УДК 004.832 **doi: 10.26906/SUNZ.2020.4.081**

С. І. Шаповалова, О. О. Мажара

Національний технічний університет України «КПІ імені Ігоря Сікорського», Київ, Україна

ВИЗНАЧЕННЯ ЕФЕКТИВНОСТІ МЕХАНІЗМІВ ЛОГІЧНОГО ВИВЕДЕННЯ

Анотація. Предметом дослідження в статті є алгоритми співставлення зі зразком, які використовуються в програмному інструментарії розробки систем, що базуються на правилах. Мета роботи - представлення особливостей вибору або генерації бенчмарків алгоритмів співставлення зі зразком в залежності від специфіки вирішуваних задач. В статті вирішуються наступні завдання: визначити проблематику тестових задач; провести аналіз концепцій базових алгоритмів співставлення зі зразком; провести аналіз існуючих бенчмарків алгоритмів співставлення зі зразком; виокремити основні підходи та методи формування бенчмарків. Методами, що аналізуються, є Rete, Treat та їх модифікації, а також методи та підходи до формування бенчмарків для аналізу продуктивності алгоритмів співставлення та систем, заснованих на правилах. Отримані наступні **результати:** для порівняльного аналізу представлено концепції базових алгоритмів співставлення зі зразком, що дозволило виокремити значимі характеристики, які впливають на продуктивність співставлення в термінах часу виконання та структури бази знань. Виокремлення характеристик відбувалося за двома підходами, які стосуються логічного виведення в системах, що базуються на правилах (rule-base) та для систем Semantic Web. Визначено базові тестові задачі, які використовуються в якості бенчмарків. Представлено основні бенчмарки алгоритмів співставлення зі зразком з відповідним визначенням специфіки області їх використання. **Висновки:** визначено проблеми аналізу ефективності механізмів логічного виведення для прикладних систем різного типу. Проведено аналіз та представлено концептуальні відмінності базових алгоритмів співставлення зі зразком, які впливають на вимоги до формування або вибору бенчмарків. На основі проведеного аналізу представлено основні характеристики бенчмарків для продукційних систем та систем Semantic Web. Визначено основні підходи та методи формування бенчмарків. Перспективним напрямком подальших досліджень вбачається вбачається створення нових тестових задач, які дозволять застосовувати представлення в термінах логіки першого порядку.

Ключові слова: продукційні системи, співставлення зі зразком, бенчмарки, Rete.

Вступ

Постановка проблеми. Виведення логічного заключення є основною задачею в автономних експертних системах. Крім цього така задача також часто постає як одна з багатьох в програмних комплексах, зокрема, в системах навігації, моніторингу та діагностики. Для реалізації цієї задачі використовуються спеціальні механізми виведення, найбільш затребувані з яких можна умовно поділити за такими основними напрямками: виведення за правило-орієнтованими базами знань (rule-based) та за базами Semantic Web.

В обох випадках найбільш ресурсоємним компонентом є співставлення зі зразком (СЗЗ). Цей етап виведення дозволяє обрати підмножину правил, які узгоджуються з поточною постановкою задачі. Для оцінювання продуктивності механізмів логічного виведення в цілому та співставлення зі зразком зокрема використовують спеціалізовані тестові задачі – бенчмарки. Однак більшість з них була запропонована для перших розробок продукційних систем (ПС, rule-based systems) та не відповідає сучасним вимогам до прикладних програмних комплексів. Сучасні рішення вимагають підходів до генерації правил та даних тестових задач, які відповідатимуть характеристикам баз знань, а також дозволятимуть визначити специфічні особливості в процесі співставлення для конкретних реалізацій алгоритмів. Актуальною є задача виокремлення таких характеристик та створення підходів до перевірки ефективності механізмів логічного виведення на основі правил.

Аналіз останніх досліджень і публікацій. Тестування ефективності систем зазвичай відбувається за двома базовими параметрами: швидкодія та використання ресурсів пам’яті. Часто ці характеристики потребують окремих реалізацій тестових баз знань. Крім того важливим аспектом є те, що реалізація тестових задач має проводитись або в рамках однієї концепції продукційних систем або з використанням декількох обчислювальних механізмів. Так, наприклад, в CLIPS представлено як продукційну так і об’єктно-орієнтовану модель. В даному програмному інструментарії об’єктно-орієнтована модель реалізовувалась іншою групою розробників як доповнення до класичного представлення і опису конструкцій фактів. В залежності від обраного формату представлення робочої пам’яті, можуть змінюватися показники навіть в рамках однієї версії продукту.

Найпоширенішими бенчмарками на сьогодні залишаються реалізації класичних задач штучного інтелекту, такі як Waltz та Manners. Однак, іноді разом с системою поширюються тести та бенчмарки, які використовувалися в процесі її розробки. Це можуть бути спрощені задачі, орієнтовані на перевірку специфічної властивості алгоритму або регресії між різними версіями системи [1]. В той же час важливо забезпечити користувачів можливістю крос-продуктного порівняння.

Waltz та Manners були створені для тестування Treat алгоритму СЗЗ [2]. Тест Manners – це задача розміщення гостей на вечері. Правила полягають у тому, що стать має чергуватися (чоловік, жінка, чоловік, жінка тощо), а гості, які сидять поруч, ма-

ють спільне хобі. Зміна кількості гостей дозволяє перевірити ефект від ускладнення проблеми. Тест Waltz знаходить співвідношення ліній двовимірних зображень з ребрами тривимірної сцени.

Дослідниками неодноразово наголошувалось на їх недостатності для визначення ефективності роботи механізмів виведення [1]. Однак досі більшість порівняльних аналізів використовують саме ці бенчмарки. Крім того варто брати до уваги, що академічні дослідження алгоритмів відмінні від їх використання в комерційних продуктах [1]. Програмний інструментарій систем, що базуються на правилах, а також деякі засоби Semantic Web, використовують Rete алгоритм для СЗЗ. Порівняльний аналіз ефективності реалізації Rete алгоритму в декількох послідовних версіях CLIPS, навіть з використанням найпростіших бенчмарків, дозволив G. Riley виявити основні недоліки поточної реалізації і наступного року представити зміни в базовому алгоритмі співставлення середовища CLIPS [3].

Більш складною задачею є перевірка ефективності систем заснованих на правилах в цілому. Спеціалізовані засоби розробки – обгортки продукційних систем – надають користувачеві реалізацію механізму логічного виведення та задають синтаксис представлення продукцій. Для порівняльного аналізу механізмів виведення (benchmarking), заснованих на правилах розробляють фреймворки, які вирішують проблему міграції тестових баз знань в межах визначеного набору обгорток [4].

Для напрямку Semantic Web data важливими задачами є генерація обсягів даних, близьких до реальних, та порівняння систем, які базуються на різних технологіях представлення/обробки правил. Створюються спеціалізовані ресурси з наборами бенчмарків та засобами їх міграції на різні системи [5, 6] Комплексність таких тестових систем накладає обмеження на представлення умовної частини правил. Тому вони частіше використовуються для порівняння відносних показників швидкодії чи затрат пам'яті, а не для аналізу базових алгоритмів виведення.

Метою статті є представлення особливостей вибору та генерації бенчмарків алгоритмів співставлення зі зразком в залежності від специфіки вирішуваних задач**.**

Для досягнення мети було вирішено такі задачі:

1. Визначити проблематику тестових задач
2. Провести аналіз концепцій базових алгоритмів співставлення зі зразком
3. Провести аналіз існуючих бенчмарків алгоритмів співставлення зі зразком
4. Виокремити основні підходи та методи формування бенчмарків

Проблематика тестових задач

З точки зору коректності та повноти перевірки можна виокремити такі складності застосування тестових задач:

- відповідність специфіці проблемних областей, для яких створювався алгоритм (області не перетинаються, задача тестує лише незначну підмножину, задача повністю відповідає специфіці продукту);
- забезпечення імплементації аналогічної якості в рамках мультипродуктового порівняння;
- узгоджене використання специфічної оптимізації (наприклад, хешування)
- забезпечення неупередженої оцінки/вимірювання в рамках експерименту, особливо якщо системи не надають однакових засобів збору статистичних даних.

Необхідним вбачається створення тестових задач, які відповідають специфіці предметних областей застосування кінцевого продукту. Це зумовлюється наявністю специфічних характеристик баз знань та з точки зору підходів до формування правил та відмінностями в тенденціях до змін робочої пам'яті. Так, наприклад, Rete алгоритм було розроблено з розрахунком на те, що системи, в яких він застосовується, матимуть властивості структурної подібності та тимчасової надмірності. Структурна подібність передбачає, що правила бази знань міститимуть умовні елементи схожої структури та значень. ПС наділена тимчасовою надмірністю, якщо в процесі логічного виведення на кожному кроці змінюється лише незначна частка фактів робочої пам'яті.

Очевидно, що не всі задачі наділені подібними характеристиками, що призвело до створення великої кількості модифікацій базового алгоритму.

В той же час, наразі не запропоновано набір бенчмарків, які дозволять порівняти ефективність цих модифікацій для баз знань з різними властивостями.

Використання реальних систем в якості бенчмарків дозволяє оцінити роботу алгоритмів в оточенні наближеному до комерційних задач. В той же час вона породжує додаткові складності як в реалізації так і в інтерпретації результатів.

Вважається доречним подібне тестування в рамках переходу від однієї версії продукту до іншої з метою перевірки на регресію. Однак, в загальному випадку, використання прикладних задач в якості бенчмарків для перевірки реалізацій різних обгорток вбачається недоцільним завдяки складності реалізації. В той же час можливим є спрощення прикладних проблем до задач, для яких можлива генерація правил БЗ та міграція на інші обгортки в адекватний проміжок часу.

Базові алгоритми співставлення зі зразком

Оскільки СЗЗ є найбільш ресурсоємним етапом логічного виведення, саме на визначення продуктивності його реалізації спрямовано більшість бенчмарків.

Продукційні системи проводять логічне висновування циклічно. На кожному циклі з множини правил бази знань обирається підмножина правил (конфліктна множина), які узгоджуються з вмі-

стом робочої пам'яті. Вибір правила, яке застосовується на поточному кроці, визначається стратегією розв'язання конфлікту. Саме на етапі формування конфліктної множини застосовується співставлення зі зразком. На цьому етапі доводиться істинність умовної частини правил, представленої у вигляді логічної зв'язки зразків. Класичним підходом вирішення цієї задачі є використання інкрементних алгоритмів співставлення зі зразком (СЗЗ). Суть цих алгоритмів полягає в збереженні інформації про попередні цикли обробки з метою зменшення необхідних обчислень. Такий підхід зумовлює необхідність компромісу у співвідношенні швидкодії до витрат на пам'ять. При цьому не завжди додаткові витрати пам'яті призводять до збільшення швидкодії.

Реалізації методів інкерментної оцінки поділяються за об'єктом обчислення та підходами до збереження стану між циклами співставлення на алгоритми жадібної оцінки (Eager Evaluation), лінивої оцінки (Lazy Evaluation) та зв'язування простору (Binding Space) [7]. Найбільшого поширення в програмних засобах розробки ПС набули алгоритми нетерплячої оцінки. Базовим серед таких алгоритмів вважається Rete [8]. Обгортки продукційних систем зазвичай використовують його модифікації.

В академічній спільноті значну увагу приділяють Treat алгоритму СЗЗ як модифікації Rete, яка грунтується на спостереженні, що іноді повторне обчислення відбувається з більшою ефективністю ніж підтримка та оновлення стану, що зберігається [2]. Сучасні модифікації Rete алгоритму знаходять своє обґрунтування в балансі між збереженням стану та повторним обчисленням.

Інкрементні алгоритми співставлення зі зразком передбачають два етапи виконання.

На першому етапі – прекомпіляції – з бази знань будується мережа потоку даних – графове представлення умовних елементів продукцій. Зазвичай прекомпіляція виконується при завантаженні/модифікації бази знань. Однак існують винятки у вигляді ПС, які здатні до навчання. Такі системи лежать поза межами даного дослідження. Наступний етап логічного виведення – виконання, коли зміни фактів робочої пам'яті подаються на мережу потоку даних і узгоджуються в її вузлах.

На рис. 1 представлено мережі потоку даних для Rete та Treat алгоритму.

Мережа потоку даних складається з двох частин: α-мережі, яка відповідає за узгодження константних значень та мережі узгодження змінних – β-мережі. Дані, які зберігаються в вузлах мережі при проходженні, відповідно називаються α та β пам'яттю.

Rete та Treat алгоритми мають однаковий підхід до побудови та проходження α-мережі. Вузли мережі будуються на основі константних значень умовних елементів. Якщо декілька правил мають однакові умовні елементи вони будуть поділяти між собою одній й ті ж вузли α-мережі. При проходженні змін фактів робочої пам'яті (додавання чи видалення) шаблони в вузлах зіставляються з шаблонами фактів. У разі відповідності інформація про узгоджений факт зберігається в узлі, формуючи α-пам'ять.

Відмінність алгоритмів Rete та Treat полягає у підході до узгодження змінних в рамках правила. В Rete алгоритмі β-мережа узгодження змінних будується на етапі прекомпіляції. Кожен join вузол цієї мережі (рис.1) містить два входи - від змінних, які необхідно узгодити, або від попереднього узгодження та змінної.

Таким чином проходження по β-мережі визначає поетапне узгодження кожної змінної правила з попередньо узгодженими. Якщо в лівій частині правила представлено n змінних , то β-мережа буде містити $(n-1)$ вузол. Узгодження змінних в β вузлах зберігається та оновлюється на кожному циклі співставлення.

В Treat алгоритмі β-мережа узгодження змінних (на рисунку 1 представлена пунктиром) будується динамічно та лише для правил, для яких відбулося узгодження в α-мережі. Іноді її називають термінальною мережею, а її вузли відповідно термінальними При цьому немає обмеження на кількість входів у вузол. Результати співставлення зберігаються лише для вузлів, які визначають правила в поточному конфліктному наборі. В залежності від реалізації Treat, узгоджені змінні зберігаються в вузлах мережі або лише в представленні конфліктного набору. Дані з узгоджень, які не призвели до активації правил видаляються.

В результаті досягнення листових вузлів β-мережі відбувається активація правила, тобто приклад правида з узгодженими значеннями змінних додається до конфліктної множини.

Процес видалення фактів відбувається симетрично процесу додавання з тією лише різницею, що правило видаляється з конфліктного набору. Кожна зміна робочої пам'яті містить відповідну мітку додавання чи видалення.

Для теоретичного порівняння ефективності алгоритмів співставлення дослідниками було запропоновано формальне представлення узгодження [9, 10].

Системи, створені на основі продукційної моделі вважаються, адаптацією класичної логіки до штучного інтелекту та інженерії знань [11]. Тому задачі логічного виведення для таких систем доцільно формалізувати саме в термінах логіки.

Подібні формалізації дозволяють розширити розуміння факторів, які впливають на ефективність роботи алгоритму, дозволяють інтерпретувати існуючі оптимізації та виявити закономірності в використанні ресурсів для різних підходів.

Так, в роботі [10], на основі формалізації представлення Treat алгоритму запропоновано уточнення формули розрахунку затрат пам'яті, запропонованої в [12].

В той же час для експериментальної демонстрації результатів оптимізації найчастіше використовуються вищезгадані Waltz та Manners бенчмарки.

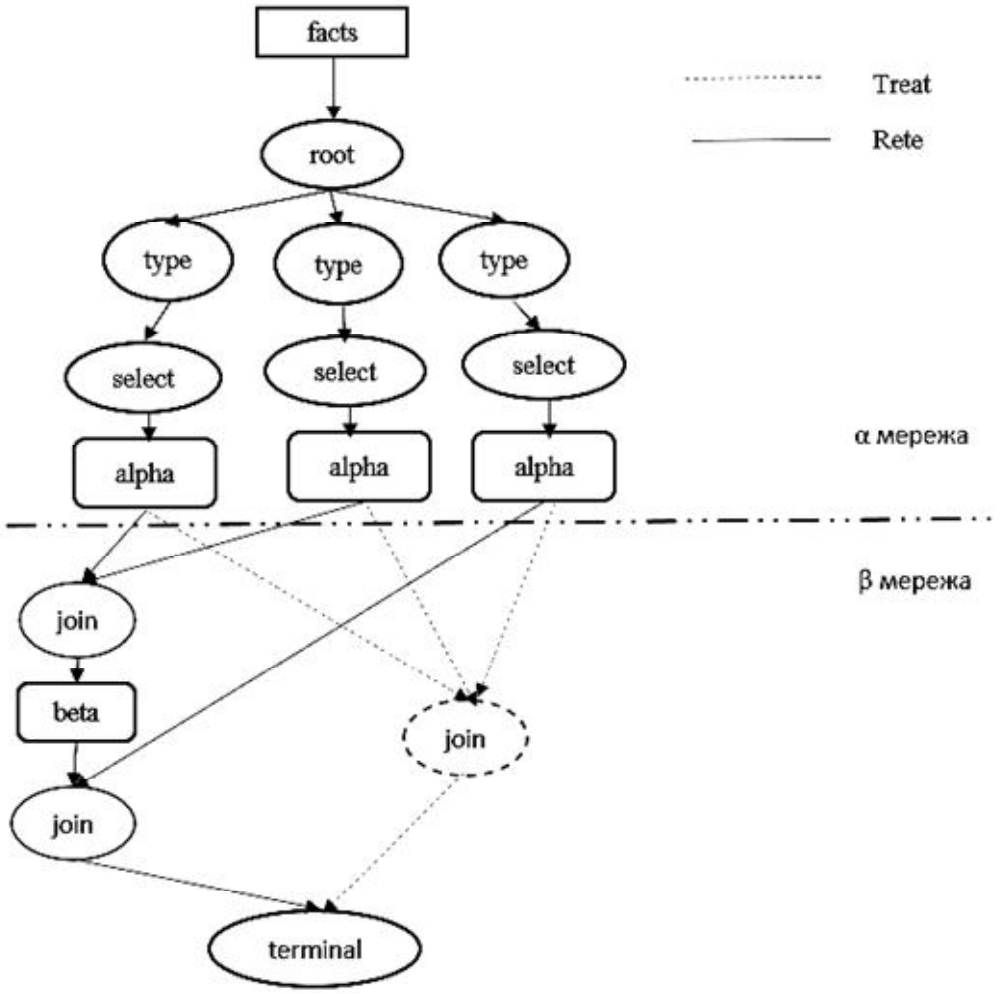

Рис. 1. Схема потоку даних за Rete та Treat алгоритмами

Бенчмарки алгоритмів співставлення зі зразком

Загальні вимоги до швидкодії та витрат пам'яті можна перевірити на прикладі класичних задач штучного інтелекту. Однак, при виборі алгоритму для специфічної задачі варто враховувати її характеристики у відповідності до домену. В роботі [7] на основі аналізу попередніх досліджень було виокремлено характеристики баз знань та властивостей часу виконання, які впливають на ефективність логічного виведення за базовими Rete та Treat алгоритмами.

Вважається доцільним формування тестових бенчмарків, які дозволять порівняти ефективність алгоритмів для заданих характеристик чи її комбінацій.

Розробка таких бенчмарків дозволить дослідити не лише кількісні показники на конкретних прикладах, але й відносні характеристики різних підходів реалізації .

До властивостей баз знань відносять кількість продукцій в системі, обмеження на представлення знань, структурні властивості антецеденту. До обмежень на представлення знань можна віднести, наприклад, наявність/відсутність аналогів кванторів існування або загальності в рамках замкнутості світу, представленого в системі. До структурних властивостей антецедентів належать: середня кількість умовних елементів, середня кількість змінних, процентна кількість негативних умовних елементів.

Доцільним вважається виокремлення бенчмарків, спрямованих на тестування продукційних систем з обмеженнями на представлення знань. Наприклад, в [13] представлено задачі, розроблені для тестування ПС, обмежених пропозиційною логікою.

В пропозиційній логіці умовні елементи лівої частини продукції є константними, тому зазвичай кожне правило виконується лише один раз. Відповідно системи такого типу зазвичай мають набагато більше правил, ніж засновані на логіці першого порядку.

Крім того, ці правила часто мають подібну структуру та елементи.

Важливо також перевірити можливість системи переходити між наборами правил подібного типу.

В даному випадку необхідно розробити бенчмарки двох типів – для тестування швидкодії СЗЗ на великих наборах схожих правил та для здатності механізмів виведення до побудови ланцюжка висновування на базі груп подібних правил.

Модифікації алгоритмів СЗЗ зазвичай спрямовані на оптимізацію або подолання/зменшення впливу якогось з недоліків алгоритму.

При цьому тестування відбувається на задачах, спеціально розроблених для демонстрації впливу цього недоліку. Таке обмеження не завжди дає змогу дослідити вплив модифікації на інші аспекти алгоритму.

Крім того, часто в роботах виконується порівняння з базовим алгоритмом, але не проводиться аналіз інших підходів з оптимізації.

Модифікації алгоритмів ставлять до бенчмарків вимоги перевірки специфічних властивостей, але в той же час, необхідність надання уявлення про зміну в загальних характеристиках. Забезпечення цих вимог вимагає розробки нових тестових задач.

Аналіз базових модифікацій СЗЗ показує, що поширеними підходами до підвищення ефективності є індексація та її окремий випадок – хешування, а також принципи ранньої відмови [14-17].

Прикладом першого підходу є β-node indexing - індексація фактів відповідно до шаблонів, з якими вони узгоджуються. Другого – Right/Left Unlinking – зупинка поширення змін по мережі потоку даних, якщо не відбулося узгодження в одному з вузлів даного правила.

Оптимізації з індексуванням розраховані на властивості часу виконання системи. Якщо система не наділена тимчасовою надмірністю, то вони можуть не давати бажаного приросту ефективності.

Right/Left Unlinking – дозволяє покращити ефективність роботи алгоритму у випадку великої кількості змінних за рахунок зменшення кількості перевірок в β мережі. Однак в дослідженні [14] показано, що для однієї з тестових задач Right Unlinking жодним чином не вплинув на результати виведення, в той час як Left Unlinking дозволив значно покращити ефективність виконання. Це зумовлювалось специфікою предметної області, де лише незначна частина практично незалежних правил могла активуватися на поточному кроці, але в процесі виведення таких активацій ставало все більше. Попередні дані про узгодження не видалялись з мережі потоку даних. При цьому в групах подібних правил перші декілька умовних елементів були однаковими, але наступні відмінними. Такі характеристики також впливали на ефективність використання спільних вузлів α пам'яті. Вищезгадані особливості призводили до лінійного зменшення ефективності Rete алгоритму з додаванням правил в базу знань, навіть з використанням оптимізації.

Даний приклад ілюструє важливість створення бенчмарків з різним набором характеристик в рамках тестування єдиної концепції.

Цікавим є підхід до перевірки ефективності продукційних систем, застосований в [14]. Для тестування було обрано 7 задач, вирішених в рамках попередніх досліджень з різних предметних областей, таких як, формування розкладу або тренування агенту прийняття рішень. Ці задачі мають вищу складність в порівнянні з класичними бенчмарками штучного інтелекту. Крім того, вони використовують різні підходи та техніки до вирішення проблем; були реалізовані різними розробниками та передбачають взаємодію з іншими компонентами системи. Кількість початкових правил, на основі яких велося подальше навчання, була в діапазоні від 48 для найпростішої задачі до 1953 для найскладнішої [14]. Однак дана ініціатива створення тестових задач на основі існуючих програмних рішень не знайшла свого продовження в подальших дослідженнях з порівняння алгоритмів.

Не зважаючи на те, що зазначена робота присвячена продукційним системам, що навчаються, можна виокремити важливі аспекти перевірки ефективності таких систем. Експерименти проводились в рамках єдиної оболонки реалізації Soar, щоб порівнювати програмні продукти, реалізовані однією мовою.

В роботі перевірялася поведінка систем зі збільшенням кількості вивчених правил. Навчання відбувалося за рахунок створення нових правил на основі результатів попереднього логічного виведення за навчальною вибіркою. Це безпосередньо впливало на кількість повторюваних умовних елементів в БЗ та формат правил. Так, системи, які намагаються на основі навчальної вибірки сформувати найбільш специфічні правила, закономірно генерують продукції з більшою кількістю умовних елементів. Кожна з систем мала вивчити не менше 100 000 правил. Такий підхід до розширення бази знань дозволяє вирішити проблему генерації тестових бенчмарків для перевірки масштабування продукційних систем. В той же час з використанням прикладних задач характеристики продукційних правил визначаються специфікою проблеми і неможливо наперед задати розподіл параметрів. Одним з підходів є генерація не лише бази знань на основі реальної навчальної вибірки, але й також генерація навчальної вибірки з наперед заданими характеристиками. В цьому випадку постає питання цілісності правил продукційної системи (верифікація поставленої задачі). З іншого боку, перевірка ефективності алгоритму співставлення не ставить вимог до семантичного наповнення бази знань.

В [14] для кожної з запропонованих тестових задач було створено генератори проблем – тестових вибірок. Автори стверджують, що розподіл параметрів моделей був наближеним до рівномірного. Деяке зміщення до певного типу правил могло призвести до незначної зміни в кількісних показниках, але відносна ефективність розглянутих в роботі алгоритмів лишається коректною.

Доцільною вважається адаптація даного підходу для перевірки ефективності алгоритмів співставлення зі зразком на здатність масштабування для БЗ значних розмірів. Виникає необхідність виокремлення прикладних задач, які можуть стати бенчмарками де-факто.

Додатковим викликом у цьому випадку є можливість тестування реалізацій різних оболонок. В

той же час, більшість сучасних оболонок продукційних систем зберігають CLIPS-подібний синтаксис.

Якщо уникати використання специфічних функцій мови, можливо забезпечити міграцію з однієї платформи на іншу з мінімальними затратами на адаптацію генерованого синтаксису.

Іншим підходом до створення тестових баз знань для різних обгорток є генерація проміжного формату представлення, так званої метамоделі, яка в подальшому транслюється в конкретні реалізації БЗ [4].

Це дозволяє створювати інструменти вимірювання затрат пам'яті та часу механізмів логічного виведення в залежності від таких характеристик як кількість правил, середня кількість умовних елементів в межах правила, типи атрибутів в умовних елементах.

Важливим в даному випадку є використання відкритого, незалежного формату для представлення правил, як наприклад XML. Запропонований в [4] підхід дозволяє порівнювати ефективність механізмів виведення як для традиційних обгорток, так і для Semantic Web. В той же час система накладає обмеження на представлення лівої (умовної) частини продукційного правила як кон'юнкції умовних елементів, які можуть містити лише позитивні терми. Додатковою складністю є реалізація трансляції з загального формату на мову оболонки. Правила транслюються по одному, тому неможливо врахувати такі важливі концепції мови як шаблони представлення фактів. Важливою перевагою рішення, запропонованого в [4], є те, що генератор гарантує існування ланцюга логічного виведення, який охоплює всі згенеровані схеми правил. Нажаль, з 2017 року система припинила оновлення.

Підходи до формування бенчмарків

Огляд підходів до оцінки ефективності механізмів логічного виведення на основі продукційної моделі представлення знань дозволяє сформувати наступні базові рішення.

Перевірка обгорток продукційних систем в цілому та алгоритмів співставлення зі зразком безпосередньо вимагає різних підходів до формування бенчмарків.

Найбільш багатообіцяючими з точки зору загального тестування є підходи на основі використання онтологій Semantic Web для генерації баз знань в єдиному загальноприйнятому форматі з подальшою трансляцією на мову обгортки. Важливою відкритою задачею є розширення засобів представлення продукцій в згенерованих базах даних та додавання трансляторів на нові мови.

Складним з точки зору реалізації, але перспективними для обох напрямків, як механізму дослідження особливостей СЗЗ, так і для створення бенчмарків, близьких до прикладних задач, є метод генерації бази знань як результату навчання продукційної системи.

В рамках тестування алгоритмів співставлення зі зразком цей метод дозволяє виявити специфічні випадки представлення баз знань в різних предметних областях. З точки зору загального тестування системи логічного висновування актуальною є задача міграції подібних бенчмарків на різні платформи зі збереженням початкової ефективності.

Обидва вищезгадані методи генерації баз знань доцільно використовувати для перевірки систем на масштабування.

Для базової перевірки ефективності алгоритму співставлення зі зразком необхідно генерувати ПС з заданими характеристиками часу виконання та структури бази знань. Класичні бенчмарки, такі як Waltz та Manners, дають уявлення про загальну ефективність алгоритму, однак є недостатніми для визначення передумов специфічної поведінки алгоритму в залежності від особливостей предметної області. В той же час, перевагою цих алгоритмів є можливість зміни навантаження на механізм виведення шляхом задання різних початкових умов.

Однак для цих задач пошук цільового висновку здійснюється або на графах простору станів (State Space Graphs), або за досить простими за структурою схемами.

Виокремивши характеристики баз знань та часу виконання, які вливають на ефективність роботи алгоритмів СЗЗ, можна зробити висновок, що їх аналіз потребує створення бенчмарків зі складною структурою лівої частини продукції. В той час, як обернену задачу спрощення формату розв'язують для продукційних моделей на основі пропозиційної логіки.

Наразі бракує тестових задач з можливістю регуляції кількості змінних, використання заперечень та кванторів.

Крім того, для наближення тестових задач до реальних, зокрема, необхідно, щоб поточний висновок приймався на основі заключень, отриманих на попередніх кроках різних напрямів пошуку. Тому структуру БЗ доцільно представляти як AND/OR-graph метаправил.

В [18] запропоновано тестову задачу за подібною схемою, яка дозволяє регулювати навантаження на механізм виведення шляхом задання різних варіантів початкових фактів робочої пам'яті.

Висновки

1. Визначено проблеми аналізу ефективності механізмів логічного виведення для прикладних систем різного типу.
2. Проведено аналіз та представлено концептуальні відмінності базових алгоритмів співставлення зі зразком, які впливають на вимоги до формування або вибору бенчмарків.
3. На основі проведеного аналізу представлено основні характеристики бенчмарків за для продукційних систем та систем Semantic Web.
4. Визначено основні підходи та методи формування бенчмарків.

Перспективним напрямком подальших досліджень вбачається створення нових тестових задач, які дозволять застосовувати представлення в термінах логіки першого порядку.

СПИСОК ЛІТЕРАТУРИ

1. Riley G. Rearchitecting CLIPS. Business Rules Knowledge Base. *October Rulefest 2008. a*vailable at: http://bizrules.info/conference/ORF2008DFW/GaryRiley_RearchitectingCLIPS_ORF2008.pdf (last accessed 20.10.20).
2. Miranker D. P. Treat: A better match algorithm for AI production systems. *ArtificialIntelligence: SixthNationalConference AAAI-87*. 1987. P.42—47.
3. Riley G. The CLIPS Implementation of the Rete Pattern Matching Algorithm. 2009. Corpus ID: 33742322
4. Bobek, S., Misiak, P. (2017), "Framework for Benchmarking Rule-Based Inference Engines", Artificial Intelligence and Soft Computing, ICAISC 2017, Lecture Notes in Computer Science, vol. 10246, Springer, Cham. P. 399-410. DOI: https://doi.org/10.1007/978-3-319-59060-8_36
5. S. Liang, P. Fodor, H. Wan, and M. Kifer, (2009), "OpenRuleBench: An analysis of the performance of rule engines", Proc. 18th International Conference on World Wide Web, P.601–610. DOI: https://doi.org/10.1145/1526709.1526790.
6. Rattanasawad, T., Buranarach, M., Saikaew, K.R., Supnithi, T. (2018), "A Comparative Study of Rule-Based Inference Engines for the Semantic Web", IEICE Transactions on Information and Systems, January 2018, P. 82-89. DOI: https://doi.org/10.1587/transinf.2017SWP0004
7. Шаповалова С.І., Мажара О.О. Вибір оптимального алгоритму співставлення зі зразком при проектуванні продукційної системи. *Східно-Європейський журнал передових технологій.* 2014, Вип.2/2(68). С. 43-49.
8. Forgy C. On the Efficient Implementation of Production System. *Ph.D. Dissertation. Carnegie Mellon University*. 1979. 210 p.
9. H. Cirstea C. Kirchner M. M., Moreau P. Production Systems and Rete Algorithm Formalisation. *Research Report: ILOG, INRIA,* 2004. URL: https://hal.inria.fr/inria-00280938/file/rete.formalisation.pdf.
10. Шаповалова С.І., Мажара О.О. Формалізація базових алгоритмів співставлення зі зразком в продукційних системах. *Східно-Європейський журнал передових технологій.* 2015. Вип.4/3(76). С. 22-27. DOI: 10.15587/1729-4061.2015.46571
11. Ligeza A. Logical Foundations for Rule-Based Systems. *Studies in Computational Intelligence (SCI).* 2006. 11.P.191-198.
12. Wright I. Marshall J. The execution kernel of RC++: RETE*, a faster RETE with TREAT as a special case. *Int. J. Intell. Games & Simulation 2 (1)*. 2003. P.36-48.URL: https://books.google.com.ua/books?id=R5EHCAAAQBAJ
13. Hicks R.C., Wright K. (2009), "Performance Testing of Propositional Logic Inference Engines", Journal of Computer Information Systems, Published online. Vol. 49, P. 122-126.
14. Doorenbos R. Production Matching for Large Learning Systems. *Ph.D. Dissertation. Carnegie Mellon University*. 1995. 208 p.
15. Hanson E.N., Hasan M.S. Gator: Anoptimized discrimination network for active database rule condition testing. *Tech. Rep. 93-036, CIS Department University of Florida.* 1993. 27 p.
16. Kim M., Lee K., Kim Y., T. Kim, Lee Y, Cho S., Lee C. RETE-ADH: An Improvement to RETE for CompositeContextAware Service. *International Journal of Distributed Sensor Networks.* 2014 —11p. 69. DOI: 10.1155/2014/507160
17. Bouaud J., Zweigenbaum P. A reconstruction of conceptual graphs on top of a production system. *Conceptual Structures: Theory and Implementation Ed. by H. Pfeiffer, T. Nagle.* 1993. Vol. 754. P.125–136.
18. Shapovalova S. Generation of test bases of rules for the analysis of productivity of logical inference engine. *Innovative Technologies and Scientific Solutions for Industries*. 2020. No. 3(13). P. 88–96. DOI: https://doi.org/10.30837/ITSSI.2020.13.088. .

Received (Надійшла) 27.08.2020
Accepted for publication (Прийнята до друку) 14.10.2020

Measuring efficiency of inference engines

S. Shapovalova, O. Mazhara

Abstract. The **subject** of research is the pattern matching algorithms that are used in software tools for developing rule-based systems. The goal of this work is to stipulate the characteristics for the selection and generation of benchmarks for pattern matching algorithms, depending on the specifics of the problems being solved. The following **tasks** have been fulfilled: determination of test task problematic, analysis of pattern matching algorithms, underlining the main approaches and methods of establishing benchmarks. The analyzed methods include Rete, Treat and their modifications, as well as approaches to the generation of benchmarks for analyzing the performance of pattern matching algorithms and rule-based systems. The following **results** were obtained. The concepts of basic pattern matching algorithms are presented to highlight significant characteristics that affect matching performance in terms of runtime and knowledge base structure. The definition of characteristics was done according to two approaches: the first approach relates to inference in systems based on rules (rule-base) and second one is used for systems of the Semantic Web. Basic test problems that are commonly used as benchmarks have been defined. The main benchmarks of the pattern matching algorithms are presented, with the corresponding definition of the specifics of their area of use. **Conclusions**. The problems of analyzing the efficiency of inference mechanisms for various types of applied systems are defined. The conceptual differences of the basic pattern matching algorithms are presented. Features which affect the requirements for the establishing or selection of benchmarks are identified. Based on the analysis, the main characteristics of benchmarks for productive systems and Semantic Web systems are provided. The main approaches and methods for the formation of benchmarks are defined. A promising direction for further research is the expansion of the proposed solution by the development of new solutions to enable representations of the generated knowledge bases in terms of first order logic.

Keywords: production systems, pattern matching, benchmarks, Rete.