Інформаційні технології

УДК 000.04

Слухай Яна Олександрівна
бакалавр комп'ютерних наук,
Національний технічний університет України
«Київський політехнічний інститут»
Слухай Яна Александровна
бакалавр компьютерных наук,
Национальный технический университет Украины
«Киевский политехнический институт»
Slukhai Y.
Bachelor of computer science
The National Technical University of Ukraine
«Kyiv Polytechnic Institute»

ПРИНЦИПИ ПОБУДОВИ ТА ПОРІВНЯЛЬНИЙ АНАЛІЗ СЕМАНТИЧНИХ РІЗОНЕРІВ
ПРИНЦИПЫ РАБОТЫ И СРАВНИТЕЛЬНЫЙ АНАЛИЗ СЕМАНТИЧЕСКИХ РИЗОНЕРОВ
PRINCIPLES OF CONSTRUCTION AND COMPARATIVE ANALYSIS OF SEMANTIC REASONERS

Анотація: Досліджено основні принципи побудови семантичних різонерів та проведено їх порівняльний аналіз.

Ключові слова: різонер, Structural reasoner, Pellet, Hermit, семантична таблиця, гіперетаблиця, пряме виведення, зворотне виведення.

Аннотация: Исследованы основные принципы построения семантических ризонеров и проведен их сравнительный анализ.

Ключевые слова: ризонер, Structural reasoner, Pellet, Hermit, семантическая таблица, гипертаблица, прямой вывод, обратный вывод.

Abstract: The basic principles of the semantic reasoners were investigated and a comparative analysis was made.

Keywords: reasoner, Structural reasoner, Pellet, Hermit, semantic tableau, hypertableau, forward chaining, backward chaining.

Семантичний різонер - частина програмного забезпечення, що здатна виводити логічні висновки з набору вибраних фактів та аксіом, а також надає можливість автоматичної підтримки таких завдань як: класифікація, налагодження, формування запитів. Правила виведення зазвичай задаються засобами мови онтологій, і часто засобами мов описової логіки [1, с. 115]. При проектуванні дуже великих онтологій (понад декілька тисяч класів) без різонера дуже важко обслуговувати великі, складні онтології і зберігати в коректному, належному вигляді.

Серед основних задач різонерів можна виділити такі: проводити класифікацію і виводити ієрархію класів, перевіряти консистентність онтології, визначати тип індивіда - належність до певного класу, визначати класи, що не перетинаються з заданим класом, визначати підкласи вибраних класів.

Перейдемо до алгоритмів, на базі яких найчастіше будується логіка різонерів.

Пряме виведення (forward chaining) є одним з двох основних методів логічного виведення. Алгоритм починається з формування ланцюжка з наявними даними і використовує правила виведення для вилучення більшої

кількості даних (від кінцевого користувача, наприклад), поки мета не буде досягнута. У системах з прямим виведенням за відомимми фактами відшукується факт, який з них випливає. Якщо такий факт вдається знайти, то він записується в базу фактів. Пряме виведення називають також виведенням, керованим даними або виведенням, керованим посиланнями правил.

Зворотний вивід (або зворотне міркування) - це метод отримання висновку, який працює в зворотному напрямку від мети. Він використовується в автоматичному доведенні теорем, машинному виведенні та інших напрямках штучного інтелекту [2, с. 146-147].

Цілям завжди відповідають висновки, у яких пізніше антецеденти розглядаються в якості нової мети. В кінцевому рахунку антецедентам повинні відповідати відомі факти (вони, як правило, визначаються як висновки, у яких завжди істинний антецедент). Таким чином, правилом виведення, яке використовується є modus ponens. Зворотний вивід часто використовується в експертних системах [3, с. 234-236].

Метод семантичних таблиць - це формальна роздільна процедура для формул логіки висловлювань і логіки предикатів, що дозволяє чисто синтаксичними засобами вирішувати семантичні проблеми формалізованих обчислень. Семантична таблиця - це дерево, вершинами якого є досліджувана формула і всі її підформули [4, с. 20-21]. Останні вершини кожної гілки - це обов'язково атоми, і такі вершини називаються листками. Семантична таблиця складного висловлювання К будується індуктивно, виходячи з семантичних таблиць підформул, що входять в висловлювання К. Кожній логічній зв'язці, що виконується у відповідній даній вершині-підформулі, зіставляється елементарна семантична таблиця у вигляді дерева, що розкриває логічну інтерпретацію зв'язки.

Алгоритм побудови гіпертаблиць базується на особливостях побудови аналітичних таблиць, але з використанням переваг головних ідей алгоритму гіпер-резолюцій [5, с. 106-107].

Оскільки метод є розширенням табличних методів, він забезпечує багату структуру для всього процесу виведення; важливі частини історії виведення зберігаються в таблиці і можуть бути використані для подальших логічних виведень [6, с. 7-8].

Було проаналізовано швидкодію трьох різонерів та якість виконання ними їх головних функцій.

Таблиця 1. Відповідність між різонерами та алгоритмами(складено автором на основі [7, с. 5])

Різонер	**Алгоритм**
Structural Reasoner	Прямого та зворотнього виводу
Pellet	Семантичних таблиць
Hermit	Гіпертаблиць

Було використано 6 тестових онтологій для оцінки роботи різонерів.

Таблиця 2. Тестові онтології(використані для тестування різонерів, наявні по відповідним посиланням у вільному доступі)

	IRI онтології
Тестова онтологія №1	http://ontology.dumontierlab.com/molecule-complex
Тестова онтологія №2	http://ontology.dumontierlab.com/unit-individuals
Тестова онтологія №3	http://ontology.dumontierlab.com/time-interval-primitive
Тестова онтологія №4	http://owl.man.ac.uk/2006/07/sssw/people.owl
Тестова онтологія №5	http://www.biopax.org/release/biopax-level1.owl

Тестова онтологія №6	http://ontology.dumontierlab.com/physics-complex-1.0.owl

Для всіх тестових онтологій найкоротший час виконання логічного виведення класової ієрархії, ієрархії властивостей даних та ієрархії властивостей об'єктів показав Pellet. Час отримання всіх всіх зв'язків «клас - підклас» онтології найкоротшим був для Structural різонера, але він для тестової онтології №4 виявив меншу кількість зв'язків, ніж Pellet та Hermit. Час отримання всіх зв'язків «клас – непересічний клас» найкоротшим також виявився для Structural, але кількість таких виявлених ним зв'язків у всіх шести випадках була меншою ніж у Pellet та Hermit. Серед останніх двох швидшим виявивився Pellet. Часу на те, щоб показати, що онтологія не має індивідуалів, найменше витратив Hermit, однак на вирахування їх кількості у екземплярних онтологіях менше часу витратив Structural, але він знову ж таки, зв'язків «індивідуал – клас» він нарахував менше, ніж Pellet чи Hermit. Перевірка онтології на консистентність найменше часу у п'яти випадках з шести зайняла у Hermit-а, лише один раз у Structural. Отже оптимальним варіантом для виведення логічних фактів на основі даного дослідження є Pellet – різонер, що реалізовує метод семантичних таблиць. Але перевірку на консистентність системи варто проводити, використовуючи Hermit.

Література:

1. Cornet R. Non-standard reasoning services for the debugging of description logic terminologies. / Cornet R., Schlobach S. - Gottlob, G., Walsh, T., eds.: IJCAI, Morgan Kaufmann , 2003. - 529 c.
2. Hayes-Roth F. - Building Expert Systems / Hayes-Roth F., Waterman D., Lenat D. - Addison-Wesley, 1983. – 254 c.

3. Kaczor K. Overview of Expert System Shells / Kaczor K., Szymon B., Grzegorz J. - Krakow, Poland: Institute of Automatics: AGH University of Science and Technology, Poland, 5 December 2010. – 334 с.
4. Бет Э. Математическая теория логического вывода / Бет Э. – М.: Наука, 1967. – 523 с.
5. Hähnle, R. Tableaux and Related Methods. Handbook of Automated Reasoning / Hähnle, R. – Volume I. Elsevier science, 2001. – 277 с.
6. Baumgartner P. Hyper Tableaux / Baumgartner P., Furbach U. - Niemela Universitat Koblenz Institut für Informatik Rheinau 1, 56075 Koblenz, Germany, 2013. – 18 с.
7. Gardiner Tom. Automated Benchmarking of Description Logic Reasoners / Gardiner Tom, Ian Horrocks, Dmitry Tsarkov. - Description Logics Workshop 2006. – 8 с.