УДК 681.3:553.98

АЛГОРИТМИ АВТОМАТИЗОВАНОГО ВИБОРУ ОПТИМАЛЬНОГО МЕТОДУ ОБМЕЖЕННЯ ПРИПЛИВУ ПЛАСТОВИХ ВОД У СВЕРДЛОВИНУ

© Яковин С.В., 2005
Івано-Франківський національний технічний університет нафти і газу

Представлено структуру інформаційної системи „PLAST", яка призначена для вибору оптимального методу обмеження припливу пластових вод (ОППВ) у свердловину. Запропоновано реалізацію фреймів та слотів. Описано модифіковані алгоритми активізації фреймів при прямому та зворотному виведенні, які дозволяють досягнути більшої швидкості обробки даних

Особливістю розробки нафтових родовищ в Україні є ріст об'ємів ремонтно-ізоляційних робіт [1]. Значна частина ремонтно-ізоляційних робіт припадає на обмеження припливу пластових вод (ОППВ) у свердловину [2]. При виборі методу ОППВ необхідно одночасно враховувати багато факторів, які погано піддаються формалізації [1]. Для вирішення такого класу задач використовуються різні методи.

В основу розробленої інформаційної системи (ІС) "PLAST", яка призначена для вибору оптимального методу обмеження припливу пластових вод (ОППВ) у свердловину, покладено ядро у вигляді процесора логічного висновку на фрейм-продукційному представлені знань. База знань зберігається у спеціальному форматі, який фактично відповідає деякому нульовому стану внутрішнього представлення бази. Основними типами даних, представлених в системі, є числовий Number, логічний Bool та текстовий String, списковий List, який містить набір String. Всі вони успадковані від батьківського класу Any (рис. 1). Батьківський клас задає типово-незалежні операції GetName, GetType, GetValueStr, GetValue, Input. Вирази представляються у вигляді двійкового дерева довільного вигляду (рис. 2).

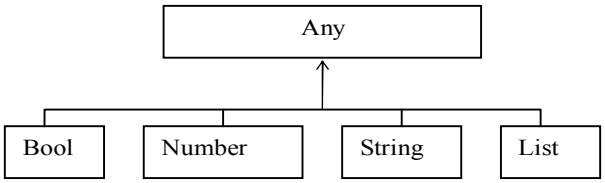

Рис. 1. Діаграма наслідування базових типів в ІС „PLAST"

Базовим класом для представлення виразів є TreeNode, в якому декларується метод compute() для обчислення значення виразу у відповідності із існуючим контекстом. Від TreeNode походять BinOpTreeNode та UnaryOpTreeNode. В свою чергу від цих класів походять операції: OpAdd, OpSub тощо.

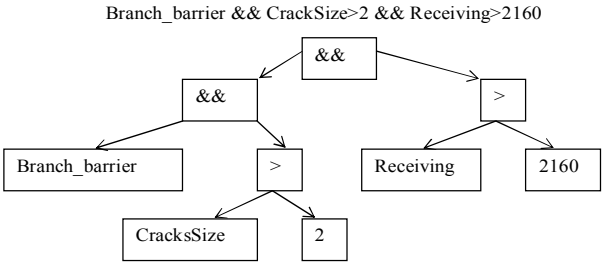

Рис. 2. Приклад представлення виразу у вигляді двійкового дерева

В ІС статичні знання про предметну область представляються у вигляді фреймової ієрархії, а як динамічні знання використовуються продукційні правила, згруповані навколо фреймів та слотів (рис. 3). Кожен фрейм в ІС представляється у вигляді екземпляра класу Frame.

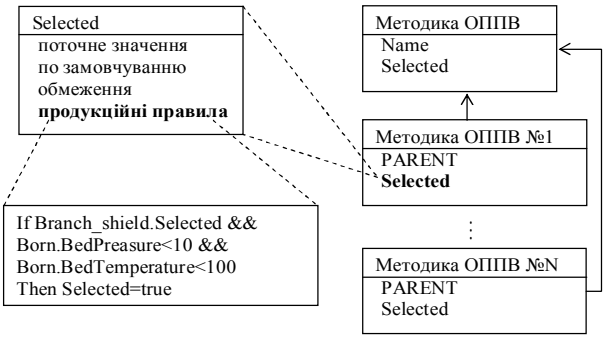

Рис. 3. Фрагмент представлення знань про методи ОППВ

Для забезпечення цілісності побудови продукційні правила зворотного виведення транслюються у відповідні процедури-запити, приєднані до слотів, що містяться в правій частині правил, а правила прямого виведення транслюються у процедури-демони, які приєднуються до слотів, що фігурують у лівій частині правил. Таким чином забезпечується спрацювування правил при зміні будь-якого із вхідних операндів. Слоти, значення яких повинні запитуватися у користувача, містять посилання на екземпляр об'єкту Input.

Правила зворотного виведення представлені одним із підкласів класу AnyAction, який реалізує процедури, що повертають значення Any, а саме: ComputeAction – для простих присвоєнь та обчислень виразів (ComputeAction містить посилання на дерево виразу, і, при активізації, обходить його обчислюючи результат, та повертає отримане значення); IfThenAction для невироджених продукційних правил з непустою множиною посилок (в цьому випадку спершу обчислюється умова, а при її істинності – і сам вираз). При запиті значення слоту в процесі виведення переглядаються всі під'єднані до нього процедури-запити і на основі евристичного правила визначається порядок їх застосування.

Для правила, яке активоване, виконується скорочене обчислення умовної частини, яке, в свою чергу, може привести до запиту значень інших слотів, індукуючи таким чином рекурсивний процес зворотного виведення. У випадку істинності умовної частини слоту присвоюється значення виразу, що міститься в правій частині. При цьому може ініціюватися наступне логічне виведення, якщо в виразі є посилання на значення інших слотів. У випадку неуспішності розглядається наступне правило.

Після присвоєння значення здійснюється перевірка обмежень, встановлених на слот (фасетні обмеження), на фрейм або фреймову ієрархію (референціальні обмеження). У випадку невиконання обмежень слоту присвоюється значення empty і застосування процедур продовжується. Таким чином здійснюється часткове повернення і застосування інших правил. При частковому поверненні не здійснюється відновлення попереднього стану у зв'язку із монотонністю процесу виведення. Після того, як знайдено значення, яке не суперечить обмеженням, розгляд правил завершується.

Блок-схема на рис. 4 показує процес обчислення значення слоту. Якщо значення слоту не вдалось отримати за допомогою приєднаних процедур, то аналогічним чином здійснюється запит значення відповідного слоту фрейма-батька. При цьому в процедуру передається посилання на поточний фрейм, який називається базовим. Таким чином, для успадкованих фреймів застосовуються всі процедури-запити (відповідно, всі продукційні правила зворотного виведення), які задані для батьків. Методи GetFirstGetAction() та GetNextGetAction() забезпечують ітеративне проходження списку правил-демонів, під'єднаних до слоту. LPR(r) та RPR(r) повертають відповідно ліву або праву частину породжуючого правила. Calculate() здійснює розрахунок в контексті заданого фрейму. Check() здійснює перевірку значення на відповідність заданим для слоту обмеженням. Activate() активує приєднані до слота процедури-демони.

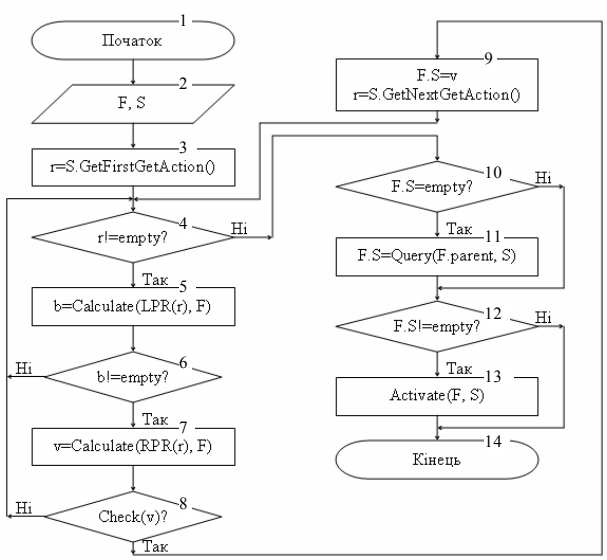

Рис. 4. Блок-схема застосування правил-запитів для реалізації зворотного виведення

Для реалізації прямого виводу в класичних продукційних системах сімейства OPS5 застосовуються різні алгоритми прямого виведення, найбільш поширеними з яких є алгоритми сімейства Rete [3], які по ефективності на порядок переважають „наївним" алгоритмом, який базується на співставленні на кожному кроці всіх правил із станом робочої пам'яті [4].

Rete-подібні алгоритми, до яких відносяться запропонований Чарлзом Форджі [3] алгоритм Rete, TREAT [5], Gator [6] та інші, базуються на побудові мереж певної конфігурації правил прямого виводу. В проміжних вузлах сітки (α та β пам'яті) зберігається інформація про істинність окремих підвиразів лівої частини правила, таким чином, при зміні значення змінної відбувається оновлення тільки невеликого фрагменту мережі.

Аналогічним чином алгоритми сімейства Rete можуть бути застосовані і до фрейм-продукційної системи. У цьому випадку правила прямого виводу

при трансляції перетворюються у мережу Rete, в кінцевих вузлах якої розміщені, з однієї сторони, процедури-демони, які приєднані до відповідних слотів, з іншої – процедури, які виконуються при істинності даного виразу. При присвоєнні слоту значення спрацьовують процедури-демони батьківських слотів, активізуючи всі пов'язані з ними правила. Для правильної обробки наслідування правилам необхідно також мати посилання на фрейм, який викликав активацію (базовий фрейм). Таким чином, з кожною активацією в мережі Rete зберігається також посилання на базовий фрейм. В ІС "PLAST" використовується алгоритм прямого виведення, який запропонований у [7], зі зміною, при якій єдина мережа для всіх правил не будується. У цьому випадку при змінні значення слоту активуються всі пов'язані з даним слотом процедури-демони, які безпосередньо пробують розрахувати вирази, які знаходяться в лівій частині продукційних правил.

Вирази, які представлені відповідними деревоподібними структурами в пам'яті, вираховуються за спеціальним алгоритмом із запам'ятовуванням проміжних результатів у вузлах-операціях дерева, які у даному випадку відіграють роль β-пам'яті, яка застосовується в алгоритмі Rete. Необхідність в α-пам'яті в даному випадку відпадає (як α-пам'ять використовуються самі слоти), так як уніфікація правил із значеннями в робочій області не здійснюється, а заміщається неявною уніфікацією за наслідуванням, яка досягається за рахунок виклику процедур-демонів всіх батьківських фреймів з передачею поточного фрейму, який викликав активацію, як контекст виклику. Таким чином, мережа утворюється неявно під'єднаними до слотів процедурами-демонами, пов'язаними з ними правилами та деревами виразів, в вузлах яких запам'ятовуються проміжні значення.

Процедури-демони застосовуються згідно визначеного для них порядку. Після вибору наступної процедури-демона для застосування перевіряється пов'язане з нею правило на предмет виконання заданих в ньому умов, і, якщо правило спрацьовує і веде до модифікації інших слотів, активізуються відповідні їм процедури-демони. Тільки тоді розглядаються наступні процедури-демони вихідного слоту. Завершимість алгоритму достатньо очевидна із-за скінченої кількості слотів та приєднаних процедур, враховуючи, що одна і та ж процедура не може спрацювати двічі. Це забезпечується занесенням базового фрейму в контекст активності приєднаної процедури. Блок-схема алгоритму прямого виведення представлена на рис. 5.

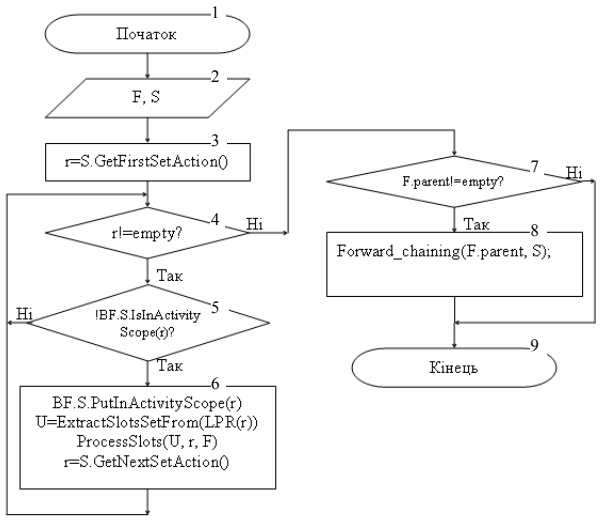

Рис.5. Блок-схема алгоритму прямого виведення, що базується на застосуванні процедур-демонів

Таким чином, застосування запропонованих алгоритмів виконання прямого та зворотнього виведення дозволяє збільшити швидкість опрацювання бази знань в інформаційній системі „PLAST". За рахунок впровадження ІС "PLAST" на підприємствах ВАТ „Укрнафта" збільшилась кількість успішно проведених водоізоляційних робіт, вивільнились людські та матеріальні ресурси на проведення експертних оцінок.

1. Блажевич В. А. и др. Ремонтно-изоляционные работы при эксплуатации нефтяных месторождений. /В. А. Блажевич, Е. Н. Умрихина, В. Г. Уметбаев. – М.:Недра, 1981.-233с. 2. Довідник з нафтогазової справи /За заг. ред. докторів технічних наук В. С. Бойка, Р. М. Кондрата, Р. С. Яремійчука. – К.:Львів, 1996.–620с. 3. Forgy C. L. RETE: A fast algorithm for the many pattern / many object pattern match problem. Artificial Intelligence, Vol. 19, No.1, 1982.–P. 17-37. 4. Girratano J., Riley G. Expert Systems: Principles and Programming. – PWS Publishing Company, Boston, 1993. (2nd Ed.). 5. Miranker D. P. TREAT: A New and Efficient Match Algorithm for AI Production Systems. Morgan Kaufmann Publishers, Inc., San Mateo, California, 1990. 6. Hanson E. N., Hasan M. S. Gator: An optimized discrimination network for active database rule condition testing. Technical Report TR-93-036, CIS Department, University of Florida, 1993. 7. Журавлева Т.Э. Гибридный инструментарий интеллектуальных систем на основе расширенного логического программирования. Дисс ... к.ф.–м.н. – М.: МАИ, 1993.–38 с.