УДК 004.832.2

ФОРМАЛІЗАЦІЯ СПІВСТАВЛЕННЯ ЗІ ЗРАЗКОМ ЗА TREAT-АЛГОРИТМОМ

К. т. н. С. І. Шаповалова, О. О. Мажара

НТУУ «Київський політехнічний інститут»
Україна, м. Київ
olyamazhara@gmail.com

Запропоновано формальне представлення Treat-алгоритму в термінах логіки першого порядку, яке дозволяє отримати характеристики обробки поточної бази знань, аналогічні з наявними для Rete-алгоритму.

Ключові слова: співставлення зі зразком, продукційна система, формалізація.

Сучасна тенденція до використання портативних пристроїв породжує необхідність створення інтелектуальних програмних засобів з підвищеними вимогами до ефективності використання обчислювальних ресурсів. Тому вдосконалення інструментарію розробки таких систем, які за своїми характеристиками можна використовувати на портативних пристроях, є актуальною. В прикладних задачах штучного інтелекту широко застосовуються продукційні системи (ПС). В [1] доведено, що базовим алгоритмом, який впливає на ефективність роботи ПС, є співставлення зі зразком. Серед існуючих алгоритмів співставлення найбільш поширеними є інкрементні алгоритми Rete, Treat та їх модифікації.

Для коректного порівняння алгоритмів співставлення необхідне їх формальне представлення. Математична модель виведення на базі логіки першого порядку в ПС дозволяє коректно оцінити ефективність їх роботи. Rete-алгоритм формально описано в термінах логіки першого порядку в дослідженнях [2, 3]. Проте для коректного порівняння Rete та Treat не існує однотипного формального опису роботи, що зумовлює необхідність його створення. Метою даної роботи є формальне представлення виконання Treat- та Rete-алгоритмів співставлення зі зразком для подальшого аналізу їх ефективності.

Робота інкрементних алгоритмів співставлення ґрунтується на запам'ятовуванні кортежів фактів, які узгоджуються з частиною антецеденту. В Rete-алгоритмі на кожному кроці виведення зберігається узгодження умов антецеденту з фактами робочої пам'яті (внутрішні тести), а також результат зв'язування змінних (перевірка узгодженості значень змінних в межах одного правила — зовнішні тести). Treat-алгоритм передбачає збереження узгодження фактів, проте зв'язування змінних обчислюється повторно на кожному кроці виведення. Реалізація інкрементного підходу базується на побудові дискретної мережі з антецедентів правил. Для виконання внутрішніх тестів в антецеденті будується альфа-мережа, в Anode-вузлах якої містяться тести для окремих шаблонів та факти, які призвели до активації поточної вершини. Підмножина таких фактів називається лівою (альфа) пам'яттю lm. Компіляцію внутрішніх тестів в мережу потоку даних можна представити єдиним чином для обох алгоритмів. Для заданого правила $l_j \in PM$ (Production memory) позначимо кон'юнкцію внутрішніх тестів q_i^j для заданого шаблону p_i. Для виконання зовнішніх тестів Rete-алгоритм передбачає додаткову бета-мережу, в Bnode-вузлах якої відбувається зв'язування змінних. Термінальна вершина Tnode зберігає всі факти, які узгоджуються з антецедентом, та передає їх на вхід до конфліктного набору CS. Кон'юнкцію всіх зовнішніх тестів позначимо як r_j. Treat-алгоритм не будує проміжних вершин для зв'язування змінних. Якщо правило було активоване на попередньому кроці співставлення, то в Tnode зберігається підмножина фактів cm_j, яка призвела до активації відповідного правила $l_j \in PM$. Додатково на кожному кроці співставлення в термінальній вершині зберігається множина

фактів *tm*, яка узгоджується з активованими шаблонами правила на поточному кроці зв'язування змінних.

Представимо в загальному вигляді правила переходу $?m \xrightarrow{a} ms!$ для вершин Treat мережі, де $?m$ — вхідний маркер у вигляді $<+|-,f>$ для факту f, що додається чи видаляється з робочої пам'яті; *ms*! — список вихідних маркерів; a — опціональні дії на поточному кроці. Для Anode правило переходу визначається наступним чином:

$$<+,f>? \xrightarrow{lm_j \leftarrow lm_j \cup \{f\}} \{+,f\}!\ \ ifq(f),$$

$$<-,f>? \xrightarrow{lm_j \leftarrow lm_j - \{f\}} \{-,f\}!\ ifq(f),$$

$$<+|-,f>? \rightarrow 0!\ \ if\neg q(f).$$

Для Tnode правило переходу визначається наступним чином $\forall f \in lm_j$:

$$<-,f>? \longrightarrow \{<-,f>\},$$

$$<+,f>? \xrightarrow{tm_j \leftarrow tm_j \cup \{f\} \wedge \sigma \leftarrow \sigma \cup \sigma' | \exists R \in PM \wedge \exists p \in p^+(R) | \sigma' p = f} 0!\ \ ifq(f)\wedge \neg r_j(tm_j),$$

$$<+,f>? \xrightarrow{tm_j \leftarrow tm_j \cup \{f\} \wedge \sigma \leftarrow \sigma \cup \sigma' | \exists R \in PM \wedge \exists p \in p^+(R) | \sigma' p = f} \{<+,tm_j>\}!\ \ ifq(f)\wedge r_j(tm_j),$$

$$<+|-,f>? \rightarrow 0!\ if\neg q(f).$$

Тоді для конфліктної множини CS

$$<-,f>? \xleftarrow{cm \leftarrow cm - \{f\}} \{<-,f>\},$$

$$<+,tm_j>? \xrightarrow{cm \leftarrow cm \cup tm_j;} \{<+,tm_j>\}!,$$

де R — продукція, σ — підстановка (визначена в стандартних термінах логіки першого порядку), p — шаблон правила, визначений як терм, p^+ — позитивний шаблон правила.

Запропоноване формальне представлення Treat-алгоритму використовується для оцінки середньої складності на основі математичної моделі, запропонованої в [3]. На його основі розширюється модель розрахунку витрат пам'яті, запропонована в [4], з врахуванням додаткових затрат на збереження пам'яті в конфліктному наборі та в термінальній вершині: $M(n) = \sum_{i=0}^{n} (lm_j + tm_{j_i} + cm_j)$, де n — кількість правил в базі знань. Таким чином, в даній роботі запропоновано формальне представлення Treat-алгоритму, яке дозволяє отримати характеристики обробки поточної бази знань, аналогічні з наявними для Rete-алгоритму. Це дозволяє обрати оптимальний алгоритм співставлення зі зразком для розв'язання поточної інтелектуальної задачі.

ВИКОРИСТАНІ ДЖЕРЕЛА

1. Forgy, C. L. On the efficient implementation of production system: PhD thesis / Computer Science Department, Carnegie Mellon University.– Pittsburg, 1979.

2. Cirstea H., Kirchner C., Moossen M., Moreau P.-E. Production systems and rete algorithm formalisation.– Режим доступу: \https://hal.inria.fr/file/index/docid/280938/filename/rete.formalisation.pdf .– 2008.

4. Albert L., Fages F. Average case complicity analyzes of the Rete multi-pattern match algorithm // Automata, Languages and Programming 5th International Colloquium Tampere.— Finland.— 1988.— P. 18–37.

3. Wright I., Marshall J.A.R. The execution kernel of RC++: RETE*, a faster RETE with TREAT as a special case // International Journal of Intelligent Games and Simulation.— 2003.— N 2(1).— P. 36–48.

O. O. Mazhara, S. I. Shapovalova

Treat algorithm formalization.

A formalization of the Treat algorithm in terms of first-order logic is proposed. It allows defining characteristics of the current knowledge base, similar to those defined for Rete algorithm. This formalization is used for choosing the optimal matching algorithm to solve the current intellectual problem.

Keywords: *production system, match, formalization*