УДК 004.825

О.О. Мажара

ПОРІВНЯННЯ TREAT ТА RETE АЛГОРИТМІВ СПІВСТАВЛЕННЯ ЗІ ЗРАЗКОМ

Анотація: У роботі розглянуто Rete та Treat алгоритми співставлення зі зразком для визначення їх переваг для специфічних прикладних задач за обраним критерієм оптимальності. Формалізовано схеми потоку даних для цих алгоритмів. Приведено порівняльну характеристику для обробки бета-мережі.

Ключові слова: співставлення зі зразком, продукційна система, .Rete алгоритм, Treat алгоритм

Вступ

Продукційні системи є одними з найбільш поширених систем для вирішення інтелектуальних задач. Під час їх проектування необхідно враховувати багато факторів, притаманних як моделі представлення знань так і розв'язуваній прикладній проблемі. Дослідження, спрямовані на виокремлення таких факторів та виявлення їхнього впливу на механізм логічного виведення є актуальною задачею.

В процесі виведення висновку в продукційних системах найбільш затратним за часом виконання та використанням ресурсів пам'яті є алгоритм співставлення зі зразком [1]. Поширеними алгоритмами співставлення, представленими в сучасних обгортках продукційних систем є Rete та Treat. Вони характеризуються високою швидкодією, низькими затратами ресурсів пам'яті. Проте в загальному випадку в спеціалізованих програмних засобах реалізації алгоритмів співставлення зі зразком не орієнтуються на специфіку розв'язуваної проблеми. Важливим для розробника продукційних систем є розуміння особливостей роботи механізму виведення з точки зору переваг та недоліків для поточної задачі. В дослідженні розглянуто процес співставлення з допомогою Rete та Treat. Виділено особливості алгоритмів, які впливають на ефективність виведення.

Метою даного дослідження є формування рекомендацій щодо випадків постановки прикладної задачі, в яких доцільно застосовувати Rete та Treat на основі різних критеріїв оптимальності.

Огляд останніх досліджень

Rete-алгоритм розроблений Charles Forgy в університеті Карнегі Меллона. Він став основою багатьох популярних обгорток продукційних систем, серед яких CLIPS, Jess, OPS та Soar [2, 3, 4, 5]. Існує ряд модифікацій даного алгоритму, запропонованих різними науковцями, з метою підвищення швидкодії, зменшення затрат пам'яті, ефективної інтеграції з базами знань, обробки продукцій з специфічним представленням антецеденту [6, 7, 8].

Treat алгоритм був розроблений Daniel P. Miranker в кінці 1980-их років [9]. Першочергово метою створення алгоритму була розробка паралельної версії Rete. Забезпечення підтримки стану, який використовувався в Rete алгоритмі, зумовлювало великі накладні витрати при паралельному обчисленні тому, що зміни станів повинні бути взаємопов'язані [9, 10]. Було запропоновано модифікації Rete алгоритму, спрямовані на уникнення взаємозв'язків в процесі співставлення та зменшення затрат пам'яті.

Науковцями проведено ряд досліджень з метою порівняння алгоритмів Rete та Treat за швидкодією та затратами пам'яті [11, 12]. В загальному випадку Treat показує кращі результати [11]. В той же час існують прикладні задачі, для яких переваги надає Rete алгоритм. Наразі не представлено узагальнених рекомендацій для вибору одного з цих алгоритмів з врахуванням специфіки прикладної задачі.

Співставлення зі зразком за допомогою Rete алгоритму

В Rete алгоритмі втілено наступні емпіричні спостереження, на основі яких була запропонована структура даних, яка лежить в його основі:

1. Продукційні системи характеризуються тимчасовою надмірністю. Зміни, які виникають в результаті запуску однієї з продукцій, зазвичай стосуються лише декількох фактів та впливають лише на декілька правил.
2. Продукційним базам знань притаманна структурна подібність. Один і той самий шаблон (умовний елемент) часто зустрічається в антецеденті більш ніж одного правила.

В Rete був запропонований підхід до збереження стану, який згодом став основою для розробки інших інкрементних алгоритмів співставлення за зразком. Основою інкрементних алгоритмів співставлення є збереження від циклу до циклу деякої інформації про стан системи (збереження стану).

При використанні Rete система будує спеціальний граф, вузлам якого відповідають частини умов правил. Шлях від кореня до листа утворює повну умову деякої продукції. Ліві частини правил компілюються в дискримінантну мережу потоку даних з взаємопов'язаними вузлами, які фільтрують інформацію (факти), коли вони поширюються через мережу. Для кожного умовного елементу правила генеруються відповідні необхідні тести та вузли, в яких виконуються один чи більше внутрішніх чи зовнішніх тестів. Rete зберігає результати внутрішніх (підтримка пам'яті) та зовнішніх (взаємозв'язки умов) тестів на протязі кожного з циклів зіставлення.

Схема потоку даних Rete складається з двох частин. На першому етапі відбуваються внутрішні тести для окремо взятих умовних елементів. З допомогою тесту еквівалентності визначається відповідність об'єкту класу, назви атрибутів та константні значення умовних елементів. Потім на основі даних цього тесту відбувається оцінка предикатів. При цьому зберігаються так звані маркери (token) – умовні елементи та факти, які з ними узгоджуються.

Внутрішні тести слугують в першу чергу як фільтри для виконання більш затратних зовнішніх тестів.

Зовнішній тест перевіряє взаємозв'язки між умовними елементами з антецеденту продукції. Він виконується, коли замість конкретних значень атрибутів використовуються змінні. Якщо змінна вперше зустрічається в умовному елементі лівої частини правила, їй може бути поставлене у відповідність будь-яке з фактичних значень. Якщо ж вона зустрічається декілька разів у різних умовних елементах, існування зв'язування між змінними має бути перевіреним. Якщо змінна з'являється в лівій частині правила лише один раз, перевірка не потрібна. Від циклу до циклу зберігаються відомості про часткові співставлення для правил та факти, якими вони зумовлені.

На рисунку 1 запропоновано схему формалізації етапів співставлення за рахунок Rete алгоритму. Описано вершини перевірки умов та дані, які зберігаються від циклу до циклу.

Бета-вершини, які виконують поєднання, що включають елементи з запереченнями, називаються вершинами з запереченням (not node). Вони забезпечують додаткову поведінку, створюючи вихідні дані, виключно у випадку, коли жоден з фактів не забезпечує повне зв'язування змінних [1].

Співставлення зі зразком за допомогою Treat алгоритму

Treat, подібно до Rete, використовує мережу для зіставлення, але він не включає підтримку взаємозв'язків між умовами. Відповідно, проміжні результати зовнішніх тестів не зберігаються, а перераховуються від циклу до циклу. Виключення підтримки взаємозв'язків між умовами значно зменшує вимоги до пам'яті в Treat алгоритмі, порівняно з Rete алгоритмом. На рисунку 2 запропоновано схему потоку даних для Treat алгоритму, яка дозволяє виділити відмінності в процесі функціонування алгоритмів.

На першому етапі роботи Treat, так само як і Rete, алгоритму відбувається перевірка зовнішніх тестів. Їх результати зберігаються від циклу до циклу. На другому етапі перевіряється зв'язування змінних. Так як інформація про часткові співставлення не зберігається від циклу до циклу, кожного разу перевіряються всі умови правила. В той же час, Treat використовує можливості збереження конфліктної множини для підвищення продуктивності за

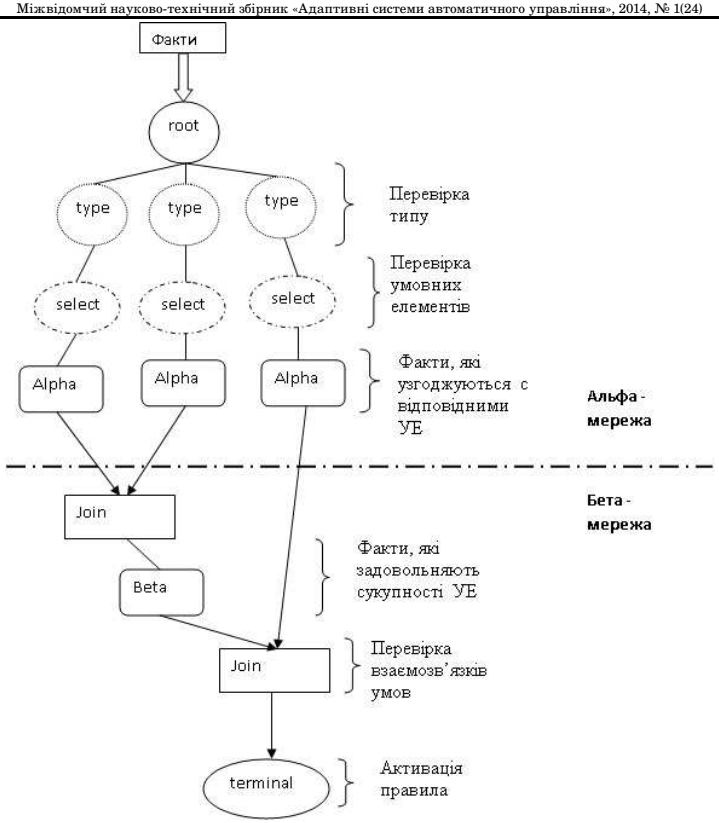

Рис. 1 – Схема потоку даних Rete

рахунок асиметричного видалення. Якщо видаляється факт, який був узгоджений з позитивним умовним елементом, переглядається конфліктна множина і відповідна продукція видаляється безпосередньо.

В таблиці 1 представлено порівняння бета-мережі для алгоритмів Treat та Rete.

Вхідна та вихідна інформація однакова для обох алгоритмів. Відмінність між мережами у тому, що Rete алгоритм формує вузли з двома вхідними дугами, забезпечуючи в кожному з них зв'язування виключно двох умовних елементів однієї продукції. При цьому в бета-пам'яті для кожного вузла зберігаються факти, які призвели до узгодження. Для Treat в бета-вузлі відбувається зв'язування для всього правила, тому кількість вхідних дуг обмежується лише особливостями представлення продукцій в базі знань.

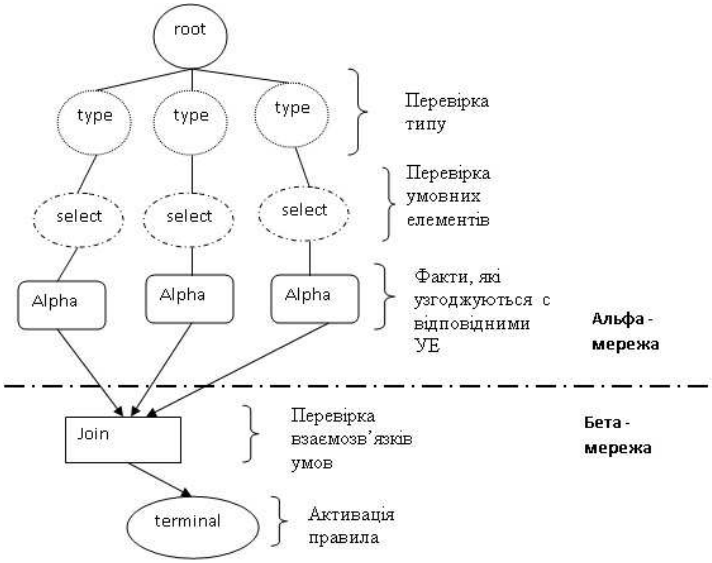

Рис. 2 – Схема потоку даних Treat

Вибір алгоритму співставлення зі зразком на основі характеристики прикладної задачі

Rete алгоритм був створений з розрахунку на те, що факти в робочій пам'яті змінюються повільно і впливають лише на незначну кількість продукцій в базі знань. В цьому випадку збереження від циклу до циклу часткових зіставлень для антецедентів на етапі бета-тестування надає значну перевагу у швидкодії, адже лише незначна кількість узгоджень потребує перевірки. В той же час, коли стан системи змінюється швидко, проміжні результати, які зберігає алгоритм, додаються й видаляються з високою частотою. Тому ефективність Rete значно знижується з додаванням фактів до робочої пам'яті. Таким чином обробка проміжних результатів внутрішніх тестів в бета-пам'яті має як переваги так і недоліки. Вона може зменшити затрати на співставлення тому, що результати попередніх циклів можуть бути використані. Проте бета-пам'ять надлишково зберігає схожу інформацію. Більш того, бета-пам'ять потенційно зберігає велику кількість результатів поєднання ,зумовлену їх невдалим впорядкуванням.

Treat алгоритм позбавлений цього недоліку, адже на етапі бета-тестування не зберігаються результати зв'язування змінних. В той же час це зумовлює необхідність повністю обчислювати зв'язування на кожному циклі співставлення, що може призвести до зниження швидкодії.

Таблиця 1

Порівняння бета-мережі для Rete та Treat

Характеристики	Rete	Treat
Задача	Визначити відповідність змінних в умовних елементах в межах однієї продукції	Визначити відповідність змінних в умовних елементах в межах однієї продукції
Вхідні дані	Результати альфа-тестів	Результати альфа-тестів
Кількість вхідних ребер для бета-вузла	Два	Рівна кількості умовних елементів продукції, які потребують зв'язування
Дані, що зберігаються для бета-тесту	Часткові співставлення	Конфліктна множина
Об'єкти перевірки	Умовні елементи, зв'язування для яких не відбулося на попередньому циклі	Всі умовні елементи, які потребують зв'язування
Вихідні дані	Конфліктна множина	Конфліктна множина

Одна з переваг Rete алгоритму це можливість поділу (sharing) частин мережі між правилами, що зменшує обчислювальну вартість та затрати пам'яті. Можливість поділу існує для вершин, в яких виконується один і той самий тест для різних правил. Це усуває надмірне тестування та збереження часткових результатів [1]. Проте затрати пам'яті все одно залишаються більшими ніж при використанні Treat алгоритму, де збереження зв'язування взагалі відсутнє.

Процес додавання і видалення фактів в Rete алгоритмі симетричний, адже той самий набір операцій виконується в обох випадках [1]. Це робить затратною операцію з видалення фактів, так як стан вершин графового представлення має бути змінений відповідно до видалених фактів, що передбачає повтор усіх розрахунків [9].

Treat алгоритм використовує збереження конфліктної множини для ефективного видалення фактів. Якщо факт, який видаляється, узгоджувався з однією з продукцій конфліктної множини, вона вилучається з набору. Це зменшує кількість перевірок в процесі співставлення та підвищує швидкодію в порівнянні з Rete.

Додавання і видалення елементів робочої пам'яті призводить до необхідності проведення зв'язувань. Чим більша кількість зв'яз-

ків, які необхідно перевірити, тим більша кількість маркерів генерується. Це явище називають ефектом довгого ланцюга [12]. За рахунок збереження результатів попереднього зв'язування Rete алгоритм дозволяє мінімізувати вплив даного ефекту. Treat алгоритм будує ланцюг зв'язування на кожному циклі роботи. При цьому перевірка зовнішніх тестів відбувається і для продукцій, які в подальшому не активуються. Це зумовлює надлишкові обчислювальні затрати, мінімізовані в Rete алгоритмі.

Таким чином, обоє алгоритмів мають як переваги так і недоліки, зумовлені особливостями їх роботи. Для різних прикладних задач властивості алгоритмів можуть по різному впливати на ефективність виведення висновку.

В таблицю 2 зведено характеристики прикладних задач та рекомендований для них алгоритм співставлення зі зразком в залежності від певного критерію оптимальності.

Таблиця 2

Доцільність використання Rete або Treat алгоритму в залежності від характеристики прикладної задачі

Характеристики задачі	Критерій оптимальності	Rete	Treat
Велика кількість умовних елементів в антецеденті (>6)	Швидкодія	+	-
Середня кількість умовних елементів антецеденту не перевищує 6	Кількість співставлень з умовними елементами	-	+
Висока тимчасова надмірність	Швидкодія	+	-
Значна кількість негативних умовних елементів	Швидкодія	-	+
	Затрати пам'яті	-	+
Умовні елементи додаються динамічно	Швидкодія	-	+
	Затрати пам'яті	-	+
Велика кількість правил в БЗ	Кількість маркерів, які генеруються	+	-
	Затрати пам'яті	-	+
Мінімальна кількість змінних в складних антецедентах	Швидкодія	+	-

На основі даної таблиці можна обрати один з алгоритмів співставлення для розв'язування прикладної задачі з точки зору заданого критерію оптимальності.

Висновки

1. Формалізовано схеми потоків даних для алгоритмів Rete та Treat. Це дозволяє розробнику зрозуміти їх концепцію та відмінності між ними.
2. Представлено характеристики формалізованих прикладних задач відповідно до критеріїв оптимальності для Treat та Rete. Виділені переваги алгоритмів по кожній з них.

Список використаних джерел

1. Forgy, C. L. On the Efficient Implementation of Production System : PhD thesis / Charles L. Forgy ; Computer Science Department, Carnegie Mellon University. – Pittsburg, 1979. – 356 p.
2. Джаррантано, Дж. Экспертные системы: принципы разработки и програмирование / Дж. Джаррантано , Г. Райли ; 4-е издание. : Пер. с англ. – М.: ООО “И.Д. Вильямс”, 2007 – 1152 с.
3. Friedman-Hill, E. Jess the Rule Engine for the JavaTM Platform / E. Friedman-Hill // Sandia National Laboratories . – Режим доступу: http://www.jessrules.com/ . – Дата доступу: 09.04.2014 р.
4. Forgy, C. L. The 0PS83 Report Charles L. Forgy ; Computer Science Department, Carnegie Mellon University. – Pittsburg ,1984 – 47 p.
5. Rosenbloom, P.S. A Preliminary Analysis of the Soar Architecture as a Basis for General / P. Rosenbloom, J. Laird, A. Newell, R. McCarl // Artificial Intelligence – 1991. – Vol. 47.– P. 289–325.
6. Sellis, T. Implementing large production systems in a DBMS environment: concepts and algorithms / T. Sellis, C. C. Lin, L. Raschid // Management of data : ACM SIGMOD international conference / ACM Press – New York, 1988. – P. 404–423.
7. Berstel, B. Extending the RETE algorithm for event management / B. Berstel // Temporal Representation and Reasoning : Ninth International Symposium / IEEE Computer Society Press – Washingtоn, 2002. – P. 49–51.
8. Doorenbos, R. Production Matching for Large Learning Systems / R.. Doorenbos ; Computer Science Department, Carnegie Mellon University. – Pittsburg, 1995. – 208 p.
9. Miranker, D. P. TREAT: A New and Efficient Match Algorithm for AI Production Systems / D. P. Miranker – London : Pitman/Morgan Kaufmann, 1990. – 144 p.
10. Miranker, D. P. The organization and performance of a treat-based production system compiler / D. P. Miranker, B. J. Lofaso // IEEE Transactions on Knowledge and Data Engeneering. – 1991 – Vol. 3(1) – P. 3–10.

11. Wang, Y. A performance comparison of the rete and treat algorithms for testing database rule conditions / Y. Wang, E. N. Hanson // Data Engeneering : Eighth International Conference / IEEE Computer Society Press – Washington, 1992, – P. 88–97.

12. Nayak, P. Comparison of rete and treat production matchers for soar (a summary) / P. Nayak, A. Gupta, P. Rosenbloom // Artificial Intelligence : Seventh National Conference (AAAI-88) / The MIT Press – Cambridge, 1988. – P. 693–698.

Отримано 36.03.2014 р.