ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ, УМОВНИХ ПОЗНАЧЕНЬ, ТЕРМІНІВ

OWL – (Web Ontology Language) мова опису онтологій

RDF – (Resource Description Framework) технологія семантичної павутини, яка включає в себе середовище опису ресурсів, визначає загальну архітектуру метаданих і призначена для забезпечення сумісності метаданих за допомогою спільної семантики, структури та синтаксису.

DL – (Description Logics) описова логіка

OIL – (Ontology Interchange Language) мова опису онтологій

SPARQL – мова запитів до RDF

ВСТУП

Інформація — це нові відомості, які прийняті, зрозумілі і оцінені користувачем як корисні; іншими словами, інформація — це нові знання, які отримує споживач (суб'єкт) у результаті сприйняття і переробки певних відомостей.

Саме інформація в сучасному світі є рушієм прогресу. «Той, хто володіє інформацією, той володіє світом» - сказав колись Уїнстон Черчілль. З часом ці слова набули ще більшої значущості, мають безліч підтверджень їх правоти. Інформація у сучасному світі – це стратегічний ресурс. І той хто вміє добувати її, а також користуватись нею, той може будувати майбутнє так, як він цього бажає.

Саме пошук інформації є причиною досліджень багатьох учених сьогодення. Створення семантичного вебу, штучного інтелекту, нейронних мереж, – все це задля пошуку інформації.

Основне питання при побудові цифрових ситеми (будь-якої бази знань) полягає в тому, як організувати великі кількості інформації так, щоб користувачі могли знайти те, що їм потрібно. Описуючи ресурси, дослідники розробили складні схеми класифікації та правила каталогізації для створення метаданих. Метадані підвищують ефективність роботи, забезпечуючи основу для пошуку, визначаючи інтелектуальний та історичний контекст.

Цифрові системи пред'являють більш високі вимоги до метаданих, ніж традиційні. Вони містять великі обсяги інформації, і доступ до неї забезпечується за допомогою великої кількості різноманітних сервісів, які допомагають користувачам визначати і задовольняти свої потреби. Більшість цих сервісів повинні забезпечуватися без допомоги людини. Таким чином, важливо щоб метадані були пристосовані до машинної обробки Крім того, загальною вимогою для таких систем є сумісність на рівні обміну даними та

спільної роботи сервісів. Щоб досягти семантичної сумісність, сенс інформації, якою обмінюються, має бути зрозумілий у всіх системах. Використання онтологій для пояснення неявного і прихованого знання - можливий підхід для досягнення цієї мети.

Онтології – більше ніж просто складний підхід до опису та класифікації інформації. Вони можуть використовуватися для підтримки функціонування і зростання нового виду цифрових систем, реалізованих як розподілені інтелектуальні системи.

Онтології дозволяють представити нові поняття так, що вони стають придатними для машинної обробки. За допомогою онтології можна "перекинути місток" між новими поняттями, з якими система ще не зустрічалася, і описами уже відомих класів, відносин, властивостей і об'єктів реального світу.

Онтології класифікують знання, інформацією ж вони стануть тоді. Коли з них можна буде виділити користь. Для чого використовуються онтології? Що корисного в класифікації якоїсь предметної області? Над онтологіями можна здійснювати логічні висновки, які можна використовувати для:

- отримання фактів, яких в цих онтологіяк немає;
- зняття неоднозначності в текстах;
- персоналізувати інформацію про об’єкти;
- пошук «смислу» в тексті;
- отримання комерційної вигоди.

Це зовсім короткий перелік можливостей, які дають логічні висновки над зібраними (створеними) онтолгіями. Процес здійснення логічних висновків (хоча б якихось) є не тривіальною задачею. Цілі комани людей по всьому світу досліджують це питання, певні напрацювання вже є, так з’являються нові мови онтологій. Рекомендації по побудові онтологій, цілі

бази онтологій Freebase, GO, DBpedia, Geonames, Musicbrainz, та інші. Ведеться досліджння алгоритмів пошуку логічних висновків над онтологіями.

Метою роботи є дослідження основних методів і алгоритмів отримання логічних висновків над онтологіями, проведення порівняння розповсюджених систем роботи з онтологіями, дослідження можливостей вдосконалення алгоритмів та надання відповідних практичних рекомендацій.

1 ОТНТОЛОЇ, РОЗВИТОК OWL

1.1 Онтології

Онтологія (філософський) – наука про буття.

Онтологія (інформатика) – представлення знань про певну предметну область (середовище, світ). Онтологію неодмінно супроводжує деяка концепція цієї області інтересів. Найчастіше ця концепція виражається за допомогою визначення базових об’єктів (індивідуумів, атрибутів, процесів) і відношень між ними. Визначення цих об'єктів і відношень між ними зазвичай називають концептуалізацією.

Отже, онтологія – це загальноприйнята і загальнодоступна концептуалізація певної області знань (світу, середовища), яка містить базис для моделювання цієї області знань і визначає шляхи для взаємодії між агентами, які використовують знання з цієї області, і, нарешті, включає домовленості про представлення теоретичних основ даної області знань.

В рамках комп’ютерних наук онтологія – це формальна назва і визначення типів, властивостей і взаємовідносин суб’єктів, що дійсно, або принципово, існують в обраному контексті (предметній області). Таким чином, вони є практичним застосуванням філософського поняття онтолгії, за допомогою таксономії.

Онтології виділяють змінні, необхідні для деякої множини обчислень і встановлення відносин між ними.

В полі штучного інтелекту, семантичних мереж, інженерних систем, розробки програмного забезпечення, біометричної інформатики, бібліотекознавства, підприємництва та інформаційної архітектури, створюються онтології для обмеження складності і організації інформації. Онтологія може бути використана для вирішення прикладних проблем.

Історично онтології виникли з гілки філософії, відомої як метафізика, яка має справу з природою реальності – того, що існує. Це фундаментальне вчення займається аналізом різних типів або форми існування, часто з особливою увагою до відносин між особливостями і універсалів, між внутрішніми і зовнішніми властивостями, а також між сутністю та існуванням. Традиційна мета онтологічного дослідження, полягає в поділі світу (класифікації), для відкриття фундаментальних категорій або видів, в які об'єкти в світі потрапляють.

У другій половині 20–го століття філософи широко обговорювали можливі методи або підходи до побудови онтологій, без фактичного будівництва будь–яких більш–менш складних онтологій. Навпаки, вчені–комп'ютерники будували кілька великих і надійних онтології, таких як WordNet і Cyc, з порівняно невеликим обговореннями, на основі яких вони були побудовані.

З середини 1970–х років дослідники в галузі штучного інтелекту (ШІ) визнали, що захоплення знання є ключем до створення великих і потужних систем ШІ. Дослідники стверджували, що вони могли б створювати нові онтології як обчислювальні моделі, які дозволяють виконувати певні види автоматизованих дослідженнь. У 1980–х роках спільнота штучного інтелекту почала використовувати термін онтології для позначення теорії модельованого світу і компонента системи знань.

На початку 1990–х років широко цитуються веб–сторінки і папери "Принципи проектування онтологій з використанням обміну знаннями" Тома Грубера [1], якому приписують навмисне визначення онтології як технічного терміна в інформатиці. Грубер ввів термін, що означає специфікацію концептуалізації: «Онтологія – це опис (як формальної специфікації програми) концепцій і відносин, які можуть формально існувати для агента або групи агентів». Це визначення узгоджується з використанням онтологій як набору

визначень понять, але більш загального характеру. Але визначення слова «онтологія» інше, ніж те, що використовується в філософії.

Відповідно до Грубера (1993): Онтології часто ототожнюють з таксономічною ієрархією класів, визначеням класів, і категорізацією відношеннь, але онтології не повинні бути обмежені цими формами. Онтології також не обмежені консервативними визначеннями – тобто визначеннями в традиційному сенсі логіки, що тільки вводять термінологію і не додають ніяких знань про світ. Щоб осмислити онтології, потрібно викласти аксіоми, які обмежують можливості інтерпретації для певних термінів [1].

Сучасні онтології структурно подібні незалежно від мови, якою вони виражені. Як уже згадувалося вище, більшість онтології описують осіб (зразки), класи (концепції), атрибути і відносини.

Загальні компоненти онтологій включають в себе:

- фізичні особи: випадки або об'єкти (основні або "базові" об'єкти);
- класи: набори, колекції, поняття, класи з програмування, типи об'єктів, або види речей;
- атрибути: аспекти, властивості, особливості, характеристики або параметри об'єктів, які клас може мати;
- стосунки: способи, в яких класи та окремі особи можуть бути пов'язані один з одним;
- функціональні терміни: складні структури, утворені з певних відносин, які можуть бути використані замість окремого терміна;
- обмеження: формально зазначені описи того, що повинно бути вірно для того, щоб твердження було прийняте в якості вхідного;
- правила: твердження у вигляді, if–then (попередник–наслідок) пропозиції, що описують логічні висновки, які можна зробити з твердження;

- аксіоми: твердження (у тому числі правила) в логічній формі, що разом складають загальну теорію, що онтологія описує галузі застосування. Це визначення відрізняється від «аксіом» в породженій граматиці і формальної логіки. У цих дисциплінах, аксіоми включають тільки твердження, що є апріорними знаннями. Поняття «аксіом» також включають теорію, отриману з аксіоматичних тверджень;
- події: зміна атрибутів або відносин.

Типи онтологій

Онтології предметної області (ПО)

Онтологія предметної області (або предметно–орієнтована онтологія) являє концепцію, яка відносяться до частини світу. Конкретні значення термінів, що застосовуються в цій області надаються онтологією. Наприклад слово «карта» має багато різних значень. Онтологія з предметною областю покер буде моделювати "гральну карту" значення слова, в той час як онтологія комп'ютерної техніки буде моделювати "перфокарту", "відеокарти".

Так онтології (ПО) представляють концепції в дуже специфічних і часто еклектичних способах, що часто несумісні. Оскільки системи, які покладаються на онтології предметних областей розширювані, їх часто необхідно об'єднати в більш загальні онтології. Це становить проблему для дизайнера онтології. Різні онтології в тій же ПО виникають через різні мови, різного напряму використання онтологій і різних уявлень про ПО (на основі культурних традицій, освіти, ідеології і т.д.).

В даний час об'єднання онтологій, які не розроблені із загального фундаменту онтології в основному є ручним процесом і, отже, забирає багато часу. Онтологій, що використовують той же фундамент онтології, щоб забезпечити набір основних елементів, за допомогою яких вказуються

значення елементів онтології можуть бути об'єднані автоматично. Є дослідження узагальнених методів для злиття онтологій, але ця область досліджень є значною мірою теоретичною[2].

Верхні онтології

Верхня онтології (фундаментальна онтологія) – це модель спільних об'єктів, які звичайно застосовуються в широкому діапазоні онтологій ПО. Вона, як правило, використовує ядро–словник, що містить терміни і пов'язані описи об'єктів, які вони використовують в різних відповідних наборах предметних областей.

Є кілька стандартизованих верхніх онтології, що доступні для використання, в тому числі BFO, метод Боро, Dublin Core, GFO, OpenCyc/ResearchCyc, SUMO, Unified Foundational Ontology (UFO), DOLCE. WordNet.

Гібридні онтології

Gellish онтологія – приклад поєднання верхньої та онтології ПО.

Онтології зазвичай кодується з використанням мов онтології. Мова опису онтологій — формальна мова, що використовується для кодування онтології. Існує кілька подібних мов :

- OWL — Ontology Web Language, стандарт W3C, мова для семантичних тверджень, розроблена як розширення RDF і RDFS;
- KIF (Knowledge Interchange Format або формат обміну знаннями) — заснований на S–виразах синтаксис для логіки;
- Common Logic — спадкоємець KIF (стандартизований — ISO/IEC 24707:2007).
- CycL — онтологічна мова, що використовується в проекті Cyc, заснована на численні предикатів із деякими розширеннями вищого порядку.

- DAML+OIL (FIPA)
- DOGMA (Developing Ontology–Grounded Methods and Applications — розробка методів на основі онтологій і додатків)
- Інші.

Для роботи з мовами онтологій існує декілька видів технологій: редактори онтологій (для створення онтологій), DBMS онтологій (для зберігання й звертання до онтології) і сховища онтологій (для роботи з декількома онтологіями).

2.1 Створення OWL

OWL [10] – нова мова онтологій для семантичних мереж, розроблена World Wide Web Consortium (W3C) Web Ontology Working Group. OWL, в першу чергу, призначена для подання інформації про категорії об'єктів і, те, як об'єкти взаємопов'язані – сортування інформації, яку часто називають онтологією. OWL може також представляти інформацію про самі об'єкти – сортування інформації, які часто сприймаються як дані.

OWL не розроблявся у вакуумі. Існуючі праці в напрямку дослідження онтологій були взяті за основу і підтримувались групою розробки. Вже було кілька мов онтології, призначених для використання в Web, OWL мала підтримувати максимальну сумісність, наскільки це можливо з існуючими мовами, у тому числі SHOE [18], OIL [12], і DAML+OIL [9].

Множинний плив на OWL створював певні складнощі в розробці. Багато технічної роботи було виконано для задоволення всіх вимог, висунутих до OWL, а також наукових знань та смаків розробників.

Далі більш детально про шлях становлення OWL

Вимоги до OWL

Як уже згадувалося вище, дизайн OWL став предметом різних впливів, починаючи від встановлениховлених формалізмів і знань, уявлення (парадигм), впливів існуючих мов онтології, існуючих мов Semantic Web.

Деякі з найбільш важливих чинників, що вплинули на дизайн OWL прийшли від його попередника DAML+OIL, з описової логіки, від парадигми кадрів і від RDF. Зокрема, формально специфікацією мови була описова логіки, структури мови (це видно з абстрактного синтаксису), вплив парадигми кадрів, RDF/XML обмін синтаксису, обраний на основі вимоги сумісності з RDF.

Кожен з цих чинників буде розглянуто більш детально далі.

2.1.1 Описова логіка

Описова логіка – сімейство з класу на основі концепції формалізованого подання знань. Вони характеризуються використанням різних конструкторів для створення складних класів із простіших, акцент на можливості розв'язання ключових проблем міркування. Описова логіка і висновки з її дослідження мали сильний вплив на конструкції OWL, особливо, на формалізацію семантики, вибір мови конструкторів та інтеграції типів даних і значень даних. Насправді OWL DL і OWL Lite (два з трьох видів OWL) можна розглядати як виразну описову логіку, з онтологіями, що є еквівалентними до бази знань описової логіки [2].

Ключовою особливістю описової логіки в тому, що вона є логікою, тобто формальною мовою з добре визначеною семантикою. Стандартна техніка для визначення описової логіки – це використання моделі теоретичної семантики, мета якої пояснювати взаємозв'язок між синтаксисом мови і предметною областю. Об’єкти в домені (предметній області) самі по собі не мають ніякого сенсу без встановлених відносин між ними. Для встановлення

відносин між об’єктами використовуються: перетин, об’єднання, доповнення, обмеження та інші операції.

Як OIL і DAML+OIL, OWL використовує описову логіку для формалізації значеннь мови. Це було визнано важливою особливістю для всіх трьох мов, так як це дозволяє онтологій та інформації, за допомогою словникового запасу, визначеного онтологією, ділитись і змінюватись без суперечок. Потреба в такого роду формальності була посилена досвідом роботи з ранніми версіями RDF і RDFS специфікації, де відсутність формальності незабаром привело до створенння мовних конструкцій таких як обмеженнями доменів і діапазонів [2].

Ще одна перевага формалізації значення мови : автоматизовані методи міркування , які можуть бути використані для перевірки узгодженості класів і онтологій, а також, відносин між ними. Це дуже важливо, якщо повна потужність онтологій буде використовуватись інтелектуальними агентами, і здатність надавати таку можливість прийняття рішення, що є головною метою створення OWL.

2.1.2 Парадигма кадрів

У контексті Semantic Web, де користувачі з широким діапазоном знань можуть очікувати на створення або зміну онтологій, читаність і загальна зручність використання є важливими факторами для мови онтологій. При проектуванні OIL, одна з мов, на яких ґрунтується OWL, ці вимоги були враховані, забезпечуючи синтаксичну базу, засновану на парадигмі рамок. Група рамки збирає інформацію про кожний клас, що робить онтології легшими для читання і розуміння, особливо, для користувачів не знайомих з описовою логікою. Парадигма рамки була використана в ряді добре відомих систем знань, включаючи Prot'eg'e, OKBC. На дизайн OIL мав вплив XOL – пропозиція для синтаксису XML для OKBC Lite (урізана версія OKBC).

В кадро–базованих мовах кожен клас описується кадром. Кадр включає в себе ім'я класу, ідентифікує більш загальний клас (або класи), що спеціалізується, і перераховує набір "слотів". Слот може складатися з пари властивість – значення, або обмеження на значення, які можуть виступати в якості ігрових "наповнювачів" (у даному контексті, значення означає або особа або значення даних). Ця структура була використана в мові OIL, з певним збагаченням синтаксису для визначення класів і слотів обмеження, з тим, щоб використовувати описову логіку по максимуму. Крім того, рамки були використані для опису властивостей, наприклад, вказавши більш загальні властивості, діапазони і доменни обмеження, транзитивність і зворотні відносини власностей [3].

Стиль кадру абстрактного синтаксису робить його набагато легшим для читання (у порівнянні з синтаксисом RDF/XML), а також легшим для розуміння і використання. Більш того, абстрактні синтаксичні аксіоми є прямою відповідністю з аксіомами описової логіки, і вони також можуть бути відображається в наборі RDF трійок.

2.1.3 Синтаксис RDF

Третім основним впливом на конструкціїю OWL була вимога збереження максимальної сумісності з існуючими веб – мовами, і, зокрема, з RDF [8]. На перший погляд, ця вимога мала смисл в RDF (і, зокрема RDF Schema), де вже включені деякі з основних особливостей класу та властивостей, основаних на мовах онтологій. Крім того, розвитку RDF передувала OWL, і це здавалося розумним, спробувати звернутися до будь–якого співтовариства користувача вже встановленому RDF [3].

Це може здатися простим: задовольнити цю вимогу просто даючи OWL синтаксис оснований на RDF. Для того, щоб забезпечити максимальну сумісність вгору, однак, було також визнано необхідним, щоб семантика OWL

онтологій також узгоджувалась з семантикою RDF [3]. Це виявилося складним завданням, оскільки значно зросла складність представлення OWL.

2.1.4 Попередники OWL

OWL – не перша веб–орієнтован мова онтологій, її дизайн був розроблений на основі кількох вже існуючих мов, включаючи RDFS, SHOE, OIL, DAML–ONT і DAML+OIL. DAML+OIL, зокрема, мав головний вплив на OWL і статут робочої групи веб–онтологій прямо вказано, що дизайн OWL повинен бути заснований на DAML+OIL. DAML+OIL в свою чергу, у великій мірі залежав від мови OIL, з додатковим впливом від DAML–ONT та RDFS [4].

2.1.4.1 SHOE

Одною з перших спроб визначення мови онтологій для розгортання в Web була SHOE. SHOE рамко–орієнтована мова з синтаксисом XML, що може бути безпечно вбудована в існуючих документах HTML. SHOE використовує URI посилання на імена, що є важливим нововведенням, це, згодом було прийнято як DAML–ONT і DAML+OIL. SHOE також використовує те, що онтології будуть щільно взаємопов'язані і можуть бути змінені. Отже, SHOE включає ряд директив, які дозволили імпортування інших онтологій, місцевого перейменування імпортованих констант, встановлення версій і сумісності інформації між онтологіями. Ця лінія мислення має важливий вплив на екстра–логічний словник OWL, який призначений для частково рішення таких питаннь. SHOE меншою мірою вплинув на синтаксичні та семантичні конструкції OWL, оскільки вона не була засновані на RDF, і не приходять з формальної семантики.

2.1.4.2 DAML–ONT

У 1999 році програма DARPA Агента мова розмітки (DAML) була ініційована з метою забезпечення основи для наступної генерації

"семантичних" Web [6]. В якості першого кроку було вирішено, що прийняття спільної мови онтологій сприятиме семантичній сумісності через різні проекти, що складають програму. RDFS (який вже був запропонований як стандарт W3C) був помічений в якості гарної відправної точки, але не був достатньо виразним, щоб задовольнити вимоги DAML. Нова мова DAML–ONT була розроблена, як продовження RDF з мовними конструкторами від об'єктно–орієнтованих і рамко–орієнтованих уявлень про мову[4].

DAML–ONT був тісно інтегрований з RDFS, і, в той же час, це було корисно з точки зору сумісності, потім це призвело до серйозних проблем в конструкції мови. Як RDFS, DAML–ONT страждав від невідповідної смисловий специфікації, і невдовзі стало зрозуміло, що це може призвести до розбіжностей, і серед людей та машин, і серед термінів онтології DAML–ONT. Крім того, DAML–ONT мав обмеження властивостей, як і RDFS, глобальної, а не локальної області, і в той час це було розумно для обмежень доменів і діапазонів, передбачених RDFS, глобальні потужності обмеження.

2.1.4.3 OIL

У той же самий час, коли DAML–ONT розроблялась, група (в основному європейських) дослідників з цілями, аналогічними дослідників DAML була розроблена інша веб–орієнтована мова онтологій під назвою OIL (Ontology Inference Laye) [4]. OIL була перша мова онтологій, що об'єднувала елементи з описової логіки, мову рамок і веб–стандартів, таких як XML і RDF. В OIL зроблено сильний акцент на формальній строгості, і мова явно розроблена таким чином, що її семантика може бути визначена за допомогою відображення SHIQ описової логіки [5]. Структура мови була, однак, на основі рамок. OIL мала обидва: і XML і RDF синтаксиси, але синтаксис RDF був розроблений для забезпечення сумісності з RDFS.

2.1.4.4 DAML+OIL

Стало очевидно, для DAML–ONT і OIL, що обом групам слід об’єднати зусилля, результатом яких було злиття DAML–ONT і OIL, створення DAML+OIL. Розвиток DAML+OIL було проведено комітетом в основному з членів двох проектних команд мови, і досить велично під назвою Joint US/EU ad hoc Agent Markup Language Committee.

Об'єднана мова має формальну семантику, отриману від DL теорії моделей. У DL надала мові конструктори OIL були збережені в DAML+OIL, але рамкова структура в значній мірі відкинулась на користь аксіом в стилі DL, які були легше інтегрувати з синтаксисом RDF.

Під впливом DAML–ONT, DAML+OIL більш тісно інтегрувався з RDF. DAML+OIL, однак,лише для тих частин, які були в RDF у відповідності зі своїм власним синтаксисом і моделі в стилі DL теорії. Це не було занадто великою проблемою, враховуючи, що RDF в той час не мали офіційно визначені смили, але стало серйозними труднощами для використання в DAML+OIL якості основи для OWL[7].

2.1.5. Проблеми

Множинний вплив на OWL призвів до ряду проблем. Деякі з цих проблем є результатом суперечливий вимог таких, як конфлікт між використанням RDF/XML як офіційного синтаксису OWL при цьому легко читаючого синтаксису. Деякі з цих проблем виникають з необхідності продовження попередніх рішеннь, а в проблемах, що виникають при розробці розширення для RDF, який включає інформацію, що не вписуються у світогляд RDF[8].

2.1.5.1 Синтаксичні проблеми

Існує ряд причин, у тому числі підтримання зв'язку з рамковою і описовою логікою, OWL повинен мати легкий для читання синтаксис, який можна буде легко зрозуміти і легко створити. Тим не менш, ця вимога є до OWL що використовують XML як нормативний синтаксису, і, крім того, використовують аким же чином, як в RDF [5]. Ця вимога вже була адресована до OIL, а потім, по DAML+OIL: OIL має і RDF/XML і XML синтаксис, у той час як DAML+OIL має тільки синтаксис RDF/XML [8].

2.1.5.2 Семантичні проблеми

Після того як питання синтаксису були розглянуті, питання, пов'язані зі змістом, все ще залишаються. RDF забезпечує значення для кожної трійки, так що OWL слід розглядати з розширенням RDF, в тому сенсі, що OWL має забезпечувати значення для трійок.

Це не було великою проблемою, коли OIL і DAML+OIL були розроблені. В OIL, зокрема, не потрудились зв'язати RDF зміст з RDF/XML синтаксисом в OIL значенні цього синтаксису – синтаксис RDF/XML для деяких OIL конструкцій робить більш–менш прийнятну збіжність з RDF значеннями цих конструктів, але це не в якому разі не справа для всіх таких конструкцій. Наприклад, OIL має особливу властивість, що використовується для зв'язку класів слотів, але зміст RDF цієї властивості, а саме стандарт змісту, що присвоєний будь–якій RDF трійці ігнорується семантикою OIL[7].

2.1.5.3 Виразна сила

Багато речей очікувалось від OWL (довгий список цілей проектування, вимоги і завдання), там було багато замовлень на виразні потужності, що дозволяли виходити за рамки, що в цілому забезпечується описовою логікою. Наприклад, багато користувачів хотіли б мати можливість зіставити інформацію з класами і властивостями і зробити класи належними до інших

класів, наскільки це можливо в RDF. Крім того, існує багато вимог для виразної сили, що виходить за рамки RDF і RDFS. Наприклад, багато користувачів хотіли, щоб мати можливість забезпечити локальне редагування для значень властивостей, як це можливо в описовій логіці. OWL мала бути розроблена, щоб якимось чином дозволяти ці види виразності, зберігаючи зв'язки зі своїм корінням. Коли DAML+OIL був розроблений, тільки тип даних підтримується RDF був літеральним : приблизно невизначені значення, наведені в рядках [3]. Таким чином, DAML+OIL забезпечував своє власне рішення для типів даних, і зробив так, що дозволяв використовувати XML. Тим не менш, будь–яке розумне рішення для типів данних, яке використовує тільки синтаксис RDF потребує допомоги від RDF (тобто, розширення синтаксису RDF), і таким чином, DAML+ OIL залишився незавершеним.

2.1.5.4 Обчислювальні завдання

Одним з аспектів OWL що відрізняє його від RDF і RDFS це те, що він підтримує багатий набір висновків. Деякі з цих висновків цілком очевиднію. Інші висновки, підтримувані OWL, проте, є досить складними, що вимагає, наприклад, обрахунків у ланцюгах властивостей. Беручи все представницькі бажання для OWL створено формалізм, де ключові проблеми виведення були нерозв'язні. Крім того, деякі аспекти RDF, такі як використання класів в якості випадках, взаємодія з іншими OWL для полегшення обчислювальних складностей, приймаючи OWL, що виходить за межі практичних алгоритмів і впроваджених систем міркування. OWL, таким чином, повинен був забезпечити вирішення цих обчислювальних проблем, зберігаючи сумісність з вгору і RDF RDFS.

2.1.6 Шляхи вирішення

Робоча група Web–онтологій провела багато частину часу для зняття основних напруженостей, що лежать в основі вищевказаних проблем. Складність полягає не в кожній проблемі окремо, але в комбінації всіх

перерахованих вище проблем і обмежень, що висунуті до на конструкції OWL. Це було б набагато легше, наприклад, для задоволення всіх вищевказаних вимог, якщо тільки в OWL могли б використовувати розширення синтаксису RDF. Якби це було дозволено, OWL міг би додати новий, природний синтаксис для своїх побудов, семантика не була б зобов'язані нести уздовж потрійний сенс RDF.

Проте життєздатним рішенням було знайдено, що задовольняє всі вищевказані вимоги. Або, насправді, правильніше буде сказати, що три рішення були знайдені, кожен з яких задовольняє майже всі з вищевказані вимоги.

OWL DL: Якщо дружній синтаксис або висновок вважається найважливішим, то OWL DL, версія OWL з вирішуваним виведення, які можуть бути записані в рамкові або описовій логіці.

OWL Lite: Якщо навіть простіший синтаксис і поступливішим висновки вважаються першорядним значенням, то OWL Lite, синтаксичний підмножина OWL DL, є доцільним.

OWL Full: Якщо сумісність з RDF і RDFS вважається найбільш необхідною властивістю, то OWL Full, синтаксичний і семантичний розширення RDFS, є доцільним.

2.1.6.1 Читабельність

OWL є не надто читабельним при записі в RDF/XML, або навіть RDF трійок. Частиною цієї проблеми є те, що RDF/XML надзвичайно багатослівні, але більша частина проблеми – читаність кодування

OWL конструкцій в RDF/XML або RDF трійках.

В рамках вирішення цієї проблеми був створений абстрактний синтаксис OWL, поряд з відображенням від абстрактного синтаксису RDF в графах. Цей абстрактний синтаксис ближче до синтаксису OIL, ніж з DAML + OIL, але без чіткого акценту на читаність, я у OIL. У цьому абстрактному синтаксис прикладу «Студент» буде написано :

```
Class(Student complete
                Person
                Restriction(enrolledIn minCardinality(1))).
```

OWL DL тоді визначатиметься як синтаксична підмножина OWL індукованого перекладу від абстрактного синтаксису RDF в графах. Тобто, граф RDF є онтологія OWL DL тільки тоді, коли це переклад якоїсь онтології в абстрактному синтаксисі. Люди та інструменти, які більше зацікавлені в читабельності, ніж в RDF/XML можуть використовувати цей абстрактний синтаксис всередині, або навіть зовні для представлення користувачів, залишаючи синтаксис RDF/XML для цілей обміну між додатками.

2.1.6.2 Робота з спотвореними графами

Оскільки OWL Full дозволяє довільні RDF графи, він повинен бути в змозі обробляти синтаксис неправильних (спотворених) OWL. (OWL DL не страждає від цієї проблеми, як це де визначено в термінах обов'язково добре освічених RDF графах, які можуть бути отримані від абстрактного синтаксису) OWL використовує розширення DAML+OIL рішення : тільки трійки, які разом утворюють добре утворені OWL конструкції, дають додатковий зміст, так :

```
:x owl:onProperty ex:child .
```

сам по собі не має ніякого особливого значення OWL.

Для обробки випадків з великою кількістю трійок, OWL знову використовує DAML + OIL вибираючи вже добре сформовані підмножини і даючи їм OWL значення.

В результаті отрмано незвичайні наслідки, наприклад :

:x owl:onProperty ex:child .

:x owl:someValuesFrom ex:Person .

:x owl:onProperty ex:friend .

:x owl:allValuesFrom ex:Doctor .

Отже, прирівнюючи розширення чотирьох різних обмежень OWL (всі можливі комбінації двох властивостей з двома класами), отримуємо майже напевно не те, що було призначено користувачем. Це рішення, однак, стверджує, монотонність, і (можливо) неінтуїтивний сенс проблеми, враховуючи, що такі потворні конструкції можна легко уникнути.

Відсутність структури в RDF графів повинна бути оброблена семантичнимих засобами, які будуть описані нижче.

2.1.6.3 Практичне забезпечення семантичної теорії для OWL

RDF тепер має теорію моделі, що повністю спирається на поняття слідування, OWL повинен забезпечити сумісність вгору з теорію моделі, що правильно обробляє висновки над OWL конструкціями. Це було б просто, якщо OWL вдалося розширити синтаксис RDF, а потім ці нові біти синтаксису могли б мали OWL тільки сенс. Тим не менш, це була вимога, щоб OWL мав синтаксис RDF, і що цей синтаксис здійснюється всією своєю RDF значенні. Ця вимога сумісності з двостороннім набагато сильніше, ніж зазвичай накладається між слабким формалізму (як логіки висловлювань) і сильного

формалізму (як логіка першого порядку), де сильнішеий формалізм дозволив розширити синтаксис слабкого формалізму.

Найбільш важким аспектом цієї проблеми є те, що в OWL синтаксичні конструкції, які кодуються у вигляді декількох трійок, повинні зберегти RDF значення для цих трійок. Як і всі трійки RDF, що виконують семантичні умови, не можуть бути виведені з нічого. Замість цього, в семантиці OWL довелося додати спеціальні обмеження, які по суті стверджують, що кожен переклад OWL повинен включати певні конструкції. (Такі обмеження, як правило, називають принципом розуміння.) Наприклад, один принцип розуміння для OWL стверджує, що кожна модель повинна включати в себе всі кінцеві списки класів; ще кажуть, що кожен такий список має бути відповідний клас перетину, вимагаючи існування певного класу, який пов'язаний в список, owl:intersectionOf властивостей.

Ці принципи розуміння підтримують слідування від

ex:John rdf:type ex:Student .

ex:John rdf:type ex:Employee .

до

_:c owl:intersectionOf :l1 .

_:l1 rdf:first ex:Student .

_:l1 rdf:rest :l2 .

_:l2 rdf:first ex:Employee .

_:l2 rdf:rest rdf:nil .

ex:John rdf:type :c .

тому що в _:l1 може бути необхідним перелік студентів та співробітників, в _:l2 може бути необхідним кінець цього списку, і _:c може бути необхідним перетин списку. Додаткові (і більш–звичайні) семантична умови вимагають, щоб приклад: ex:John belong to :c, finishing, отримав усе необхідне.

2.1.6.4 Уникнення парадоксів

Принципи розуміння дуже потужні, так як вони створюють щось з нічого (або, принаймні, щось з дуже малого). Ця сила може легко призвести до серйозних проблем.

Наприклад, парадокс Рассела – є парадокс саме через принципи розуміння, вбудовані в ранні версії теорії множин. Це рання версія теорії множин мала принципи розуміння (усвідомлення) структури речей, які задовольняються формулою з однієї вільною змінною, наприклад, формула буття людського, x ∈ людина, може бути використаний для побудови набору людей, {x | x ∈ людина}. На жаль, з формули, не належать до себе, x ∉ X, виникає безліч {x | x ∉ X }. Цей набір викликає проблеми там, де він існує, тому що це неможливо визначити, чи належить він до себе. Принцип розуміння він існує скрізь, тим самим викликаючи цю ранню версію теорії множин.

Аналогічна ситуація може виникнути з OWL. Здається, природно, хочеться, щоб кругові OWL, як конструкції, наприклад, класів, екземпляри створюються тільки для інших екземплярів класу, наприклад, в :

_:c owl:onProperty ex:child .

_:c owl:allValuesFrom _:c .

які можуть бути використані в наївному уявленні деяких аспектів біології. Однак, маючи принципи розуміння для таких кругових класів, можнае легко призвести до вимоги існування класів, як :

_:c owl:onProperty rdf:type .

_:c owl:allValuesFrom _: d .

_:d owl:complementOf _:l .

_:l rdf:first _:c .

_:l rdf:rest rdf:nil ,

які не мають відношення до типу самого класу. Об'єкти, що належать до цього класу не можуть належати йому ж, і навпаки, якщо принципи розуміння потрібні для існування цього класу, то кожна OWL онтологія буде парадоксальною.

Щоб уникнути цих парадоксів, принципи розуміння в OWL ніколи не вимагають наявності круглих ланцюгів посиланнь, як описано вище. Тим не менш, це не означає, що висновки, які можна очікувати, такі як ті, за участю конструкцію ex:child вище, наприклад, мають людину без дітей ставляться до такої конструкції, які не дійсні в OWL. При розробці цих принципів розуміння буловитрачено величезну кількість зусиль (більша частина яких : визначення основних правил принципів).

2.1.6.5 Збереження розв’язності

OWL Full нерозв'язна (з ряду причин), і навіть OWL DL може бути легко нерозв'язною, якщо в ній буде включено певну конструкцію, що призводить до нерозв'язності в описовій логіці. Тому OWL DL була ретельно створена так,щоб залишатись вирішуваною, і не включають в себе, наприклад, відносини між рольовими ланцюгами, які могли б призвести до нерозв'язних шляхів вбудовуванням проблемниху слів у OWL DL.

Це не означає, що висновки в OWL DL не складні. OWL DLмає складну проблему втілення, а припущення в SHOIN(D) має найгірший

недетерміностійкий експоненціальний час (NExpTime) складності, і OWL DL повинен мати ту ж складність. Гірше того, є як ще не відомі "практичні" повні алгоритми виведення в SHOIN(D), тобто такі, які, швидше за все, добре працюють на певних видах проблем, з якими стикаються в типових додатках. При відсутності такого алгоритму, поведінка OWL DL міркувань, ймовірно, буде менш передбачуваною (і з точки зору часу, необхідного для відповіді на запити, і використання системних ресурсів), і вони можуть іноді повертатися відповідь "невідомо" у відповідь на запити.

OWL Lite краще в цьому відношенні. Висновок в SHIFT (D) має гіршому випадку детермінованою експоненціальне час (ExpTime) складності [11], і в OWL Lite має ту ж складність. Крім того, існують практичні оптимізовані алгоритми для виведення в OWL Lite, такі як алгоритм, що лежить в основі факт описової логічної систем FaCT [22] і RACER [14]. Ці системи показали роботу з реальними додатками, і дали можливість роботи з великими онтологіями.

2.1.7 OWL

OWL як описова логіка OWL DL – стиль використання OWL, на основі описової логіки – дуже близький до SHOIN(D) Описової логіки, яка сама є розширенням впливового SHOQ (D) Описової логіки [23] (розширений зворотними ролями і обмеженний некваліфікованою кулькістю обмеженнь). OWL DL може формувати описи класів, типів даних, фізичних осіб і значень даних з використанням конструкцій (Рис 33). У цій таблиці перший стовпець показує OWL абстрактний синтаксис для, в той час як друга колонка дає стандартний синтаксис описової логіки.

OWL DL використовує ці описи для формування конструкції в аксіомах, які надають інформацію про класи, властивості і приватних осіб (Рис 34). Знову ж таки, кадр як абстрактний синтаксис дається в першій колонці, а стандартний синтаксис описової логіки – в другому стовпці.

2.1.7.1 Семантика для OWL DL

Формальна семантика, дуже схожа на семантику, передбачену в описовій логіці, для цього стилю, використовується OWL.

Оскільки OWL має типи данних, семантика для OWL дуже подібна з описовою логікою. Яка також включає типи данних, зокрема SHOQ (D). Проте конкретні типи даних, що використовуються в OWL взяті з RDF Schema і XML типів даних . Значення даних, такі як "44"^^xsd:integer, таким чином, означають, що вони будуть означати як значення даних XML–схеми.

Специфічні значення, що надаються OWL DL описам показано у третій колонці (Рисунок 1.1), де Δ^{τ} є областю осіб у моделі і Δ_D^{τ} це область значень даних. Як звичайно, значення аксіом в термінах обмежень на моделі, як показано в третій колонці на (Рисунок 1.2)

Abstract Syntax	DL Syntax	Semantics
Descriptions (C)		
A (URI reference)	A	$A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$
`owl:Thing`	$\top$	$\texttt{owl:Thing}^{\mathcal{I}} = \Delta^{\mathcal{I}}$
`owl:Nothing`	$\bot$	$\texttt{owl:Nothing}^{\mathcal{I}} = \{\}$
`intersectionOf`(C_1 C_2 ...)	$C_1 \sqcap C_2$	$(C_1 \sqcap D_1)^{\mathcal{I}} = C_1^{\mathcal{I}} \cap D_2^{\mathcal{I}}$
`unionOf`(C_1 C_2 ...)	$C_1 \sqcup C_2$	$(C_1 \sqcup C_2)^{\mathcal{I}} = C_1^{\mathcal{I}} \cup C_2^{\mathcal{I}}$
`complementOf`(C)	$\neg C$	$(\neg C)^{\mathcal{I}} = \Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}$
`oneOf`(o_1 ...)	$\{o_1, \ldots\}$	$\{o_1, \ldots\}^{\mathcal{I}} = \{o_1^{\mathcal{I}}, \ldots\}$
`restriction`(R `someValuesFrom`(C))	$\exists R.C$	$(\exists R.C)^{\mathcal{I}} = \{x \mid \exists y.\langle x,y\rangle \in R^{\mathcal{I}} \text{ and } y \in C^{\mathcal{I}}\}$
`restriction`(R `allValuesFrom`(C))	$\forall R.C$	$(\forall R.C)^{\mathcal{I}} = \{x \mid \forall y.\langle x,y\rangle \in R^{\mathcal{I}} \rightarrow y \in C^{\mathcal{I}}\}$
`restriction`(R `hasValue`(o))	$R : o$	$(\forall R.o)^{\mathcal{I}} = \{x \mid \langle x, o^{\mathcal{I}}\rangle \in R^{\mathcal{I}}\}$
`restriction`(R `minCardinality`(n))	$\geqslant n\, R$	$(\geqslant n\, R)^{\mathcal{I}} = \{x \mid \sharp(\{y.\langle x,y\rangle \in R^{\mathcal{I}}\}) \geqslant n\}$
`restriction`(R `minCardinality`(n))	$\leqslant n\, R$	$(\geqslant n\, R)^{\mathcal{I}} = \{x \mid \sharp(\{y.\langle x,y\rangle \in R^{\mathcal{I}}\}) \leqslant n\}$
`restriction`(U `someValuesFrom`(D))	$\exists U.D$	$(\exists U.D)^{\mathcal{I}} = \{x \mid \exists y.\langle x,y\rangle \in U^{\mathcal{I}} \text{ and } y \in D^{\mathbf{D}}\}$
`restriction`(U `allValuesFrom`(D))	$\forall U.D$	$(\forall U.D)^{\mathcal{I}} = \{x \mid \forall y.\langle x,y\rangle \in U^{\mathcal{I}} \rightarrow y \in D^{\mathbf{D}}\}$
`restriction`(U `hasValue`(v))	$U : v$	$(U : v)^{\mathcal{I}} = \{x \mid \langle x, v^{\mathcal{I}}\rangle \in U^{\mathcal{I}}\}$
`restriction`(U `minCardinality`(n))	$\geqslant n\, U$	$(\geqslant n\, U)^{\mathcal{I}} = \{x \mid \sharp(\{y.\langle x,y\rangle \in U^{\mathcal{I}}\}) \geqslant n\}$
`restriction`(U `maxCardinality`(n))	$\leqslant n\, U$	$(\leqslant n\, U)^{\mathcal{I}} = \{x \mid \sharp(\{y.\langle x,y\rangle \in U^{\mathcal{I}}\}) \leqslant n\}$
Data Ranges (D)		
D (URI reference)	D	$D^{\mathbf{D}} \subseteq \Delta_{\mathbf{D}}^{\mathcal{I}}$
`oneOf`(v_1 ...)	$\{v_1, \ldots\}$	$\{v_1, \ldots\}^{\mathcal{I}} = \{v_1^{\mathcal{I}}, \ldots\}$
Object Properties (R)		
R (URI reference)	R	$R^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$
	R^-	$(R^-)^{\mathcal{I}} = (R^{\mathcal{I}})^-$
Datatype Properties (U)		
U (URI reference)	U	$U^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta_{\mathbf{D}}^{\mathcal{I}}$
Individuals (o)		
o (URI reference)	o	$o^{\mathcal{I}} \in \Delta^{\mathcal{I}}$
Data Values (v)		
v (RDF literal)	v	$v^{\mathcal{I}} = v^{\mathbf{D}}$

Рисунок 1.1 – OWL DL опис, діапазони даних, властивості, індивіди, і значення даних

Abstract Syntax	DL Syntax	Semantics
Class(A partial C_1 ...C_n)	$A \sqsubseteq C_1 \sqcap \ldots \sqcap C_n$	$A^{\mathcal{I}} \subseteq C_1^{\mathcal{I}} \cap \ldots \cap C_n^{\mathcal{I}}$
Class(A complete C_1 ...C_n)	$A = C_1 \sqcap \ldots \sqcap C_n$	$A^{\mathcal{I}} = C_1^{\mathcal{I}} \cap \ldots \cap C_n^{\mathcal{I}}$
EnumeratedClass(A o_1 ...o_n)	$A = \{o_1, \ldots, o_n\}$	$A^{\mathcal{I}} = \{o_1^{\mathcal{I}}, \ldots, o_n^{\mathcal{I}}\}$
SubClassOf(C_1 C_2)	$C_1 \sqsubseteq C_2$	$C_1^{\mathcal{I}} \subseteq C_2^{\mathcal{I}}$
EquivalentClasses(C_1 ...C_n)	$C_1 = \ldots = C_n$	$C_1^{\mathcal{I}} = \ldots = C_n^{\mathcal{I}}$
DisjointClasses(C_1 ...C_n)	$C_i \sqcap C_j = \bot, i \neq j$	$C_i^{\mathcal{I}} \cap C_j^{\mathcal{I}}\{\}, i \neq j$
Datatype(D)		$D^{\mathcal{I}} \subseteq \Delta_{\mathbf{D}}^{\mathcal{I}}$
DatatypeProperty(U super(U_1)...super(U_n)	$U \sqsubseteq U_i$	$U^{\mathcal{I}} \subseteq U_i^{\mathcal{I}}$
domain(C_1) ...domain(C_m)	$\geqslant 1\, U \sqsubseteq C_i$	$U^{\mathcal{I}} \subseteq C_i^{\mathcal{I}} \times \Delta_{\mathbf{D}}^{\mathcal{I}}$
range(D_1) ...range(D_l)	$\top \sqsubseteq \forall U.D_i$	$U^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times D_i^{\mathcal{I}}$
[Functional])	$\top \sqsubseteq \leqslant 1\, U$	$U^{\mathcal{I}}$ is functional
SubPropertyOf(U_1 U_2)	$U_1 \sqsubseteq U_2$	$U_1^{\mathcal{I}} \subseteq U_2^{\mathcal{I}}$
EquivalentProperties(U_1 ...U_n)	$U_1 = \ldots = U_n$	$U_1^{\mathcal{I}} = \ldots = U_n^{\mathcal{I}}$
ObjectProperty(R super(R_1)...super(R_n)	$R \sqsubseteq R_i$	$R^{\mathcal{I}} \subseteq R_i^{\mathcal{I}}$
domain(C_1) ...domain(C_m)	$\geqslant 1\, R \sqsubseteq C_i$	$R^{\mathcal{I}} \subseteq C_i^{\mathcal{I}} \times \Delta^{\mathcal{I}}$
range(C_1) ...range(C_l)	$\top \sqsubseteq \forall R.C_i$	$R^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times C_i^{\mathcal{I}}$
[inverseOf(R_0]	$R = (^- R_0)$	$R^{\mathcal{I}} = (R_0^{\mathcal{I}})^-$
[Symmetric]	$R = (^- R)$	$R^{\mathcal{I}} = (R^{\mathcal{I}})^-$
[Functional]	$\top \sqsubseteq \leqslant 1\, R$	$R^{\mathcal{I}}$ is functional
[InverseFunctional]	$\top \sqsubseteq \leqslant 1\, R^-$	$(R^{\mathcal{I}})^-$ is functional
[Transitive])	$Tr(R)$	$R^{\mathcal{I}} = (R^{\mathcal{I}})^+$
SubPropertyOf(R_1 R_2)	$R_1 \sqsubseteq R_2$	$R_1^{\mathcal{I}} \subseteq R_2^{\mathcal{I}}$
EquivalentProperties(R_1 ...R_n)	$R_1 = \ldots = R_n$	$R_1^{\mathcal{I}} = \ldots = R_n^{\mathcal{I}}$
AnnotationProperty(S)		
Individual(o type(C_1) ...type(C_n)	$o \in C_i$	$o^{\mathcal{I}} \in C_i^{\mathcal{I}}$
value(R_1 o_1)...value(R_n o_n)	$\langle o, o_i \rangle \in R_i$	$\langle o^{\mathcal{I}}, o_i^{\mathcal{I}} \rangle \in R_i^{\mathcal{I}}$
value(U_1 v_1)...value(U_n v_n))	$\langle o, v_i \rangle \in U_i$	$\langle o^{\mathcal{I}}, v_i^{\mathcal{I}} \rangle \in U_i^{\mathcal{I}}$
SameIndividual(o_1 ...o_n)	$o_1 = \ldots = o_n$	$o_i^{\mathcal{I}} = o_i^{\mathcal{I}}$
DifferentIndividuals(o_1 ...o_n)	$o_i \neq o_j, i \neq j$	$o_i^{\mathcal{I}} \neq o_j^{\mathcal{I}}, i \neq j$

Рисунок 1.2 – OWL DL Аксіоми і факти

Семантика OWL DL дійсно включає деякі незвичайні (для описової логіки) аспекти. Анотації даються прості самостійні значення, що не показано тут, які можуть бути використані для зв'язку інформації з класами, властивостями, та особами, в манері, сумісній з семантикою RDF. Онтології також живуть в семантиці і можуть бути інформовані анотацією. Нарешті, owl:imports дається значення, яке включає знаходження посиланнь онтології (якщо це можливо) і додавання його сенсу у значенні поточної онтології.

Це робить OWL DL семантичною Web мовою, тому, це не його семантика, які цілком стандартна для описової логіки, але замість того, щоб використовувати URI посилань на імена, використовується XML Schema тип

даних для значень даних, і можливість підключення до документи в мережі інтернет.

2.1.7.2 OWL: OWL Lite

OWL DL пов'язаний з SHOIN(D) дуже виразною описовою логікою. Ця описова логіка досить важка для уяви наївних користувачів, оскільки можна будувати складні описові логічні описи, використовуючи, наприклад, об'єднання і доповнення. Суттєвою причиною складності використання SHOIN(D) є те, що вона має NExpTime складності, ось чому важко будувати інстументи навіть без міркування, оскільки опис їх досить складний.

З цих причин, підмножина OWL DL була ідентифікована, як та, що повинна бути легше на всіх перерахованих вище показниках; це підмножина називається в OWL Lite. OWL Lite забороняє об’єднання і доповнення, обмежує перетин, неявних перетинів в аксіоми класу кадрів, як, обмежує всі вбудовані описи назв концепції, не дозволяює людям показаувати в описах або аксіом класу, і обмеження потужностей в 0 або 1.

Ці обмеження в OWL Lite схожі на описову логіку SHIF(D). Як SHIF(D), ключові висновки в OWL Lite можуть бути обчислені в гіршому випадку за експоненціальний час (ExpTime), і вже є кілька оптимізованих міркувань для логік, еквівалентній OWL Lite. Це поліпшення поставляється з відносно невеликою втратою в виразній силі, хоча синтаксис OWL Lite є більш обмеженим, ніж OWL DL, ще можливо висловити складні описи, вводячи нові імена класу та експлуатуючи неявні заперечення, введені дезз'юнктністю аксіом. Використовуючи ці методи, всі описи OWL DL можуть бути захоплені в OWL Lite, за винятком тих, які містять або окремі імена або значення потужності більше, ніж 1.

2.1.7.3 OWL Full як розширення RDF

OWL DL і OWL Lite є розширеннями обмеженого використання RDF і RDFS, тому що, на відміну від RDF і RDFS, вони не дозволяють класи, які будуть використовуватися в якості фізичних осіб, а мовні конструктори не можуть бути застосовані до самої мови. Для користувачів, які потребують цих можливостей, версія OWL, що сумісна з RDF і RDFS була створена; ця версія називається OWL Full. В OWL Full, всі комбінації RDF і RDFS допускаються. Наприклад, в OWL Full, можна накласти кардинальність обмеження на rdfs:subClassOf, якщо це бажано.

OWL Full містить OWL DL, але йде далеко за межами стандартних рамок описової логіки. Штраф виплачується тут два рази. По–перше, розмірковування в OWL Full нерозв'язні (бо обмеження, необхідні для того, щоб підтримувати розв’язність OWL DL не можуть бути застосовані до OWL Full). По–друге, абстрактний синтаксис для OWL DL є недостатнім для OWL Full, і офіційний OWL синтаксис, RDF/XML, має бути використаним.

2.1.7.4 Майбутні розширення

Очевидно, OWL – це не останнє слово в мовах отнологій для семантичного вебу. Ряд особливостей були вже ідентифікований в документі OWL вимоги [17], і багато інших знаходяться в стадії обговорення. У цьому розділі коротко (і не вичерпно) перелічено деякі з цих можливих розширень і удосконалень в OWL:

2.1.7.5 Модулі та імпорт

Імпорт онтологій, що визначається іншими буде нормою на Semantic Web. Тим не менш, імпорт з OWL об'єктів є дуже не тривіальним : вона дозволяє імпортувати всю онтологію, в обране місце. Навіть якщо необхыдно використовувати невелику частину іншого онтології, ми змушені імпортувати всю цю онтологію. Модуль–конструкції в мовах програмування засновані на понятті приховування інформації: модуль обіцяє забезпечити деяку

функціональність із зовнішнім світом (експорт–застереження модуля), але модуль імпорту не потрібно здійснювати, як ця функціональність досягається. Це відкрите питання дослідження, що відповідне поняття приховування інформації для онтологій може бути, і як він може бути використаний в якості основи для гарного імпортного будівництва

За замовчуванням багато практичні системи подання знань дозволяють успадкував значення бути скасоване більш конкретних класів фантастичні С в ієрархії, лікування успадковані значення як значення за замовчуванням. Хоча широко використовується в практичній КР, консенсусу не було досягнуто на правому формалізації для немонотонної поведінки значення за замовчуванням.

Семантика OWL даний час приймає стандартну логічну модель відкритого припущення : заява не може бути передбачена правда на основі її недоведеності. Очевидно, на величезному і лише частково пізнаному World Wide Web це правильне припущення. Тим не менш, протилежний підхід (замкнутий світ припущення: твердження вірне, коли його заперечення не можуть бути доведені) також корисний в деяких додатках. Замкнутий світ припущення тісно пов'язаний з поняттям дефолтів, і призводить до тієї ж немонотонної поведінки.

Унікальні імена припущення. Типові програми баз даних припущень, що люди з різними іменами дійсно різні люди. OWL слід має логічну парадигму, де це не так. Якщо дві людини (або класи, або властивості) мають імена і далі різні, то ми, як і раніше можемо отримати від висновку, що вони повинні бути однаковими. Як з незамкненою припущенням, що не унікальні імена припущення найбільш правдоподібні, щоб на World Wide Web, але, як і колись, існують ситуації, коли унікальні імена припущення корисні. Більш тонко, можна вказати частини онтології, для яких припущення робиться.

Процесуальна прихильність. Загальна концепція в представлення знань є визначенням значення терміна не через явних де визначень в мові (як це робиться в OWL), але є шматок коду, який буде виконаний для обчислення значення терміна. Хоча широко використовується це поняття не піддається дуже добре інтеграції в системи з формальної семантики, і не були включені в OWL.

Ланцюжки нерухомості, правила. Як зазначалося вище, з міркувань разрешимости, OWL в даний час не дозволяють складу властивостей, але, звичайно, у багатьох додатках це корисно операції. Ще більш загальному, хотілося б, щоб визначити властивості, як загальними правилами (роги або іншим чином) в порівнянні з іншими властивостями. Таке об'єднання на основі правил DL в стилі даного часу активної області дослідження.

2.1.8 OWL 2

Логічним продовженням OWL стала мова OWL 2. OWL 2 Web Ontology Language, неофіційно OWL 2, є мовою онтологій для Semantic Web з формально визначеним смилсом. OWL 2 онтології надають можливыст працювати з класами, властивостями, індивидами і значеннями даних, також зберігати їх в Semantic Web документах. OWL 2 онтології можуть бути використані разом з інформацією, написаною в RDF, OWL 2 онтологій самі в першу чергу обмінюються документами в RDF.

Нижче (Рисунок 2.1) представлений короткий огляд мови OWL 2, де показано її основні стандартні блоки і те, як вони співвідносяться один з одним. Еліпс в центрі представляє абстрактне поняття онтології, яке можна розглянути, або як абстрактну структуру, або як граф RDF. Нагорі знаходяться різні конкретні синтаксиси, які можуть бути використані для серіалізациі та обміну онтологіями. Внизу малюнка розташовуються дві семантичні специфікації, які визначають значення онтологій OWL 2[9].

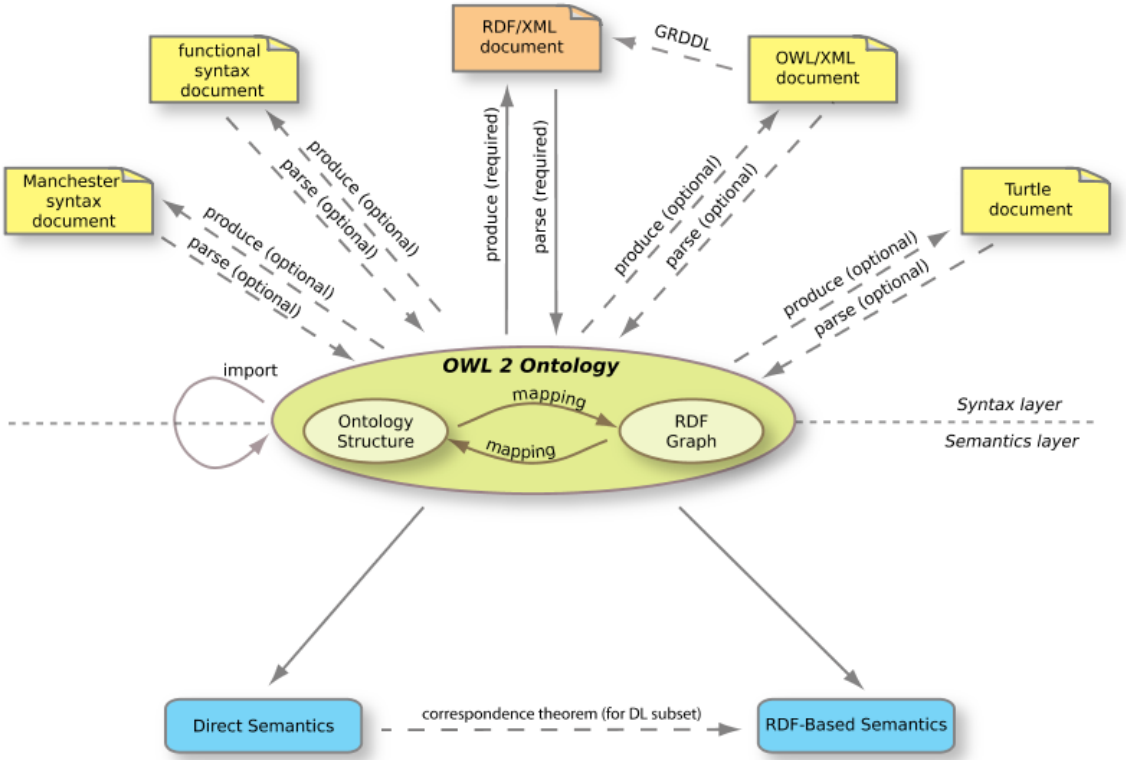


Рисунок 2.1 – Структура OWL 2

На даний момент актуальною вважається друга версія мови OWL, в якій визначаються такі різновиди :

– OWL 2 DL призначений для користувачів, яким потрібна максимальна виразність при збереженні повноти обчислень (всі логічні висновки, припускаються тієї чи іншої онтологією, будуть гарантовано вичіслімих) і разрешаемості (всі обчислення завершаться за певний час). OWL DL включає всі мовні конструкції OWL, але вони можуть використовуватися тільки згідно певним обмеженням (наприклад, клас може бути підкласом багатьох класів, але не може сам бути представником іншого класу). OWL DL так названий через його відповідності описова логіка - дисципліні, в якій розроблені логіки, складові формальну основу OWL. Існує три підмножини OWL DL, звані "профілями":

– OWL 2 EL, призначений для використання в додатках з великою кількістю властивостей і класів. На EL-онтологіях основні алгоритми логічного висновку гарантовано завершуються за поліноміальний час.

– OWL 2 QL, особливо корисний для онтологій, що містять безліч індивідів. У цьому профілі основний акцент припадає на забезпечення високої швидкості запитів до даних - вони відпрацьовують за логарифмічне час.

– OWL 2 RL призначений для запуску алгоритмів, заснованих на мовах правил. Включає відмінний від EL набір засобів і позиціонується як мова, що дозволяє підвищити виразність існуючих RDFS-онтологій.

– OWL 2 Full призначений для користувачів, яким потрібна максимальна виразність і синтаксична свобода RDF без гарантій обчислення.

Наприклад, в OWL Full клас може розглядатися одночасно як збори 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 буде в змозі здійснювати повну підтримку кожної особливості OWL Full.

У першій версії мови також були присутні підмножина OWL Lite, покликана обмежити виразність мови і підвищити швидкість алгоритмів. У новій версії стандарту OWL Lite відсутня.

OWL 2 EL дозволяє поліноміальні алгоритми часу для всіх стандартних завдань обгрунтувань (reasoning tasks); це є особливо підходящим для додатків, де необхідні дуже великі онтології, і де виразна сила може бути обміняна на гарантії виконання. OWL 2 QL дозволяє кон'юнктівние запити, що вимагають відповіді в LogSpace (точніше, AC0), з використанням стандартної технології реляційної бази даних; це є особливо підходящим для додатків, де відносно легкі онтології використовуються для того, щоб організувати велику кількість індивідів і де це буде корисно чи необхідно отримати доступ до

даних безпосередньо через реляційні запити (наприклад, SQL). OWL 2 RL дозволяє реалізації поліноміальних алгоритмів часу обгрунтувань (polynomial time reasoning algorithms), що використовують правила розширених технологій баз даних працюють безпосередньо на RDF-триплетах; це є особливо підходящим для додатків, де відносно легкі онтології використовуються для того, щоб організувати велику кількість індивідів і де це буде корисно чи необхідно працювати безпосередньо на даних у формі RDF-триплетів.

Будь-яка з онтологій EL, QL або RL OWL 2 є, звичайно ж, також онтологією OWL 2 і може бути інтерпретована за допомогою або Природною або заснованої на RDF Семантики. При використанні OWL 2 RL, заснована на правилах реалізація може працювати безпосередньо на RDF-триплетах і так само може бути застосований до довільного RDF-графу, тобто, до будь-якої онтології OWL 2. У цьому випадку, міркування завжди буде звуком (то Тобто, тільки коректні відповіді на запити будуть обчислені), але воно не може бути повним запитом (тобто, не гарантується, що всі коректні відповіді на запити будуть обчислені). Теорема PR1 з документа про профіль проте стверджує, що (в загальному), коли онтологія узгоджується з структурним визначенням OWL 2 RL, підходяща реалізація заснована на правилах виконує основні атомарні запити, буде і звуком і повним запитом.

2.1.9 Висновки (по OWL)

Через амбітні цілі дизайну для OWL, множинний вплив на OWL, а також через структурні вимоги, що обмежують OWL, розвиток OWL не обійшовся без проблем. Завдяки наполегливій роботі і компромісу, ці проблеми в значній мірі подолані, в результатом чого є мова онтології, яка дійсно є частиною семантичного вебу.

Не було можливим одночасно задовольнити всі обмеження на OWL, саме тому два стилі, що використовують OWL були розроблені, кожен з яких підходить для різних обставин.

Якщо виразна мова онтологій з визначенням висновків є головною метою, то OWL DL стиль вказується. Цей стиль за допомогою OWL втрачає деяку сумісність з RDF, перважно маючи відношення до використання класів і властивостей, як окремих осіб, але зберігає виразну і корисну мову онтологій. OWL DL також має кадроподібний синтаксис, як альтернативний синтаксис, який може використовуватися для роботу з OWL.

Навіть якщо OWL DL близький описової логіки, він включає в себе функції, які фантастичні твердо розмщені в семантичному вебі. OWL DL використовує механізми типів даних з RDF, і багато з побудованих в XML–схемах типів даних. OWL DL використовує RDF URI посилання, імена, у тому числі імена з RDF, RDF схема і XML–схеми типів даних, які мають відношення. Втілення в OWL DL сумісний з висновками в RDF і RDFS.

Якщо простіше, мова онтологій є головною метою, то в OWL Lite підмножина OWL DL може бути використана. Ця підмножина мови усуває деякі з речей, які можна сказати в OWL DL, але має ефективно–поступливий висновок, тісно пов'язаний з висновками в вже реалізованих, в декількох логічних описових системах, таких як FaCT [9] і RACER [14].

Якщо, з іншого боку, зворотна сумісність з RDF є основною проблемою, то слід обирати стиль OWL Full. Цей стиль поширюється RDF і RDF схеми повною мовою онтології, з добре визначеними обмеження права розпорядження власністю відносин, який простягається в RDF і RDFS, уникаючи будь–яких парадоксів, які можуть виникнути. Тим не менш, втілення OWL Full нерозв'язне, яке може бути значною проблемою в деяких обставинах. Крім того, зручний альтернативний синтаксис не є адекватним для OWL Full, так як RDF/XML повинні використовуватися для OWL Full.

Ці стилі, що використовуються в OWL забезпечують онтології шар для семантичного вебу, значно розширючи можливості RDF і RDFS, і розширюючи корисність Semantic Web.

Отже, звичайно, залишаються важливі питання, які навмисно не обробляються OWL, але які є нескінченно актуальні для випадків використання в семантичних веб:

– OWL уникає будь–яких зв’язків з немонотонностями (наприклад, міркування за замовчуванням і локалізованих закритих світових припущеннях);

– OWL обмежена виразність виключає такі операції, як власність–ланцюжки, або, більш загальні, аксіоми зі змінними (наприклад, правил);

– Механізм імпорту OWL є досить грубим, і не підтримує операції з визначеною зернистістю (наприклад, імпорт частин онтології);

– OWL об'єднує типи даних в дуже чистому вигляді, але немає ніякого поняття операцій на цими типами даних (наприклад, числової арифметики або строкових операцій).

Розширення поточного Semantic Web з деяких або всіх цих функцій потрібно не тільки зусилля по стандартизації, але, також, встановлення значимих досліджень проблем в суспільстві.

3.1 Висновки

Онтології мають велике майбутнє. Об’єм дослідженнь, що проводиться в цьому напрямку є доказом цього твердження.

Онтології розробляються і можуть бути використані при вирішенні різних завдань, у тому числі для спільного застосування людьми або програмними агентами, для можливості накопичення та повторного використання знань в предметній області, для створення моделей і програм, що оперують онтологіями.

Онтології можуть бути використані скрізь, де потрібна обробка даних, що враховує їх семантику. У силу початкової орієнтованості мови OWL на машинну обробку, правильне застосування онтологій може, з одного боку, істотно спростити і, з іншого боку, відкрити нові можливості в розробці додатків, що вирішують завдання автоматизованої обробки і доступу до даних.

Однією з переваг OWL-онтологій є доступність інструментів, які можуть робити логічні висновки. Побудова чіткої і працездатної системи логічних висновків — непроста справа, доступніше побудувати онтології. На сьогодні існують приклади онтологій (Protege [15], KAON2 [16] і Chimaera [18]), побудовані у сферах традиційно науковомістких галузей, починаючи від хімічної обробки, і закінчуючи конструюванням машинобудівних підприємств. Крім того, для специфічних наукових галузей існує низка ініціатив щодо побудови великомасштабної онтології. Однією з таких галузей є генетика, де багато зусиль було спрямовано на створення спільної термінології та визначень, щоб дозволити вченим керувати їхніми знаннями [2]. Ці зусилля дають уявлення про те, як онтології можуть відігравати значну роль у підтримці науковців.

Технологія SPARQL дозволяє отримувати дані з розподілених джерел і може бути як засіб інтеграції різнорідної інформації. У специфікації SPARQL відсутні недоліки, властиві традиційним мовам запитів, зокрема, не накладаються обмеження на формат даних. Завдяки цьому стає можлива взаємодія між ресурсами різного типу. Намагатися використовувати семантичну мережу без SPARQL — це все одно, що працювати з реляційною базою даних без мови структурованих запитів SQL. Тобто, SPARQL перетворює доступ до даних у деяку подібність Web-сервісу[2].

Одна з найбільш важливих завдань, яку можна вирішити, використовуючи онтології - це семантичний пошук. В даний час проблема пошуку інформації у великих масивах порівнюється з проблемою

Вавилонської вежі. Ця проблема ускладнюється ще й тим, що існуючі пошукові механізми здійснюють пошук інформації без урахування семантики слів, що входять до запиту, а також контексту, в якому вони використовуються.

Основоположними характеристиками інформаційно-пошукових систем є повнота і релевантність результатів пошуку. Повнота пошуку тісно пов'язана з оперативністю охоплення інформації системою. Створена одного разу база даних Інтернет-ресурсів є «зліпком» стану Мережі в конкретний момент. Якщо ця база не буде оновлюватися постійно і оперативно, присутні в ній посилання на документи стануть мертвими. Крім того, відсутність оперативності, оновлення баз даних не дозволить користувачеві відстежувати останні зміни в його предметної області. Повнота охоплення ресурсів Мережі - це один з двох головних аспектів характеристики повноти мережевий інформаційно-пошукової системи [11].

2 АЛГОРИТМИ ОТРИМАННЯ ЛОГІЧНИХ ВИСНОВКІВ

Для дослідження онтологій розробляються алгоритми (методи) що є досить індивідуальними для обраної предметної області. Проте метою будь-якого такого алгоритму, є отримання логічного висновку (нової інформації).

2.1 Метод прямого нечіткого висновку

У 1965 р в журналі «Information and Control» була опублікована робота Л. Заде під назвою «Fuzzy sets». Ця назва переведено на російську мову як нечіткі множини. Спонукальним мотивом стала необхідність опису таких явищ і понять, які мають багатозначним і неточний характер. Відомі до цього математичні методи, які використовували класичну теорію множин і двозначну логіку, не дозволяли вирішувати проблеми цього типу [11].

За допомогою нечітких множин можна формально визначити неточні і багатозначні поняття, такі як «висока температура» або «велике місто». Для формулювання визначення нечіткої множини необхідно задати так звану област міркувань. Наприклад, коли ми оцінюємо швидкість автомобіля, ми обмежимося діапазоном X = [0, Vmax], де Vmax - максимальна швидкість, яку може розвинути автомобіль. Необхідно пам'ятати, що X - чітке безліч

Основні поняття

Нечіткою множиною A в деякому непустому проторі X називается множина пар

$$A = \{\langle x, \mu_A(x)\rangle \,|\, x \in U\},$$

де

$$\mu_A : U \to [0,1]$$

— функція приналежності нечіткої множини A. Ця функція приписує кожному елементу x ступінь його належності нечіткій множині A.

Продовживши попередній приклад, розглянемо три неточні формулювання:

- «Мала швидкість автомобіля»;

- «Середня швидкість автомобіля»;

- «Велика швидкість автомобіля».

На малюнку представлені нечіткі множини, що відповідають наведеним формулювань, за допомогою функцій приналежності.

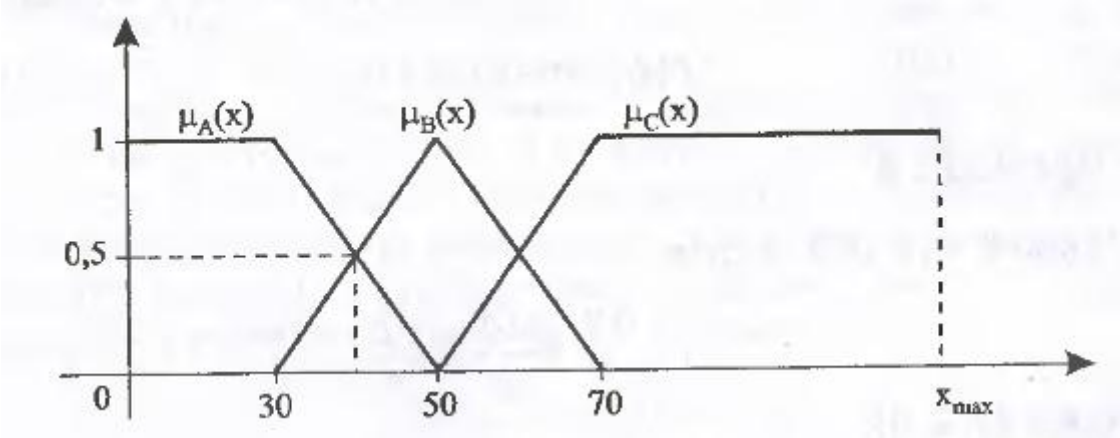


Рисунок 2.1 Нечітка логіка

У фіксованій точці X = 40 км / год. функція приналежності нечіткої множини «мала швидкість автомобіля» приймає значенням 0,5. Таке ж значення приймає функція прінадлежностінечеткого безлічі «середня швидкість автомобіля», тоді як для безлічі «велика швидкість автомобіля» значення функції в цій точці дорівнює 0.

Функція T двох змінних T: [0, 1] x [0, 1] –> [0, 1] называется T– нормою, якщо:

— є не зростаючою, щодо обох аргументів : T(a, c) < T(b, d) для a < b, c < d;

— є коммутативною : T(a, b) = T(b, a);

— задовольняє умові зв'язності : T(T(a, b), c) = T(a, T(b, c));

— задовольняє граничні умови : T(a, 0) = 0, T(a, 1) = a.

Прямий нечіткий висновок

Під нечітким висновком розуміється процес, при якому з нечітких посилок отримують деякі слідства, можливо, теж нечіткі. Наближені міркування лежать в основі здатності людини розуміти природну мову, розбирати почерк, грати в ігри, що вимагають розумових зусиль, загалом, приймати рішення в складній і не повністю певному середовищі. Ця здатність міркувань в якісних, неточних термінах відрізняє інтелект людини від інтелекту обчислювальної машини [12].

Основним правилом виводу в традиційній логіці є правило modus ponens, згідно з яким ми судимо про істинність висловлювання B по істинності висловлювань A і A -> B. Наприклад, якщо A - висловлювання «Степан - космонавт», B - висловлювання «Степан літає в космос» , то якщо істинні висловлювання «Степан - космонавт» і «якщо Степан - космнавт, то він літає в космос», то істинно і висловлювання «Степан літає в космос».

Однако, в отличие от традиционной логики, главным инструментом нечеткой логики будет не правило modus ponens, а так называемое композиционное правило вывода, весьма частным случаем которого является правило modus ponens.

Нехай, маємо криву y=f(x) і задано значення x=a. Тоді із того, що y=f(x) і x=a, ми можемо зрозуміти, що y=b=f(a).

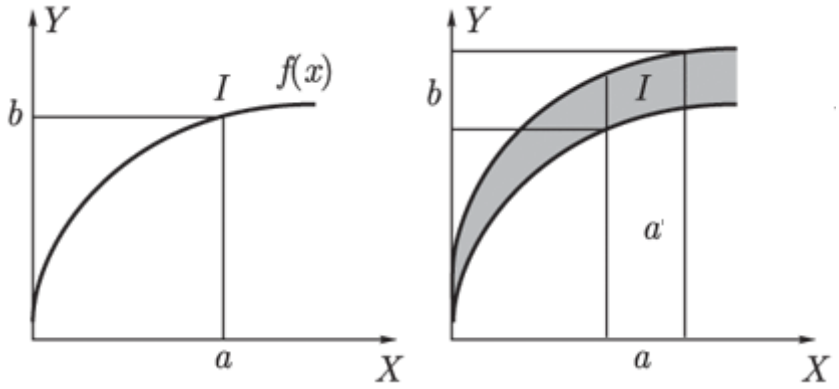


Рисунок 2.2 Нечітка логіка : прямий нечіткий висновок

Узагальнимо тепер цей процес, припустивши, що a - інтервал, а f (x) - функція, значення якої суть інтервали. У цьому випадку, щоб знайти інтервал y = b, відповідний інтервалу a, ми спочатку побудуємо множину a 'з основою a і знайдемо його перетин I з кривою, значення якої суть інтервали. Потім спроектуємо цей перетин на вісь OY і отримаємо бажане значення y вигляді інтервалу b. Таким чином, з того, що y = f (x) і x = A - нечітка підмножина осі OX, ми отримуємо значення y у вигляді нечіткої підмножини B осі OY.

Нехай U і V - два універсальних безлічі з базовими змінними u і v, відповідно. Нехай A і F - нечіткі підмножини множин U і U x V. Тоді композиційне правило виводу стверджує, що з нечітких множин A і F слід нечітка множина B = A * F.

Нехай A і B - нечіткі висловлювання і m (A), m (B) - відповідні їм функції приналежності. Тоді імплікації A -> B буде відповідати деяка функція приналежності m (A -> B). За аналогією з традиційною логікою, можна припустити, що

$$A \to B \equiv \neg A \vee B.$$

Тоді

$$\mu_{A \to B}(x, y) = \max\{1 - \mu_A(x), \mu_B(y)\}.$$

Однак, це не єдине узагальнення оператора імплікації, існують і інші.

Розглянута система правил нечітких продукцій не претендує на закінченість і служить лише ілюстрацією застосування основних ідей нечіткої логіки. У цьому зв'язку слід зазначити, що реальні експертні системи, засновані на правилах нечітких продукцій, можуть містити сотні окремих правил. У цьому випадку власне процес отримання нечітких висновків може перетворитися на серйозну проблему, що має самостійне значення і вимагає додаткових коштів для свого конструктивного рішення.

2.2 Алгоритм поширення активації

Оригінальним методом на підтримку семантичного пошуку є поширення активації – “spread activation”. Поширення активації працює над екземплярамионтологій доменів. Відношенням між концепціями онтології – властивостям концепцій у термінах OWL присвоюються чисельні ваги. Поширення активації призначене для виявлення тісно пов’язаних концепцій у екземплярі онтології за множиною заданих концепцій та відповідних значень активації. Ці початкові значення є результатами традиційного пошуку, застосованого до ресурсів (вузлів RDF – графа), анотованих категоріями із онтології. Алгоритм активації поширення суттєво залежить від домену. Відносна вага шляху в графі до певного екземпляра концепції може бути оцінена тільки у добре визначеному контексті. Тому, залежно від домену, на роботу алгоритму активації поширення накладаються різні обмеження (максимальна довжина шляху, максимальна кількість вузлів, які розгортаються алгоритмом тощо) [13]. Новаторство такого підходу в інтеграції традиційного пошуку та технологій поширення активації.

Головна проблема традиційних пошукових машин у тому, що релевантність ресурсу встановлюється за ключовими словами. Щодо семантичного пошуку, то вимога бути ознайомленим із концепціями домену ускладнює висловлення користувачем його інформаційних потреб. Із цих причин доцільно поставити у відповідність кожній концепції домену множину ключових слів. Через поширення активації поза результатами традиційного пошуку ми видобуваємо екземпляри концепцій, котрі пов’язані з концепціями, асоційованими із ключовими словами. Алгоритм поширення активації працює як “дослідник концепцій”. При заданій множині активованих концепцій та спеціальних обмежень на поширення, активація протікає у мережі та виявляє концепції, тісно пов’язані із початковими. Кожному екземпляру відношення приписується чисельна вага, оскільки використання семантичної інформації поряд із символьною повинне

покращити пошук. Таким чином, у роботі поширення активації працює із “гібридною мережею” [13]. Для автоматичного визначення ваги відношень пропонуються три метрики – кластерна, специфічності, комбінована.

Кластерна метрика. Ця метрика призначена на визначення подібності між екземпляром Cj та екземпляром концепції Ck.

$$W(C_j, C_k) = \frac{\sum_{i=1}^{n} n_{ijk}}{\sum_{i=1}^{n} n_{ij}}. \qquad (2.1)$$

nij дорівнює одиниці, якщо екземпляри концепцій Cj та Ck пов’язані між собою та нулю у протилежному випадку. nijk дорівнює одиниці, якщо обидві концепції – Cj та Ck одночасно пов’язані із Ci, інакше – нулю. За цими міркуваннями, вага W(Cj ,Ck) означає відсоток концепцій, із якими пов’язана Ck, за умови, що Cj також знаходиться у відношенні із цими концепціями. Ця міра показує, що концепції, які знаходяться в багатьох спільних відношеннях із іншими концепціями, є більш схожими. Аналог показника W(Cj ,Ck) у IR моделі – df.

Метрика специфічності. Ця метрика відповідає показнику idf у IR. Метрика специфічності слугує для визначення специфічності відношення (наскільки часто воно є вживаним у домені) і задається такою формулою:

$$W(C_j, C_k) = \frac{1}{\sqrt{n_k}} \qquad (2.2)$$

Значення nk еквівалентне числу екземплярів цього відношення, для яких концепція Ck є вузлом призначення (у термінах RDF-триплетів {<підмет>, <предикат>, <додаток>} – додатком). Тому вага відношення між концепціями Cj та Ck обернено пропорційна числу відношень, до котрих

входить Ck. Чим більшою є кількість відношень, у яких Ck виступає додатком, тим менша вага кожного з цих відношень.

Комбінована метрика. Комбінована метрика є добутком кластерної та метрики специфічності (за зразком добутку df · idf у IR моделі).

Алгоритм стартує із початкової множини екземплярів концепцій, здобутих традиційним пошуком. Цим вершинам присвоюються активаційні ваги (зазвичай вага активаційного вузла встановлюється в одиницю, а звичайного вузла – в нуль), під час поширення активуються інші вузли. Усі початкові вершини заносяться до черги з пріоритетами у не-зростаючому порядку значень ваг. Надалі із черги видобувається і обробляється вершина з найбільшим пріоритетом. Якщо поточний вузол задовольняє всі обмеження, то він поширює активацію на своїх сусідів. Нехай початкова вершина – i, а вершина призначення – j, тоді поширення активації до сусідів відбувається відповідно до такої формули, де I – вхід, O – вихід:

$$I_j(t+1)+ = O_i(t) \cdot w_{ij} \cdot f_{ij} \cdot (1-\alpha) \qquad (2.3)$$

Функція ваги вершини i (Oi(t)) додається до поточної ваги вхідної вершини j, помножена на вагу відношення між поточною і наступною вершиною wij (кластерну метрику), відносну вагу відношення fij (метрика специфічності) та α – відсоток вже активованих відношень на шляху від початкової вершини до j. Значення всіх цих трьох множників залежать від домену. Поступово всі активовані алгоритмом вузли додаються до черги пріоритетів. Процес продовжується до тих пір, доки не досягнуто бажаного стану (наприклад, до знаходження заздалегідь вказаної кількості екземплярів концепцій або у черзі пріоритетів немає вершин для обробки). По завершенні процесу всі знайдені вершини упорядковуються за порядком активації. Зауважимо, що алгоритм обробляє кожну вершину лише один раз; у термінах часової складності становить O(E) , де E – потужність множини відношень

графа екземпляра онтології. Операції над чергою пріоритетів мають часову складність $O(\log V)$, де V – кількість концепцій графа.

Тому тотальна часова складність алгоритму поширення активації становить $O(E*\log(V))$ ·

Активаційні ваги. Зауважимо, традиційна пошукова машина рангує результати за рівнем релевантності запиту. Кожній вершині RDF-графа присвоюється дійсне число з інтервалу (0;1] – ранг. Це значення виступає в ролі активаційної ваги вузла для алгоритму поширення активації. Саме ця множина вузлів RDF-графа є активаційною множною для алгоритму поширення активації.

Обмеження. Однією із проблем алгоритмів поширення активації є те, що коли процес не контрольований, то як результат пошуку може бути відібраний неадекватний граф. На вирішення цієї проблеми скеровані обмеження.

Приклади обмежень:

Обмеження на тип концепції: активуватись повинні тільки вершини певного типу.

Обмеження розгортання: повинні відбиратись тільки ті концепції, котрі знаходять у певних відношеннях із більшою/меншою кількістю концепцій, ніж задано.

Обмеження на довжину шляху: алгоритм поширення активації не повинен видобувати концепції, котрі знаходять на більшій відстані від вхідних концепцій, аніж задано (експериментально з'ясовано, що оптимальна довжина шляху знаходиться у межах від одиниці до трьох).

Схема семантичного пошуку із поширенням активації. Користувач задає пошукову фразу ключовими словами. припустимо, що усі документи мають семантичні анотації, із кожною концепцією асоційовано список ключових слів. Традиційна пошукова машина знаходить документи, семантичні анотації котрих містять екземпляри концепцій, заданих користувачем. Надалі над множиною здобутих семантичних анотацій (графа екземплярів концепцій) залучається до дії компонента поширення активації, і система презентує користувачу остаточні результати (Рисунок 2.3).

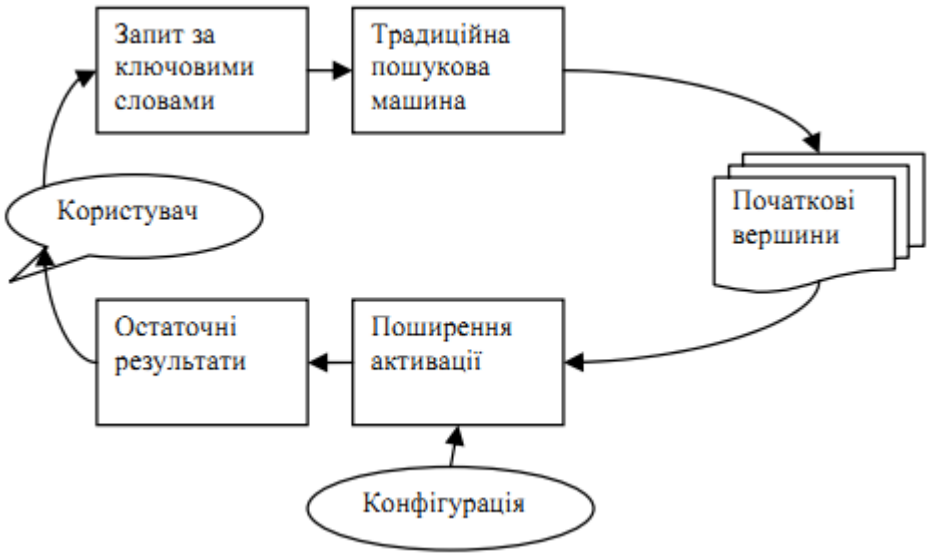


Рисунок 2.3 – семантичний пошук із поширенням активації

2.3 ELK-Reasoner

ELK розроблений для забезпечення високої продуктивності міркування підтримку OWL DL онтологій. Основна увага в системі (I) широке висвітлення можливостей OWL EL, (II) висока продуктивність міркувань, і (III) легко розширюваність і використання. У цьому відношенні, ELK вже може пропонувати переваги порівняно з іншими системами OWL EL, про які міркування згадувалося вище. Наприклад, на сьогоднішній день, ELK єдина система, яка може використовувати кілька процесорів/ядер для прискорення процесу міркування, що дозволяє класифікувати SNOMED CT в якості лише 5 секунд на стандартних апаратних засобів [24]. Ця стаття не передбачає яких-

небудь нових теоретичних результатів або експериментальні порівняння з іншими інструментами; ті можуть бути знайдені в наших попередніх роботах [25].

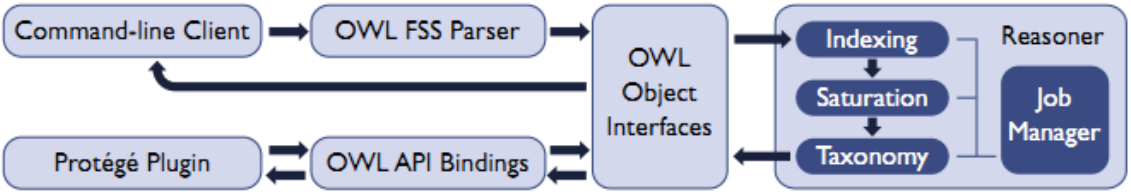


Рисунок 2.4 – Базові модулі ELK та інформаційні потоки при класифікації онтології

2.3.1 Системний огляд алгоритму

ELK є гнучкою системою, яка може бути використана в різних конфігураціях. Вона підтримує модульну структуру програми, яка організована за допомогою менеджера збірки Apache Maven для Java. Maven можна використовувати для автоматичного завантаження, конфігурацій, і побудови ELK і його залежностей, але є також вбудовані пакети для найбільш поширених конфігурацій. Модульна структура також відокремлює наслідко-основні міркування від решти компонентів, що полегшує розширення системи з новими функціями мови.

Description Logic Preliminaries

Коротко про описову логіку EL +, яка забезпечує більш лаконічний синтаксис, ніж OWL для обговорення міркування ELK. Співвідношення між OWL і DL пояснювалось вище розділі 1.

Словник EL + складається з рахункового нескінченної наборів (атомних) функцій і атомних понять. Комплексні поняття і аксіоми визначені рекурсивно, використовуючи конструктори в Рисунок 2.5 . Ми використовуємо букви R, S ролей, C; D, E для понять і, B для атомних понять.

Концепція еквівалентності C≡ D скорочує два включення C € D і D € C. З онтології створюється кінцевий набір аксіом.

	Syntax	Semantics
Roles:		
atomic role	R	$R^{\mathcal{I}}$
Concepts:		
atomic concept	A	$A^{\mathcal{I}}$
top	$\top$	$\Delta^{\mathcal{I}}$
conjunction	$C \sqcap D$	$C^{\mathcal{I}} \cap D^{\mathcal{I}}$
existential restriction	$\exists R.C$	$\{x \mid \exists y : \langle x, y\rangle \in R^{\mathcal{I}} \wedge y \in C^{\mathcal{I}}\}$
Axioms:		
concept inclusion	$C \sqsubseteq D$	$C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$
role inclusion	$R \sqsubseteq S$	$R^{\mathcal{I}} \subseteq S^{\mathcal{I}}$
role composition	$R_1 \circ R_2 \sqsubseteq S$	$\langle x, y\rangle \in R_1^{\mathcal{I}} \wedge \langle y, z\rangle \in R_2^{\mathcal{I}} \rightarrow \langle x, z\rangle \in S^{\mathcal{I}}$

Рисунок 2.5 – Синтаксис і семантика EL +

Враховуючи онтології O, ми пишемо $\sqsubseteq_{\mathcal{O}}^{*}$ для найменших рефлексивного бінарного відношення перехідної над ролями таких, як $R \sqsubseteq_{\mathcal{O}}^{*} S$ має місце для всіх $R \sqsubseteq S \in \mathcal{O}$. Будемо говорити, що концепція C відбувається негативно (позитивно) в онтології O, якщо C синтаксичної підвиразу D (відповідно. E) для деякого аксіома $D \sqsubseteq E \in \mathcal{O}$.

EL + має семантику Тарського стилю. Інтерпретація $\mathcal{I}$ складається з непорожньої безлічі $\Delta^{\mathcal{I}}$, що називається областю $\mathcal{I}$ і функція $\cdot^{\mathcal{I}}$інтерпретація, що привласнює кожен R бінарне відношення $R^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$ і для кожного A встановлює набір $A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$. Функція інтерпретації розширюється до складних концепцій, як показано на Рисунку (Рисунок 2.5). більше детально про синтаксис та семантику алгоритму написано в [25].

2.3.2 Основні частини алгоритму

Три основні частини, з яких складається алгоритми, що мають ядро модуля міркування ELK's (Рисунок 2.4) : індексація, насиченість і таксономічна структура.

2.3.2.1 Індексація

Завдання етапу індексації є створення внутрішнього подання в онтології O сайті, який підходить для перевірки побічних умов для правил виведення. Як показано на Рисунок 2.4, вхід, який використовується для індексації онтології в легкому уявлення ELK в OWL виразів і аксіом, які, можливо, були отримані з OWL FSS файлів, об'єктів OWL API, або будь-якого іншого джерела. У той час як уявлення ELK з OWL вірно захоплює структуру онтології з точки зору особливостей мови OWL, індексована онтологія представляє тільки опис логічної семантики, що має важливе значення для застосування правил виводу. Для цього, система індексованих об'єктів використовується для представлення DL (замість OWL) ролі і концепції вираження. Ці об'єкти є основою для всіх подальших внутрішніх індексних структур.

Індексування являє собою легку задачу, яка може бути виконана за допомогою одного рекурсивного обходу через структуру кожної аксіоми в онтології. Так як можна вважати, одну аксіому в той час, він може бути запущений навіть до того, як вся онтологія стане відомою для міркувача. У ELK, індексація виконується на другому потоці, паралельно навантаженні аксіоми. Крім того, якщо нові аксіоми додаються в онтології, достатньо, просто індексуіати їх, без перезавантаження всієї онтології.

2.3.2.2 Насиченість

На цьому етапі ми попередньо обчислити зворотним проходом транзитивне замикання $\sqsubseteq^*_{\mathcal{O}}$ включення ролі аксіоми в O, які необхідні для ефективного застосування правил $\mathsf{R}^+_\exists$ і $\mathsf{R}_\circ$ зберігати його в таблиці під назвою hier (для ієрархії ролей).

$$\mathsf{hier} = \{\langle R, S\rangle \mid R \sqsubseteq^*_{\mathcal{O}} S\} \qquad (2,1)$$

Так кількість ролей в реальних онтологій, як правило, набагато менше, ніж число понять, будь-який розумний алгоритм для обчислення транзитивного замикання може бути використаний без значного впливу на загальну продуктивність системи. Приклад :

$$\mathsf{hier} = \{\langle R, R\rangle, \langle R, S\rangle, \langle S, S\rangle\}$$

Далі обчислюємо дедуктивне закриття. Нагадаємо, що правила виведення працюють з трьома типами аксіом init(C), $C \sqsubseteq D$, $C \overset{R}{\rightarrow} D$. Такі аксіоми зберігаються в окремих таблицях init, subs, links :

axiom	internal representation
$\mathsf{init}(C)$	$C \in \mathsf{init}$
$C \sqsubseteq D$	$\langle C, D\rangle \in \mathsf{subs}$
$C \overset{R}{\rightarrow} D$	$\langle C, R, D\rangle \in \mathsf{links}$

Рисунок 2.6 Таблиці аксіом

Основна частина алгоритму насичення показано в алгоритмі 1 (Рисунок 2.7): для класифікації, він ініціалізує обчислення з ініціалізації (A) для всіх атомних понять, що відбуваються в онтології, а потім повторно обробляє аксіоми з todo, поки черга не спорожніє. Здійснення процесу функції (axiom) для кожного із трьох типів аксіом показано в алгоритмі 2. Зверніть увагу, що алгоритм 2 часто перебирає приєднується таблиць; оптимізації таких ітерацій має важливе значення для забезпечення ефективності алгоритму[26].

Algorithm 1: Main body of the saturation algorithm

```
1 for each IndexedAtomic A do
2     todo.add(init(A));
3 while todo ≠ ∅ do
4     axiom ← todo.poll();
5     process(axiom);
```

Рисунок 2.7 – Алгоритм 1

Algorithm 2: process(axiom)

```
1  process(init(C)):
2  if init.add(C) then
3      todo.add(C ⊑ C);                                   // rule R0
4      if top.negOccurs > 0 then
5          todo.add(C ⊑ top);                             // rule R⁺⊤
6  process(C ⊑ D):
7  if subs.add(⟨C, D⟩) then
8      if D instanceOf IndexedConjunction then
9          todo.add(C ⊑ D.firstConj);
10         todo.add(C ⊑ D.secondConj);                    // rule R⁻⊓
11     if D instanceOf IndexedExistential then
12         todo.add(init(D.filler));
13         todo.add(C --D.role--> D.filler);              // rule R⁻∃
14     for each D2, E with subs(C, D2) ∧ negConjs(D, D2, E) do
15         todo.add(C ⊑ E);                               // rule R⁺⊓
16     for each D1, E with subs(C, D1) ∧ negConjs(D1, D, E) do
17         todo.add(C ⊑ E);                               // rule R⁺⊓
18     for each E, F, R, S with links(E, R, C) ∧ negExis(S, D, F) ∧ hier(R, S) do
19         todo.add(E ⊑ F);                               // rule R⁺∃
20     for each E with conceptIncs(D, E) do
21         todo.add(C ⊑ E);                               // rule R⊑
22 process(E --R--> C):
23 if links.add(⟨E, R, C⟩) then
24     for each D, F, S with subs(C, D) ∧ negExis(S, D, F) ∧ hier(R, S) do
25         todo.add(E ⊑ F);                               // rule R⁺∃
26     for each D, R2, S1, S2, S with links(C, R2, D) ∧ roleComps(S1, S2, S) ∧
       hier(R, S1) ∧ hier(R2, S2) do
27         todo.add(E --S--> D);                          // rule R∘
28     for each D, R1, S1, S2, S with links(D, R1, E) ∧ roleComps(S1, S2, S) ∧
       hier(R1, S1) ∧ hier(R, S2) do
29         todo.add(D --S--> C);                          // rule R∘
```

Рисунок 2.8 – Алгоритм 2

2.3.2.4 Таксономічна структура

Фаза насичення концепція обчислює повний транзитивно-замкнуту категоризацію відносини. Тим не менше, очікується, вихід класифікації є таксономія, яка містить тільки пряму категоризації між вузлами, що представляють класи еквівалентності атомних понять. Таким чином, комп'ютерна категоризація між атомними поняттями повинна бути транзитивно зменшена. На етапі першої, необхідно відкинути всі категоризації, отримані за допомогою алгоритму насичення, які включають неатомарние поняття. Таким чином, в решти цього розділу, ми можемо припустити, що всі поняття атомарні. Приклад таксономії об’єктів показано на (Рисунок 2.9)

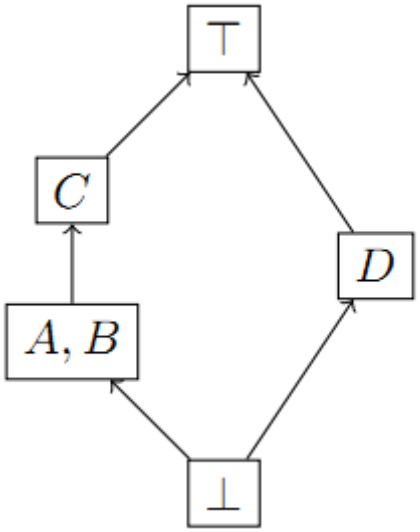


Рисунок 2.9 – Приклад таксономії

Більш детальніше про ELK можна дізнатись тут [25]. Подальше вдосконалення даного алгоритму полягає в перегляді логіки міркування, обчислення дедуктивного закриття, розробці нових правил обробки аксіом.

3 ПРОГРАМНІ ЗАСОБИ ДЛЯ РОБОТИ З ОНТОЛОГІЯМИ

На сьогоднішній день існує не менше десятка зарубіжних систем, що відносяться до класу інструментів онтологічного інжинірингу, які підтримують різні формалізми для опису знань і використовують різні машини виведення з цих знань .

Незважаючи на досить велику кількість систем, їх призначення орієнтоване на вузьке коло вирішення проблем. Кожна виконує своє призначення, тому їх застосування в експертних системах стає неможливим.

Найбільш відомі з них – це: Protégé, CYC, KAON2, OntoEdit, KADS22[16].

3.1 Protégé

Protégé – це вільний, відкритий редактор онтологій і фреймворк для побудови баз знань.

Платформа Protégé підтримує два основних способи моделювання онтологій за допомогою редакторів Protégé–Frames і Protégé–OWL. Онтології, побудовані в Protégé, можуть бути експортовані в безліч форматів, включаючи RDF (RDF Schema), OWL і XML Schema.

Protégé має відкриту, легко розширювану архітектуру за рахунок підтримки модулів розширення функціональності.

Protégé підтримується значним спільнотою, що складається з розробників і вчених, урядових і корпоративних користувачів, що використовують його для вирішення завдань, пов'язаних зі знаннями, в таких різноманітних галузях, як біомедицина, збір знань та корпоративне моделювання.

Protégé доступний для вільного скачування з офіційного сайту разом з плагінами і онтологіями [16].

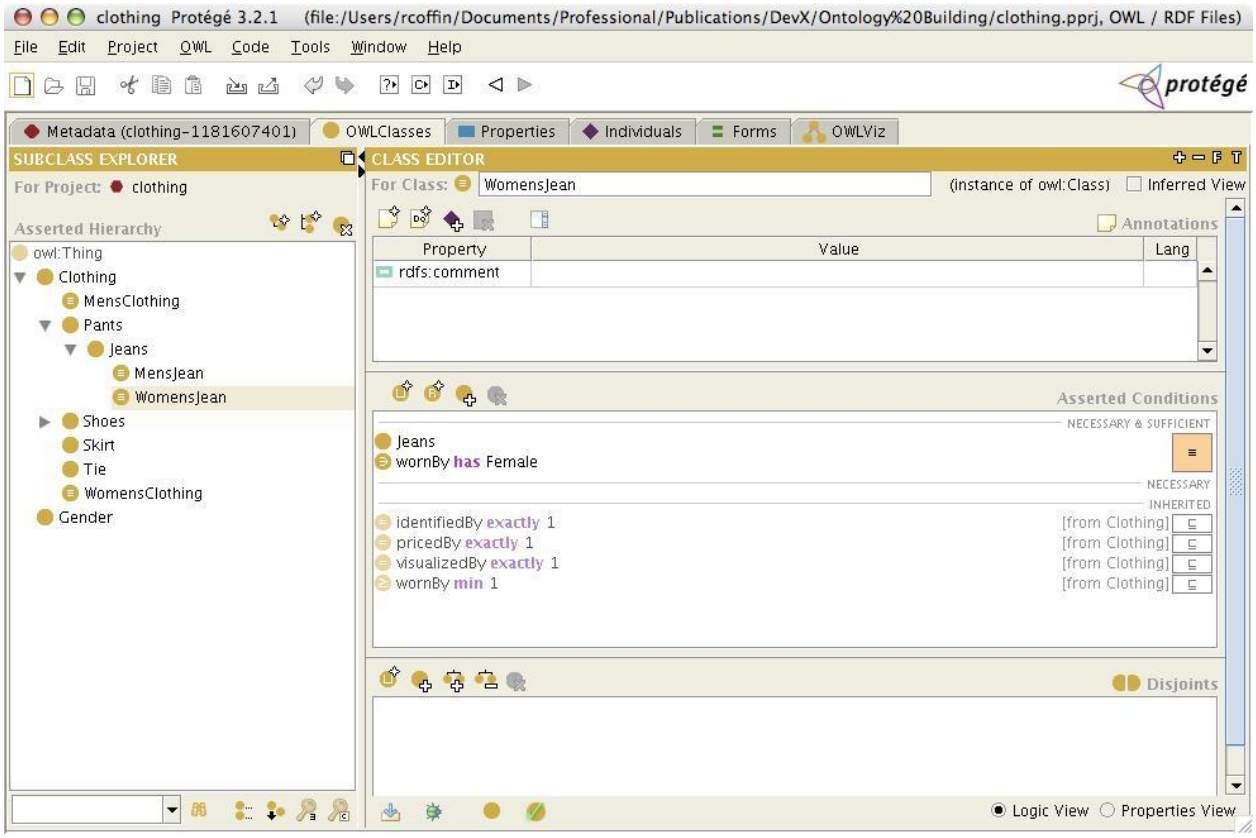


Рисунок 3.1 – Приклад діалогового вікна рдактора онтологій Protégé

3.2 CYC

CYC – проект по створенню об'ємної онтологічної бази знань, що дозволяє програмам вирішувати складні завдання з області штучного інтелекту на основі логічного висновку та залучення здорового глузду.

Проект розпочав Дуглас Ленат в 1984 році в Microelectronics and Computer Technology Corporation. Назва «Cyc» (утворене від англ. Encyclopedia і промовлене як «цик») є зареєстрованою торговою маркою компанії Cycorp, Inc в Остіні, якою управляє Ленат і створеної для розробки CYC. База знань є власністю компанії, однак невелика частина бази, призначена для встановлення загального словника для програм автоматичного

міркування, була випущена як OpenCYC під відкритою ліцензією. Пізніше, CYC стала доступною для дослідників II під спеціальною дослідницької ліцензією як ResearchCyc.

Типовим прикладом знань у базі є «Всяке дерево є рослиною» і «Рослини смертні». Якщо запитати «вмирають чи дерева?», Машина логічного висновку може зробити очевидний висновок і дати правильну відповідь. База Знань (англ. Knowledge Base або KB) містить більше мільйона занесених туди людьми тверджень, правил і загальновживаних ідей. Вони формулюються на мові CYCL, який заснований на обчисленні предикатів і має схожий з Ліспі синтаксис. Англомовні користувачі жартують що вони «велосипедисти» (від англ. Cyclist – велосипедист).

Велика частина сьогоднішньої роботи в проекті CYC все ще пов'язана з інженерією знань – опис фактів про навколишній світ вручну і реалізація ефективних механізмів логічного висновку на основі цих знань. Однак ведеться робота над тим, щоб дати системі CYC можливість самостійно спілкуватися з користувачами на природній мові, і над прискоренням процесу поповнення бази за допомогою машинного навчання [17].

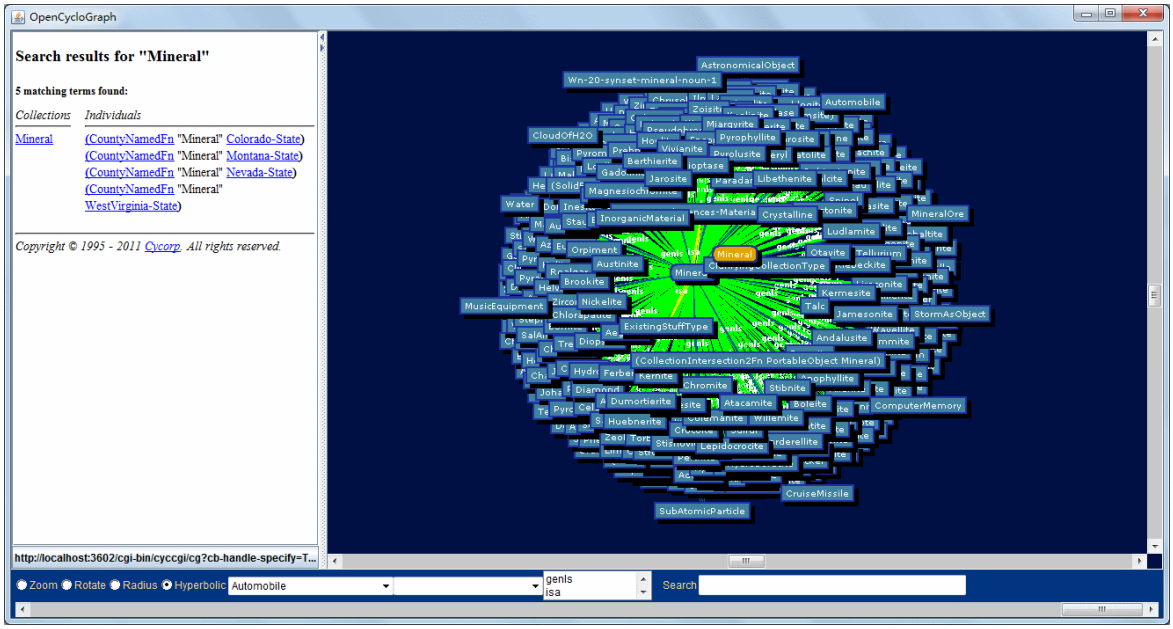


Рисунок 3.2 – Приклад діалогового вікна рдактора онтологій CYC

3.3 OntoEdit

OntoEdit – спочатку був розроблений в інституті AIFB (Institute of Applied Informatics and Formal Description Methods) Університету Karlsruhe (зараз комерціалізувати Ontoprise GmbH) виконує перевірку, перегляд, кодування і модифікацію онтологій. В даний час OntoEdit підтримує мови представлення: FLogic, включаючи машину виведення, OIL, розширення RDFS і внутрішню, засновану на XML, серіалізацию моделі онтології використовуючи OXML – мова представлення знань OntoEdit (OntoEdit's XML–based Ontology representation Language). До достоїнств інструмента можна віднести зручність використання; розробку онтології під керівництвом методології і за допомогою процесу логічного висновку; розробку аксіом; розширювану структуру за допомогою плагінів, а також дуже гарну документацію [19].

Існує дві версії OntoEdit: вільно поширювана OntoEdit Free (обмежена 50 концептами, 50 відносинами і 50 екземплярами) і ліцензована OntoEdit Professional (немає обмежень на розмір). Природно, що OntoEdit Professional має більш широкий набір функцій і можливостей (наприклад, машину виведення, графічний інструмент запитів, більше модулів експорту та імпорту, графічний редактор правил, підтримка баз даних JDBC і т.д.).

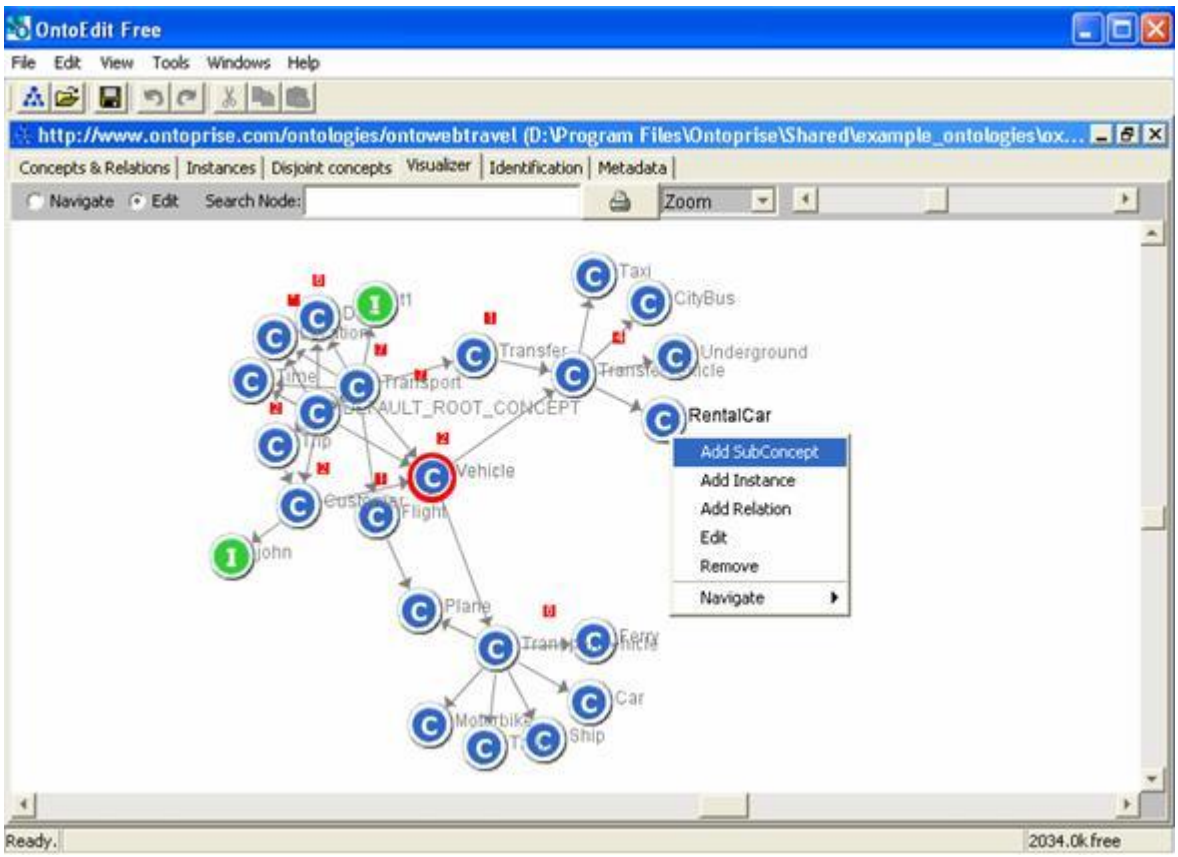


Рисунок 3.3 – Приклад діалогового вікна рдактора онтологій OntoEdit

3.4 KAON2

KAON2 (http://kaon2.semanticweb.org) – інфраструктура для управління онтологіями OWL–DL, SWRL, і F–Logic. Запити можуть бути сформульовані на SPARQL [4].

KAON2 надає наступні можливості: API для програмного керування OWL–DL, SWRL, і F–Logic онтології; автономний сервер, що забезпечує доступ до онтології в, використовуючи RMI; висновок двигуна для відповіді кон'юнктівних запитів (виражається з використанням синтаксису SPARQL); інтерфейс DIG, надаючи доступ до таких інструментів, як Protege; модуль для вилучення примірників онтології з реляційних баз даних [20].

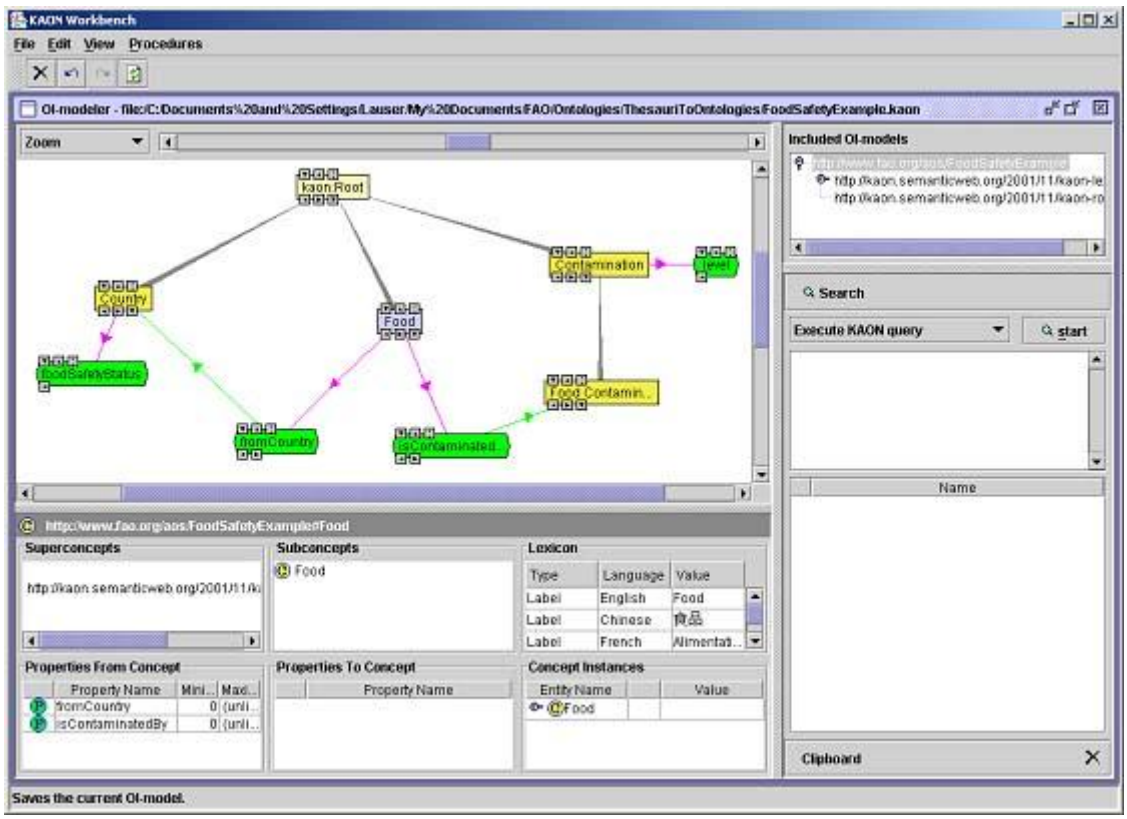


Рисунок 3.4 – Приклад діалогового вікна рдактора онтологій KAON2

3.5 KADS22

KADS22 – інструмент підтримки проектування моделей знань згідно методології CommonKADS. Онтології складають частину таких моделей знань (інша частина – моделі виводу). Моделі CommonKADS визначені в CML (Conceptual Modeling Language). KADS22 – інтерактивний графічний інтерфейс для CML з наступними функціональними можливостями: синтаксичний аналіз файлів CML, друк, перегляд гіпертексту, пошук, генерація глосарію та генерація HTML [21].

Великий внесок у сферу штучного інтелекту приніс кандидат наук за спеціальністю «Системи та засоби штучного інтелекту» Щербак С.С. , Створивши блог, де обговорюються різних проблем штучного інтелекту, онтологій і Semantic Web (Shcherbak.net).

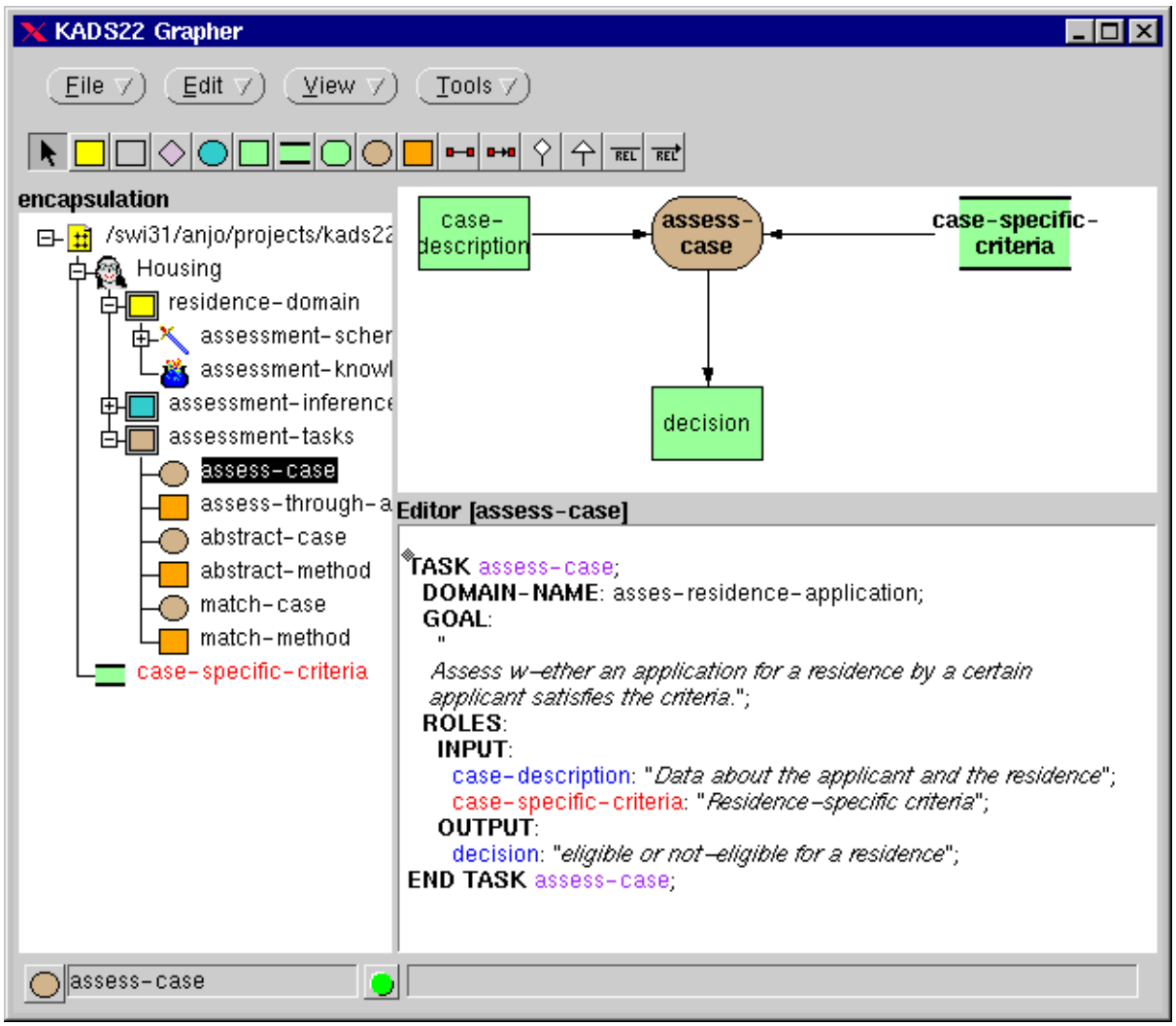


Рисунок 3.5 – Приклад діалогового вікна рдактора онтологій KADS22

3.6 Приклади отримання рішень

Нижче приведені деякі приклади отримання логічних висновків.

- Запит до freebase на мові MQL синтаксично схожий на JSON-запит,

```
[{
  "type": "/people/person",
  "name": "Barack Obama",
  "*": [{}]
}]
```

Він задає питання до онтології : обери мені всі об’єкти з типом people/person в яких імя Barack Obama і покажи всю доступну інформацію про них. С

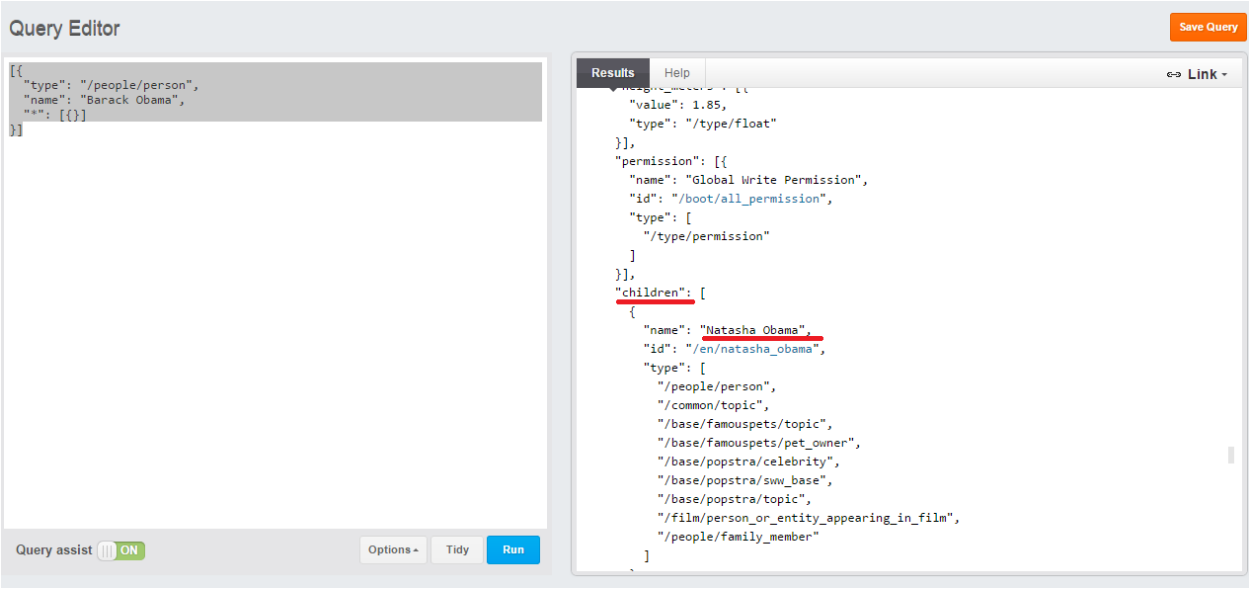


Рисунок 3.6 – приклад запиту до онтології за інформацією (знайдена інформація про дитину Барака Обами Наташу)

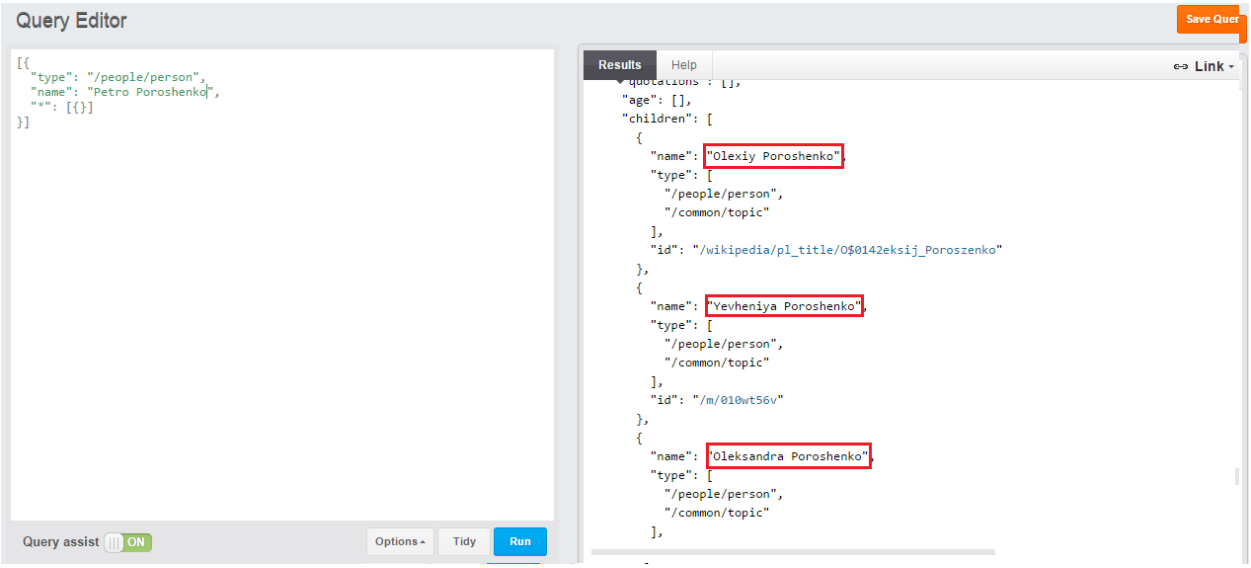


Рисунок 3.7 – приклад запиту до онтології за інформацією (знайдена інформація про дітей Петра Порошенка)

- Запит до онтології GO знайти по імені : molecula

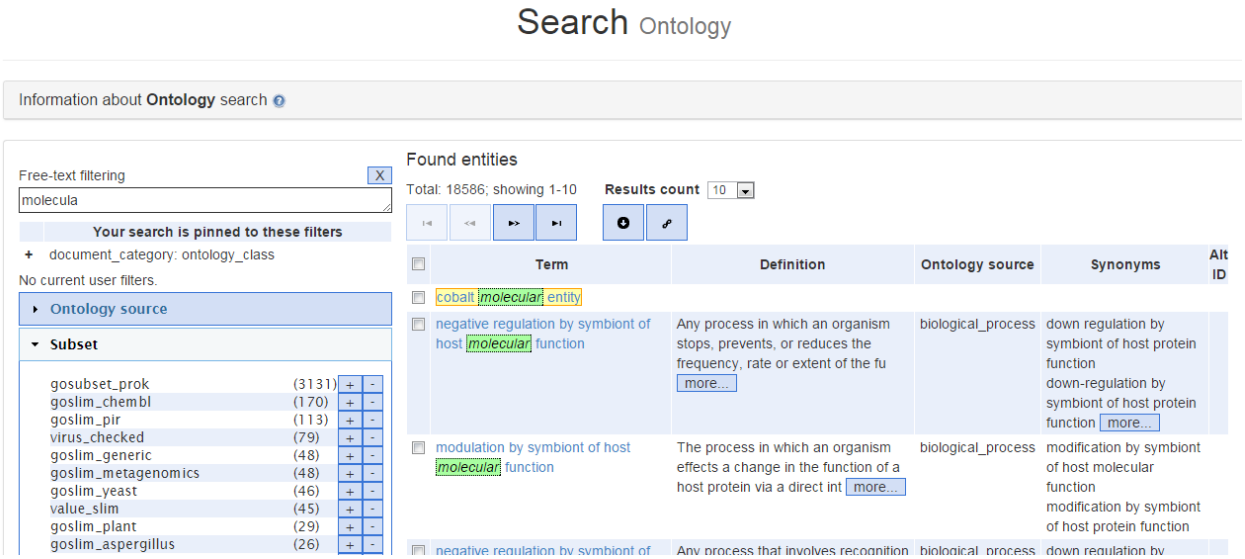


Рисунок 3.8 – приклад запиту до онтології за інформацією (знайдена інформація про дітей Петра Порошенка)

- Запит до локальної бази, використовуючи Protégé

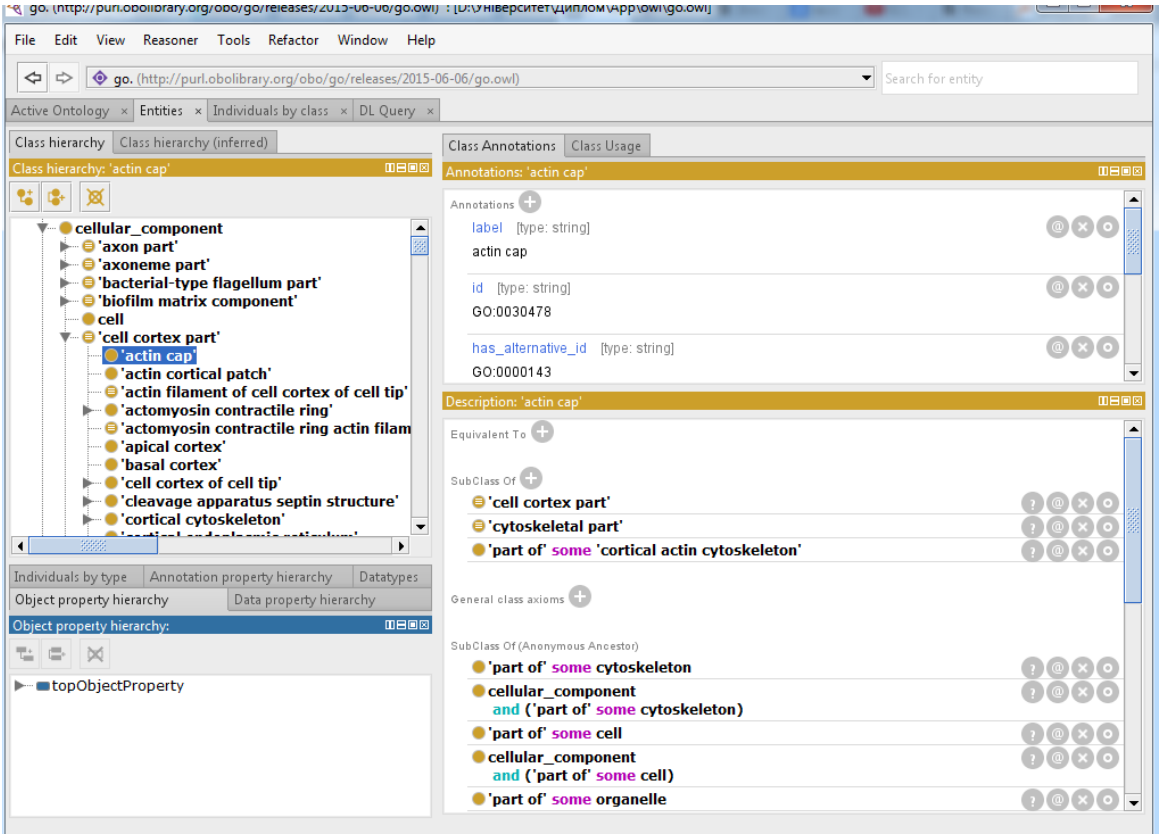


Рисунок 3.9 Запит в Protégé

3.7 Висновки

Широкий спектр технічних засобів роботи з онтологіями надає можливість дослідникам знаходити нові шляхи для пошуків логічних висновків в своїх дослідженнях. Програмні засоби дють можливість зливати (мерджити) між собою онтології, для отримання єдиної онтології з певної області знань.

Сумісне використання людьми або програмними агентами загального розуміння структури інформації є однією з найзагальніших цілей розробки онтологій [22]. Наприклад, нехай, декілька різних Web-сайтів містять інформацію з медицини або надають інформацію про платні медичні послуги, що сплачуються через Інтернет. Якщо ці Web-сайти спільно використовують і публікують одну і ту ж базову онтологію термінів, якими вони всі користуються, то комп'ютерні агенти можуть добувати інформацію з цих різних сайтів та накопичувати її. Агенти можуть використовувати накопичену інформацію для відповідей на запити користувачів або як вхідні дані для інших застосувань.

Забезпечення можливості використання знань у предметній галузі сприяло бурхливому розвитку у вивченні онтологій. Наприклад, для моделей багатьох різних предметних галузей необхідно сформулювати поняття часу, яке включає поняття часових інтервалів, моментів часу, відносних мір часу і т.ін. Якщо одна група вчених детально розробить таку онтологію, то інші можуть повторно використовувати її у своїх предметних галузях. Крім того, якщо нам необхідно створити велику онтологію, ми можемо інтегрувати декілька існуючих онтологій, які описують частини великої предметної галузі. Ми також можемо повторно використовувати основну онтологію, таку, як UNSPSC [23], і розширювати її для опису предметної галузі, що нас цікавить.

Створення явних допущень у предметній галузі, покладених в основу реалізації, надає можливість легко змінювати ці допущення при зміні наших знань про предметну галузь. Жорстке кодування припущень про світ мовою програмування призводить до того, що ці припущення не тільки складно знайти і зрозуміти, але й складно змінити, особливо непрограмістові. Крім того, явні специфікації знань у предметній галузі корисні для нових користувачів, які мають засвоїти значення термінів з цієї галузі.

Відокремлення знань предметної галузі від оперативних знань — це ще один варіант загального застосування онтологій. Ми можемо описати завдання конфігурації об'єкту і його компонентів відповідно до необхідної специфікації і запровадити програму, яка формує цю конфігурацію незалежною від об'єкту і самих компонентів. Після цього ми можемо розробити онтології компонентів і характеристик комп'ютерів, на яких базується об'єкт, та застосувати цю програму для морфологічного синтезу нестандартних конфігурації об'єктів.

Аналіз знань у предметній галузі можливий, якщо є декларативна специфікація термінів. Формальний аналіз термінів надзвичайно цінний як під час спроби повторного використання існуючих онтологій, так і при їх розширенні. Часто онтологія предметної галузі сама по собі не є метою. Розробка онтології схожа на визначення набору даних і їх структури для використання іншими програмами. Методи розв'язання завдань, доменно-незалежні застосування і програмні агенти використовують як дані онтології і бази знань, побудовані на основі цих онтологій [23].

Для кожної предметної галузі існує безліч онтологій залежно від степеня деталізації її структури. Прийнято говорити про спектр онтологій, який

містить:

• онтології малої структуризації — тахономії (наприклад, ієрархія Yahoo, біологічна тахономія), схеми баз даних і схеми метаданих (ebXML, WSDL);

• онтології середньої структуризації — тезауруси (WordNet, CALL, DTIC) і концептуальні моделі (Моделі ОО, UML);

• онтології високої структуризації — логічні концептуальні моделі TOVE, CYC, семантичний Web).

Є можливість обрати існуючу, чи розробляти власну онтологію, або створити власну і доовнити її власною.

4 РЕАЛІЗАЦІЯ АЛГОРИТМУ МІРКУВАННЯ

4.1 Алгоритм

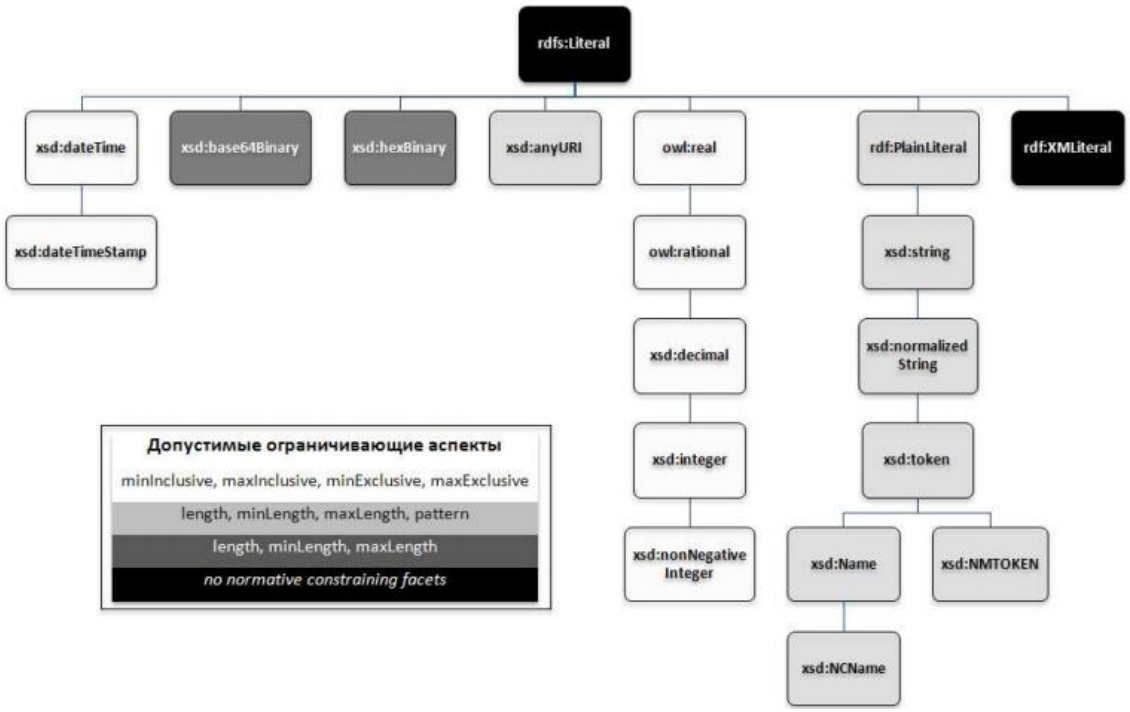
За основу для можливої модернізації був взятий алгоритм ELK-Reasoner, оскільки він має практичну реалізацію, і найбільш пристосований для роботи з OWL онтологіями.

Типізовані вирази з використанням конкретних типів даних матимуть першорядну роль в Семантичній павутині і необхідні для складання практично застосовних баз знань. Пропоную модифікацію високопродуктивного логічного процесора ELK з підтримкою типізованих властивостей в рамках профілю OWL 2 EL. Проведені випробування показали виняткову швидкість класифікації великих онтологій зі значною кількістю примірників і типізованих властивостей, що відкриває нові перспективи для застосування логічних процесорів на практиці.

ELK підтримує основні завдання логічної обробки онтологій, в тому числі перевірку на протиріччя, TBox-класифікацію, і обчислення конкретного типу іменованих примірників (FBox-реалізація). ELK на пряму не підтримує комплексні (анонімні) вираження, проте досить легко додати подібну підтримку шляхом додавання до вихідної онтології нового іменованого класу, еквівалентного анонімному висловом, для кожного такого виразу.

За результатами тестування [25] ELK здатний класифікувати онтологію SNOMED CT, яка містить більше 300 тисяч концептів, за кілька секунд, лідируючи в своєму класі. Але в ELK відсутня підтримка типізованих властивостей. Недавній аналіз використання OWL онтологій в мережі Інтернет показав, що типізовані властивості залишаються надзвичайно важливим елементом використовуваних в мережі онтологій, незважаючи на їх обмежену підтримку поширеними логічними аналізаторами.

Багаторазово висловлювалася думка, що типізовані вираження найчастіше є основною складовою баз знань, а їх частка в планованої Семантичної Павутині може скласти більше половини всіх виразів.



Рисуно 4.1 Допустимі типи даних в онтологіях OWL 2 EL профіля

Згідно специфікації мови OWL 2, профіль OWL 2 EL передба чає використання 19-і типів даних, більшість з яких виразно в рамках специфікації XSD - мови опису структури XML - документів. Рисуно 4.1 резюмує всі підтримувані в рамках профілю типи даних, встановлює їх спадковість і дозволені обмежувальні аспекти.

Важливо відзначити, що профіль OWL 2 EL не передбачає використання наступних типів даних: xsd:double, xsd:float, xsd:nonPositiveInteger, xsd:positiveInteger, xsd:negativeInteger, xsd:long, xsd:int, xsd:short, xsd:byte , xsd:unsignedLong, xsd:unsignedInt, xsd:unsignedShort, xsd:unsignedByte, xsd:language і xsd:boolean. Причина цього полягає в тому, що список дозволених профілем типів був складений таким чином, що б перетин просторів значень кожного з них з будь-яким іншим з типів був або порожнім, або нескінченним. Подібне обмеження допомагає забезпечити необхідні обчислювальні характеристики логічного аналізу EL онтологій[27].

4.2 Реалізація

Для реалізації підтримки вищезазначених аксіом з типізований виразами, було додано нові правила логічного висновку, описані на (Рисуно 4.2),

$$\frac{}{A \sqsubseteq \bot} \quad A \sqsubseteq \exists r_+ \in O,\ r_+ \rightarrow_D \bot$$

$$\frac{A \sqsubseteq \exists r^+}{A \sqsubseteq B} \quad \exists r_- \sqsubseteq B \in O,\ r_+ \rightarrow_D r_-$$

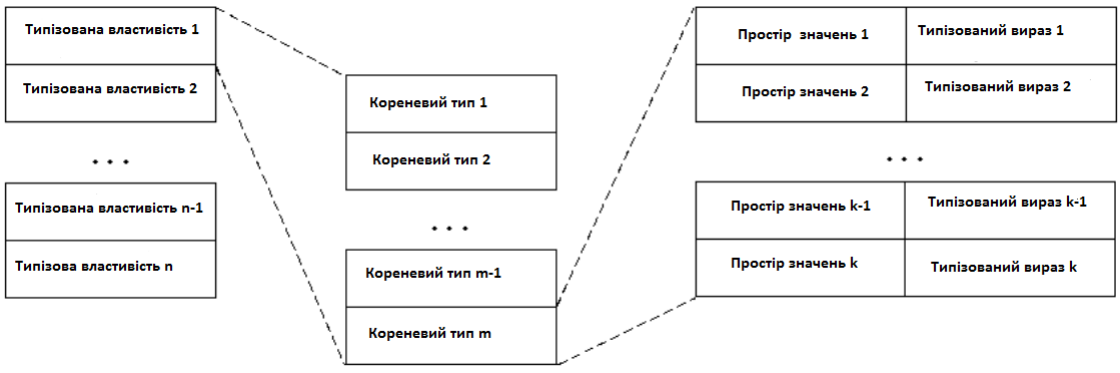
$$\frac{A \sqsubseteq B}{A \sqsubseteq \exists r^+} \quad B \sqsubseteq \exists r_+ \in O$$

Рисунок 4.2 Правила обробки типізованих виразів

де O – онтологія, A, B ∈ NC, r ∈ NR, NR - нескінченна безліч атомарних ролей, NC - безліч атомарних концептів, вираз r + → D ⊥ означає, що з обмеження слід порожня множина, в той час як вираз r + → D r- означає, що обмеження праворуч від задовольняє обмеження зліва від нього. Знаки + і - вказують на те, що вираз зустрічається праворуч і ліворуч від ⊑ відповідно[27].

Наступним завданням буа індексація типізовані вираження вихідної онтології таким чином, щоб максимально швидко робити пошук відповідних виразів ∃r ⊑ B ∈ O і дозвіл операції r + → D r-.

Було вирішено зберігати типізовані вираження наступним чином (Рисуно 4.3). Для кожного типізованого властивості в онтології зберігається контейнер кореневих типів.



Рисуно 4.3 – Реєстр типізованих виразів

Кореневим будемо називати такий тип даних, який є найбільш загальним типом своєї категорії. Кореневої тип стоїть на вершині ієрархії родинних йому типів даних і розділяє з ними спільний простір значень.

Для кожного кореневого типу створюється ще один контейнер, що містить пари значень <Простір значень, типізований вираз>. У нашому випадку типізований вираз це r ⊑ B.

Після індексації, коли ELK входить у фазу насичення, для кожного виразу r + проводиться пошук вираження r- яке його поглинає[27]..

4.3 Приклад роботи

Для перевірки працездатності модифікацій використовувались існуючі онтології, що відповідають профілю OWL 2 EL.

Перша, повна онтологія, містила 1087 124 аксіом, 65 класів, 33 об'єктних властивості, 109 типізованих властивостей і 131637 примірники. Другий, усічений варіант онтології відрізнявся тим, що містив 230 670 аксіом і 36191 екземпляр.

У Таблиця 4.1наведені результати тестування.

Таблиця 4.1 – Результати тестування

Логічний аналізатор	t, мс (усічена онтологія)	t, мс (повна онтологія)
ELK	8368	66912
ELK^{I}	6278	15350

Проаналізувавши продуктивність ELK, стає зрозумілим що можливо значно прискорити його роботу за рахунок деякої втрати повноти логічного аналізу строкових літералів. Так як введення типізації в онтології, що містить велику кількість типізованих властивостей з типом xsd: string, скористались іншим, більш простим алгоритмом їх обробки який не уточнює фактичний тип строкових літералів. Конкретно для тестової онтології це не вплинуло на коректність отриманих результатів, але скоротило час класифікації на 25% і 77% для усіченої та повної онтології відповідно.

4.4 Висновки

Представлені зміни до логічного аналізатору ELK відкривають дорогу до його широкого використання в тих випадках, коли необхідно швидко обробляти великі масиви знань представлені у вигляді OWL онтологій. Представлена тут підтримка типізованих властивостей і конкретних типів даних дозволяє застосовувати логічний аналіз над фактичними і конкретними даними такими, наприклад, як записи баз даних, зовнішні інформаційні системи, служби каталогів LDAP, документи RDF і багато інших подібних джерела.

Незважаючи на деякі обмеження, накладені профілем OWL 2 EL, засновані на ньому онтології мають достатню практичну експресивність і завдяки високопродуктивним логічно аналізаторам подібних ELK, можуть бути ефективно використані в різноманітних прикладних областях.

5 ОХОРОНА ПРАЦІ

5.1 Вступ

У даному розділі проаналізовані умови праці на робочому місці, на якому розроблювався програмний продукт на основі санітарних норм України.

У ДР розроблено програмний продукт для роботи з блоком відеокодеку H.264. Даний програмний продукт передбачено застосовувати, як частину відеокодеку, для подальшого вдосконалення. Використовуватись ПП буде в офісах розробників програмного забезпечення.

5.2 Опис приміщення

Приміщення, в якому розроблявся програмний продукт розташоване на третьому поверсі п’ятиповерхового будинку. Приміщення має однобічне природне освітлення та загальне штучне освітлення. Вікно обладнане завісками, орієнтовано на північ, площа засклення 40%. Стіни обклеєні світлими шпалерами, стеля біла, підлога вкрита світлим паркетом з природнього матеріалу.

Приміщення загальною площею 30 м^2, ширина якого складає 5 м, довжина – 6 м, висота стелі – 3 м (Рисунок 6.1). В приміщенні працює 4 людини.

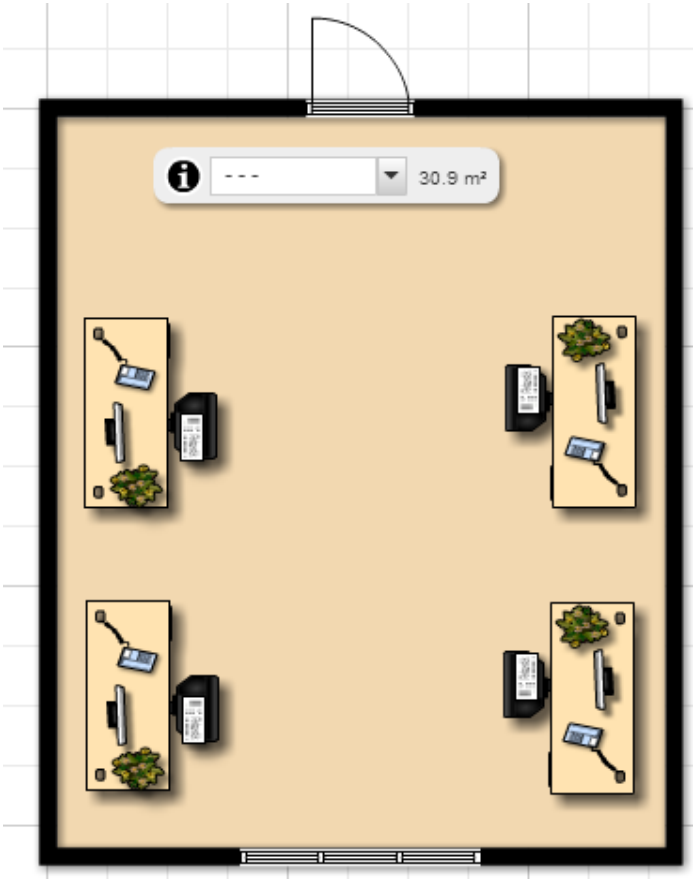


Рисунок 5.1 – План робочого приміщення.

Виходячи з того, що в приміщенні працює 4 людини, отримаємо наступні дані, наведені в таблиці 5.1:

Таблиця 5.1 – Фактичні та нормативні значення площі та об’єму приміщення з розрахунку на одне робоче місце

Параметр приміщення	Нормативний	Фактичний
Площа, $м^2$	6 і більше	7,5
Об’єм, $м^3$	20 і більше	22,5

Аналізуючи дані, наведені в таблиці 1, зробимо висновок, що розміри приміщення відповідають існуючим вимогам [27].

5.3. Напруженість праці користувача ПЕОМ

Робота користувача ПЕОМ за показниками напруженості трудового процесу [29] при використанні програмного продукту розробленого у ДР характеризується:

за показником інтелектуального навантаження – 1 ступінь напруження (класи 1–4, в середньому клас 3: частково творча і частково здійснюється за алгоритмами, вимагає оцінок взаємопов'язаних параметрів, по часу – робота за індивідуальним планом),

за сенсорним навантаженням – 2 ступінь напруження (клас 4: робота на ПЕОМ більше 4 годин на добу),

за емоційним навантаженням — оптимальна (клас 1: відповідальність за окремі елементи роботи і відсутність ризиків для життя і безпеки),

за монотонністю навантажень – 2 ступінь напруження (клас 4: робота на ПЕОМ характеризується постійним активним виконанням різних операцій),

за режимом праці – допустима (клас 2: близько 8 годин на добу).

Отже, характер робіт в середньому відповідає класу 3 (напружена робота 1 ступеня), тому необхідно періодично та в достатній кількості вживати заходів щодо збереження здоров'я: розробити та виконувати комплекс вправ, забезпечити здорове харчування під час періодів відпочинку.

5.3.1. Повітряне середовище

Для постійних робочих місць, якими є робочі місця операторів ПК, встановлені оптимальні параметри мікроклімату (згідно з [30]), а при неможливості їх дотримання використовують допустимі параметри. Небезпечним фактором є мікроклімат приміщення, хоча приміщення знаходиться не з сонячної сторони, влітку у приміщенні дуже жарко, а взимку холодно.

Через те, що приміщення належить до I(а) категорії (виконуються легкі фізичні роботи),необхідно дотримуватись наступних вимог(Таблиця 5.2) :

Таблиця 5.2 – Параметри мікроклімату для приміщень з ПК

Період року	Параметр мікроклімату	Норма	Реальна величина
Холодний	Температура повітря в приміщенні Відносна вологість Швидкість руху повітря	22...24 °С 40... 60% 0,1 м/с	24 °С 55% 0,1 м/с
Теплий	Температура повітря в приміщенні Відносна вологість Швидкість руху повітря	23...25 °С 40...60% 0,1 м/с	25 °С 55% 0,1 м/с

Для створення й автоматичної підтримки в приміщенні оптимальних значень температури, вологості, чистоти і швидкості руху повітря незалежно від зовнішніх умов, у холодний час року використовується водяне опалення, у теплий час року застосовується кондиціонування повітря. Для охолодження приміщення використовується кондиціонер ORION MCH–09 з потужністю 2,75 кВт. та рівнем шуму 39Дб. Вибір кондиціонеру проводився по потужності з урахуванням необхідної продуктивності по об’єму повітря, що подається до кімнати. Під час розрахунку потужності охолодження враховуються всі

можливі теплопотоки та тепловиділення у приміщенні при найбільш несприятливих умовах, основними параметрами при розрахунку мають бути розмір вікна та кількість комп'ютерів у приміщенні.

Таблиця 5.3 – Норми подачі свіжого повітря в приміщення з ПК

Характеристика приміщення	Об'ємна витрата свіжого повітря, що подається в приміщення, на одну людину в годину
Об'єм до 20 $м^3$ на людину	Не менше 30
20... 40 $м^3$ на людину	Не менше 20
більше 40 $м^3$ на людину	Може бути використана природна вентиляція

5.3.2. Освітлення робочого місця

Робота, що виконується з використанням обчислювальної техніки, має багато недоліків. Серед факторів, що пливають на організм людини в процесі роботи, світло займає одне з найважливіших місць. Оскільки, відомо, що майже 50% всієї інформації про довкілля людина одержує через органи зору. Під час здійснення будь–якої трудової діяльності втомлюваність очей, в основному. Залежить від напруженості процесів, що супроводжують зорове сприйняття інформації. До таких процесів відносяться адаптація, акомодація та конвергенція.

Адаптація – пристосування ока до зміни умов освітлення (рівня освітленості).

Акомодація – пристосування ока до зрозумілого бачення предметів, що знаходяться від нього на неоднаковій відстані за рахунок зміни кривизни кришталика.

Конвергенція – здатність ока при розгляданні близьких предметів займати положення, при якому зорові осі обох очей перетинаються на предметі.

В приміщенні використовується природне і штучне освітлення. Природне: вікно площею *3,6 м*^{2}.

Природне освітлення здійснюється через вікно(а) та являється боковим освітленням.

Розрахунок для кожного вікна:

Знайдемо коефіцієнт природної освітленості за формулою:

$$i = \frac{S_{Bik}}{S_n} = \frac{3.6}{30.0} = 0.12$$

де $S_{вік}$ – загальна площа вікон, м^2; $S_п$ – площа підлоги, м^2

Отримане значення i=0.12 менше ніж встановлено нормами [27], тобто природного освітлення не вистачає для нормальної роботи (0.12 < 0.15). Тому потрібно використовувати ще й штучне освітлення.

У кімнаті встановлено 4 світильники ламп денного світла типу F18/33 (18Вт/Т8) із світловим потоком лампи = 1150 лм.

Кожен з столів оснащений настільною лампою DE LUX ДЕКОР TF–05

Освітлення задовольняє нормам.

6.3.3 Випромінювання

Усі монітори ПК вироблені на основі рідко – кристалічної матриці, підсвітка якої здійснюється неоновою лампою, тому у приміщенні відсутні інфрачервоні, ультрафіолетові та електромагнітні випромінювання.

5.3.4. Шум та вібрація

Шум один з основних факторів, що негативно впливає на людей під час роботи, відпочинку. Збільшення потужності устаткування, насиченість виробництва високошвидкісними механізмами, різке збільшення транспортного потоку приводить до збільшення рівня шуму як у побуті так і на виробництві.

Шкідливий вплив шуму на організм людини досить різноманітний. Реакція і сприйняття шуму людиною залежить від багатьох факторів: рівня інтенсивності, частоти (спектрального складу), тривалості дії, тимчасових параметрів звукових сигналів, стану організму.

Працюючі в умовах тривалого шумового впливу випробують зниження пам'яті, запаморочення, підвищену стомлюваність, дратівливість і ін. До об'єктивних симптомів шумової хвороби відносяться: зниження слухової чутливості, зміна функцій травлення, що виражається в порушенні кислотно–лужного балансу у шлунку, серцево–судинній недостатності, нейроендокринному розладі. Відмічаються порушення зорового та вестибулярного апарату.

Основним джерелом шуму в приміщенні є кондиціонер, з рівнем шуму 39дБА. Отже, фактичний обмірюваний рівень шуму склав 39дБА, що (згідно з [31]) задовольняє вимогам (до 50 дБА для програмістів ПЕОМ).

Медичними дослідженнями встановлено, що вібрація є подразником периферичних нервових закінчень, розташованих на ділянках тіла людини, що сприймають зовнішні коливання. Навіть невеликі рівні вібрації, що можуть

спостерігатися в офісних приміщеннях працівників аналізованої галузі негативно впливають на рівень працездатності, можливість концентрації, на нервову систему.

В приміщенні відсутні джерела вібрації. Негативний вплив на працездатність і роботу людини в приміщенні немає.

5.4. Електробезпека

У приміщенні використовується однофазна мережа електропостачання (фазовий, нульовий робочий і нульовий захисний дріт з напругою 220 В і частотою 50 Гц). Проводка проведена прихованим способом, вимикачі і розетки захищені пластмасовими корпусами, світильники розташовані на висоті 3 м.

Розетки типу "Європа" із заземляючими контактами і мають маркування по напрузі. Заземляючі контакти розеток мають з'єднання із заземляючим контуром приміщення. Пристрої, електропроводи і кабелі мають апаратуру захисту від струмів короткого замикання і інших аварійних режимів. Нульовий захисний провідник використовується для заземлення (занулення) електроприймачів. Площа перерізу нульового провідника і нульового захисного провідника є більше площі перерізу фазового дроту. Пристрої підключаються до електромережі тільки за допомогою справних штепсельних з'єднань і електророзеток заводського виготовлення, які мають спеціальні контакти для підключення нульового захисного дроту. Обслуговуючий персонал ознайомлений з правилами техніки безпеки при роботі з обчислювальною технікою.

Отже, стан електробезпеки відповідає вимогам [28].

5.5. Ергономіка робочого місця

Висота над рівнем підлоги робочої поверхні, на якій працює користувач, складає 750 мм. Розміри поверхні столу 1400 х 1100 мм^2. Під столом є простір для ніг з розмірами по глибині 600 мм. Робочий стілець користувача оснащений підйомно–поворотним механізмом (висота регулюється в межах 400 – 550 мм), регулюванням нахилу. Під столом є підставка для ніг.

Екран ВДТ розташований на відстані 600 мм від очей, клавіатура – на відстані 200 мм від краю стола. Кут нахилу екрана регулюється і в середньому становить 30 град.

Робоче місце не відповідає вимогам [27] .

5.6 Психофізичне розвантаження

Працююча за комп'ютером людина тривалий час повинна зберігати відносно стале положення, що негативно позначається на хребті й циркуляції крові у всьому організмі (застій крові). Особливо сильно застій крові виражений на рівні органів малого таза й кінцівок. При тривалих порушеннях циркуляції крові порушується живлення тканин і ушкоджуються стінки судин, що у свою чергу призводить до їхнього розширення.

Тривала робота на клавіатурі приводить до перенапруги суглобів кисті й м'язів передпліччя.

Робота за комп'ютером припускає переробку великого масиву інформації й постійну концентрацію уваги, тому при тривалій роботі за комп'ютером нерідко розвивається розумова втома й порушення уваги. Тому, тривала робота за комп'ютером, часто є причиною хронічного стресу.

Усе частіше з'являються повідомлення про виникнення комп'ютерної залежності. Дійсно, тривала робота за комп'ютером, робота в Інтернеті можуть викликати подібні психічні розлади.

Таким чином, людина, яка тривалий час працює за комп'ютером зазнає реального ризику серцево–судинних захворювань, різних захворювань очей, рухового апарата, органів шлунково–кишкового тракту, психічних розладів.

Для зменшення впливу шкідливих факторів роботи з комп’ютером слід проводити сеанси розвантаження, що основані на методах саморегуляції і виконанні нескладних фізичних вправ із словесним самонавіюванням. Головну увагу слід приділити до навичок розслаблення м’язів (реласкації). Для таких сеансів має бути відведена спеціальна кімната, з відповідним інтер’єром. В кімнаті має бути передбачено місце для не складних фізичних вправ та нетривалого, проте дуже ефективного відпочинку. Також слід звернути увагу на кольорове оформлення кімнати, воно має бути спрямованим на розслаблення працівника.

Відпочинок після роботи з комп'ютером має відбуватись в регламентовану перерву, індивідуальну перерву, або в кінці робочого дня.

Також, періодичні медичні огляди мають проводитися раз на два роки комісією в складі терапевта, невропатолога та офтальмолога.

Нижче наведено варіанти вправ для очей, а також вправ для поліпшення кровообігу [33].

1.Вихідне положення (В.п.) – сидячи, руки на колінах. Закрити очі, сильно напруживши очні м'язи, на рахунок "раз–шість", потім відкрити очі, подивитись вгору на рахунок "сім–вісім", подивитись на рахунок "дев'ять–десять". Повторити 5 разів.

2. В.п. те саме. Робити колові рухи очима, фіксуючи погляд в таких положеннях: додолу–вліво–вгору–вправо–додолу. Повторити 5 разів. Потім те саме 5 разів у зворотному напрямі.

3. В.п. те саме. Закрити очі на рахунок "раз–два", відкрити очі і подивитися на кінчик носа на рахунок "три–чотири". Повторити 5 разів.

1. В.п. – основна стійка (о.с.). На рахунок "раз" – руки за голову, лікті розвести, голову нахилити назад. На рахунок "два" – лікті вперед. На рахунок "три–чотири" – руки розслаблено опустити вниз, голову нахилити вперед. Повторити 4–6 разів у повільному темпі.

2. В.п. – стійка "ноги порізнь", пальці стиснуті в кулаки. На рахунок "раз" – різкий мах лівою рукою назад, правою – вгору назад. На рахунок "два" – різко змінити положення рук. Повторити 6–8 разів у середньому темпі.

3. В.п. – сидячи на стільці. На рахунок "раз–два" – плавно відвести голову назад, на рахунок "три–чотири" – голову нахилити вперед, плечі не піднімати. Повторити 4–6 разів у повільному темпі.

5.7. Висновки

Аналіз умов праці в розглянутому робочому приміщенні показав, що: умови праці з ПЕОМ відповідають вимогам [27] , оскільки мінімальна площа та об’єм із розрахунку на одну людину в приміщенні не менше нормативних значень.

Освітленість приміщення відповідає нормам.

Шум у приміщенні в межах норм.

Електробезпека відповідає вимогам [28].

Ергономіка робочого місця відповідає вимогам [27], що сприяє продуктивній роботі працівника.

З урахуванням показників напруженості трудового процесу (див. п.6.3.) рекомендовано проведення комплексів вправ для очей, опорно–рухового апарату, кровообігу.

ВИСНОВКИ

В ході роботи здійснена спроба дослідження онтологій. А конкретніше отримання логічних висновків над ними.

Складність роботи полягала у тому, що теоретична база хоч і активно розвивається, і є перспективною, проте не надто добре вивчена. Пошук матеріалу ускладнювався мовою джерел (мова переважно англійська).

В ході роботи було досліджено шлях розвитку мови онтологій OWL. Адже вона в даний час є основною, для створення онтологій.

Також в ході роботи були проаналізовані алгоритми, за домпомогою можна отримувати нову інформацію з онтологій. Цей процес доволі складний, і вимагає великих об'ємів роботи. Проте майбутні перспективи використання цих напрацювань в створенні семаничного вебу надихає.

В результаті роботи було досліджено модифікацію одного з існуючих алгоритмів прийняття рішення, а саме ELK-Reasoner(a).

В запропонованому варіанті алгоритма отримано задовільний гарний результат скорочення часу обробки онтології, за рахуновк використання типізації.

Подальшу розробку в напрямку модернізації існуючого алгоритмів прияняття рішення вважаю доцільною. Ідея маніпуляції з данними, їх більш чітка ієрахізація, разом з класифікацією, а також розробка правил відбору за типами може дати ще більш серйозний виграш у часі при роботі з великими онтологіями.

Рекомендую розвивати науково-дослідницьку роботу в напрямку роботи з онтологіями, оскільки в далекоглядні перспективі можливості створеного апарату прияняття рішень над структурованими даними є необмеженими. Шляхи реалізації такого потенціалу : починаючи від електронних бібліотек, до розумних перекладачів, закінчуючти штучним інтелектом.

ПЕРЕЛІК ПОСИЛАНЬ

1. Berners-Lee T., Hendler J., Lassila O. The Semantic Web // Scientific American 2001. — May. —P. 598–602.

2. Згуровський М.З., Петренко А.І. Е-наука на шляху до семантичного Грід. Частина 2 семантичний WEB- і семантичний грід // Системні дослідження та ін-формаційні технології. - 2010. - № 1. - С. 26-38.

3. Теленик С.Ф. Логічний підхід до інтеграції програмних застосувань підтримки міждисциплінарних наукових досліджень / С.Ф.Теленик, О.А.Амонс, К.В.Єфремов, В.Т.Лиско // Наукові вісті НТУУ «КПІ». – 2013. – № 5. – С. 53-72.

4. SPARQL Query Language for RDF [Електронний ресурс]. Режим доступу: http://www.w3.org/TR/rdf-sparql-query/. - Электронна публікація. – Дата доступу : 14.05.2015

5. J. Bao, D. Caragea, and V. Honavar. Modular ontologies - a formal investigation of semantics and expressivity. In R. Mi-zoguchi, Z. Shi, and F. Giunchiglia (Eds.): Asian Semantic Web Conference 2006, LNCS 4185, pages 616–631, 2006.

6. B. C. Grau, B. Parsia, and E. Sirin. Working with multiple ontologies on the semantic web. In International Semantic Web Conference, pages 634, 2004.

7. Gruber, T. (1995). "Toward Principles for the Design of Ontologies Used for Knowledge Sharing". International Journal of Human–Computer Studies 43 (5–6): 907–928. doi:10.1006/ijhc.1995.1081.

8. OWL Web Ontology Language guide. W3C working draft. W3 Consortium, 2003. URL: http://www.w3.org/TR/2003/WD–owl–guide–20030331/ (дата обращения: 23.05.2015).

9. Офіційний сайт : OWL 2 Web Ontology Language. – Режим доступу : http://www.w3.org/TR/owl2-overview/ - Электронна публікація. – Дата доступу : 14.05.2015.

10. Matuszek, Cynthia, M. Witbrock, R. Kahlert, J. Cabral, D. Schneider, P. Shah and D. Lenat. Searching for Common Sense: Populating Cyc from the Web. In Proceedings of the Twentieth National Conference on Artificial Intelligence, Pittsburgh, Pennsylvania, July 2005

11. Рутковская Д., Пилиньский М., Рутковский Л. Нейронные сети, генетические алгоритмы и нечеткие системы: Пер. с польск. И. Д. Рудинского. — М.: Горячая линия — Телеком, 2006. — 452 с.: ил.

12. Zadeh L. A. Fuzzy Sets, Information and Control, 1965, vol. 8, s. 338–353

13. Глибовець М.М. Семантичний пошук із поширенням активації// Наукові праці. Комп'ютерні техології Випуск 77. Том 90 - 2012 - С. 197-204

14. Тарануха В.Ю. Інтелектуальна обробка текстів: [навчальний посібник]/ В. Ю.Тарануха. – Київ: електронна публікація на сайті факультету, 2014. – 80 с.

15. Розпутний М.В. Засоби здійснення логічних висновків над онтологіями / М.В. Розпутний// Системний аналіз та інформаційні технології: материалы 17-й Міжнародної науково-технічної конференції SAIT 2015, Київ, 22-25 червня 2015 р. – 2015. – с.171

16. The Protégé Ontology Editor and Knowledge Acquisition System. – Режим доступу : http://protege.stanford.edu - Электронна публікація – Дата доступу : 14.05.2015.

17. KAON2 - infrastructure for managing ontologies. – Режим доступу : http://kaon2.semanticweb.org - Электронна публікація. – Дата доступу : 14.05.2015.

18. Derivo Semantic Systems. – Режим доступу : http://www.derivo.de/en/ - Електронна публікація – Дата доступу : 14.05.2015.

19. Home of Smarter Solutions - Cycorp. – Режим доступу : http://www.cyc.com - Електронна публікація – Дата доступу : 14.05.2015.

20. Болотова В.А., Инструментальные средства создания баз знаний на основе системы онтологий/ Інтернет–ресурс. − Режим доступу: http://masters.donntu.org/2010/fknt/bolotova/diss/ Дата доступу : 14.05.2015.

21. Бажанова А.И., Разработка онтологической модели для семантического поиска информации в электронной библиотеке/ Інтернет–ресурс. − Режим доступу: http://masters.donntu.org/2011/fknt/bazhanova/diss/index.htm Дата доступу : 14.05.2015.

22. База знань по геронтології, OntoEdit Інтернет–ресурс. − Режим доступу: http://gerontology–explorer.ru/5ba7e2cf–4d1e–498b–a1f5–9b6d15d6e67d.html Дата доступу : 14.05.2015.

23. База знань по геронтології, KADS22 Інтернет–ресурс. − Режим доступа: http://gerontology–explorer.ru/aa5ada45–6fd3–42c3–8af3–02247f604571.html Дата доступу : 14.05.2015.

24. Semantic tools, KADS22 Інтернет–ресурс. − Режим доступа: http://www.semantictools.ru/tools/12––sw.html Дата доступу : 14.05.2015.

25. Kazakov, Y., Krötzsch, M., Simančík, F.: Unchain my EL reasoner. In: Proc. 23rd Int. Workshop on Description Logics (DL'10). CEUR Workshop Proceedings, vol. 745. CEUR-WS.org (2011)

26. Kazakov, Y., Krötzsch, M., Simančík, F.: Concurrent classification of EL ontologies.n: Proc. 10th Int. SemanticWeb Conf. (ISWC'11). LNCS, vol. 7032. Springer (2011)

27. Поспішний О.С. Ефективний алгоритм логічного аналізу OWL 2 EL онтологій з типізованими виразами на базі логічного процесору ELK/О.С. Поспішний, С.Г. Стіренко//Том 1, № 20 (2012) : Адаптивні системи автоматичного управління, Київ, 22 травня 2012 р. - 2012 - с. 106-115

28. Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно–обчислювальних машин ДСанПіН 3.3.2.007–98 (затверджено Постановою Головного державного санітарного лікаря України від 10.12.1998 р. № 7).

29. Правила охорони праці під час експлуатації електронно–обчислювальних машин. НПАОП 0.00–1.28–10 (затверджено наказом Державного комітету України з промислової безпеки, охорони праці та гірничого нагляду від 26.03.2010р. № 65)

30. Гігієнічна класифікація праці за показниками шкідливості та небезпечності факторів виробничого середовища, важкості та напруженості трудового процесу (затверджено наказом МОЗ України від 27.12.2001р № 528)

31. Санітарні норми мікроклімату виробничих приміщень : ДСН 3.3.6.042–99. – [Чинний від 2000–01–01]. – К. : МОЗ України, 2000. – 42 с. – (Національні стандарти України).

32. Санітарні норми виробничого шуму, ультразвуку та інфразвуку : ДСН 3.3.6.037–99. – [Чинний від 2000–01–01]. – К. : МОЗ України, 2000. – 37 с. – (Національні стандарти України).

33. Лапін В.М. Безпека життєдіяльності людини. – Львів; К. – 1999. – 186 с.