12. Денинг, В. Диалоговая система «человек–ЭВМ». Адаптация к требованиям пользователя [Текст] / В. Денинг, Г. Эссинг, С. Маас. – М.: Мир, 1984. – 110 с.
13. Скакун, С. В. Нейросетевая модель пользователей компьютерных систем [Текст] / С. В. Скакун, Н. Н. Куссуль // Кибернетика и вычислительная техника. – 2004. – Вып. 143. – С. 55–68.
14. Петровский, А. Б. Пространства множеств и мультимножеств [Текст] /А. Б. Петровский. – М: Едиториал УРСС, 2003. – 248 с.
15. Вовк, О. Л. Применение мультимножеств для ранжирования [Текст] / О. Л. Вовк // Вестник ХНТУ. Работы молодых ученых. – 2008. – № 1(30). – С. 498–502.
16. Петровский, А. Б. Метрические пространства мультимножеств [Текст] / А. Б. Петровский // Доклады Академии наук. – 1995. - Т. 344, № 2. - С. 175–177.
17. Anonymous Web data from www.microsoft.com / [Electronic resource] / Available at: http://kdd.ics.uci.edu/databases/msweb/msweb.data.html
18. Breese, J. S. Empirical analysis of predictive algorithms for collaborative filtering [Text] : proc. of the fourteenth conf. / J. S. Breese, D. Heckerman, C. Kadie // Uncertainty in Artificial Intelligence (UAI-98). Morgan Kaufmann, San Francisco, 1998. – P. 43–52.

УДК 004.825

ВИБІР ОПТИМАЛЬНОГО АЛГОРИТМУ СПІВСТАВЛЕННЯ ЗІ ЗРАЗКОМ ПРИ ПРОЕКТУВАННІ ПРОДУКЦІЙНОЇ СИСТЕМИ

С. І. Шаповалова
Кандидат технічних наук, доцент*
E-mail: lana@ aprodos.ntu-kpi.kiev.ua
О. О. Мажара
Аспірант*
E-mail: olyamazhara@gmail.com
*Кафедра автоматизації проектування енергетичних процесів і систем
Національний технічний університет України «Київський політехнічний інститут»
вул. Політехнічна, 6, м. Київ, Україна, 03056

Запропонована сукупність характеристик продукційної системи, які залежать від формалізованого представлення прикладної задачі та впливають на ефективність логічного виведення. На основі даних характеристик запропоновано схему визначення оптимального за швидкодією та затратами пам'яті алгоритму співставлення зі зразком на етапі проектування продукційної системи. Приведено приклад вибору алгоритму для задачі діагностування баштової градирні

Ключові слова: продукційна система, співставлення зі зразком, представлення антецедента, обгортка продукційних систем

Предложена совокупность характеристик продукционной системы, которая зависит от формализованного представления прикладной задачи и определяют эффективность логического вывода. На основании данных характеристик предложено схему определения оптимального по быстродействию и затратам памяти алгоритма сопоставления с образцом на этапе проектирования продукционной системы. Приведен пример выбора алгоритма для задачи диагностирования башенной градирни

Ключевые слова: продукционная система, сопоставление с образцом, представление антецедента, оболочка продукционных систем

1. Вступ

Подальша інтелектуалізація сучасного програмного забезпечення визначає широке використання систем, які базуються на знаннях. Продукційні системи є одним з розповсюджених різновидів таких систем.

В процесі функціонування продукційної системи найбільш затратною за ресурсами пам'яті та часу є задача співставлення зі зразком [1 – 3]. Враховуючи циклічність процесу виведення, розв'язання цієї задачі повторюється багато разів. В [4] доведено, що процес співставлення у системі може займати 90 % усього часу виконання програми.

При проектуванні продукційної системи необхідно враховувати багато характеристик вирішуваної прикладної задачі та використовуваного алгоритму

співставлення. Ці характеристики в багатьох випадках взаємозалежні, а іноді створюють суперечливі вимоги до представлення інформації. Тому створення на основі цих характеристик формальних рекомендацій з організації процесу логічного виведення, оптимального за швидкістю обробки антецедентів продукцій, є актуальною задачею, яка має практичне значення.

2. Постановка проблеми

При аналізі існуючих алгоритмів співставлення зі зразком основною проблемою є те, що в них представлені і використані різні показники ефективності. Зазвичай не надається інформація про вплив на ефективність виведення та можливість застосування запропонованих параметрів оцінки з точки зору використання ресурсів і швидкодії. Метрики можуть бути упередженими для алгоритму, його реалізації або контекстних функцій. Тобто характеристики для порівняння можуть бути підібрані таким чином, що заздалегідь надаватимуть перевагу одному з алгоритмів.

Цілями дослідження є:

1) формалізація принципів представлення характеристик продукційної системи та антецедентів продукцій для обрання концепції алгоритму співставлення зі зразком, який забезпечить оптимальний за швидкістю процес логічного виведення при розв'язанні прикладної задачі;

2) наведення прикладу визначення алгоритму співставлення зі зразком за запропонованими рекомендаціями.

В дослідженні виокремлено критерії, які відповідають оптимальності використання кожного з існуючих алгоритмів співставлення. Проблема полягає в визначенні на їх основі оптимального за швидкодією та затратами пам'яті алгоритму співставлення зі зразком з урахуванням особливостей прикладної задачі.

3. Літературний огляд

Базовим підходом для усіх ефективних методів співставлення є інкрементний підхід [5 – 7]. Проте він породжує постійний компроміс у співвідношенні швидкодії до витрат на пам'ять. Збільшення швидкодії продукційної системи призводить до збільшення затрат пам'яті [8].

Основою інкрементних алгоритмів співставлення є збереження від циклу до циклу деякої інформації про стан системи, відоме як збереження стану. Ефективність використання комбінації декількох методів до збереження стану була вперше досліджена McDermott [9], який висловив припущення, що для деяких продукційних систем повторне виконання тестів коштуватиме менше, ніж збереження графового представлення попередньо узгоджених умовних елементів.

Це припущення стало поштовхом для розробки нових алгоритмів співставлення зі зразком з меншим використанням пам'яті [10].

Проблема вибору параметрів для порівняння Rete та Treat алгоритмів розглядалась в роботах [10, 11], які дають різні результати в залежності від вибору метрик. Кореляція переваг певного алгоритму в залежності від складу правила розглядалася в роботах [11, 12].

Дослідження показують, що в залежності від середньої кількості умовних елементів в продукціях може змінюватися швидкодія алгоритмів. Так в роботах [8, 11 – 13] порівнювались алгоритми Treat та Rete. Експериментальні результати п'яти різних тестових програм, реалізованих з допомогою оболонки продукційних систем OPS5, показали, що швидкість алгоритму Treat, краща ніж у Rete майже в два рази у більшості випадків [10]. В той же час протилежні результати було продемонстровано в роботі [2]. Інше порівняння алгоритмів було зроблене в контексті обгортки продукційних систем SOAR [11]. Важливою відмінністю між застосунками SOAR та OPS5 є складність правил перших. Ця обставина породжує ефект довгого ланцюга та посилює ефект крос-продукцій. На противагу роботі Miranker, в цьому дослідженні Rete перевершує Treat алгоритм. Було зазначено, що дані результати зумовлені складністю правил в SOAR та її наслідками, які мають більше недоліків для Treat.

Wang і Hanson досліджували ефективність Rete та Treat в контексті баз даних, які мають свою власну базу знань [12]. Вони виміряли час обробки маркерів і Treat показав себе краще ніж Rete у більшості випадків. Варіації затрат для різного впорядкування зв'язування високі, що робить його нестійким, в той час як Treat показує стійкі результати навіть без оптимізації. Це дозволило зробити висновок, що завдяки швидкодії та збереженню пам'яті Treat є кращим алгоритмом співставлення для поєднання баз даних з продукційними системами.

Таким чином, в різних порівняльних дослідженнях одних і тих самих алгоритмів перевагу віддають почергово Treat і Rete алгоритмам. Узагальнених висновків, які дозволять обирати алгоритм співставлення зі зразком на етапі проектування продукційної системи з урахуванням особливостей вирішуваних прикладних задач, досі не існує.

4. Обрання оптимального алгоритму співставлення зі зразком при проектуванні продукційних систем

4. 1. Алгоритми співставлення зі зразком

Базові концепції алгоритмів співставлення зі зразком здебільшого запропоновані в 1990-х роках. Згодом до них вносилися лише модифікації. Алгоритми співставлення зі зразком поділяють на 2 основні класи в залежності від того, чи зберігають вони відомості про попередню конфліктну множину або обчислюють її на кожному циклі співставлення [10]. Проте даної класифікації недостатньо для формування рекомендацій вибору одного з них.

В даному дослідженні розглядається три основні типи алгоритмів співставлення зі зразком. В табл. 1 зведено об'єкти обчислень даних алгоритмів та приклади їх реалізацій, запропоновані різними науковцями.

Таблиця 1

Алгоритми співставлення зі зразком

№	Тип	Об'єкт обчислення	Приклади алгоритмів		
			Назва	Автор	Рік
1	Алгоритми нетерплячої оцінки (Eager Evaluation Algorithms)	Повний конфліктний набір (усі активовані правила) на кожному циклі співставлення	Gator	Eric N. Hanson Mohammed S. Hasan [14]	1993
			Rete	Charles Forgy [4]	1979
			Treat	Daniel P. Miranker [10]	1990
			Rete*	Ian Wright, James Marshall [8]	2003
2	Алгоритми лінивої оцінки (Lazy Evaluation Algorithms)	Тільки одна активація правила згідно зі стратегією розв'язання конфлікту (новизни)	Leaps	Daniel P. Miranker [10]	1990
3	Алгоритми зв'язування простору (Binding Space Algorithms)	Повний конфліктний набір (усі активовані правила) на кожному циклі співставлення	Matchbox	Mark Perlin, Jean-Marc Debaud [15]	1989
			HAL	Pou-Yung Lee, Albert Mo Kim Cheng [16]	2002
			GridMatch	Jack Tan, Manish Maheshwari, Jaideep Srivastava [17]	1990

У всіх інкрементних алгоритмів формується графове представлення процесу співставлення. Алгоритми відрізняються способом формування такого графу та правилами його обробки. Зазвичай процес співставлення відбувається в два етапи. На першому (внутрішній тест) перевіряється узгодженість окремих умовних елементів з фактами робочої пам'яті. На другому (зовнішній тест) відбувається зв'язування змінних – перевірка на відповідність змінних в межах правила. Спільним для всіх алгоритмів також є збереження стану від циклу до циклу.

Однак різні підходи до інформації, яка зберігається, зумовлюють різницю в характеристиках алгоритмів та їх чутливості до проблем, які виникають під час співставлення. В залежності від цих характеристик виникають вимоги до алгоритмічного забезпечення продукційної системи і в першу чергу до алгоритму співставлення.

4. 2. Характеристики продукційної системи

В даному дослідженні виділено характеристики продукційної системи, які залежать від прикладної задачі та впливають на ефективність механізму виведення.

Ці характеристики можна розбити на підкласи в залежності від сфери дії (рис. 1).

Загальні вимоги до продукційної системи дозволяють визначитися лише з основними характеристиками необхідного алгоритму. Так, необхідність забезпечення навчання в системі передбачає формування повної конфліктної множини на кожному циклі співставлення зі зразком. Це дозволяє одразу відмовитися від алгоритмів лінивої оцінки, які не надають такої можливості.

Варто приймати до уваги, що не завжди є можливість отримати властивості часу виконання програми. В такому випадку слід звертатися до алгоритмів співставлення, які розраховані на низьку тимчасову надмірність та обробку значних проміжних результатів внаслідок недостатньої селективності умовних елементів. Селективним (конкретним) називається умовний елемент, який характеризується тим, що узгоджується з якомога меншою кількістю фактів робочої пам'яті та породжує якомога більшу кількість зв'язувань змінних, які обмежують інші умовні елементи [18].

Рис. 1. Характеристики продукційної системи

4. 3. Проблеми процесу співставлення

Деякі з проблем співставлення можна вирішити шляхом вдосконалення представлення бази знань, інші ж вимагають наявності певних особливостей механізму співставлення. В табл. 2 зведені основні проблеми та існуючі шляхи їх вирішення через оптимізацію запису продукцій в базі знань.

Таблиця 2

Проблеми співставлення зі зразком

Назва	Суть проблеми	Принципи оптимізації
Низька тимчасова надмірність [10]	Зниження ефективності системи зі збільшенням кількості фактів, які змінюються від циклу до циклу	Відсутні
Крос-продукційний Ефект (Cross-Product Effect) [11, 19]	Генерація великої кількості проміжних результатів на етапі зовнішнього тестування	Обмежувальні перші (Restrictive first)
		Групування змінних (Group variables)
Довголанцюговий ефект (Long-Chain Effect) [11, 19, 2]	Збільшення витрат на виконання зовнішніх тестів у випадку складних правил, які мають велику кількість умовних елементів	Непостійні останні (Volatile last)
Проблема утиліт (Utility Problem) [20, 4]	Уповільнення процесу виведення зі збільшенням бази знань	Відсутні

Порядок послідовності умовних елементів в правилі грає критичну роль для співставлення. Він визначає порядок виконання зовнішніх тестів (з'єднань). Якщо порядок з'єднання неоптимальний, виконується додаткова потенційно непотрібна робота. Окрім генерації великої кількості результатів зв'язування, можливі великі затрати на обчислення для визначення правил, які не будуть активованими. Для оптимізації порядку умовних елементів в антецеденті використовують евристики – підходи, що з високою вірогідністю дозволять пришвидшити процес співставлення. В той же час, це не строгі правила, вони не спроможні гарантувати, що обране рішення найбільш ефективне. В роботах [18, 19, 21, 22] описано евристики для впорядкування умовних елементів правил з метою генерації кращого порядку зв`язування та зменшення витрат на співставлення.

В табл. 3 представлено евристики, які допомагають оптимізувати процес співставлення шляхом впорядкування умовних елементів в антецеденті продукції.

Оптимізація представлення правил в базі знань дозволяє вплинути на процес співставлення і, отже, повинна бути врахована розробником продукційної системи. Використання декількох евристик може створювати колізію. Наприклад, умовні елементи, які зіставляються з часто змінюваними фактами (і повинні бути розміщені останніми) можуть також бути обмежувальними (і повинні бути розміщені першими) [18]. Крім того, для їх ефективного застосування можуть знадобитися апріорні знання, такі як селективність умовних елементів або частота зміни фактів. Нажаль такі знання можуть бути отримані лише під час роботи програми.

На вибір алгоритму співставлення впливає можливість їх отримання.

Оптимізатор для алгоритму Gator [14] використовує інформацію, зібрану під час виконання програми, та оціночну модель для обчислення оптимального порядку зв'язування. Алгоритми співставлення, які не покладаються на статичну структуру зв'язування, такі як, наприклад, Treat [10], мають можливість для динамічної оптимізації порядку зв'язування. Використовується "недорога" евристика впорядкування, так як накладні втрати на динамічне визначення оптимального порядку зв'язування в кожному циклі важко обчислити [10, 21].

Таблиця 3

Принципи розташування умовних елементів

Принцип оптимізації продукції	Правило оптимізації	Пояснення дії
Обмежувальні перші	Умовні елементи з більшою селективністю розміщуються першими	Зменшується кількість проміжних результатів та кількість зв'язувань з іншими умовними елементами
Групування змінних	Умовні елементи, які мають спільні змінні, повинні бути згруповані	Зменшується кількість проміжних результатів зв'язування
Впорядкування умовних елементів	Подібні умовні елементи різних правил розміщуються першими	Зменшується обсяг результатів внутрішніх та зовнішніх тестів
Негативні першими	Негативні умовні елементи розміщуються в антецеденті так рано, як це можливо	Зменшуються розміри результатів зв'язування
Непостійні останніми	Умовні елементи, які зіставляються з часто змінюваними фактами, розміщуються останніми	Зменшується кількість операцій зв'язування та змін в часткових співставленнях правила

4. 4. Модифікації алгоритмів співставлення зі зразком

Для створення продукційних систем зазвичай використовують спеціалізовані програмні засоби – обгортки. Вони забезпечують користувача незалежним від предметної області механізмом виведення. Доцільно використовувати такі готові рішення при створенні продукційних систем. В той же час документальний супровід більшості обгорток продукційних систем не містить формального опису механізму виведення та особливостей реалізації методів співставлення зі зразком та розв'язання конфліктів.

Деякі з обгорток продукційних систем постачаються з відкритим програмним кодом. Це надає можливість аналізувати та модифікувати методи, які вони використовують. При проектуванні продукційних систем, зна-

ючи характеристики бази знань, можна скористатися вже існуючими модифікаціями алгоритмів. В даному дослідженні вивчалися найпоширеніші з них. В табл. 4 представлені деякі з таких обгорток та підходи до модифікації алгоритмів співставлення зі зразком.

Не враховуються властивості часу виконання тому, що на етапі проектування немає можливості їх отримати. Розрахунок виконується як сума оцінок за кожною характеристикою. Максимальне значення відповідає оптимальному алгоритму.

Таблиця 4

Модифікації класичних алгоритмів співставлення зі зразком

Алгоритм	Мета модифікації	Обгортка продукційної системи	Посилання
Rete	Підвищення швидкодії	CLIPS	[23]
	Використання в системах керування базами даних (DBMS) з метою роботи з базами знань з великою кількістю правил	OPS5	[24]
	Інтеграція обробки подій (events) в продукційних системах	ILOG JRules	[25]
	Вирішення проблеми утиліт в продукційних системах, здатних до навчання	OPS5	[26]
	Підвищення швидкодії	OPS5 SOAR	[27], [28]
	Підвищення швидкодії шляхом інтеграції з мультипроцесорною архітектурою CUPID	OPS5	[29]
LEAPS	Підвищення швидкодії та створення нової специфікації в термінах контейнерно-курсорної абстракції (container-cursor programming abstractions)	OPS5	[30]

Отже, при виборі алгоритму співставлення зі зразком доцільно виділити найбільш значимі вимоги до продукційної системи та її відповідні характеристики. Оболонки продукційних систем, які поширюються з відкритим програмним кодом дозволяють модифікувати алгоритми співставлення зі зразком згідно з поточними потребами.

4. 5. Схема визначення оптимального алгоритму співставлення зі зразком

На основі виділених характеристик (рис. 1) розроблено 2 схеми визначення оцінки алгоритму співставлення зі зразком: відповідно для загальних вимог до продукційної системи (рис. 2) та особливостями представлення антецедентів продукцій (рис. 3).

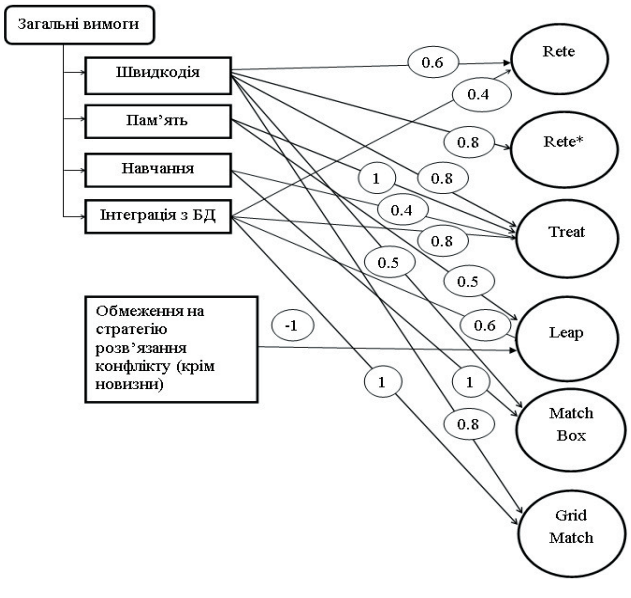

Рис. 2. Схема визначення пріоритету алгоритму співставлення за зразком за загальними вимогами до продукційної системи

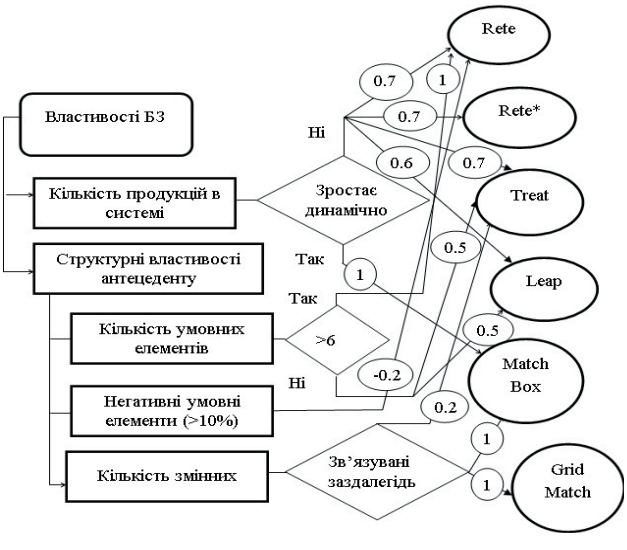

Рис. 3. Схема визначення пріоритету алгоритму співставлення за зразком за властивостями бази знань

5. Апробація результатів

Теоретичні засади, розглянуті в даній роботі, використовуються при реалізації програмного модуля визначення технічного стану споруд АЕС на основі даних візуального огляду, технічного паспорту та нормативної документації. Даний модуль реалізується у вигляді продукційної системи з допомогою спеціалізованих засобів розробки, а саме середовища CLIPS [18]. Вибір даного середовища був зумовлений тим, що воно забезпечує інтеграцію з зовнішніми програмними застосунками, поширене за вільною ліцензією, має відкритий програмний код, можливість співпраці з розробниками.

Формалізація знань для кожного об'єкту діагностики проводилася з урахуванням принципів опти-

мізації, приведених в табл. 3. Це дозволило визначити характеристики, які необхідно врахувати для вибору алгоритму співставлення зі зразком. Для прикладу беремо задачу діагностування баштової градирні. В табл. 5 наведено значення потрібних характеристик.

Таблиця 5

Характеристики прототипного модулю

Характеристика	Значення
Витрати пам'яті	Мінімальні
Забезпечення навчання системи	Відсутнє
Інтеграція з базою даних	Необхідна
Кількість продукцій в системі	Від 100 до 200
Середня кількість умовних елементів	5
Кількість змінних	Від 0 до 4
Кількість негативних умовних елементів	8 %

Згідно технічного завдання до системи не ставиться підвищених вимог щодо швидкодії.

Кількість правил в базі знань – 178. Правила можуть додаватися в систему лише інженером знань. Динамічне додавання правил неможливе. Кількість негативних елементів антецедентів не перевищує 8 %. Середня кількість умовних елементів в продукціях – 5. Кількість змінних в антецедентах варіюється від 0 до 4. В умовних елементах може відбуватися виклик функцій, тому зв'язування змінних неможливо обрахувати заздалегідь.

За даними характеристиками найбільш сприятливим для розробки є Treat алгоритм. Він забезпечує мінімізацію затрат пам'яті при збереженні високої швидкості обробки : 1.0 (Пам'ять) + 0.8 (Можливість інтеграції з БД) + 0.5 (Кількість умовних елементів) + 0.2 (Динамічне зв'язування) = 2.5. Низькі затрати на пам'ять зумовлені тим, що алгоритм не зберігає результати зовнішніх тестів. В той же час доведено, що для баз даних з продукціями, які містять незначну кількість умовних елементів (< 6), Treat надає значні переваги у швидкодії [10].

На основі даного висновку при програмній реалізації було модифіковано використовуваний в обгортці CLIPS Rete алгоритм. За основу для внесення змін було обрано підходи до збереження пам'яті, які використовує Treat алгоритм.

7. Висновки

1. Виділено характеристики продукційної системи, які залежать від прикладної задачі і впливають на вибір алгоритму співставлення зі зразком, запропоновано виділити характеристики структурних властивостей антецеденту.

2. Представлено формалізацію принципів оптимального представлення продукцій для запобігання проблем співставлення зі зразком.

3. На основі значень виділених характеристик продукційної системи запропоновано схему визначення алгоритму співставлення зі зразком, оптимального для поточної задачі.

4. Наведено приклад апробації цієї схеми для вибору алгоритму для розв'язання задачі визначення технічного стану баштової градирні АЕС на основі візуального обстеження, технічного паспорту і нормативної документації.

Результати дослідження можуть використовуватись при проектуванні продукційної системи, призначеної для вирішення заздалегідь формалізованої прикладної задачі.

Література

1. Tamble, M. Uni-rete: Specializing the rete match algorithm for the unique-attribute representation [Text] / M. Tamble, D. Kalp, P. S. Rosenbloom. – Scholl of Computer Science, Carnegie Mellon University. – Pittsburg : Computer Science Department, 1991. – 30 p.
2. Gupta, A. Parallelism in Production Systems [Text] / A. Gupta. – London : Pitman, 1987. – 255 p.
3. Friedman-Hill, E. Jess in Action: Java Rule-Based Systems [Text] / E. Friedman-Hill. – Greenwich : Manning Publications Co, 2003. – 480 p.
4. Forgy, C. L. On the Efficient Implementation of Production System : PhD thesis [Text] / C. L. Forgy. – Computer Science Department, Carnegie Mellon University. – Pittsburg, 1979. – 356 p.
5. Brant, D. A. Effects of database size on rule system performance: Five case studies. [Текст] : sevententh inter. conf. / D. A. Brant, T. Gose, B. Lofaso, D. P. Miranker // Morgan Kaufmann Publishers Inc. – San Francisco, 1991. – P. 287–296.
6. Miranker, D. P. The organization and performance of a treat-based production system compiler [Text] / D. P. Miranker, B. J. Lofaso // IEEE Transactions on Knowledge and Data Engeneering. – 1991. – № 3(1) – P. 3–10.
7. Perlin, M. Incremental binding-space match: The linearized matchbox algorithm [Text] : third IEEE inter. conf. / M. Perlin // Tools for Artificial Intelligence. IEEE Computer Society Press. – San Jose,1991. – P. 468–477.
8. Wright, I. The execution kernel of rc++: Rete*, a faster rete with treat as special case [Text] / I. Wright, J. Marshall // International Journal of Intelligent Games and Simulation. – 2003. – № 2(1). – P. 36–48.
9. McDermott, J. The efficiency of certain production system implementations [Text] / J. McDermott, A. Newell, J. Moore // Pattern-Directed Inference Systems. – 1978. – № 67. – P. 155–176.
10. Miranker, D. P. TREAT: A New and Efficient Match Algorithm for AI Production Systems [Text] / D. P. Miranker. – London : Pitman/Morgan Kaufmann, 1990. – 144 p.

11. Nayak, P. Comparison of rete and treat production matchers for soar (a summary) [Text] : seventh nat. conf. (AAAI-88) / P. Nayak, A. Gupta, P. Rosenbloom // Artificial Intelligence. The MIT Press. – Cambridge, 1988. – P. 693–698.
12. Miranker, D. P. Treat: A better match algorithm for AI production systems [Text] : sixth nat. conf. (AAAI-87) / D. P. Miranker // Artificial Intelligence. The MIT Press. – Seattle, 1987. – P. 42–47.
13. Wang, Y. A performance comparison of the rete and treat algorithms for testing database rule conditions [Text] : eighth inter. conf. / Y. Wang, E. N. Hanson // IEEE Computer Society Press. – Washington, 1992. – P. 88–97.
14. Hanson, E. N. Gator: An optimized discrimination network for active database rule condition testing [Text] / E. N. Hanson, M. S. Hasan. – University of Florida. – Gainesville : CIS Departement, 1993. – 27 p.
15. Perlin, M. Match box: Fine-grained parallelism at the match level [Text] : IEEE inter. workshop / M. Perlin, J. Debaud // IEEE Computer Society Press. – Fairfax, 1989. – P. 428–434.
16. Lee, P. Hal: A faster match algorithm [Text] / P. Lee, A. M. K. Cheng // IEEE Transactions on Knowledge and Data Engeneering. – 2002. – № 14 (5). – P. 1047–1058.
17. Tan, J. Gridmatch: A basis for integrating production systems with relational databases [Text] : IEEE inter. workshop / J. Tan, M. Maheshwari, J. Srivastava // IEEE Computer Society Press. – Herndon, 1990. – P. 400–407.
18. Джаррантано, Дж. Экспертные системы: принципы разработки и програмирование [Текст] / Дж. Джаррантано, Г. Райли; 4-е издание.: Пер. с англ. – М.: ООО «И.Д. Вильямс», 2007 – 1152 с.
19. Scales, D. J. Efficient matching algorithms for the soar/ops5 production system [Text] / D. J. Scales. – Scholl of Computer Science, Carnegie Mellon University. – Pittsburg : Computer Science Department, 1986. – 36 p.
20. Doorenbos, R. Production matching for large learning systems [Text] / R. Doorenbos. – Computer Science Department, Carnegie Mellon University. – Pittsburg, 1995. – 208p.
21. Ishida, T. Parallel, Distributed and Multiagent Production Systems [Text] / T. Ishida. – Michigan : Springer, 1994. – 172 p.
22. Kang, J. A. Shortening matching time in ops5 production systems [Text] / J. A. Kang, A. M. K. Cheng // IEEE Transactions on Software Engineering. – 2004. – № 30 (7). – P. 448–457.
23. Liu, D. Rule Engine based on improvement Rete algorithm [Text] : inter. conf. / D. Liu, Gu T., Xue J. // Apperceiving Computing and Intelligence Analysis. IEEE Computer Society Press. – Chengdu, 2010. – P. 346–349.
24. Sellis, T. Implementing large production systems in a DBMS environment: concepts and algorithms [Text] : ACM SIGMOD inter. Conf. / T. Sellis, C. C. Lin, L. Raschid // ACM Press. – New York, 1988. – P. 404–423.
25. Berstel, B. Extending the RETE algorithm for event management [Text] : ninth inter. symposium / B. Berstel // Temporal Representation and Reasoning. IEEE Computer Society Press. – Washington, 2002. – P. 49–51.
26. Doorenbos, R. Production Matching for Large Learning Systems [Text] / R. Doorenbos. – Computer Science Department, Carnegie Mellon University. – Pittsburg, 1995. – 208 p.
27. Kang, J. A. Shortening matching time in OPS5 production systems [Text] / J. A. Kang // IEEE Transactions on Software Engineering. – 2004. – № 30(7). – P. 448–457.
28. Scales, D. Efficient Matching Algorithms for the SOARlOPS Production System [Текст] / D. Scales. – Computer Science Department, Stanford University. – Stanford, 1986. – 58 p.
29. Kelly, M. An evaluation of DRete on CUPID for OPS5 matching [Text] : 11th inter. joint conf. / M. Kelly, R. Seviora // Morgan Kaufmann Publishers. – San Francisco, 1989. – P. 84 – 90.
30. Batory, D. The LEAPS Algorithms [Text] / D. Batory. – Computer Science Department, The University of Texas. – Austin, 1994. – 15 p.