Засоби здійснення логічних висновків над онтологіями

Виконав : Розпутний Михайло Васильович
Керівник : Булах Богдан Вікторович

Мета дипломної роботи

- дослідження, систематизація і порівняння 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.

- **Завдання**, які вирішуються в роботі для досягнення поставленої мети:

- 1. Провести дослідження існуючих методів і алгоритмів отримання логічних висновків над онтологіями.

- 2. Надати практичні рекомендації по проблемі отримання логічних висновків: удосконалення існуючих алгоритмів, розробці нових підходів.

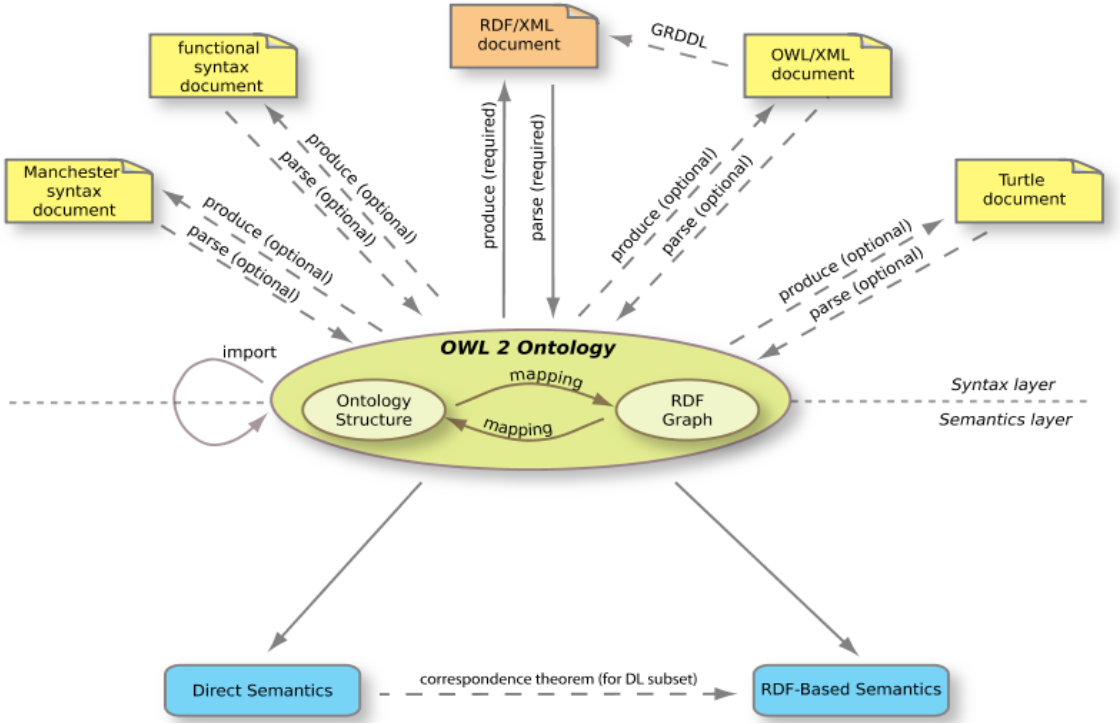
- 3. Експериментально дослідити ефективність алгоритму отримання логічних висновків над онтологіями.
- .

Актуальність роботи

- Отримання нової інформації з вже зібраної, структурованої, логічний аналіз такої інформації розкривають широкий спектр практичного застосування : створення електронних бібліотек, розумних перекладачів, систем прийняття рішень та ін. прикладів з області штучного інтелекту.
- Майбутні напрацювання можуть приносити комерційну вигоду, так наприклад такі відомі гравці на світовому ринку працюють в такому ж напрямку : IBM, Google, Yandex, Bing та інші.
- Пошук «смислу» в текстах (машиною!)
- Процес автоматизації отримання логічних висновків є ключовим елементом в архітектурі семантичних додатків, від його ефективності напряму залежить успішність семантичних систем у їх протистоянні з традиційними продуктами на базі СУБД, оскільки бази знань здатні постійно накопичувати чималі об'єми інформації, а алгоритми здійснення логічних висновків набагато витратніші в часі за алгоримтми вибірки в реляційних СУБД.

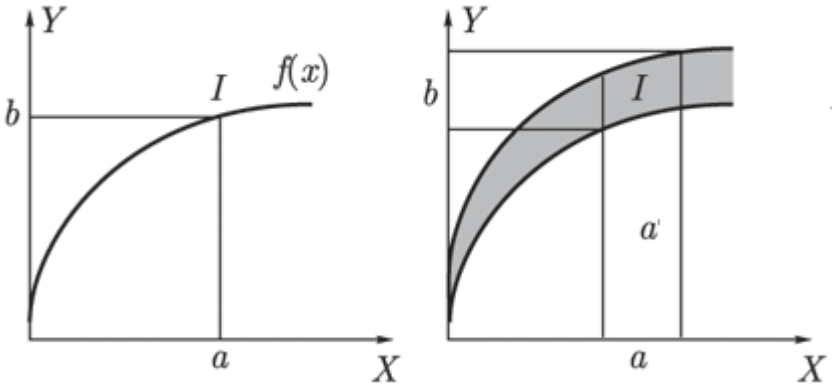
Кілька слів про OWL

- RDF/XML
- Description Logic (DL)
- OWL (OWL-Lite, OWL-DL, OWL-Full)



- OWL 2 DL
  - OWL 2 EL
  - OWL 2 QL
  - OWL 2 RL
- OWL 2 Full

Метод прямого нечіткого висновку



- 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.

Недолік: OWL не підтримує на належному рівні нечіткі висловлювання

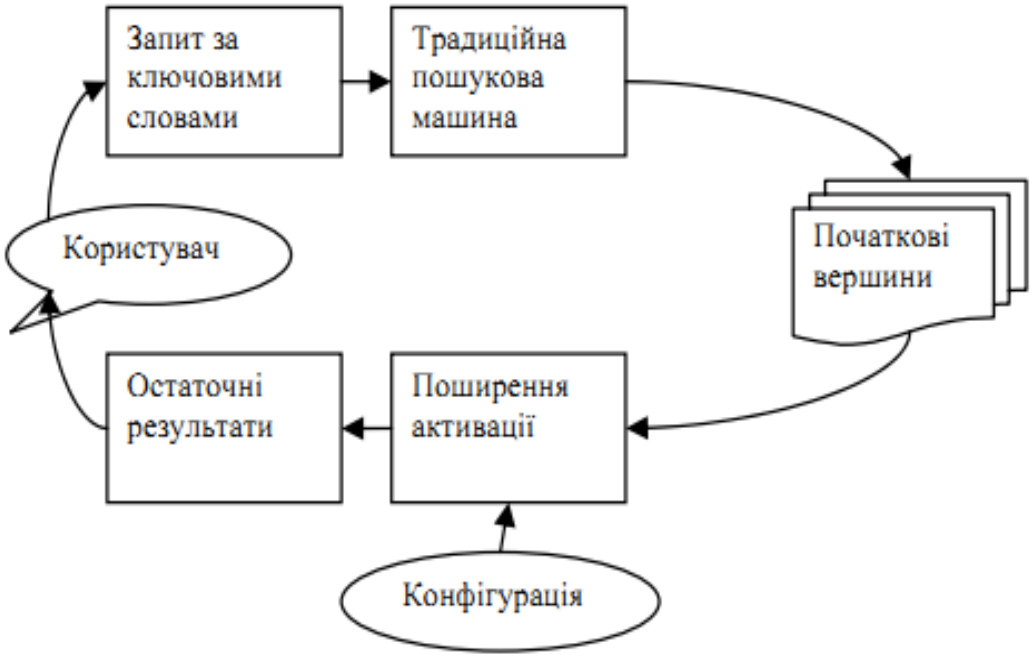
Алгоритм поширення активації (“spread activation”)

- 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”.
- Кожному екземпляру відношення приписується чисельна вага, оскільки використання семантичної інформації поряд із символьною повинне покращити пошук.
- Для автоматичного визначення ваги відношень пропонуються три метрики – кластерна, специфічності, комбінована.

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

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

- Алгоритм стартує із початкової множини екземплярів концепцій, здобутих традиційним пошуком.
- Цим вершинам присвоюються активаційні ваги, 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.



ELK-Reasoner

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∘
```

Модернізований алгоритм

- За основу для можливої модернізації доцільно обрати алгоритм ELK-Reasoner, оскільки він має практичну реалізацію, і найбільш пристосований для роботи з OWL онтологіями.
- За результатами тестування [24] 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.**
- Однак, типізовані вираження найчастіше є основною складовою баз знань, а їх частка в планованої Семантичної Павутині може скласти більше половини всіх виразів.
- Для реалізації підтримки типізованих виразів слід додати віподвідні нові правила логічного висновку, а також здійснити індексацію типізованих виразів вихідної онтології таким чином, щоб максимально швидко робити пошук відповідних виразів.

Порівняння результатів

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

Висновки. Напрямки подальшої роботи

- Було досліджено шлях розвитку мови онтологій OWL. Вона в даний час є основною для створення онтологій. В той же час, успішність семантичних систем напряму залежить від ефективності алгоритмів здійснення логічних висновків, оскільки бази знань здатні постійно накопичувати чималі об'єми інформації, а алгоритми здійснення логічних висновків набагато витратніші в часі за алгоримтми вибірки в реляційних СУБД.

- Були проаналізовані алгоритми, за домпомогою можна отримувати нову інформацію з онтологій, зокрема, spread activation та ELK reasoner. Також досліджувались шляхи їх удосконалення, зокрема підтримка типізованих відношень у ELK reasoner. Виявилось, що при цьому можливо досягнути також пришвидшення роботи алгоримту за рахунок правильної індексації типізованих виразів.

- Результат магістерської дисертації можна розглядати як основу для подальшої розробки систем, в яких будуть використовуватись алгоритм прийняття рішень над онтологіями.

Дякую за увагу!