;

Підмножина містить три аксіоми, які впливають на консистентність $K1$:

{ProductOwner ⊑ Employee ⊓ ¬Developer, Employee ⊑ Developer ⊓ Person, ProductOwner(tom)}. Таким чином,

I_{MI} ({ProductOwner ⊑ Employee ⊓ ¬Developer, Employee ⊑ Developer ⊓ Person, ProductOwner(tom)}) = |{{ProductOwner ⊑ Employee ⊓ ¬Developer, Employee ⊑ Developer ⊓ Person, ProductOwner(tom)}}| = 1.

Враховуючи те, що інші підмножини не впливають на відповідність, для них значення мінімальної міри невідповідності дорівнює 0. Тому для обчислення значення Шеплі використовуємо тільки одну підмножину. Відтак:

$S^{Imi}_{ProductOwner \sqsubseteq Employee \sqcap \neg Developer}(K1)$= ((3 – 1)!(3 – 3)!/3!) * (1 – 0) = 2!0!/3! = 1/3;

$S^{Imi}_{Employee \sqsubseteq Developer \sqcap Person}(K1)$= ((3 – 1)!(3 – 3)!/3!) * (1 – 0) = 2!0!/3! = 1/3;

$S^{Imi}_{ProductOwner(tom)}(K1)$= ((3 – 1)!(3 – 3)!/3!) * (1 – 0) = 2!0!/3! = 1/3;

Для $K2$ отримуємо:

I_{MI} ({ProductOwner ⊑ Employee ⊓ ¬Developer, Employee ⊑ Developer ⊓ Person, ProductOwner(tom)}) = |{{ProductOwner ⊑ Employee ⊓ ¬Developer, Employee ⊑ Developer ⊓ Person, ProductOwner(tom)}}| = 1.

I_{MI} ({ProductOwner ⊑ Employee ⊓ ¬Developer, Employee ⊑ Developer ⊓ Person, ProductOwner(tim)}) = |{{ProductOwner ⊑ Employee ⊓ ¬Developer,

Employee ⊑ Developer ⊓ Person, ProductOwner(tim)}}| = 1.

I_{MI} ({ProductOwner ⊑ Employee ⊓ ¬Developer, Employee ⊑ Developer ⊓ Person, ProductOwner(tom), ProductOwner(tim)}) = |{{ProductOwner ⊑ Employee ⊓ ¬Developer, Employee ⊑ Developer ⊓ Person, ProductOwner(tom)}, {ProductOwner ⊑ Employee ⊓ ¬Developer, Employee ⊑ Developer ⊓ Person, ProductOwner(tim)}}| = 2.

Відтак:

$S^{Imi}_{ProductOwner \sqsubseteq Employee \sqcap \neg Developer}(K2) = ((3-1)!(4-3)!/4!)*(1-0)! + ((3-1)!(4-3)!/4!)*(1-0)! + ((4-1)!(4-4)!/4!)*(2-0)! = 2!1!/4! + 2!1!/4! +3!0!*2/4! = 1/12 + 1/12 +2/4 = 2/3;$

$S^{Imi}_{Employee \sqsubseteq Developer \sqcap Person}(K2) = ((3-1)!(4-3)!/4!)*(1-0)! + ((3-1)!(4-3)!/4!)*(1-0)! + ((4-1)!(4-4)!/4!)*(2-0)! = 2!1!/4! + 2!1!/4! +3!0!*2/4! = 1/12 + 1/12 +2/4 = 2/3;$

$S^{Imi}_{ProductOwner(tom)}(K2) = ((3-1)!(4-3)!/4!)*(1-0)! + ((3-1)!(4-3)!/4!)*(1-1)! + ((4-1)!(4-4)!/4!)*(2-1)! = 2!1!/4! + 3!0!/4! = 1/12 + 1/4 = 4/12 = 1/3;$

$S^{Imi}_{ProductOwner(tim)}(K2) = ((3-1)!(4-3)!/4!)*(1-1)! + ((3-1)!(4-3)!/4!)*(1-0)! + ((4-1)!(4-4)!/4!)*(2-1)! = 2!1!/4! + 3!0!/4! = 1/12 + 1/4 = 4/12 = 1/3.$

Для $K3$ отримуємо

$S^{Imi}_{ProductOwner \sqsubseteq Employee \sqcap \neg \exists manage.Developer}(K3) = 0;$

$S^{Imi}_{Employee \sqsubseteq Developer \sqcap Person}(K3) = 0;$

$S^{Imi}_{ProductOwner(tom)}(K3) = 0.$

2.2.5 Розрахункова міра невідповідності

Розрахункова міра невідповідності також використовує множину $MI(K)$.

Визначення 29. Нехай $MI(K)$ – множина мінімально неконсистентних підмножин з бази знань K, $\alpha \in K$ – аксіома. Розрахункова міра невідповідності $I_S : (K, \alpha) \rightarrow R_{\geq 0}^{\infty}$ визначається як

$$I_S(K, \alpha) = |MI(K)| - |MI(K - \alpha)|.$$

Приклад 17. Використовуємо бази знань з прикладу 2.

MI(K1) := {{ProductOwner ⊑ Employee ⊓ ¬Developer, Employee ⊑ Developer ⊓ Person, ProductOwner(tom)}};

MI(K2) := {{ProductOwner ⊑ Employee ⊓ ¬Developer, Employee ⊑ Developer ⊓ Person, ProductOwner(tom)}, {ProductOwner ⊑ Employee ⊓ ¬Developer, Employee ⊑ Developer ⊓ Person, ProductOwner(tim)}};

MI(K3) := ∅ .

Три аксіоми з $K1$ впливають на консистентність $K1$:

MI({ProductOwner ⊑ Employee ⊓ ¬Developer, Employee ⊑ Developer ⊓ Person, ProductOwner(tom)}) = |{ProductOwner ⊑ Employee ⊓ ¬Developer, Employee ⊑ Developer ⊓ Person, ProductOwner(tom)}| = 1.

I_S($K1$, ProductOwner ⊑ Employee ⊓ ¬Developer) = 1 – 0 = 1;

I_S($K1$, Employee ⊑ Developer ⊓ Person) = 1 – 0 = 1;

I_S($K1$, ProductOwner(tom)) = 1 – 0 = 1.

Для $K2$ маємо наступне:

MI({ProductOwner ⊑ Employee ⊓ ¬Developer, Employee ⊑ Developer ⊓ Person, ProductOwner(tom)}) = |{{ProductOwner ⊑ Employee ⊓ ¬Developer, Employee ⊑ Developer ⊓ Person, ProductOwner(tom)}}| = 1;

MI({ProductOwner ⊑ Employee ⊓ ¬Developer, Employee ⊑ Developer ⊓ Person, ProductOwner(tim)}) = |{{ProductOwner ⊑ Employee ⊓ ¬Developer, Employee ⊑ Developer ⊓ Person, ProductOwner(tim)}{| = 1;

MI({ProductOwner ⊑ Employee ⊓ ¬Developer, Employee ⊑ Developer ⊓ Person, ProductOwner(tom), ProductOwner(tim)}) = |{{ProductOwner ⊑ Employee ⊓ ¬Developer, Employee ⊑ Developer ⊓ Person, ProductOwner(tom)}, {ProductOwner ⊑ Employee ⊓ ¬Developer, Employee ⊑ Developer ⊓ Person, ProductOwner(tom), ProductOwner(tim)}}| = 2.

Відтак:

I_S($K2$, ProductOwner ⊑ Employee ⊓ ¬Developer) = 2 – 0 = 2;

I_S($K2$, Employee ⊑ Developer ⊓ Person) = 2 – 0 = 2;

I_S($K2$, ProductOwner(tom)) = 2 – 1 = 1.

I_S (K2, ProductOwner(tim)) = 2 – 1 = 1.

Для K3 маємо:

I_S (K3, ProductOwner ⊑ Employee ⊓ ¬∃manage.Developer) = 0;

I_S (K3, Employee ⊑ Developer ⊓ Person) = 0;

I_S (K3, ProductOwner(tom)) = 0.

Висновки до розділу 2

1. Проаналізовано способи обчислення міри невідповідності онтології в описовій логіці та показано, що їх можна ефективно використовувати і для OWL онтологій.

2. Проведена класифікація способів обчислення міри невідповідності онтології за одиницею обчислення неконсистентності: на рівні онтології та на рівні аксіоми.

3. На основі проведеного дослідження запропонований спосіб обчислення MI-Шеплі міри невідповідності, який відрізняється від існуючих використанням мінімально неконсистентних підмножин, що у випадках, коли таких підмножин не більше 25% від загальної кількості, призводить до меншого часу виконання обчислення.

РОЗДІЛ 3. ІМПЛЕМЕНТАЦІЯ МЕТОДІВ ОБЧИСЛЕННЯ МІРИ НЕВІДПОВІДНОСТІ

Обчислення міри невідповідності онтологій OWL реалізується в програмному забезпеченні, яке складається з 15 вибраних методів, що зазначені в попередньому розділі. Розроблене ПЗ можна використовувати і для реальних онтологій. Бібліотеки для створення інструменту будуть пояснені в даному розділі. Основні алгоритми, що використовуються для реалізації інструмента також зазначені в розділі.

В даній дисертаційній роботі використовуємо Java як мову програмування для розробки програмного забезпечення з Java SE Development Kit 8 в Intelij Idea як IDE (Integrated Development Environment). Характеристики ПК, на якому виконувалась розробка, наступні:

1. Операційна система: Windows 10 Enterprise
2. Процесор: Intel (R) Core (TM) i5-6200U процесор 2,40 ГГц
3. ОЗП (Random Access Memory): 8.00 ГБ
4. Тип системи: 64-розрядна операційна система

3.1 Бібліотеки для імплементації

Для реалізації методів обчислення міри неконсистентності OWL 2 онтології необхідні бібліотеки, які будуть описані в даному пункті.

3.1.1 OWL API

OWL API (Application Programming Interface) використане для розробки ПЗ для обробки OWL2 онтологій. OWL API має відкритий віхідний код та може бути використане для створення і керування онтологіями OWL та OWL2.

Остання версія цього API зосереджена на OWL2. API надає методи для завантаження, обробки та збереження OWL файлів. У даній роботі використовуємо версію 3.5.0 OWL API.

3.1.2 OWL Explanation

OWL Explanation – бібліотека OWL, побудована для розробки

обґрунтування. Ми можемо отримати тип пояснення для втілення в онтології. Оскільки ця бібліотека дозволяє отримати мінімальну підмножину онтології, достатню для обґрунтування, використовуємо її, щоб знаходити мінімально неконсистентні підмножини онтологій. Це неохідно для деяких методів обчислення міри невідповідності, які озглянуті в розділі 2, що використовують підмножини. В даній роботі використовуємо версію OWL Explanation 1.1.2.

3.1.3 OWL Reasoner

Reasoner грає важливу роль у розробці онтології, написаної в OWL. Головна функція Reasoner – формування відповідей на запити для знаїодження певних класів та індивідів, які відповідають певному запиту. Крім того, Reasoner допомагає користувачам інтегрувати декілька онтологій та зберегти їх якість.

Приклад 18. На вхід подається OWL-файл. Наступний програмний коди виводить множину мінімально неконсистентних підмножини.

```
inputOntologyFile = new input file
rf = new ReasonerFactory
manager = createOWLOntologyManager
df = getOWLDataFactory with manager
ontology = loadOntologyFromOntologyDocument with manager from
inputOntologyFile
reasoner = createReasoner of ontology with rf
explainInconsistency=new
InconsistentOntologyExplanationGeneratorFactory of rf to
createExplanationGenerator of ontology
explanationsOfMIKSet = getExplanations of df with explainInconsistency
Show explanationsOfMIKSet
```

Для реалізації методів обчислення міри невідповідності онтологій в даній роботі використовуємо HermiT і JFact як Reasoner. Обидва вони мають настпні функції: реалізація, класифікація, здійсненність, втілення та узгодженість. Використовуємо два Reasoner для перевірки ефективності

методів з різними Reasoner. Нижче наведено пояснення причини вибору цих OWL-Reasoner.

3.1.3.1 Hermit

HermiT – це OWL 2 Reasoner, який можна запустити в командному рядку або OWL API. Оскільки він підтримує функції оцінки неконсистентності, він відомий як один з небагатьох Reasoner, яикй повністю підтримує OWL2. Тому в даній роботі розглядаємо цей Reasoner для побудови інструменту вимірювання міри невідповідності.

3.1.3.2 JFact

Інший OWL-Reasoner, який використовуємо в даній роботі, – JFact. Це реалізація Fact ++ на Java як відкритого джерела C ++ на основі табличного Reasoner для OWL 2. JFact можна використовувати разом з OWL API для підтримуваного інтерфейсу. Однак Fact ++ може бути запущений з командного рядка або OWL API. Використовуємо JFact 1.2.1 для побудови інструменту вимірювання міри невідповідності.

3.1.4 Інші бібліотеки

Крім описаних вище, використовуються також наступні бібліотеки:

1. Telemetry

Telemetry використовується для запису вкладених даних, які можуть бути корисними для використання OWL Explanation. Використовуємо Telemetry 1.0.0, яка побудована під тим самим розробником, що і OWL Explanation.

2. Apache Commons Lang

Оскільки стандартні бібліотеки Java відомі через відсутність достатніх методів для маніпулюванняо основними класами, додаємо бібліотеку Apache Commons Lang. Це пакет з класами утиліти Java для класів, що знаходяться в ієрархії java.lang, наприклад, java.lang.Exception, для обробки винятків.

3. Java Collections Framework

Колекції в Java використовуються через Java Collections Framework.

Це набір інтерфейсів та класів, які можуть бути корисними для збору, зберігання та обробки даних, наприклад множина і список. Ця бібліотека використовується для зберігання аксіом.

3.2 Алгоритми реалізації

У цьому розділі описані деякі основні алгоритми для побудови ПЗ: булеан бази знань, мінімальна непослідовність, максимальна послідовність і мінімальна корекція підмножини. Кожен з них розглянемо нижче.

3.2.1 Алгоритм пошуку булеану бази знань

Деякі з описаних методів обчислення міри невідповідності онтології потребують доступу до підмножин бази знань. Всі можливі підмножини бази знань повинні бути перераховані, щоб перевірити, які з них відповідають вимогам обчислень.

Визначення 30. Нехай K – база знань, яка складається з множини аксіом. Булеан бази знань K – це множина всіх підмножин K, включаючи пусту множину та безпосередньо K.

Кількість підмножин K досягає значення 2^n.

Приклад 19. Використовуємо базу знань *K1* з попередніх прикладів.

$K1$ = {ProductOwner ⊑ Employee ⊓ ¬Developer, Employee ⊑ Developer ⊓ Person, ProductOwner(tom)}

Булеан $K1$:

{{}, {ProductOwner ⊑ Employee ⊓ ¬Developer}, {Employee ⊑ Developer ⊓ Person}, {ProductOwner(tom)}, {ProductOwner ⊑ Employee ⊓ ¬Developer, Employee ⊑ Developer ⊓ Person}, {Employee ⊑ Developer ⊓ Person, ProductOwner(tom)}, {ProductOwner ⊑ Employee ⊓ ¬Developer, ProductOwner(tom)}, {ProductOwner ⊑ Employee ⊓ ¬Developer, Employee ⊑ Developer ⊓ Person, ProductOwner(tom)}}

Як описано в попередньому розділі методи обчислення міри невідповідності онтології, що використовують булеан бази знань наступні: MC-міра невідповідності, nc-міра невідповідності, міра невідповідності I_{DMCS}

та міри невідповідності MI-Шеплі. Таким чином, алгоритм булеану бази знань використовується для цих методів.

Алгоритм 1. Алгоримт пошуку булеану бази знань
Input: Set of axioms in knowledge base (stored in originalSet) Output: Powerset of the knowledge base 1: procedure POWERSET(ORIGINALSET) 2: sets = new HashSet of OWL axiom set 3: if originalSet is empty then 4: add new HashSet to sets 5: return sets 6: end if 7: list = new ArrayList(originalSet) 8: head = get first element in list 9: rest = new HashSet(subList of list) 10: for set in powerSet(rest) do 11: newSet = new HashSet 12: add head to newSet 13: addAll set to newSet 14: add newSet to sets 15: add set to sets 16: end for 17: return sets 18: end procedure

3.2.2 Алгоритм мінімальної неконсистентності

11 із 15 методів обчислення міри невідповідності онтології використовують мінімально неконсистентні підмножини бази знань, тобто $MI(K)$. Ці методи описані в розділі 2.1.2, mv-міра невідповідності, і згадані в розділі 2.2. Використовуємо алгоритм 2 для пошуку мінімально неконсистентних підмножин.

Алгоритм 2. Алгоритм пошуку мінімально неконсистентних підмножин
Input: Ontology axiom set Output: Set of minimal inconsistent subsets 1: procedure 2: Compute powerset of the ontology 3: Generate each subset of the powerset 4: if Subset M is inconsistent then 5: Search whether there is M’ M 6: if M0 is consistent then 7: M is minimal inconsistent subset 8: return M 9: end if 10: end if 11: end procedure

3.2.3 Алгоритм максимальної консистентності

Методи обчислення міри невідповідності онтології, що використовують максимально консистентні підмножини – це MC-міра невідповідності і nc-міра невідповідності. Кожна з них відноситься до різних визначень максимально послідовної підмножини. Нижче описаний Алгоритм 3 для пошуку максимально послідовних підмножин для MC-міри невідповідності та Алгоритм 4 для пошуку максимально послідовних підмножин nc-міри невідповідності. Алгоритм 3 визначає максимальну консистентність на основі послідовних підмножин, які не мають консистентних множин з підмножин. Це робиться шляхом збору послідовних підмножин у списку і видалення тих, що не є максимальними. Алгоритм 4 визначає максимальну консистентність на основі максимального розміру зібраних послідовних підмножин.

Алгоритм 3. Алгоритм пошуку максимально консистентних підмножин для MC-міри невідповідності
Input: List of axiom sets (stored in MCKcandidate) Output: Set of maximal consistent subsets 1: procedure ELIMINATE_NOTMCK(MCKCANDIDATE) 2: MCK = new HashSet of OWL axiom set 3: Sort MCKcandidate 4: for i = 0 to size of MCKcandidate do 5: candidateI = get i from MCKcandidate 6: boolean flag = true 7: for MCKcand in MCK do 8: if MCKcand containsAll candidateI then 9: flag = false 10: break 11: end if 12: end for 13: if flag then 14: add candidateI to MCK 15: end if 16: end for 17: return MCK 18: end procedure

Алгоритм 4. Алгоритм пошуку максимально консистентних підмножин для nc-міри невідповідності
Input: Set of consistent subsets of knowledge base Output: Maximal consistent subsets size 1: procedure 2: Get the size of each subset 3: Collect the size of the consistent subsets 4: Get the maximum size of the subsets 5: end procedure

3.2.4 Алгоритм підмножини з мінімальною корекцією

У цьому підрозділі описуються алгоритми пошуку мінімальної підмножини корекції (MCS), що використовується мірою невідповідності I_{DMCS}. Алгоритми поділяються на дві частини: алгоритм перевірки консистентності всіх виділених підмножин і основний алгоритм для отримання MCS.

Алгоритм 5. Пошук списку, в якому містяться неконсистентні підмножини
Input: List of values with value "true" for consistent subset and "false" for inconsistent subset (stored in arrayList) Output: Yes for all values of subsets: "false", otherwise no 1: procedure DOESLISTCONTAINALLFALSE(ARRAYLIST) 2: String falseValue = "false" 3: for str in arrayList do 4: if str is not equal to falseValue then 5: return false 6: end if 7: end for 8: return true 9: end procedure

Алгоритм 6. Обчислення мінільної підмножини корекції

```
Input: Ontology axiom set
Output: Minimal correction subsets
1: procedure
2: ontologyAxiomSet = new HashSet of OWLAxiom
3: for subset M in powerSet of ontologyAxiomSet do
4: AxiomOntology = createOntology
5: remove all axioms of AxiomOntology
6: add all axioms of ontologyAxiomSet to AxiomOntology
7: remove subset M in AxiomOntology
8: reasoner = createReasoner(AxiomOntology)
9: Show whether ontology minus subset M consistent?
10: AxiomOntology2 = createOntology
11: remove all axioms of AxiomOntology2
12: add all axioms of ontologyAxiomSet to AxiomOntology2
13: consistentValue = new ArrayList of string
14: for Ci in subset M do
15: remove subset M in AxiomOntology2
16: add axiom Ci to AxiomOntology2
17: reasoner2 = createReasoner(AxiomOntology2)
18: Show whether ontology minus (subset M minus axiom Ci) consistent?
if ontology minus (subset M minus axiom Ci) is consistent then
20: add "true" to consistentValue
21: else
22: add "false" to consistentValue
23: end if
24: if ontology minus subset M is consistent then
25: if doesListContainAllFalse(consistentValue) == true then
26: add subset M to MCSes
27: end if
28: end if
29: end for
30: end for
31: end procedure
```

Висновки до розділу 3

1. Проведено порівняльний аналіз існуючих та запропонованого способів обчислення міри невідповідності онтології та виявлено, що час виконання обчислення пропорційний розміру онтології, а у випадку запропонованого MI-Шеплі способу час виконання залежить від кількості мінімально неконсистентних підмножин онтології.

2. Розроблений спосіб обчислення міри неконсистентності, який використовує особливості Java бібліотек, розроблених для OWL2 онтологій, що можуть бути використані для побудови системи обчислення міри невідповідності онтології різними способами.

РОЗДІЛ 4. ТЕСТУВАННЯ ТА ОЦІНКА ОТРИМАНИХ РЕЗУЛЬТАТІВ

У цьому розділі зазначена оцінка розробленого програмного забезпечення. Тести виконуються на комп'ютері з специфікацією, зазначеною в попередньому розділі. Результати експерименту також описані в даному розділі.

4.1 Тестові випадки

Були обрані 25 OWL-файли для тестових випадків, взяті з бібліотеки DL SROIQ онтологій. Тестові випадки містять суперечливі онтології. Оскільки ми будемо використовувати їх для показу оцінок у наступному розділі, вони перераховані і зображені в таблиці 4.1. Всі наступні таблиці та графіки в даній роботі стосуються порядку онтологій, перерахованих в таблиці 4.1. Наприклад, якщо посилання на № 10 в діаграмі або в іншій таблиці, це буде посилання на онтологію Rdfbased-sem-char-irreflexive-inst.

Таблиця 4.1 Тестові випадки

Номер	Ім’я онтології
1	DisjointClasses-002
2	New-Feature-AsymmetricProperty-001
3	New-Feature-BottomObjectProperty-001
4	New-Feature-IrreflexiveProperty-001
5	New-Feature-NegativeObjectPropertyAssertion-001
6	New-Feature-TopObjectProperty-001
7	Rdfbased-sem-bool-complement-inst
8	Rdfbased-sem-char-asymmetric-inst
9	Rdfbased-sem-char-asymmetric-term
10	Rdfbased-sem-char-irreflexive-inst
11	Rdfbased-sem-class-nothing-ext

Продовження таблиці 4.1	
12	Rdfbased-sem-eqdis-different-sameas
13	Rdfbased-sem-eqdis-disclass-eqclass
14	Rdfbased-sem-eqdis-disclass-inst
15	Rdfbased-sem-ndis-alldifferent-fw
16	Rdfbased-sem-ndis-alldifferent-fw-distinctmembers
17	Rdfbased-sem-ndis-alldisjointclasses-fw
18	TestCase:WebOnt-I4.5-002
19	TestCase:WebOnt-Nothing-001
20	TestCase:WebOnt-Restriction-001
21	TestCase:WebOnt-Restriction-002
22	TestCase:WebOnt-Thing-003
23	TestCase:WebOnt-description-logic-001
24	TestCase:WebOnt-description-logic-002
25	TestCase:WebOnt-description-logic-003

4.2 Вимірювання

Онтологічні тестові випадки, описані в попередньому розділі, подаються як вхідні дані до інструменту вимірювання. Отримуємо результати оцінювання, які будуть розглянуті в даному розділі. Для того, щоб оцінити час роботи методів обчислення, проводимо експеримент з тестування “чорних ящиків” як звичайний тип тестування програмного забезпечення, який вивчає функціональність інструменту, фокусуючись на вхідних та вихідних даних. Проведені два види тестування. По-перше, використовуємо тестові випадки для тестування системи для OWL-reasoner: HermiT і JFact. Результати цього тестування представлені в першому підрозділ. По-друге, проводимо пестування продуктивності, оскільки будемо оцінювати час роботи засобу

вимірювання з урахуванням розмірів онтологічних тестів і значення невідповідності. У другому підрозділі будуть показані результати цього тестування. Крім того, представлена статистика тестових випадків та кількісного значення невідповідності.

4.2.1 Час роботи

Оскільки використовуємо два типи OWL-reasoner, зобразимо як час роботи програмного забезпечення залежить від розміру онтологій і від їх невідповідності.

4.2.2.1 Порівняння з розміром онтології

Порівняння часу роботи з розміром онтологій (тобто кількістю логічних аксіом в онтологіях) представлено на графіку, наведеною на рисунку 4.1. Графік представлений з набором онтологій з найменшого розміру до найбільшого розміру онтологій, тобто в зростаючому порядку. Пік часу виконання отриманий за номером 25 (TestCase: WebOnt-description-logic-003) з розміром 15 аксіом. Для виконання обчислень необхідно 5400 секунд і 5160 секунд для HermiT і JFact, відповідно. Нижчий пік часу виконання отримується при обробці онтології № 18 (TestCase: WebOnt-I4.5-002) з розміром 13 аксіом, що потребує 675.225 секунд і 645.464 секунди для роботи з HermiT і JFact, відповідно.

Найменший час виконання – це, як правило, близько 10 секунд, який необхідний для обробки однієї аксіоми. Ці дані про час роботи відпосяться до онтологій 1, 2, 3 та 4. За цим графіком видно, що чим більший розмір онтології, тим вище час її обробки (рисунок 4.1).

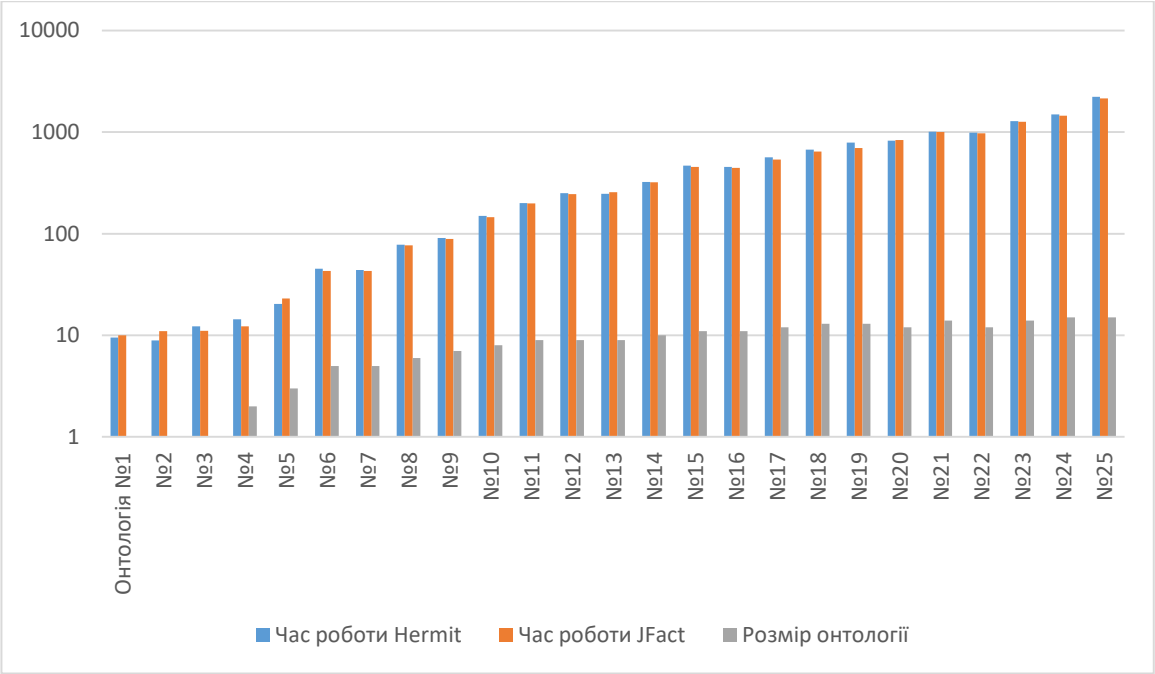


Рисунок 4.1. – Порівняння часу роботи та розміру онтологій.

4.2.2.2 Порівняння зі значенням невідповідності

Порівняємо час роботи прогрманого забезпечення та значення невідповідності на основі онтологій та аксіом для 25 баз знань. За цією оцінкою видно, наскільки пов'язане значення невідповідності онтологій і час виконання обробки з HermiT та Jfact. Усі наведені нижче графіки відображають дані у таблиці 2 додатку 3. Порядок онтологій в діаграми зберігатиме висхідний порядок розміру онтологій, як у попередньому підрозділі.

Міри невідповідності на основі онтологій

Вимоги щодо невідповідності на основі онтологій будуть перелічені в наступних пунктах, щоб дати оцінку кожному з них щодо часу роботи та значення невідповідності

Критичне вимірювання невідповідності

Оскільки ми розглядаємо лише неконсистентні онтології, всі їх критичні занчення невідповідності будуть 1. Оскільки всі значення константні, цей метод не має суттєвого впливу на тривалість роботи. Вимірювання з HermiT і JFact дійсно однакові, отже, це стосується обох обробників (Рисунок 4.2).

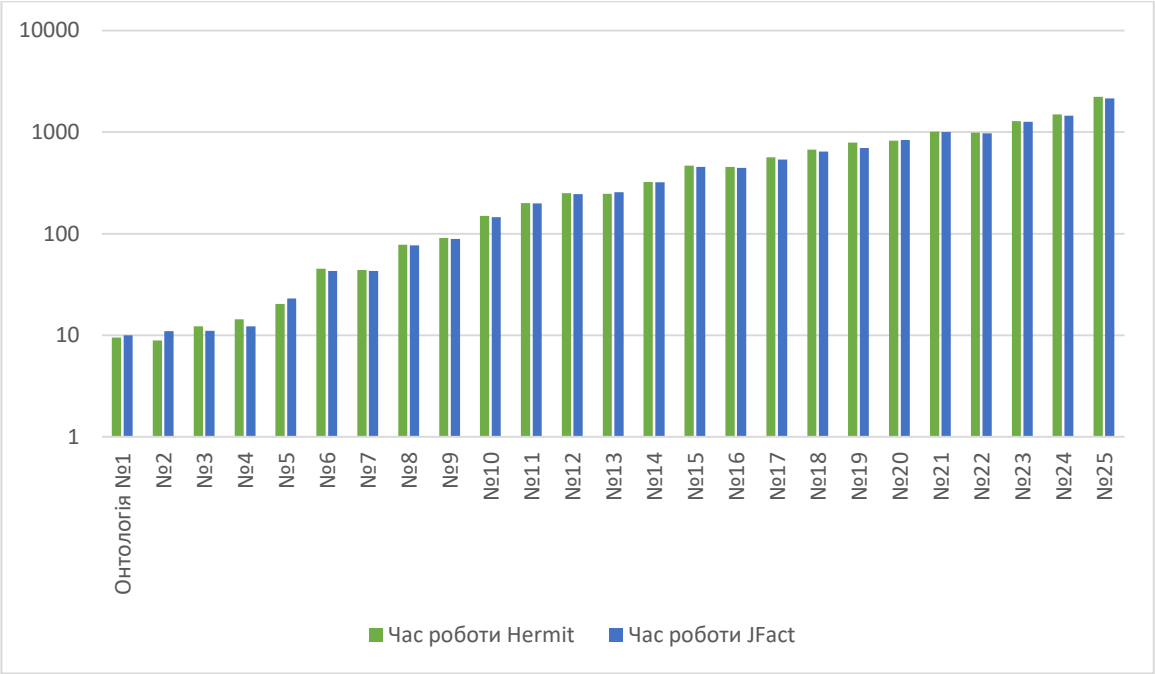


Рисунок 4.2. – Порівняння часу роботи для обчислення критичної міри невідповідності

Мі-міра невідповідності

MI-міра невідповідності, як відомо, є мірою, яка використовує мінімально непослідовні підмножини, тобто $MI(K)$. На діаграмі 4.3 показано, що хоча час роботи коливається, значення невідповідності змінюється небагато. Або ж з HermiT або з JFact, обчислення розміру підмножин не сильно впливає на час роботи. Отже, в цьому пункті ми стверджуємо, що MI-міра невідповідності також не дає значного впливу на час роботи.

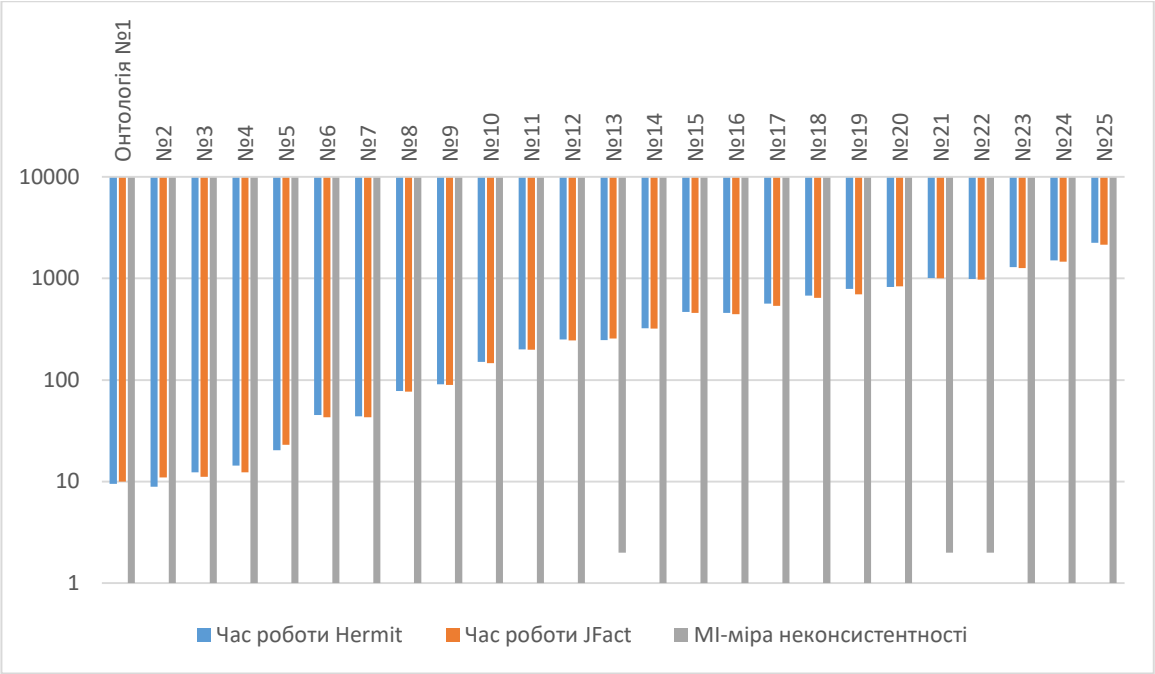


Рисунок 4.3. – Порівняння часу роботи для обчислення MI-міри невідповідності

MI^{C}-міра невідповідності

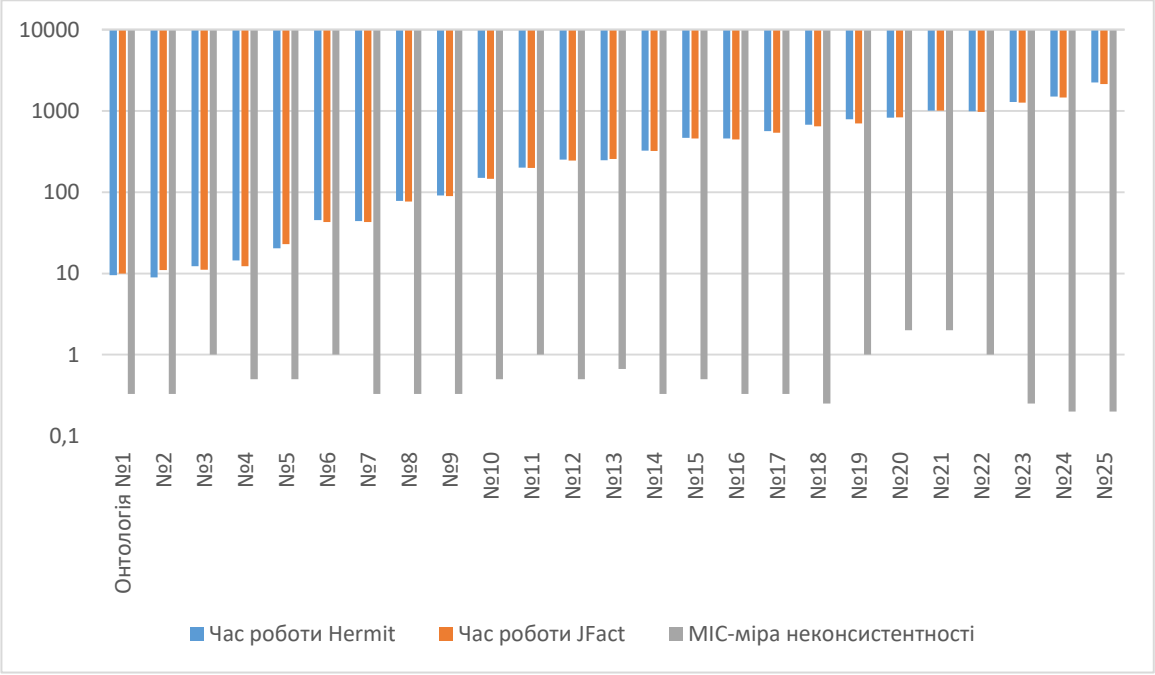


Рисунок 4.4. – Порівняння часу роботи для обчислення MI^{C}-міри невідповідності

MI^{C}-міра невідповідності вимагає доступу до $MI(K)$ в процесі обчислення.

Графік на рисунку 4.4 показує, що значення невідповідності, отримані за допомогою цього показника, впливають на час роботи. Видно, що більше значення невідповідності онтології обчислюється за менший час. Обидва обробники як HermiT, так і JFact, мають однакове відображення онтологій, для яких обчислюється MI^C-міра невідповідності.

D^f-міра невідповідності

На рисунку 4.5 представлений результати обчислення D^f-міри невідповідності. Найменші значення невідповідності онтологій спостерігаються на онтологіях 18 та 25. Хоча час роботи алгоритму для цих баз знань має найбільше значення. За діаграмою видно, що нижча ступінь міри невідповідності онтології збільшує час виконання.

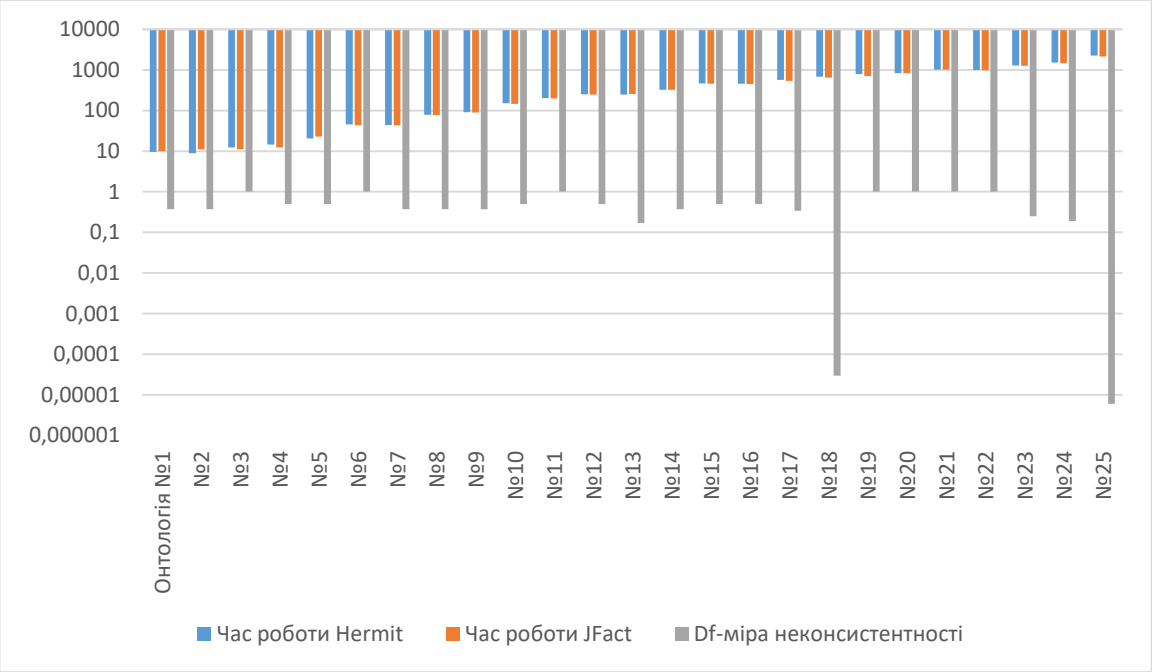


Рисунок 4.5. – Порівняння часу роботи та значення D^f -міри невідповідності.

Проблемна міра невідповідності

Як правило, час виконання адгоритму обчислення проблемної міри невідповідності не залежить від стіпені невідповідності. Це можна побачити на діаграмі на рисунку 4.6, яка зображує результати роботи HermiT та Jfact.

Коефіцієнт несумісності для міри невідповідності

Коефіцієнт несумісності для міри невідповідності також використовує

MI(*K*), тому ступінь несумісності не впливає на час роботи. Великий розмір онтології не означає, що вона має більше мінімально неконсистентних підмножин. Причому, коли ступінь неконсистентності висока, час роботи не обов'язково високий. У більшості тестових випадків високий ступінь невідповідності передбачає низький час роботи (рисунок 4.7).

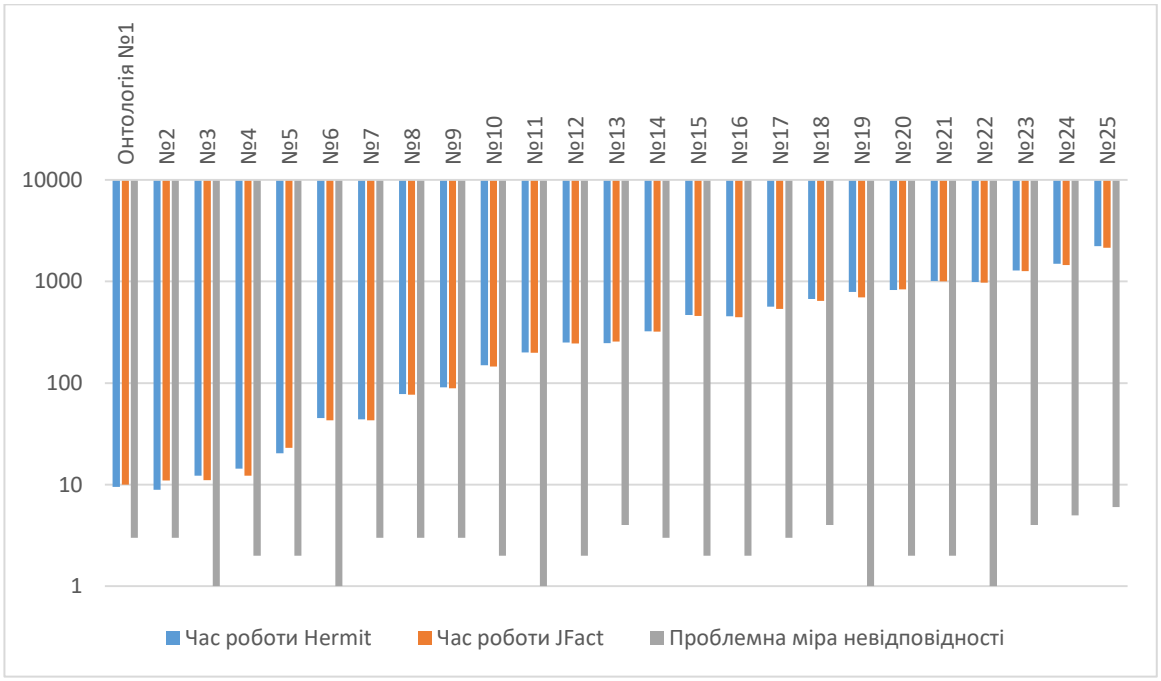


Рисунок 4.6. – Порівняння часу роботи та значень проблемної міри невідповідності.

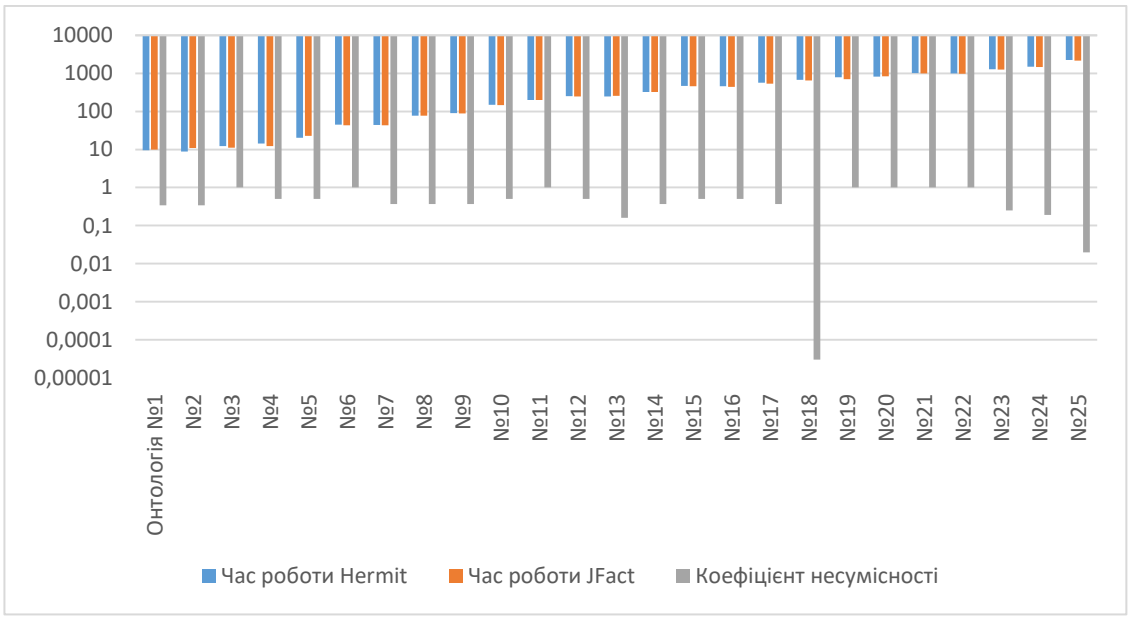


Рисунок 4.7. – Порівняння часу роботи та значень коефіцієнту несумісності для міри невідповідності.

MC-міра невідповідності

Ступінь невідповідності онтології, обчислена за допомогою MC-міри невідповідності, не є прямо пропорційною до часу виконання цього методу. Це видно з графіка на рисунку 4.8, де час роботи зростає, але ступінь невідповідності залишається незмінною.

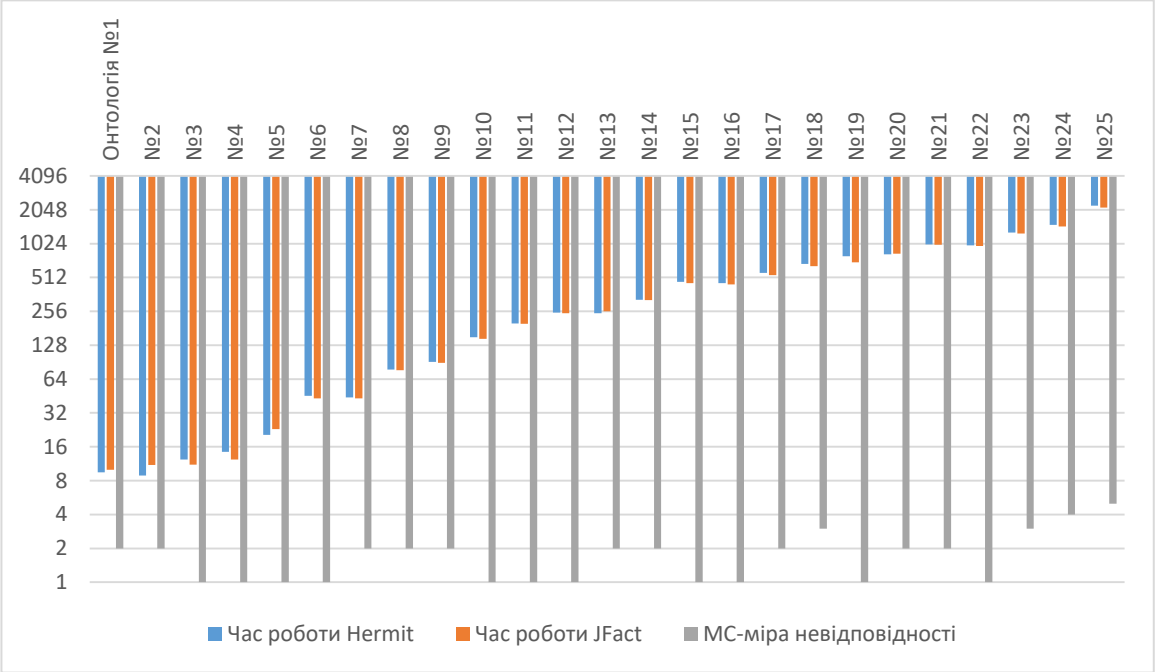
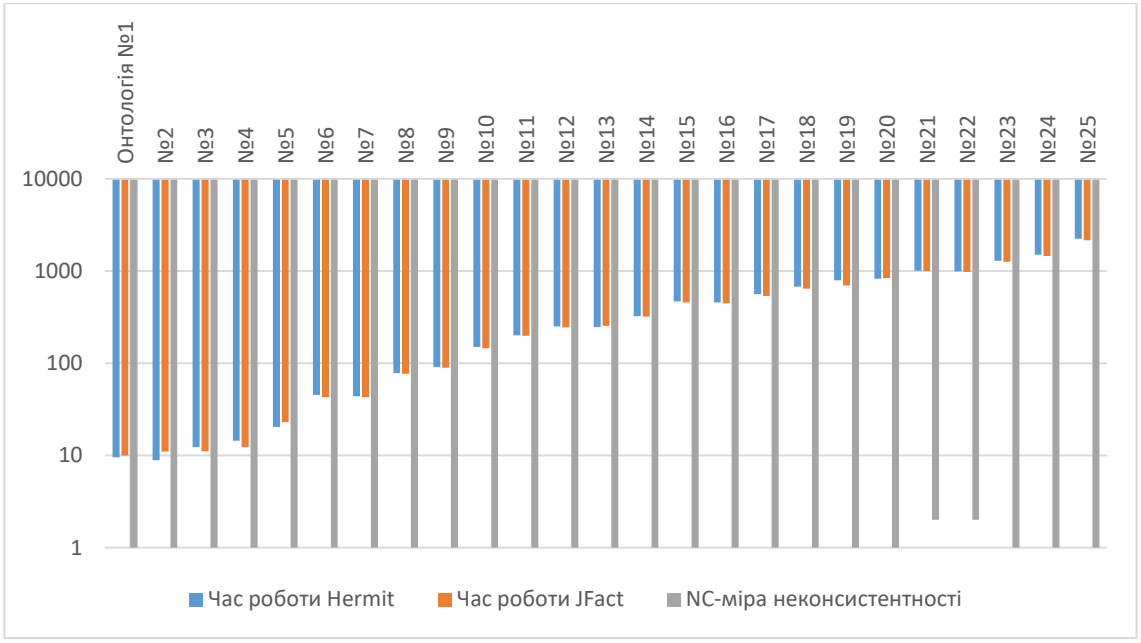


Рисунок 4.8. – Порівняння часу роботи та значень MC-міри невідповідності

NC-міра невідповідності

Рисунок 4.9 показує, що nc-міра невідповідності обчислює значення консистентності, майже однакові для всіх онтологій. Час обчислення для HermiT та JFact також суттєво не відрізняється.



нок 4.9. – Порівняння часу роботи та значень NC-міри невідповідності

MV-міра невідповідності

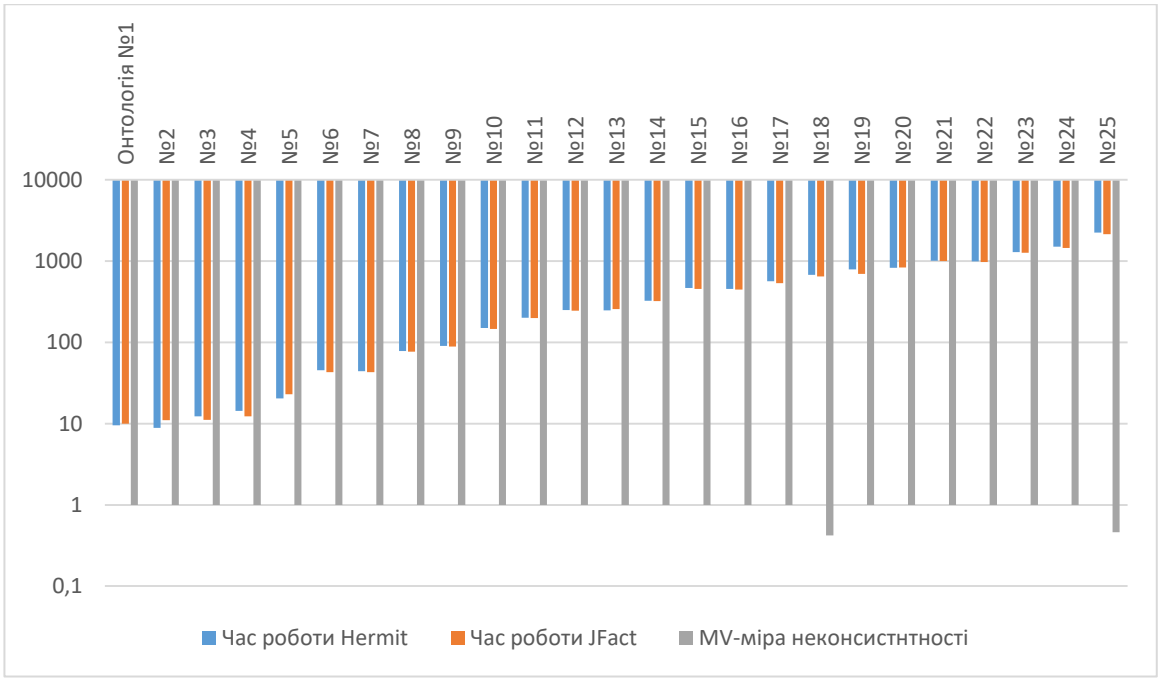


Рисунок 4.10. – Порівняння часу роботи та значень з MV-міри невідповідності

Тривалість роботи для обчислення MV-міри невідповідності буде тим меншою, ступінь неконсистентності онтології вища. Це показано на рисунку 4.10. Крім того, максимальне значення несумісності (тобто значення 1) можна

отримати в середньому за 2 секунди.

ID_{MCS}-міра невідповідності

Пік часу роботи, отриманий для онтології №25 не супроводжується піком значення невідповідності онтології. Як правило, для ID_{MCS}-міри невідповідності час роботи не залежить від ступеню невідповідності. Це справедливо і для HermiT, і для JFact, що показано на графіку на рисунку 4.11.

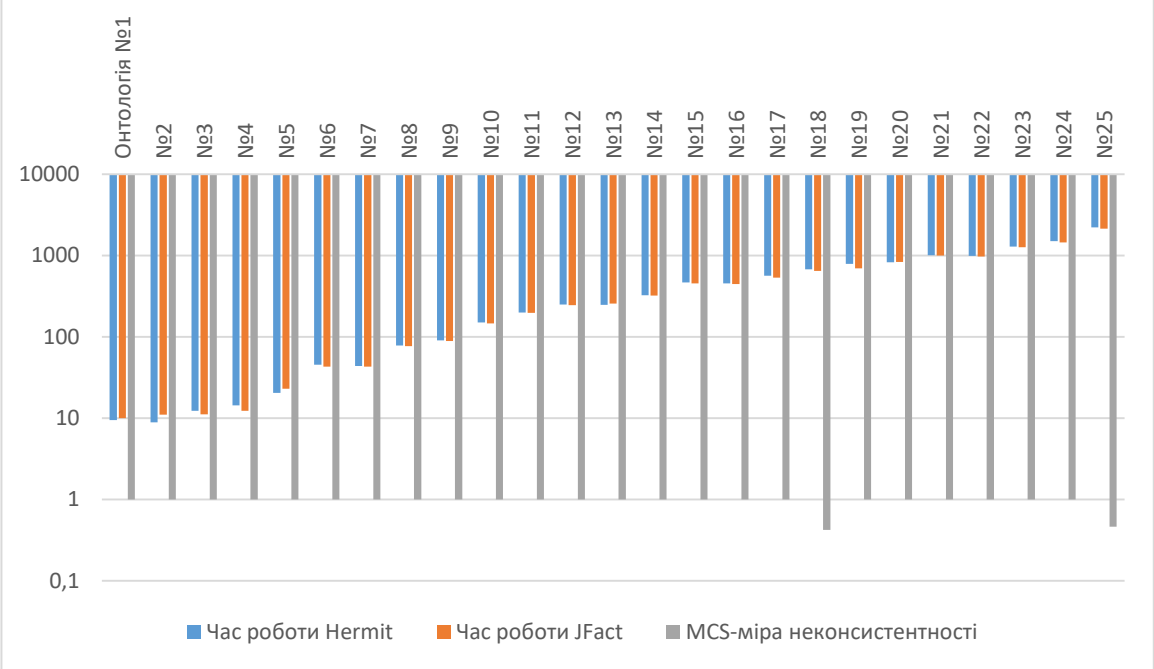


Рисунок 4.11. – Порівняння часу роботи та значення ID_{MCS}-міри невідповідності

Міра неконсистентності на основі аксіоми

Методи обчислення міри невідповідності на основі аксіоми складаються з 5 алгоритмів, кожен з яких використовує $MI(K)$. Крім того, обчислення MIV_D, $MIV_{\#}$, MIV_C збігаються з розрахунками кричичної міри невідповідності, MI та MI_C відповідно. Відтак характеристики MIV_D-міри невідповідності відповідає характеристикам критиіної міри невідповідності I_d , які описані з попередніх пункцтах. Так само як і характеристики $MIV_{\#}$ та MIV_C, які відповідають MI та MI_C. На основі прикладів обчислення міри невідповідності MI-Шеплі у попередньому розділі можна зробити висновок, що значення S^I_a має ті ж показники, що і MIV_C. Проте значення міри неконситентності MI-Шеплі відрізняється від отриманих раніше. MI-Шеплі та MIV_C відрізняються для

деяких аксіом, наприклад, в онтології № 18.

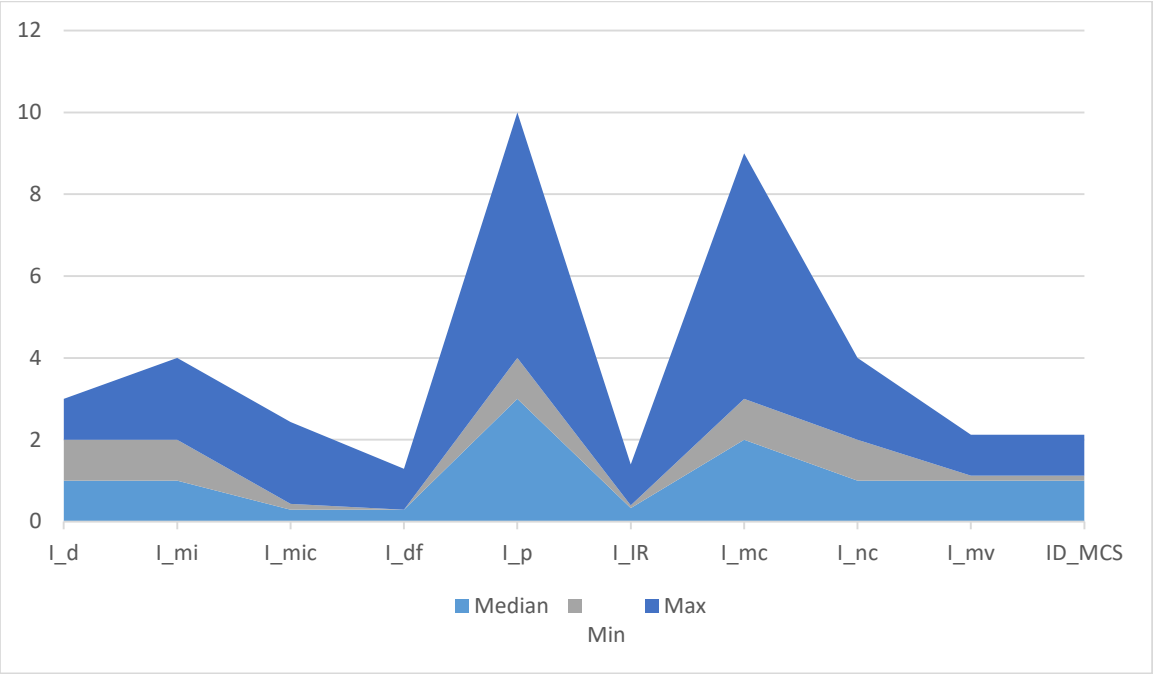
Значення розрахункової міри невідповідності аксіом I_S, такі ж, як значення $MIV_{\#}$, за винятком значень 15 аксіом з онтології № 25, які не можуь бути розрахові цим методом.

Отже, загалом значення I_S відповідають $MIV_{\#}$, де значення невідповідності не є прямо пропорційними до часу виконання. Більш детальну інформацію про неконсистентність на основі аксіоми можна отримати в таблиці B.1.

4.2.3 Описова статистика тестових випадків

В даному розділі описуються основні особливостей збіру даних з тестових випадків, які включають мінімуми, максимуми, середні значення, медіану, моду та діапазон. Найменше та найбільше значення серед зібраних даних приймається як мінімум і максимум, відповідно. Середнє значення – середнє значення зібраних даних, яке розраховується як сума даних, поділена на загальну кількість зібраних точок даних. Мода – це значення, яке найчастіше з'являється в наборі даних. Медіана – це середнє значення в наборі даних, яке отримується шляхом перерахування даних за зростанням та вибором значення в середині списку. Всі показники зображені на діаграмах нижче, де вони представляють статистику для значенб невідповідності на основі онтологій та аксіом.

Рисунок 4.12 показує, що проблемна міра невідповідності I_p має найбільше значення медінаи з більшості статистичних даних, за якою слідує MC-міра невідповідності міри I_{MC}. На рисунку 4.13, що відображає статистичні дані для методів обчислення міри невідповідності на основі аксіоми, показано, що мінімум для всіх методів – значення 0. Q1 і Q3 у діаграмах означає перший і третій квартелі даних, відповідно.



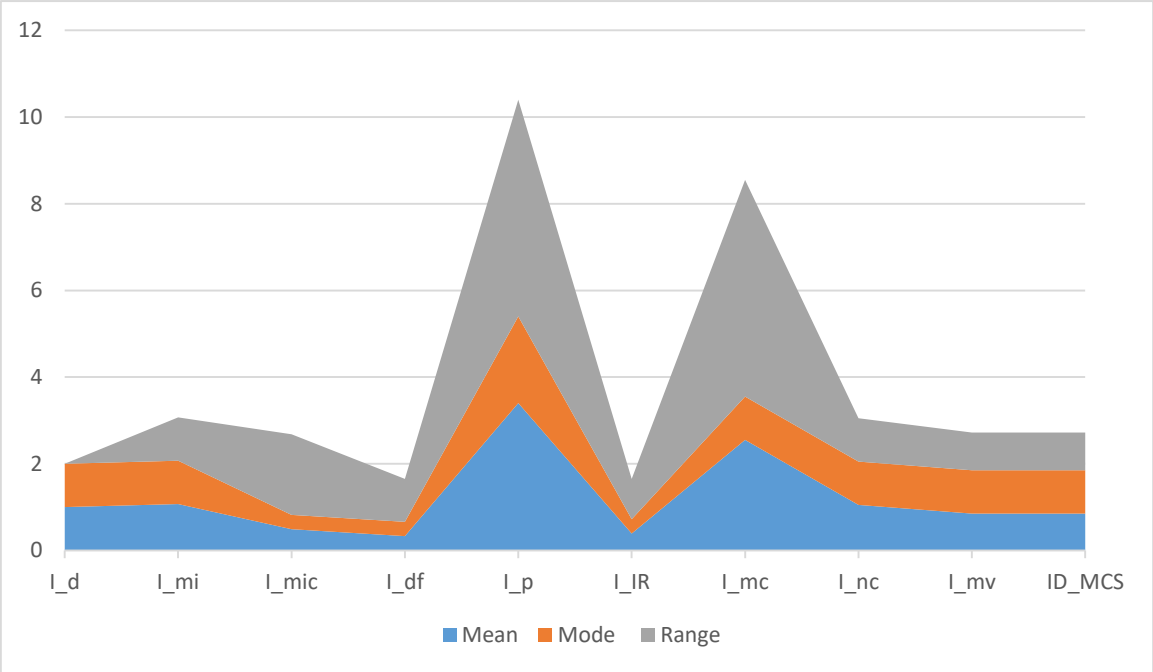


Рисунок 4.12. – Описова статистика тестових випадків для вимірювання невідповідності на основі онтологій.

Як видно, мінімум належить методам на основі аксіоми, найбільші значення невідповідності отримані для I_d, I_{MI}, I_p, I_{MC} та I_{nc}. Найменший максимум отриманий для I_d, I_{Df}, I_{IR}, I_{mv}, ID_{MCS}, MIV_D, MIV_C і S^I_a. Найменше і найбільше значення належить MIV_C та I_p, відповідно. Найменший середній показник – MIV_C і S^I_a, тоді як найбільша медіана – I_p. Найменша і найбільша мода отримана для тих ж методів, що і найбільша і найменша медіана.

Найменший і найбільший діапазон у методів I_d і I_p, відповідно (рисунок 4.13).

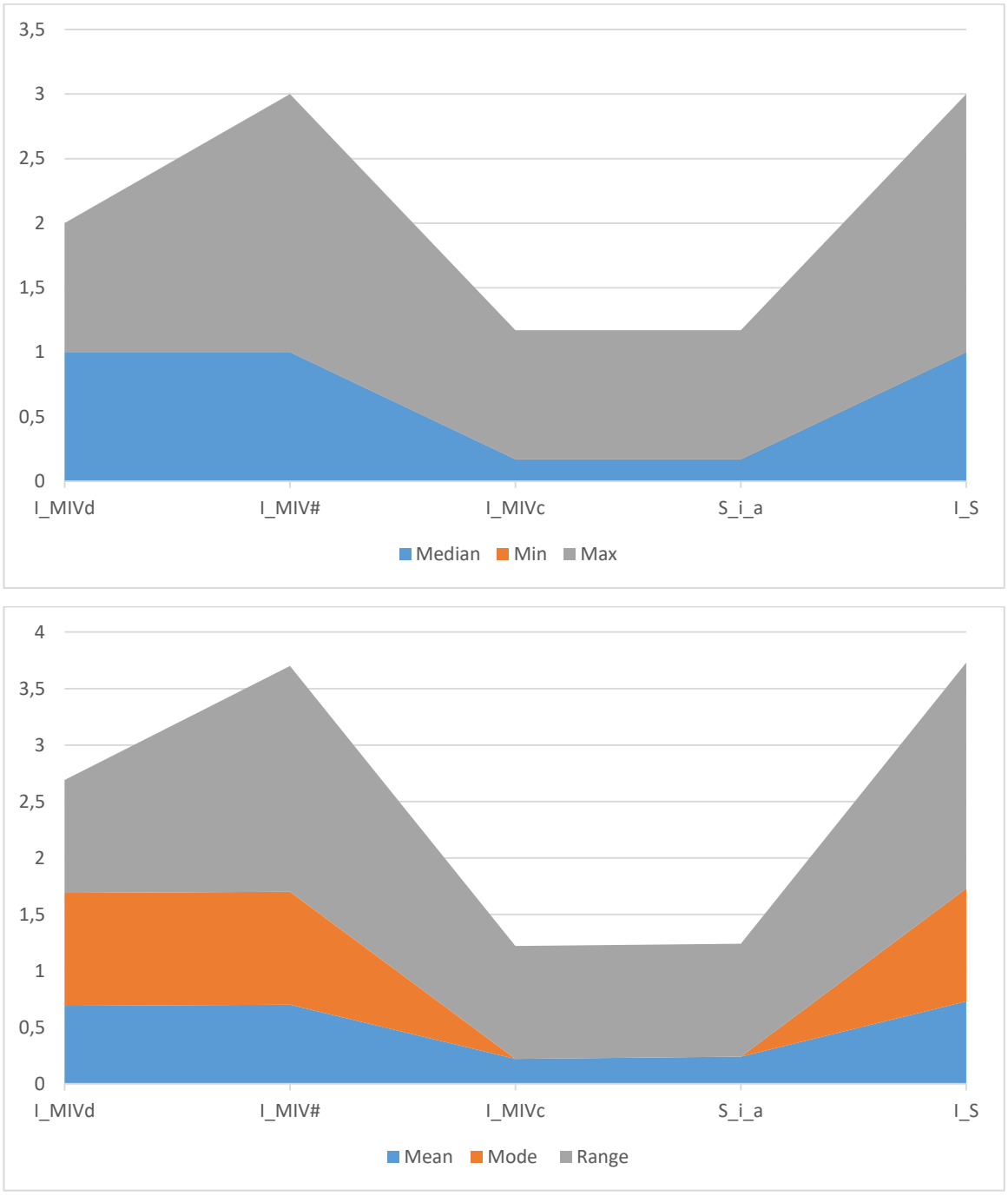


Рисунок 4.13. – Описова статистика тестових випадків для методів обчислення міри невідповідності на основі аксіоми.

4.2.4 Значення міри невідповідності

У цьому пункті розглянемо отримані за допомогою описаних методів значення невідповідностей. Оскільки розглянуто 15 підходів, перерахуємо результати для кожного з них. 23 з 25 тестові випадки є неконсистентними онтологіями, які можуть дати вихідні значення, тому розглянемо лише ці 23

онтології. Для оцінки методів обчислення міри невідповідності на основі аксіоми є 83 аксіоми для 23 онтологій, які будуть описані в даному розділі. Дані по всім наведеним нижче графікам представлені в додатках: таблиці А.6 – А.20.

4.2.4.1. Критична міра невідповідності

Критична міра невідповідності має значення лише 0 та 1 для консистентної та неконсистентної онтології відповідно. Оскільки онтології в усіх 23 тестових випадках є непослідовними, всі вони мають значення 1 критичної міри невідповідності. На рисунку 4.14 показана діаграма занчень.

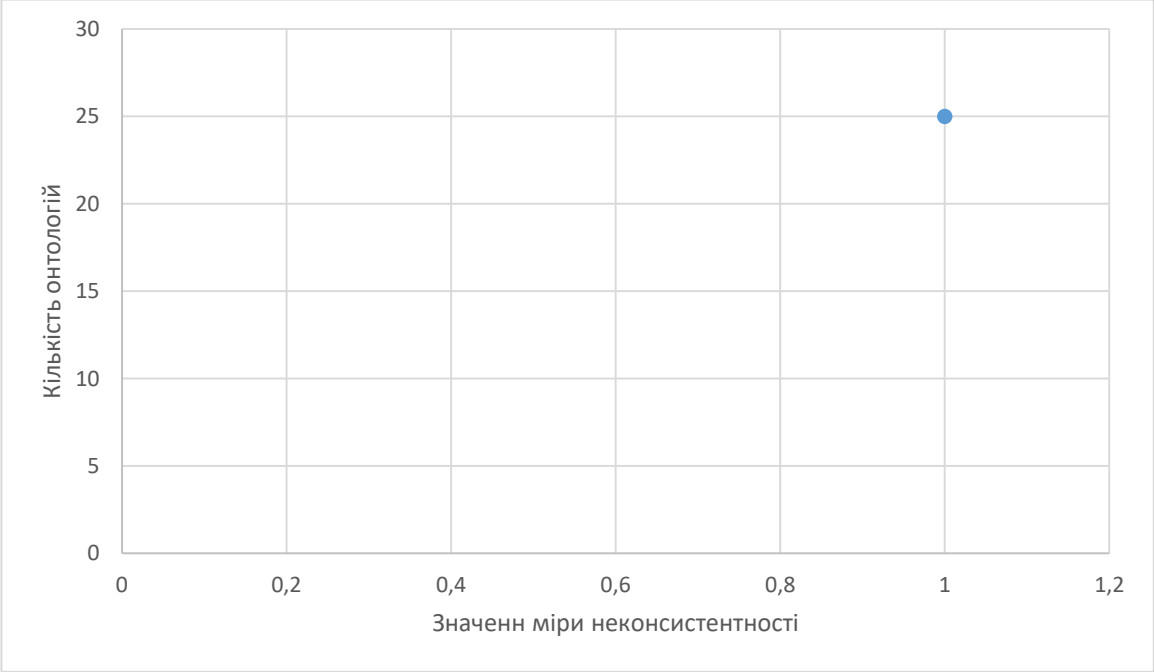


Рисунок 4.14. – Значення критичної міри невідповідності для тестових випадків.

4.2.4.2 MI-міра невідповідності

Оскільки онтології в усіх 23 тестових випадках є непослідовними, кожена з них має мінімально неконсистентні підмножини, тобто $MI(K)$. Серед 23 тестів, для 18 онтологій розмір $MI(K) = 1$ та для 5 онтологій – $MI(K) = 2$. Це відображено на діаграмі 4.15.

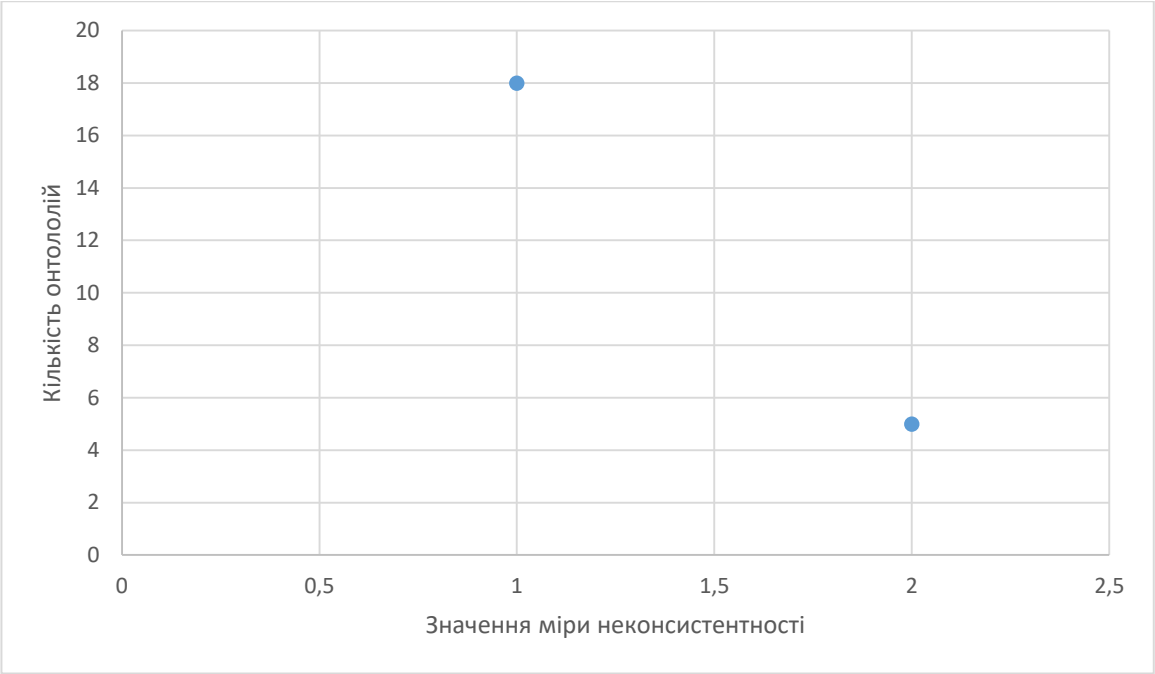


Рисунок 4.15. – Значення MI-міри невідповідності для тестових випадків.

4.2.4.3 MI^C- міра невідповідності

Метод MI^C невідповідності має більше різновидів значень, ніж попередні два підходи. За цим метод значення неконсистентності однієї онтології – 0.667, дев'яти – 0.334. Це відображено на рисунку 4.16.

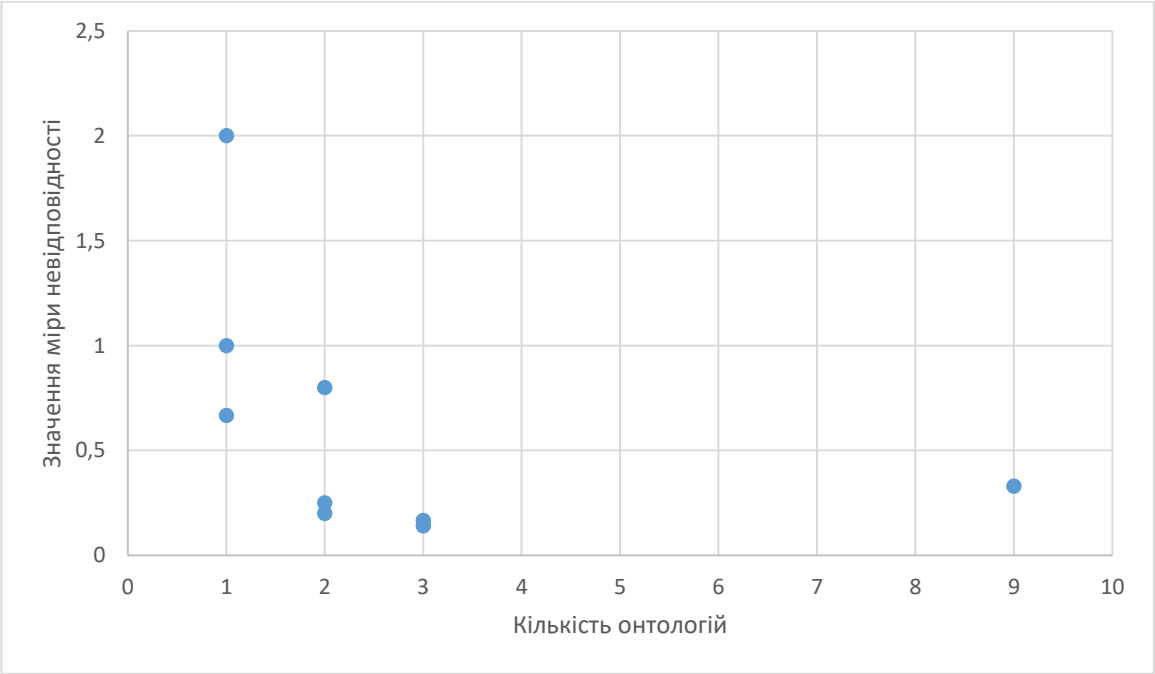


Рисунок 4.16. Значення MI^C- міри невідповідності для тестових випадків.

4.2.4.4 D_f-міра неконсистентності

Як і попередня міра, D_f-міра неконсистентності також обчислює більше різновидів значенньь невідповідності. Найбільша кількість онтологій отримана зі значеннями 0,337 та 1, кількість таких онтологій – 4. Хоча отримано найменшу кількість онтологій зі значенням міри невідповідності 1. Рисунок 4.17 відображає описані дані.

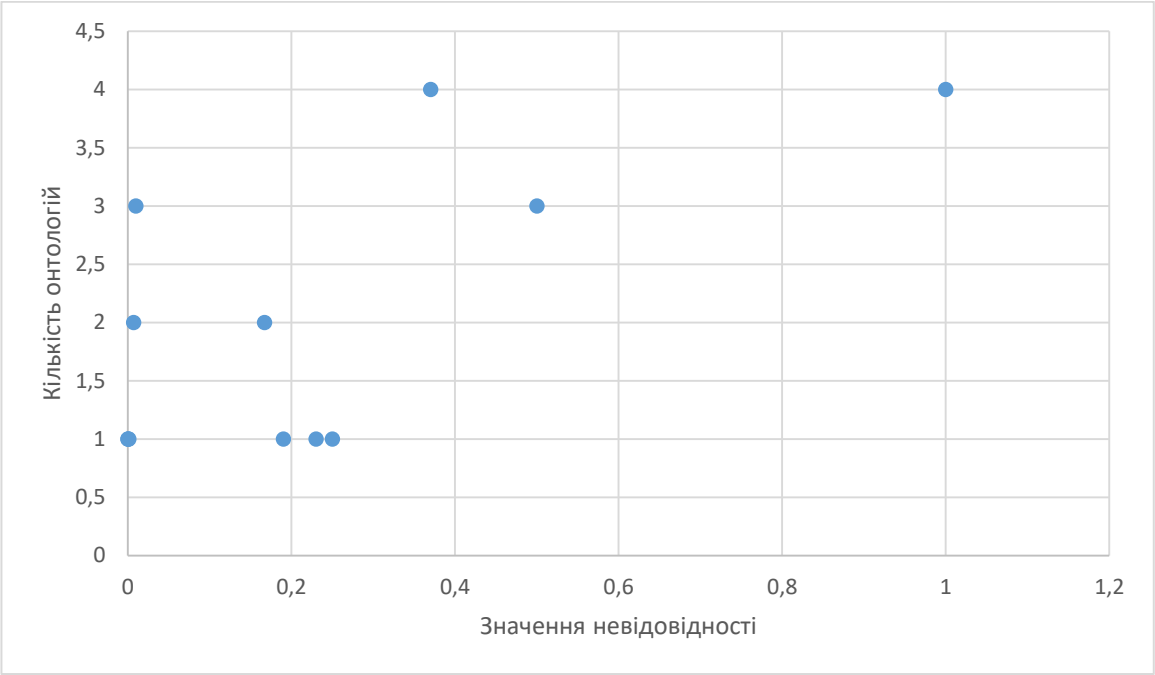


Рисунок 4.17. – Значення D_f-міри неконсистентності для тестових випадків.

4.2.4.5. Проблемна міра невідповідності

Оскільки проблемна міра невідповідності використовує кількість аксіом в $MI(K)$, значення, отримані за допомогою цього підходу завжди більше 2. Графік на рисунку 4.18 відображає додаткову інформацію про отримане значення за допомогою цього методу.

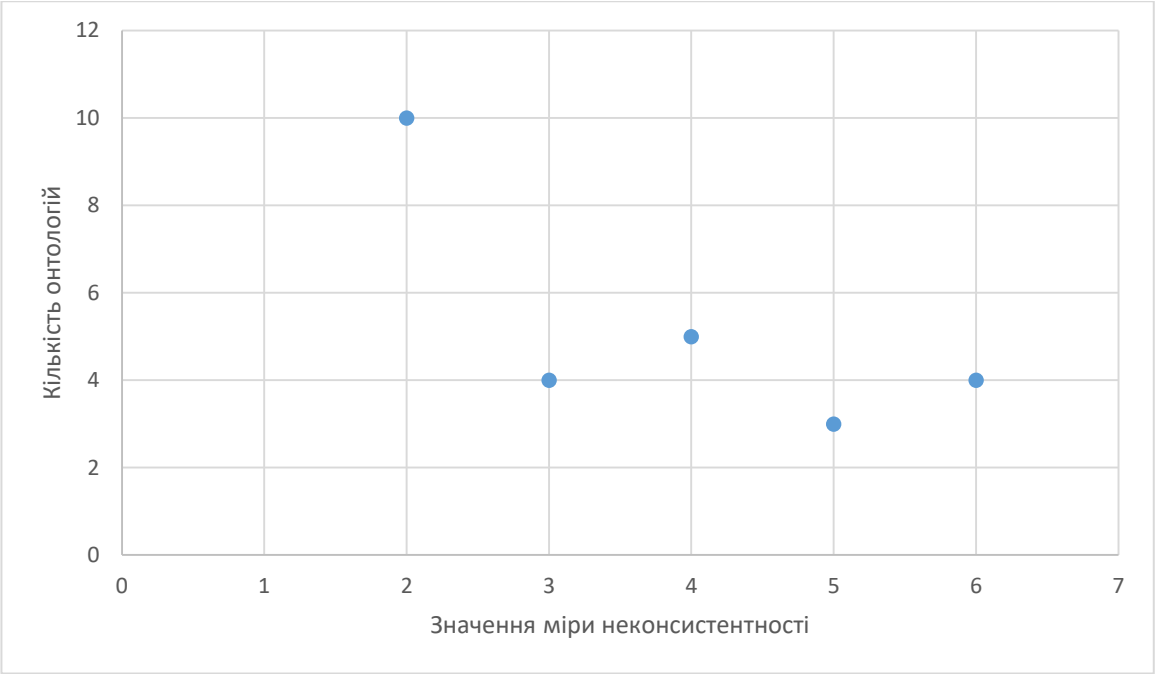


Рисунок 4.18. – Значення проблемної міри невідповідності для тестових випадків.

4.2.4.6 Коефіцієнт несумісності для міри невідповідності

Величина коефіцієнту несумісності для міри невідповідності має діапазон значень між 0 і 1. Для тестових випажків коефіцієнт несумісності дорівнює від 0.067 до 1 для одної та семи онтологій відповідно. Найбільша кількість онтологій – 8 має значення 0.334. Більш детальну інформацію про результати обчислення за цим методом можна отримати на графіку на рисунку 4.19.

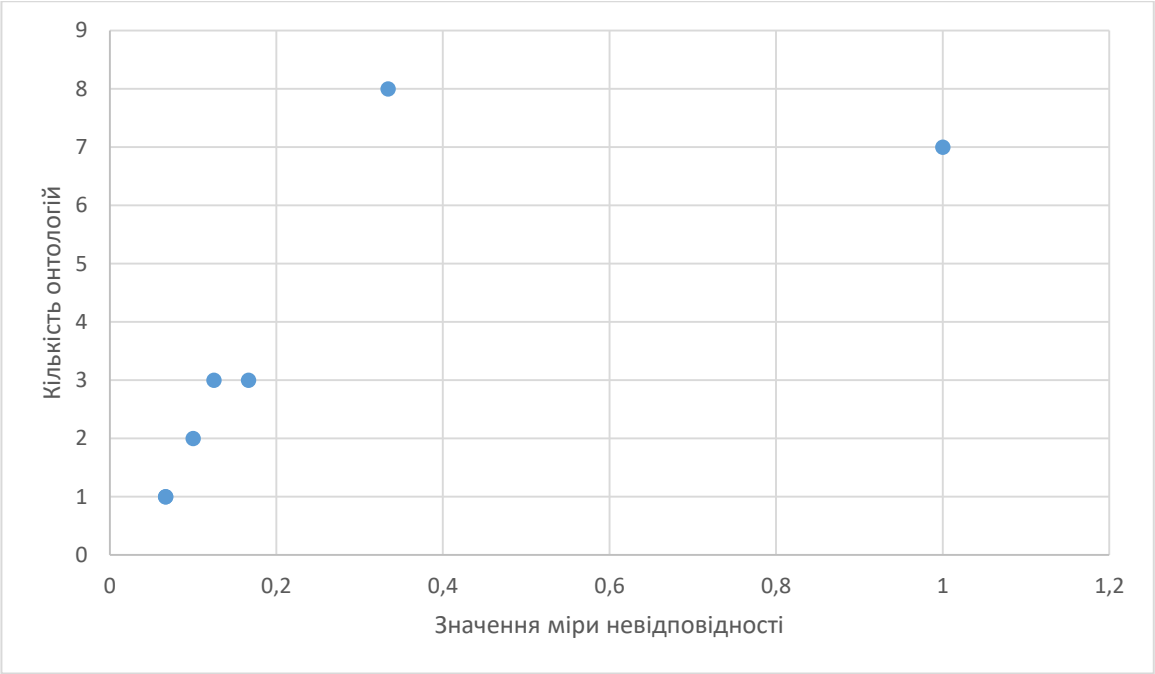


Рисунок 4.19. – Значення коефіцієнту несумісності для міри невідповідності для тестових випадків.

4.2.4.7 MC-міра невідповідності

Найменше значення міри невідповідності виявлене в 13 онтологіях, де воно дорівнює 1. Найвища величина несумісності для цього методу обчислена для трьох онтологій та дорівнює 6. Кожне отримане значення та його кількість можна побачити на графіку 4.20.

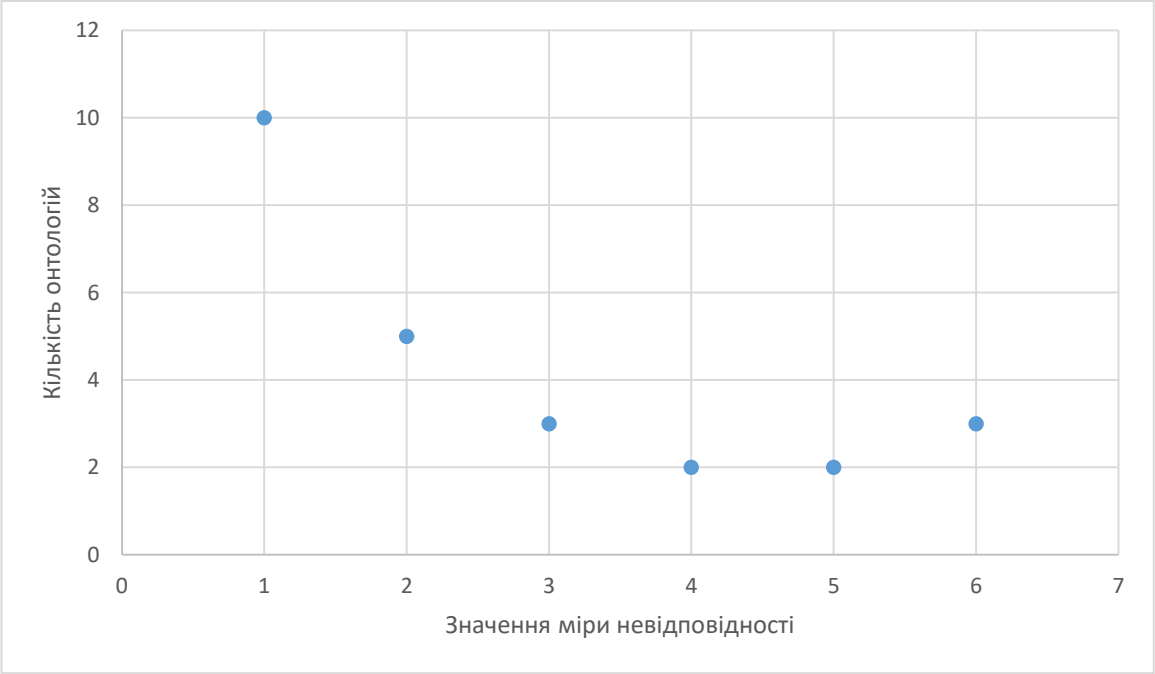


Рисунок 4.20. – Значення MC-міри невідповідності для тестових випадків.

4.2.4.8 NC-міра невідповідності

Результатом обчислення NC-міри невідповідності є лише два різновиди значень: 1 і 2, кожне з них обчислене для 21 і 2 онтологій, відповідно. Отже, графік цієї міри є простим, Рисунок 4.21.

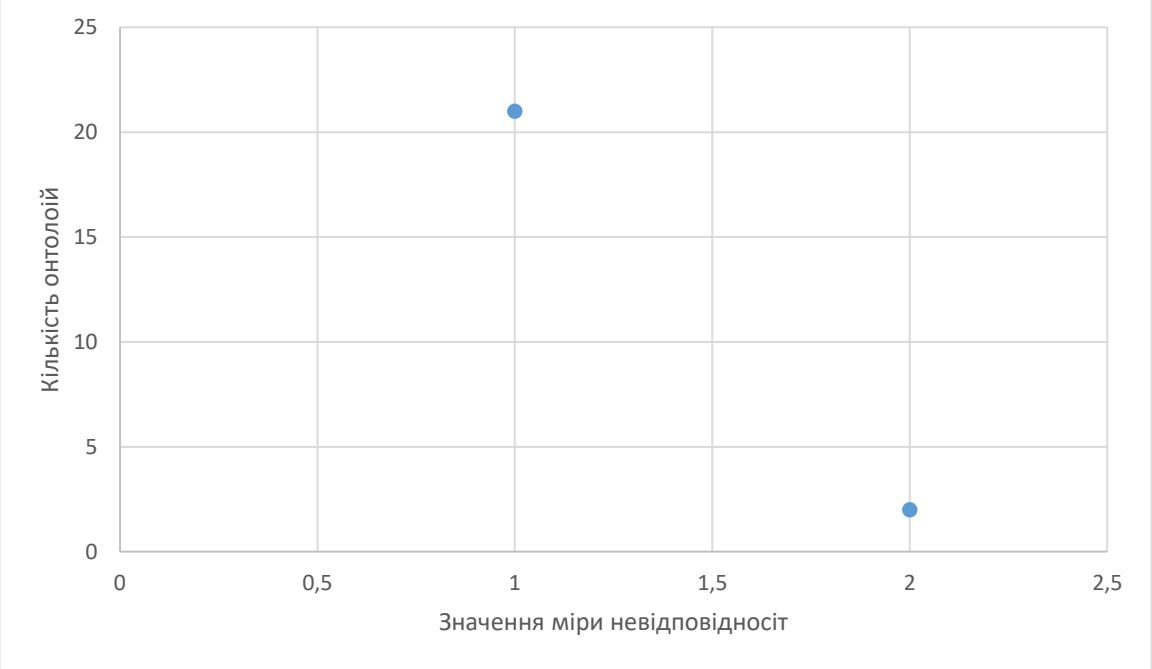


Рисунок 4.21. – Значення NC-міри невідповідності для тестових випадків.

4.2.4.9 MV-міра невідповідності

Значення невідповідності тестових випадків, отриманих за допомогою MV-міри невідповідності достатньо рузноманітні. Протое більшість величин 17 мають значення 1. Графік на рисунку 4.22 це відображає.

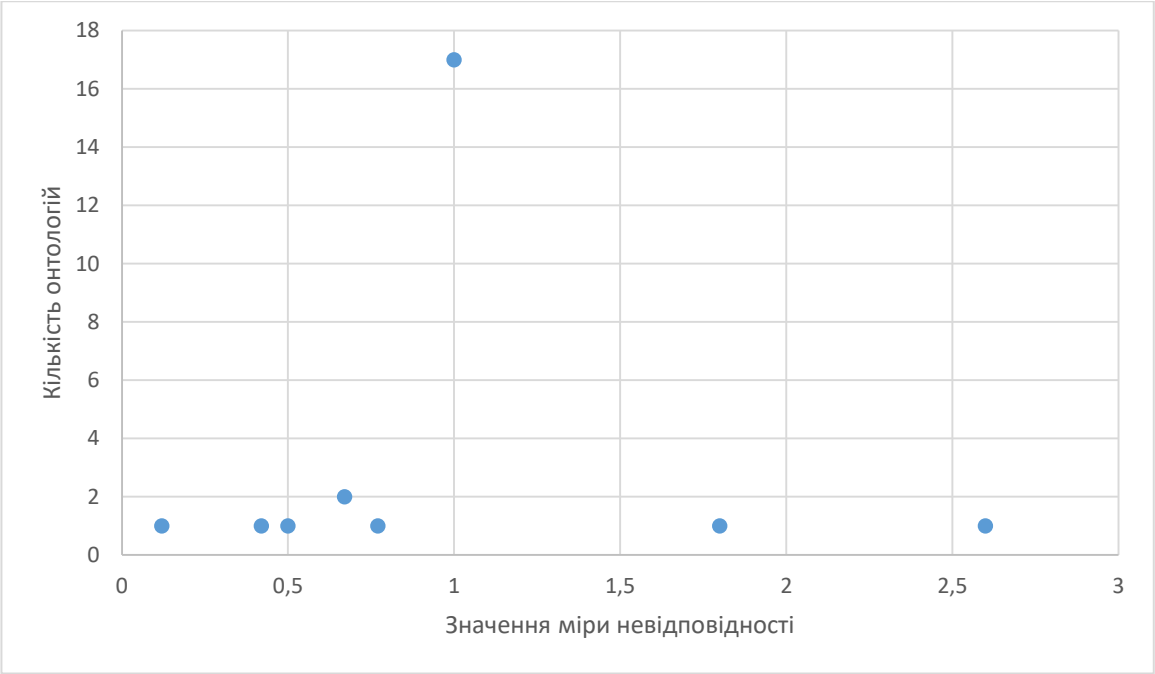


Рисунок 4.22. – Значення MV-міри невідповідності для тестових випадків.

4.2.4.10 ID_{MCS}-міра невідповідності

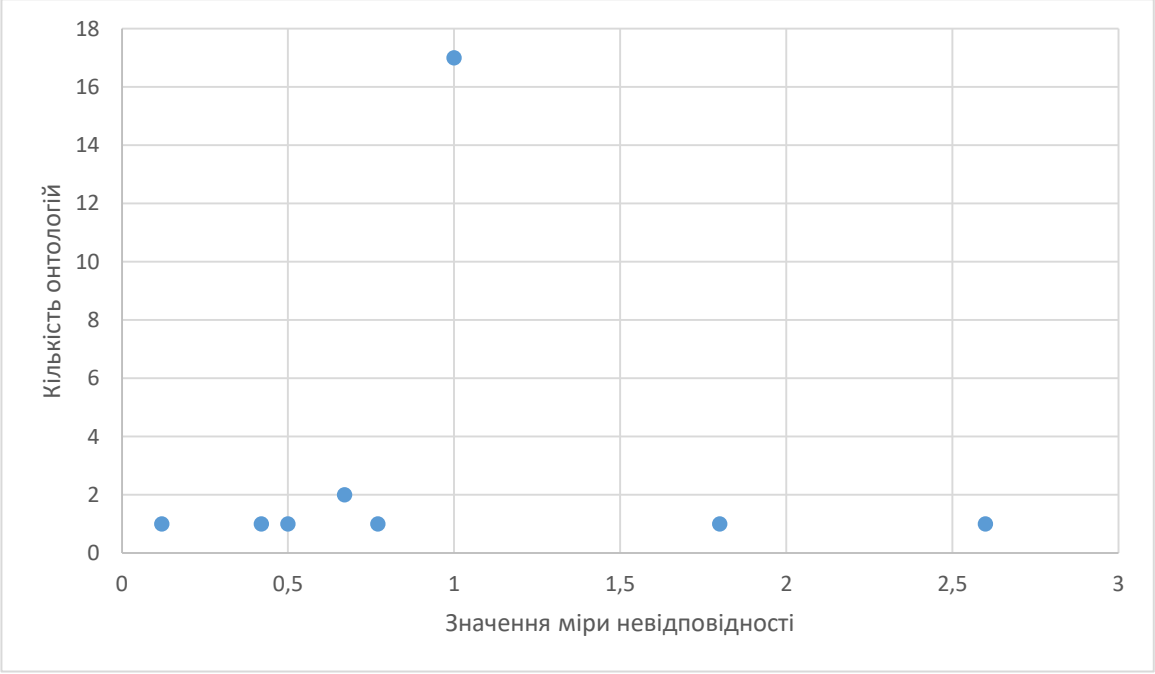


Рисунок 4.23. – Значення ID_{MCS}-міри невідповідності для тестових випадків.

За результатом роботи методу ID_{MCS}-міра невідповідності були отримані ті ж значення та їх кількість, як для MV-міри невідповідності. Отже, діаграма,

яка зображує отримані дані(Рисунок 4.23), буде такою ж як і на рисунку 4.22.

4.2.4.11 MIV_D- міра невідповідності

Невідповідності для аксіом баз знань отримуємо для 83 аксіом з 23 тестових онтологічних тестів. Ті аксіоми, що входять до мінімально неконсистентних підмножин, мають значення 1, їх 47. Решта 36 має значення 0, що означає, що ці аксіоми є вільними, які не належать $MI(K)$. На Рисуноку 4.24 зображена описані результати.

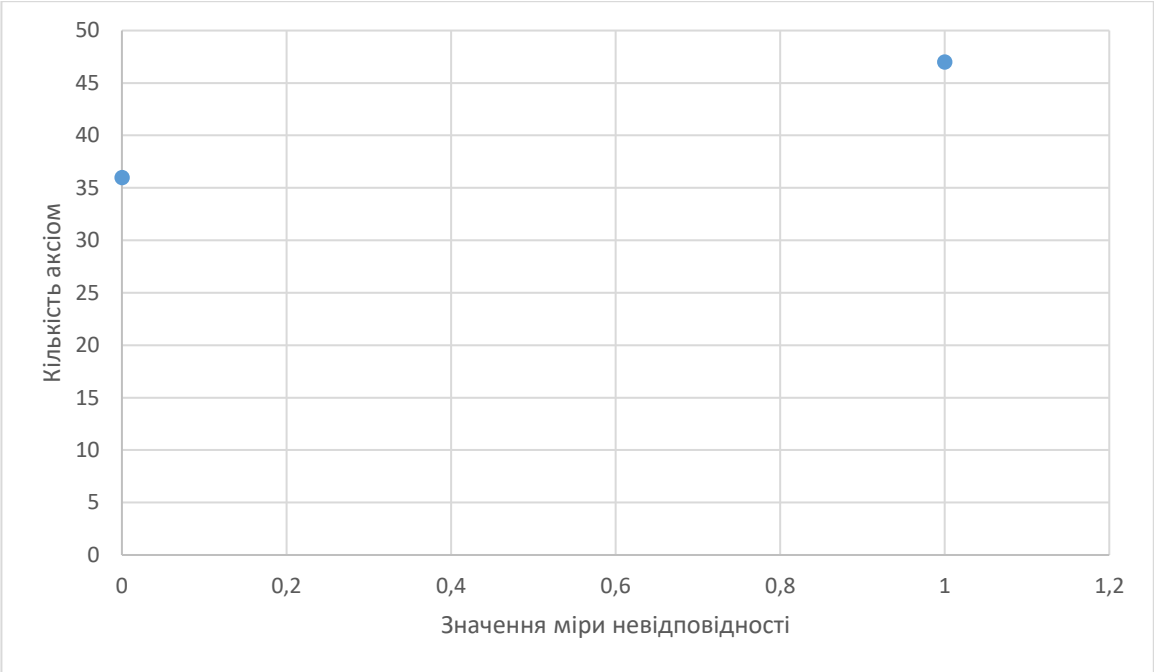


Рисунок 4.24. – Значення MIV_D-міри невідповідності для тестових випадків.

4.2.4.12 $MIV_{\#}$- міра невідповідності

Існує 7 аксіом з 23 тестових випадків, які не можуть бути обчислені, 66 аксіом, які можуть бути обчислені та мають значення, відображені на рисунку 4.25.

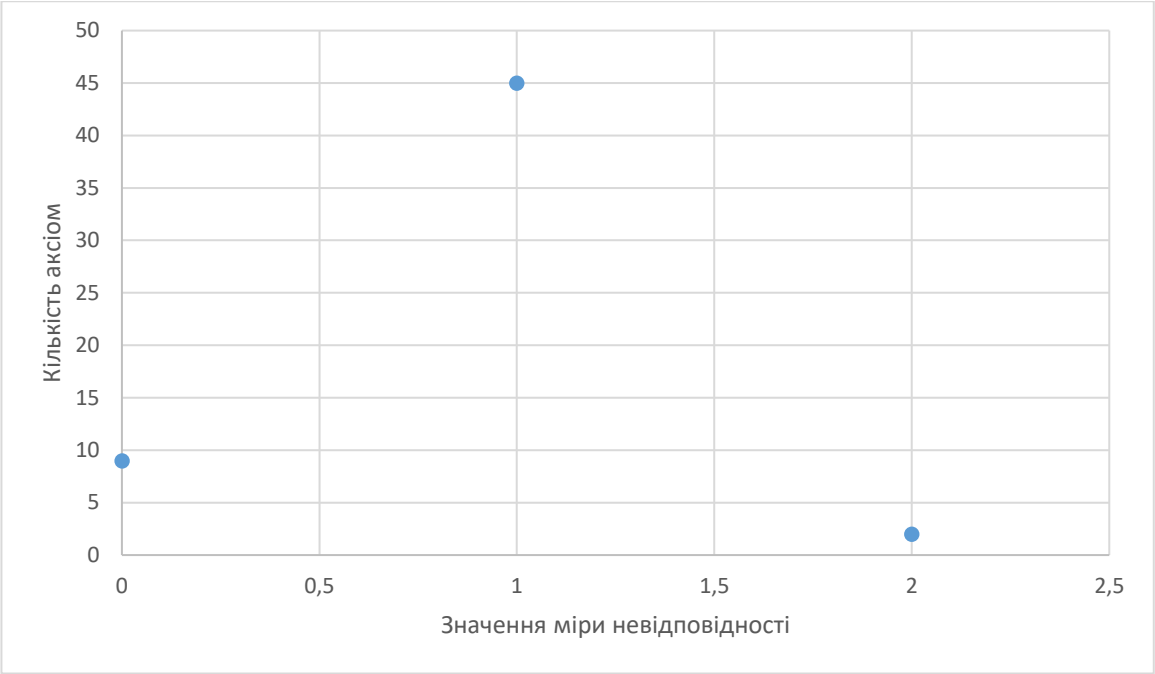


Рисунок 4.25. – Значення $MIV_{\#}$-міри невідповідності для тестових випадків.

4.2.4.13 MIV_C- міра невідповідності

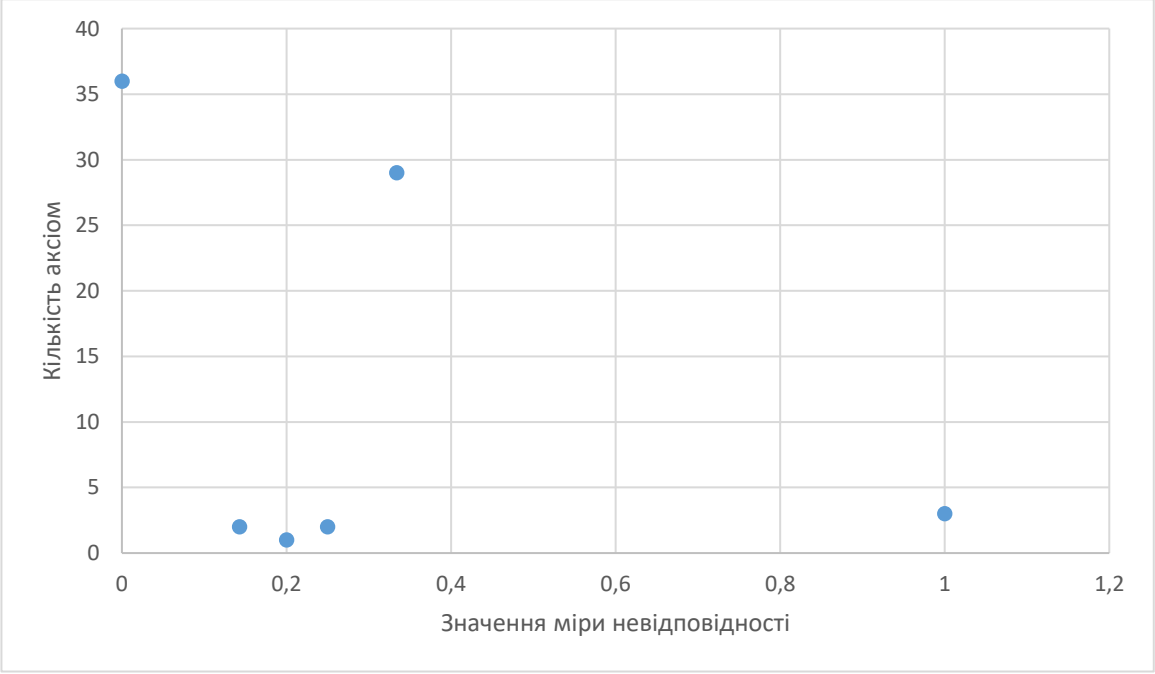


Рисунок 4.26. – Значення MIV_C-міри невідповідності для тестових випадків.

MIV_C- міра невідповідності обчислює значення, які коливаються в діапазоні значень між 0 і 1. Оскільки існує 36 вільних аксіом, існує 36 аксіом,

які мають значення 0. Інші значення знаходяться в діапазоні між 0.143 та 1, де найбільша кількість онтологій, що складається з 29 аксіом, має значення 0.334. Графік на рисунку 4.26 відображає отримані дані.

4.2.4.14 MI-Шеплі міра невідповідності

В тестових випадках існує 79 аксіоми, для ких можна очислити MI-Шеплі міру невідповідності, та 4 аксіоми, які не можуть бути захоплені даним методом. Серед 79 аксіом, існує 34 аксіоми з значеннями 0 і 6 аксіом з значенням 1. Крім того, інші аксіоми мають різні значення невідповідності. Графік на рисунку 4.27 отримані результати.

4.2.4.15 Розрахункова міра невідповідності

Результатом обчислення розрахункової міри невідповідності є: 9 аксіом 23 онтологій, які не можуть бути захоплені; значення 0 для 20 аксіом, значення 1 для аксіоми 52 і значення 2 для 2 аксіом. Результат можна побачити на рисунку 4.28.

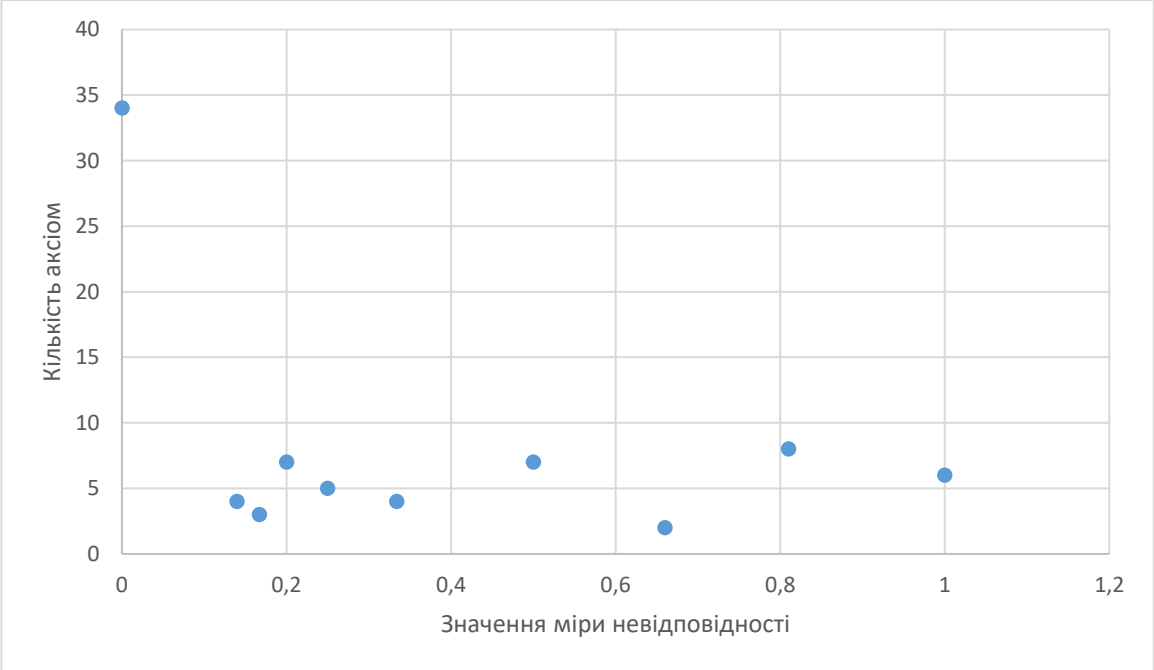


Рисунок 4.27. – Значення MI-Шеплі міри невідповідності для тестових випадків.

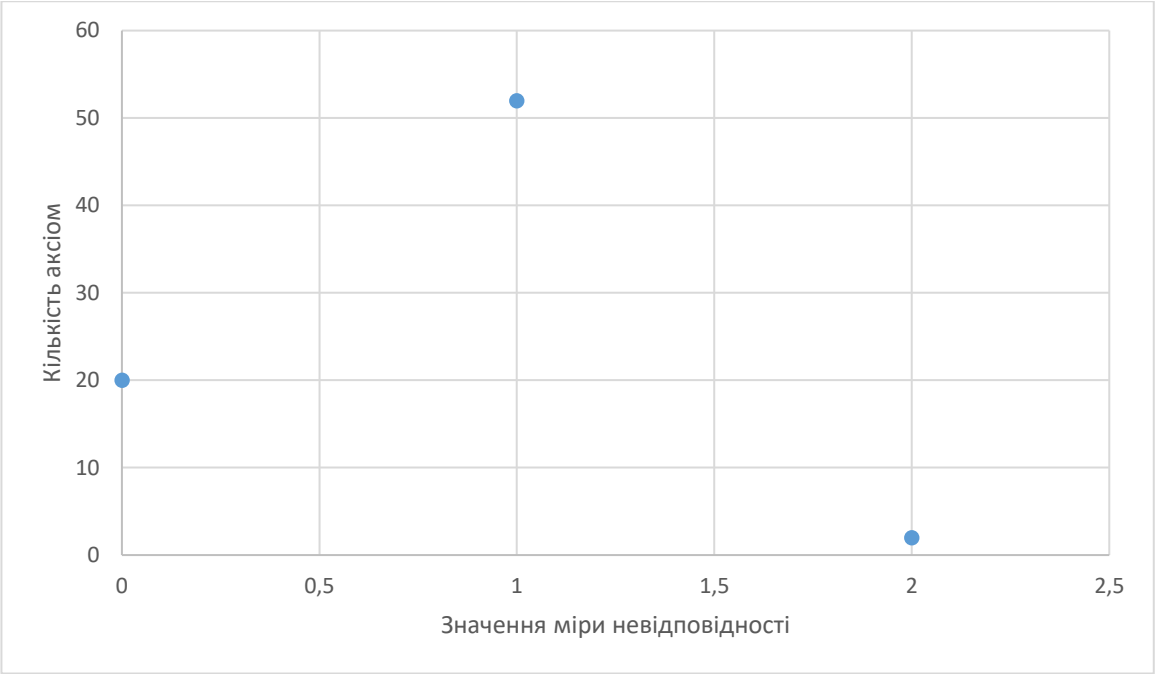


Рисунок 4.28. – Значення розрахункової міри невідповідності для тестових випадків.

Висновки до розділу 4

Проаналізовано способи обчислення міри неконсистентності онтологій на 25 тестових випадках та отримані наступні результати:

1. Критична міра невідповідності обчислюється за найменший час, проте вона не дає рекомендацій щодо ступеню невідповідності онтології, а лише відображає факт неконсистентності.

2. Результати обчислення міри невідповідності I_{Df} є найбільш різноманітними, час виконання відповідає розміру онтології, тому можна зробити висновок про доцільність використання даної міри в загальному випадку.

3. Аналіз розробленого способу обчислення міри невідповідності MI-Шеплі показав, що цей спосіб дає виграш в часі виконання в в середньому в 1,5 рази лише для онтологій, кількість мінімально неконсистентних підмножин в яких є невеликою (10 — 25%).

ВИСНОВКИ

1. Проведено аналітичний огляд існуючих методів обчислення міри невідповідності для описової логіки. Представленно 15 підходів до обчислення міри невідповідності для OWL2 онтологій, які переносяться з описової логіки. Показано, що недоліком існуючих підходів є те, що більшість підходів має невелику кількість унікальних результучих значень міри невідповідності, що не дозволяє надавати рекомендації щодо виправлення невідповідності.

2. Проведена класифікація методів на дві групи. За допомогою підходів першої групи можно обчислити міру невідповідності на основі онтології, другої групи – на основі аксіом.

3. Розроблений метод обчислення міри невідповідності на основі значення Шеплі, який відрізняється від уснуючих використанням мінімально неконсистентних підмножин.

4. Розроблене програмне забезпечення, яке обчислює міру невідповідності за допомогою двох обробників: HermiT і JFact, за результатами роботи якого можно зробити наступні висновки:

а) В загальному випадку reasoner HermiT має потребує більше часу для обчислення міри невідповідності, ніж Jfact. Однак, оскільки різниця в часі виконання не перевищує 2 секунд, можна вважати її незначною.

б) З точки зору часу обчислення з порівнянні з розміром онтологій, чим більший розмір онтології, тим вище час обчислення міри її неконсистентності.

в) Критична міра невідповідності завжди розраховується за найменший час, хоча не дає рекомендацій щодо ступеню неконсистентності. Idf, I_{IR}, I_{mv} та I_{DMCS} методи мають тим більший час виконання, чим нижча ступінь несумісності. Крім того, оскільки немає різниці в отриманих значеннях непослідовності для деяких методів, можна стверджувати, що міра невідповідності mv ідентична мірі невідповідності I_{DMCS}, $MIV_{\#}$ міра невідповідності ідентична мірі невідповідності MC.

г) Найрізноманітніші значення міри невідповідності були

обчислені за допомогою міри I_{df}, відповідно цей метож дає найбільш точні рекомендації щодо вилучення аксіом, які спричиняють неконсистентність. I_{Df} і S^i_a мають найбільш кількість обчислень серед методів на основі онтологій та аксіом, відповідно. I_{Df}, I_{IR}, I_{mv} та I_{DMCS}, як правило, використовуються найчастіше, оскільки дозволяють досить точно визначити рівень невідповідності онтологій та мають прямо пропорційну залежність між часом виконання та розміром онтології.

г) Розроблений метод обчислення міри неквідповідності дає виграш в часі виконання лише для онтологій з невеликою кількістю мінімально неконсистентних підмножин. Якщо ж кількість аксіом, що входять в ці підиножини перевищує 25% від загальної кількості аксіом, MI-Шеплі метод має більший час виконання. Тому в такому випадку доцільне використання методу обчислення D_f-міри неконсистентності.

СПИСОК ВИКОРИСТАНИХ ЛІТЕРАТУРНИХ ДЖЕРЕЛ

1. Hunter A. Measuring Inconsistency Through Minimal Inconsistency Sets / A. Hunter, S. Konieczny // Proceedings of the Eleventh International Conference on Principles of Knowledge Representation and Reasoning. – Washington, DC, USA, AAAI Press, 2008. – P.358-366.

2. Hunter A. Shapley Inconsistency Values / A. Hunter, S. Konieczny // Proceedings of the 10th International Conference on Knowledge Representation. – Washington, DC, USA, AAAI Press, 2006. – P.249-259.

3. Hunter A. Approaches to Measuring Inconsistent Information. Inconsistency Tolerance / A. Hunter, S. Konieczny // Lect. Notes Comput. Sci., vol. 3300, Springer, 2010. – P.189-234.

4. Akama S. Why Paraconsistent Logics? / S. Akama, C. Newton // Paraconsistent Engineering, Springer International Publishing. – Switzerland, 2016. – P.105.

5. Guohui X. Inconsistency Measurement Based on Variables in Minimal Unsatisfiable Subsets / X. Guohui, Y. Ma // Proceedings of the 20th European Conference on Artifical Intelligence. – 2012. – P. 203-214.

6. Guohui X. Computing Inconsistency Measurements under Multi-Valued Semantics by Partial Max-SAT Solvers // Proceedings of the 12th International Conference on the Principles of Knowledge Representation and Reasoning. – 2010. – P.324-356.

7. Besnard P. Quasi-Classical Logic: Non-trivializable Classical Reasoning from the Inconsistent Information / P. Besnard, A. Hunter // Proceedings of ECSQARU. – 2009. – P.44-51.

8. Patel-Schneider P. A Four-Valued Semantics for Terminological Logics // Artificial Intelligence, Elsevier Science Publishers B.V. -North Holland, 2007. – P. 319-351.

9. Thimm M. On the Expressivity of I

10. nconsistency Measures // Artificial Intelligence 234. – Elsevier, 2016. – P.120-151.

11. Ma Y. Computing Inconsistency Measure Based on Paraconsistent Semantics / Y. Ma, Q. Guilin, H. Pascal // J. Log. Comput. 21 (6), 2011. – P.1257-1281.

12. Grant J. Analysing Inconsistent First-Order Knowledgebases / J. Grant, A. Hunter //Artificial Intelligence 172 (8-9), 2008. – P.1064-1093.

13. Ma Y. Measuring Inconsistency for Description Logics Based on Paraconsistent Semantics // Proceedings of the 20th InternationalWorkshop on Description Logics, 2007. – P.258-279.

14. Ma Y. Distance-based Measures of Inconsistency and Incoherency for Description Logics / Y. Ma, H. Pascal // Proc. 23rd Int. Workshop on Description Logics. – Waterloo, Canada, 2010. – P.482-502.

15. Deng X. Measuring Inconsistencies in Ontologies / X. Deng, H. Volker, N. Shiri // European Semantic Web Conference,2007. – P.326-340.

16. Zhou L. Measuring Inconsistency in DL-Lite Ontologies // The 2009 IEEE/WIC/ACM International Conferences onWeb Intelligence (WI'09). – Milan, Italy, September 2009. – P.349-356.

17. Priest. G. Logic of Paradox // J. Philos. Log. 8. -2001.- P. 219-241.

18. Patel-Schneider P. What is OWL (and Why Should I Care)? // American Association for Artificial Intelligence (AAAI), 2004.

19. Rudolph S. Foundation of Description Logics // Reasoning Web 2011. – P.76-136.

20. Kedian M. A Syntax-based Approach to Measuring the Degree of Inconsistency for Belief Bases // International Journal of Approximate Reasoning 52 (7), 2011. – P.978-999.

21. Grant J. Measuring Consistency Gain and Information Loss in Stepwise Inconsistency Resolution / J. Grant, A. Hunter // Proceedings of the 11th European Conference on Symbolic and Quantitative Approaches to Reasoning with Uncertainty, 2011. – P. 362-373.

22. D. Dragan Measures of Inconsistency and Defaults // International Journal of Approximate Reasoning 51, 2010. – P. 832-845.

23. Дзицюк Є. Модифікація алгоритмів MI та Шеплі для обчислення міри неконсистентності онтології / Є. Дзицюк, М.Орлова: матеріали 8-ої міжнар. наук.-техн. конф. ПМК’2016, 20-22 кві 2016, Київ, Україна / НТУУ «КПІ», Ф-т прикладної матем. – К., 2016. – 345с. – Парал. тит. арк. англ.

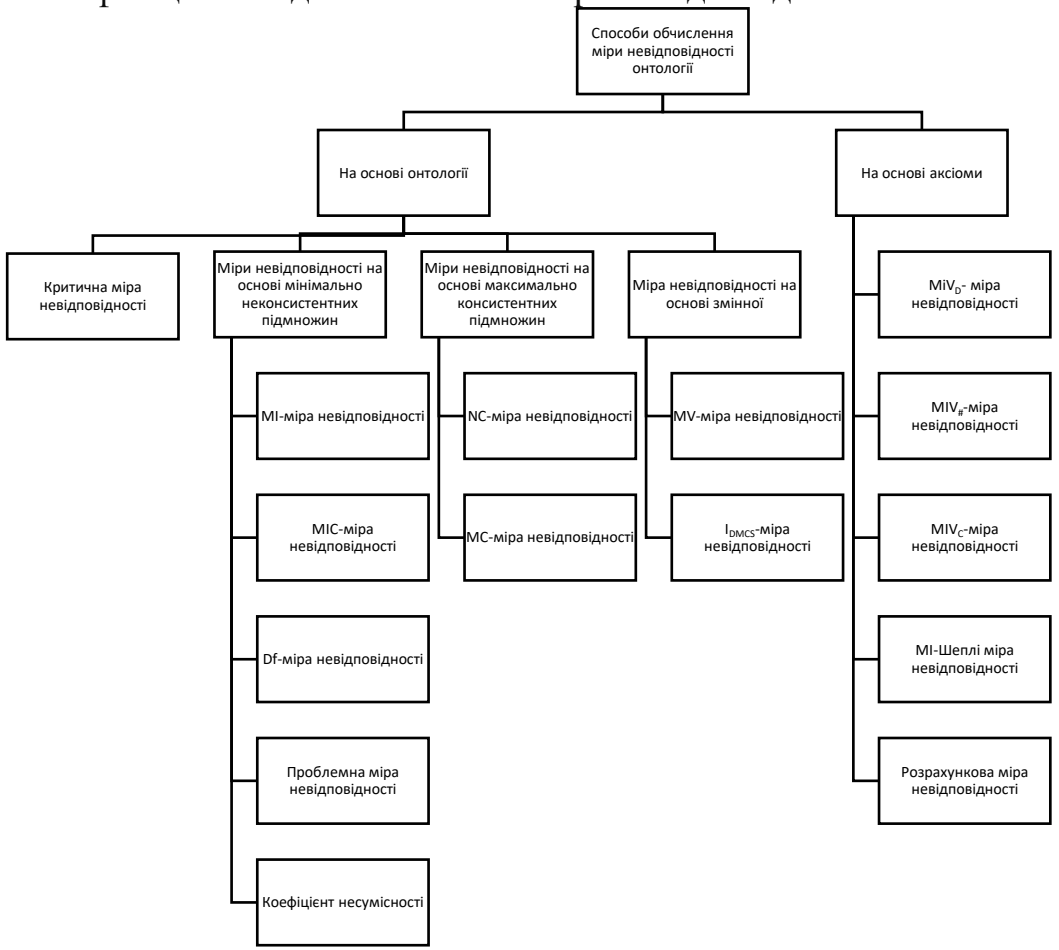
24. Дзицюк Є. Поєднання алгоритмів MIV та Шеплі для обчислення невідповідності онтологій / Є. Дзицюк, М.Орлова // Міжнародний науковий журнал "Інтернаука". – 2018. – №3.

25. Орлова М. Інтелектуальний аналіз даних в хмарних інфраструктурах / М. Орлова, Є. Дзицюк, Є. Багінський: матеріали 7-ої міжнар. наук.-техн. конф. ПМК’2016, 20-22 кві 2016, Київ, Україна / НТУУ «КПІ», Ф-т прикладної матем. – К., 2016. – 345с. – Парал. тит. арк. англ.

ДОДАТКИ

Додаток 1. Копії графічних матеріалів

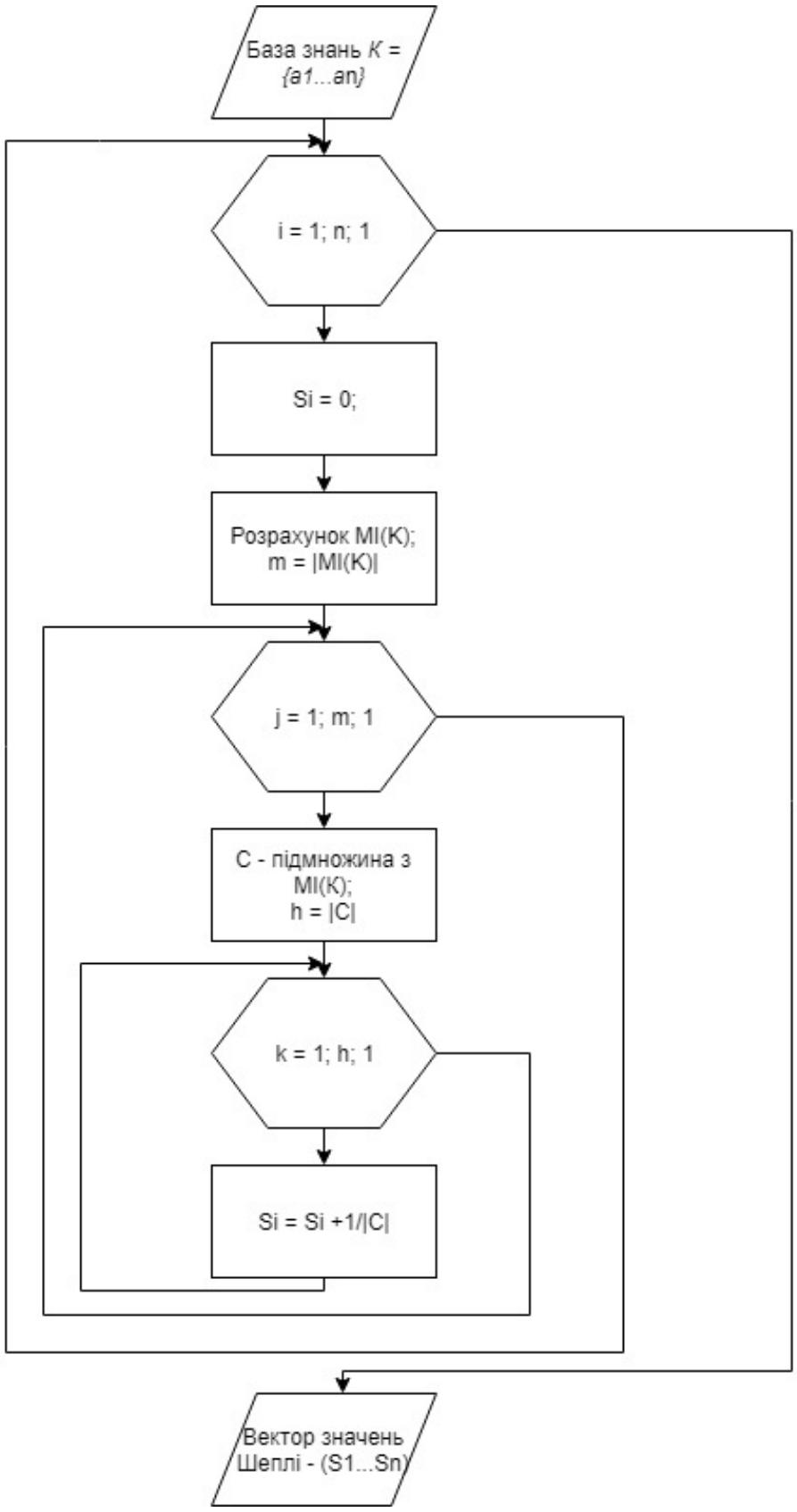
Класифікація методів обчислення міри невідповідності онтологій.



Дзицюк Є.В.

КВ-61м

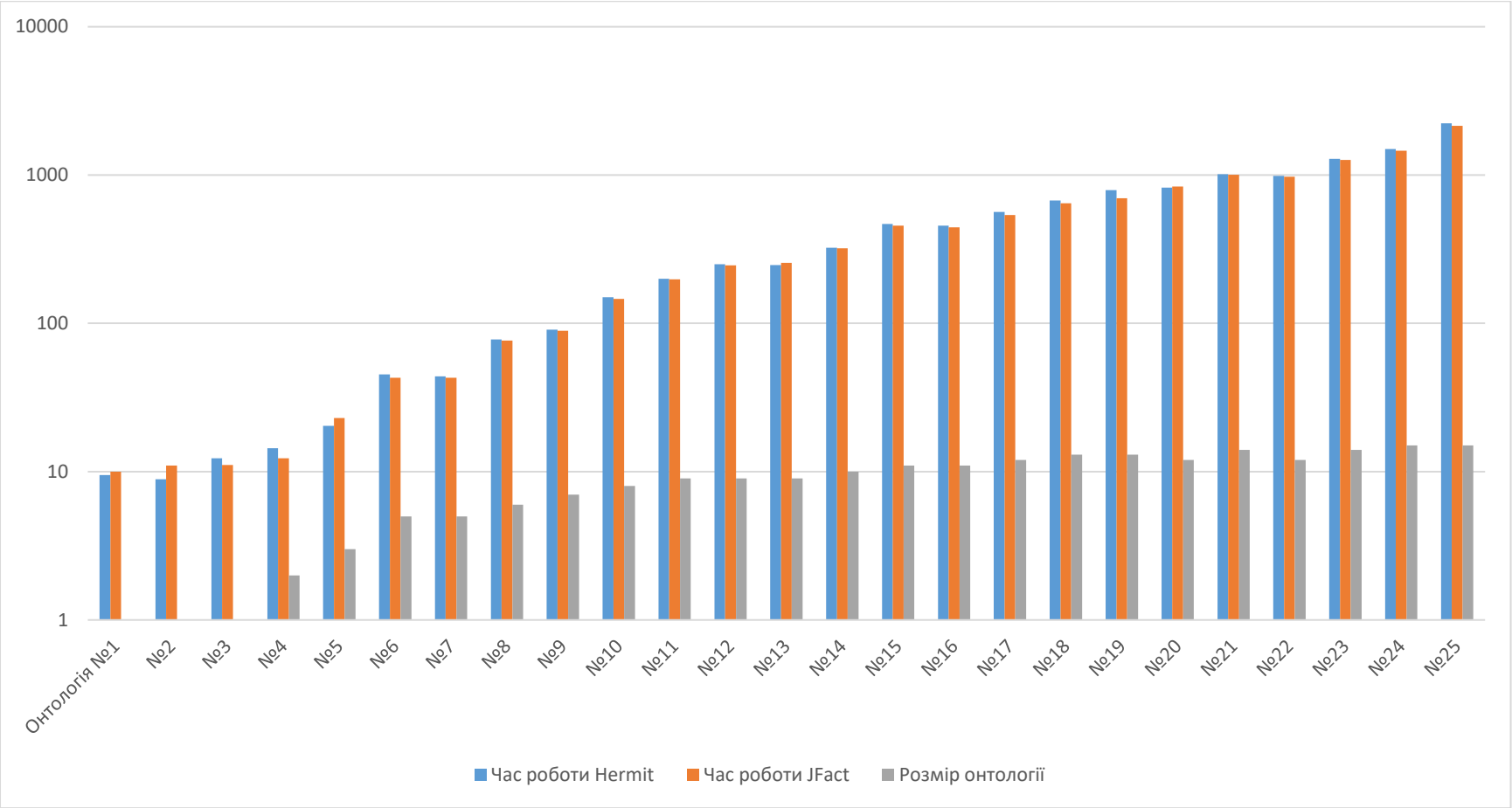
Блок схема розробленого способу обчислення міри невідповідності онтології



Дзицюк Є.В.

КВ-61м

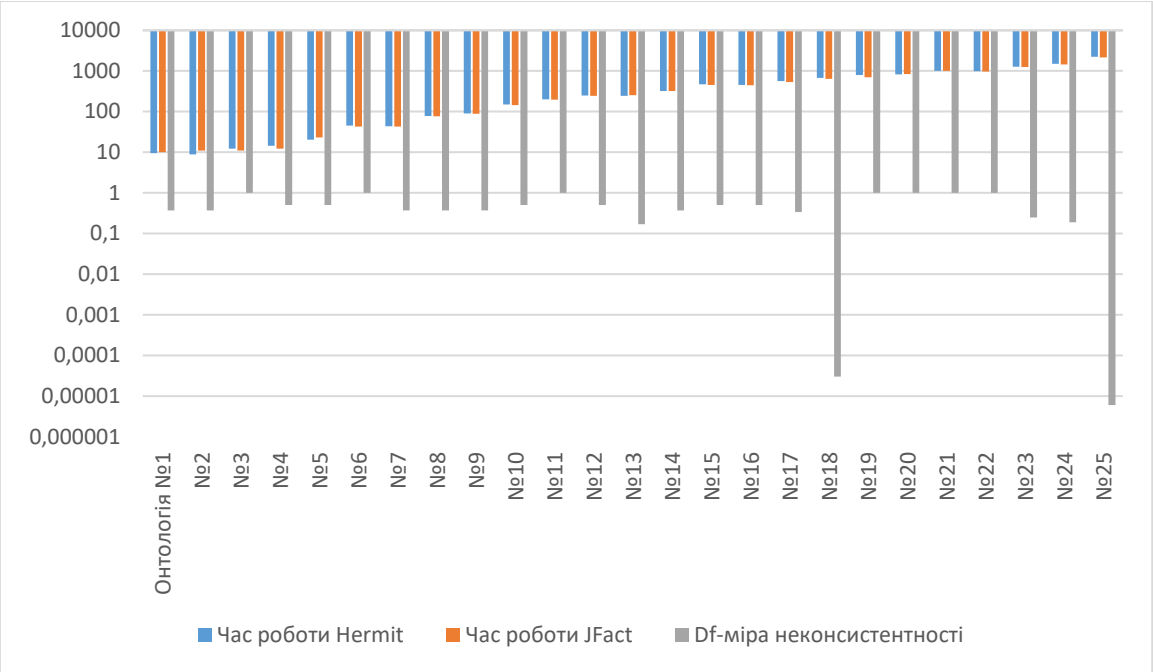
Порівняння часу роботи існуючих та розробленого способів обчислення міри невідповідності та розміру онтологій.

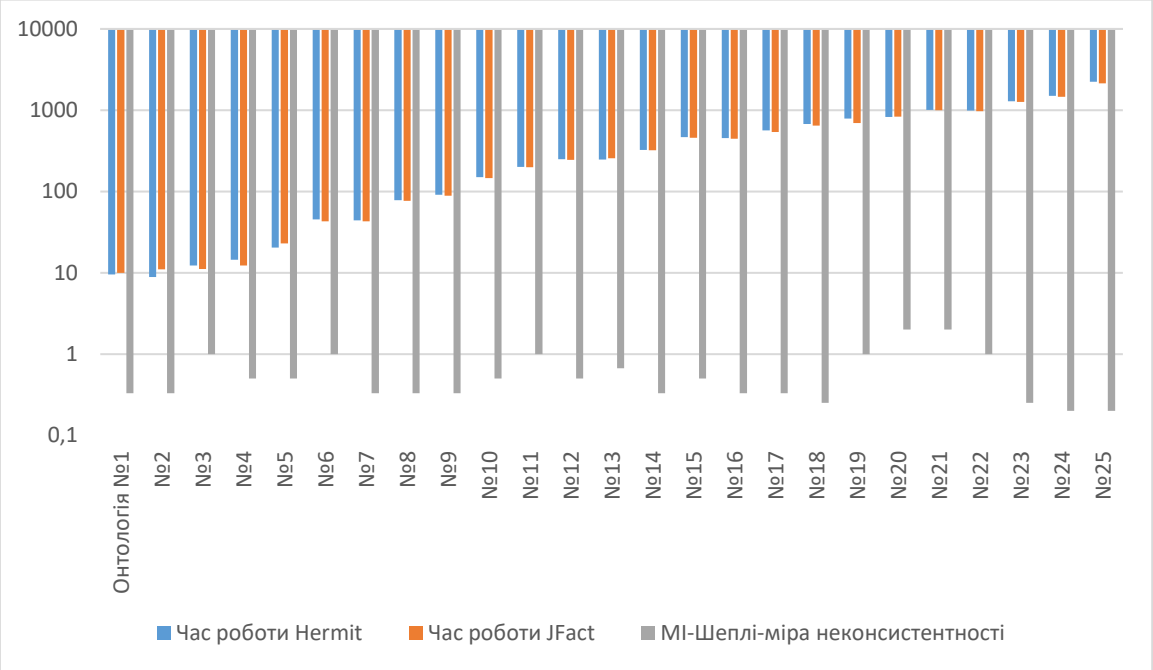


Дзицюк Є.В.

КВ-61м

Порівняння часу роботи та значень MI-Шеплі та D^f -міри невідповідності.

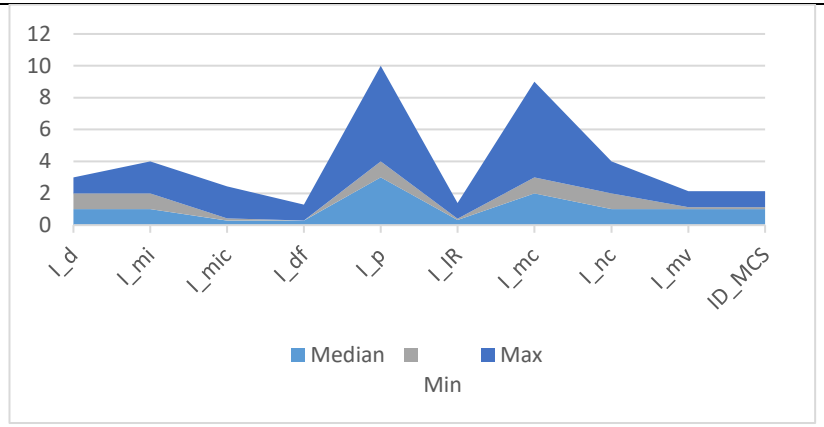


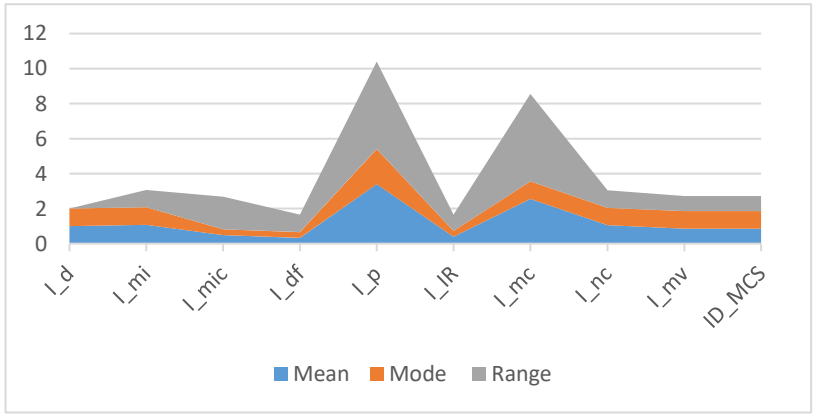


Дзицюк Є.В.

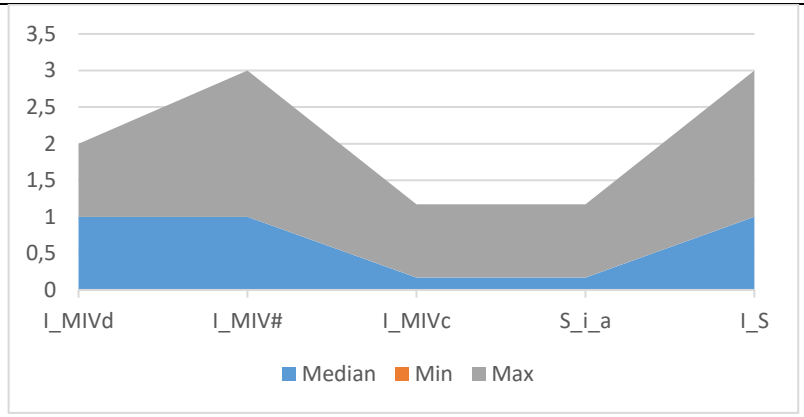
КВ-61м

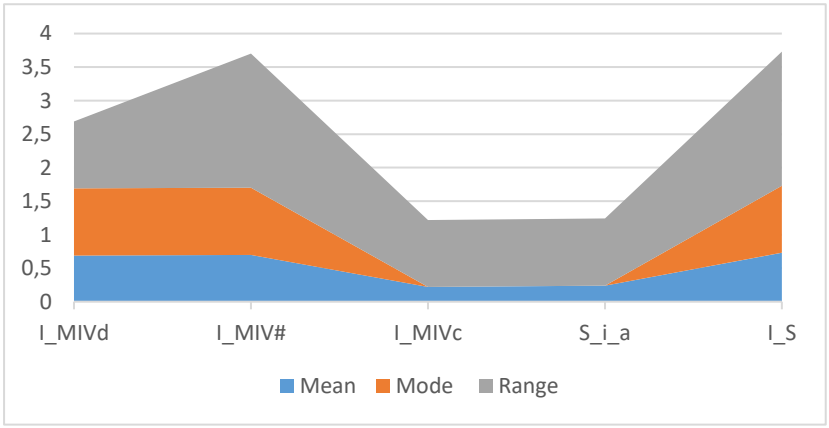
Описова статистика тестових випадків для способів вимірювання невідповідності.





Описова статистика тестових випадків для вимірювання невідповідності на основі онтологій.



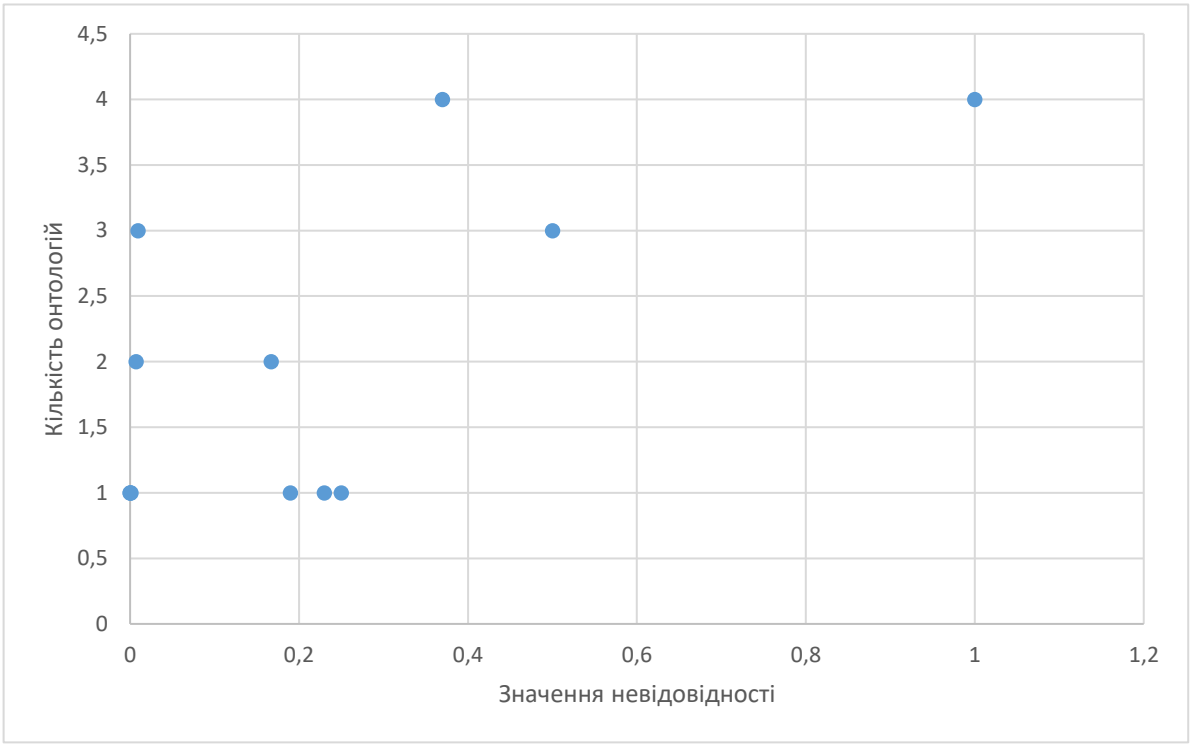


Описова статистика тестових випадків для вимірювання невідповідності на основі аксіоми

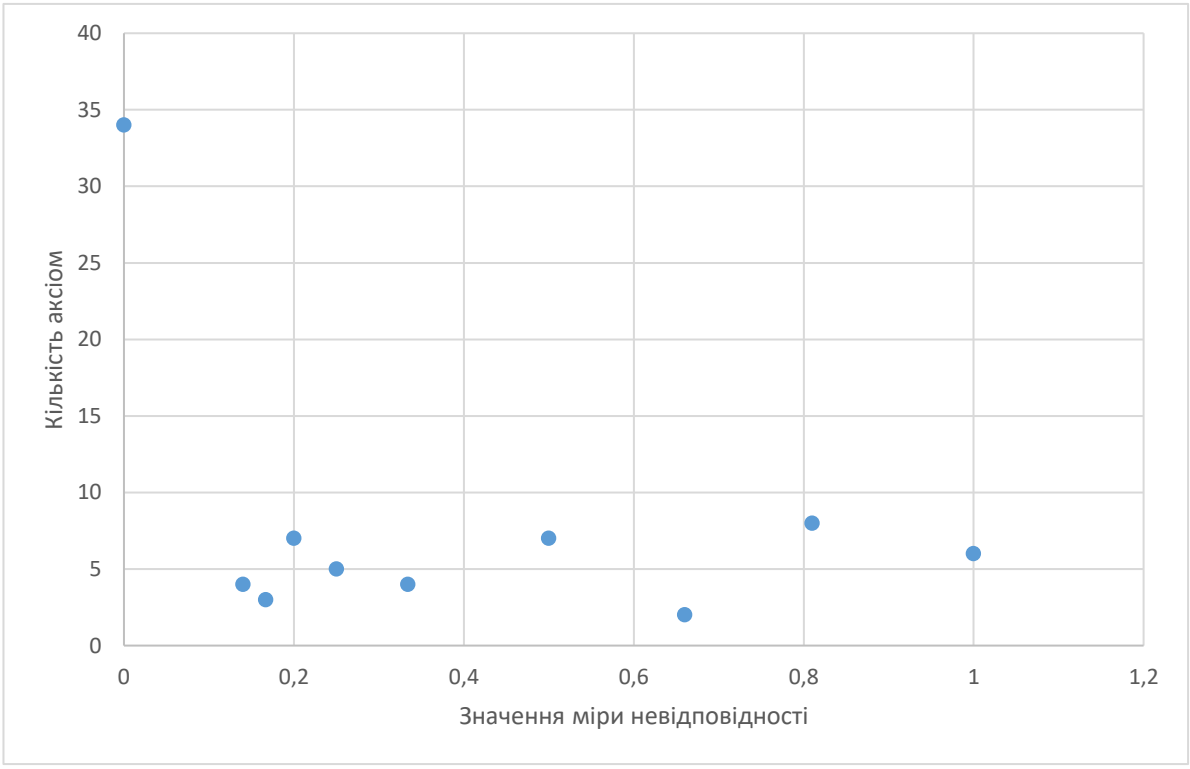
Дзицюк Є.В.

КВ-61м

Значення MI-Шеплі та D_f-міри неконсистентності для тестових випадків



Значення D_f-міри неконсистентності для тестових випадків



Значення MI-Шеплі міри невідповідності для тестових випадків.

Дзицюк Є.В.

КВ-61м

Додаток 2. Фрагмент лістингу розробленого програмного забезпечення

```
public static <T> Set<Set<T>> powerSet(Set<T> originalSet) {

    Set<Set<T>> sets = new HashSet<Set<T>>();
    if (originalSet.isEmpty()) {
        sets.add(new HashSet<T>());
        return sets;
    }
    List<T> list = new ArrayList<T>(originalSet);
    T head = list.get(0);
    Set<T> rest = new HashSet<T>(list.subList(1, list.size()));
    for (Set<T> set : powerSet(rest)) {
        Set<T> newSet = new HashSet<T>();
        newSet.add(head);
        newSet.addAll(set);
        sets.add(newSet);
        sets.add(set);
    }
    return sets;
}
private static void addSubsets(ReasonerFactory rf6, ArrayList<Integer>
consistentSubsetSize, HashSet<Set<OWLAxiom>> consistentSubset,
HashSet<Set<OWLAxiom>> inconsistentSubset, Set<OWLAxiom> s) throws
OWLOntologyCreationException {
    OWLOntologyManager manager6;
    OWLOntology AxiomOntology6;
    Set<OWLAxiom> axiomsToRemove6;
    AddAxiom addAxiom6;
    manager6 = OWLManager.createOWLOntologyManager();
    AxiomOntology6 = manager6.createOntology();

    axiomsToRemove6 = AxiomOntology6.getAxioms();

    if (axiomsToRemove6 != null) {
        manager6.removeAxioms(AxiomOntology6, axiomsToRemove6);
    }

    for (OWLAxiom axiomOfS : s) {
        addAxiom6 = new AddAxiom(AxiomOntology6, axiomOfS);
        manager6.applyChange(addAxiom6);
    }

    OWLReasoner reasoner6 = rf6.createReasoner(AxiomOntology6); // for
    // hermit
    // and
    // JFact
    boolean consistent = reasoner6.isConsistent();
    if (!consistent) {
        System.out.println("C: " + s);
        System.out.println("Is C consistent? " + consistent);
    }
    if (consistent) {
        consistentSubset.add(s);
        consistentSubsetSize.add(s.size());
    }

    if (!consistent) {
        inconsistentSubset.add(s);
    }
}
private static void getMIVSharp(HashSet<OWLAxiom> ontologyAxiomSet,
Set<Set<OWLAxiom>> arrayOfExplanationSet, Set<Set<OWLAxiom>> setOfMinMIK) {
    for (OWLAxiom axiomInK2 : ontologyAxiomSet) {
        setOfMinMIK.clear();
```

```
        for (Set<OWLAxiom> MinMIK : arrayOfExplanationSet) {
            if (MinMIK.contains(axiomInK2)) {
                setOfMinMIK.add(MinMIK);
            }
        }
        System.out.println("MIV_Sharp(K," + axiomInK2 + ") = " +
setOfMinMIK.size());

    }
}
private static void getMKCandidate(ArrayList<Set<OWLAxiom>> MCKcandidate,
private static float getSumOfSize(ArrayList<Integer> explanationSizeList,
float sumOfSize, HashSet<OWLAxiom> MIKAxiomSet, HashSet<OWLClass>
MIKClassSet, HashSet<OWLNamedIndividual> MIKIndividualSet,
HashSet<OWLObjectProperty> MIKObjectPropertySet, HashSet<OWLAxiom> topBottom,
Set<Set<OWLAxiom>> arrayOfExplanationSet, Explanation<OWLAxiom> explanation)
{
    Set<OWLAxiom> arrayOfExplanation;
    Set<OWLClass> inconsistentClass;
    Set<OWLNamedIndividual> inconsistentIndividual;
    Set<OWLObjectProperty> inconsistentObjectProperty;
    float sizeOfM;
    float onePerSizeOfM;// is
    arrayOfExplanation = explanation.getAxioms(); // arrayOfExplanation
    System.out.println("-----------------------------------------------------
------------------------");
    System.out.println("MI(K) subset: " + arrayOfExplanation);
    arrayOfExplanationSet.add(arrayOfExplanation); // arrayOfExplanationSet
    System.out.println("-----------------------------------------------------
------------------------");

    System.out.println("Axioms causing the inconsistency: ");
    for (OWLAxiom causingAxiom : arrayOfExplanation) {
        System.out.println(causingAxiom);
        MIKAxiomSet.add(causingAxiom);
        if ((causingAxiom.isBottomEntity()) || (causingAxiom.isTopEntity()))
{
            topBottom.add(causingAxiom);
        }

        inconsistentClass = (Set<OWLClass>)
causingAxiom.getClassesInSignature();
        // System.out.println("theClass: " + theClass);
        MIKClassSet.addAll(inconsistentClass);
        inconsistentIndividual = (Set<OWLNamedIndividual>)
causingAxiom.getIndividualsInSignature();
        MIKIndividualSet.addAll(inconsistentIndividual);
        inconsistentObjectProperty = (Set<OWLObjectProperty>)
causingAxiom.getObjectPropertiesInSignature();
        MIKObjectPropertySet.addAll(inconsistentObjectProperty);
    }
    sizeOfM = arrayOfExplanation.size();
   // System.out.println("M size: " + sizeOfM);
    explanationSizeList.add((int) sizeOfM);
    onePerSizeOfM = (float) 1 / sizeOfM;
  //  System.out.println("One per M size: " + onePerSizeOfM);
    sumOfSize = sumOfSize + onePerSizeOfM;
    return sumOfSize;
}
```

Додаток 3. Публікації за темою магістерської дисертації