УДК 004.942

ІМПЛЕМЕНТАЦІЯ CSP-КОНЦЕПЦІЙ ДЛЯ ІНТЕЛЕКТУАЛЬНОГО АНАЛІЗУ ДАНИХ НАФТОГАЗОВОЇ ПРЕДМЕТНОЇ ОБЛАСТІ

Л.І. Випасняк, Б.І. Шпакодрай, В.І. Шекета

Івано-Франківський національний технічний університет нафти і газу,
76019, Івано-Франківськ, вул. Карпатська, 15/1416.
Тел.: +38 097 988 6938, факс +38 034–224 2139,
e-mail: lyubic@gmail.com

Представлено аналіз проблеми застосування CSP-концепції для виконання інтелектуального аналізу даних засобами технології Constraint Logic Programming. Введено формально-логічне обґрунтування процедури ефективного керування множиною обмежень на прикладі задачі нафтогазової предметної області.

In this paper the analysis of CSP conception implementation problem for intelligible data analysis by means of Constraint Logic Programming is presented and formal logical substantiation of effective control routine for constraints set is introduced on the base of example data from oil&gas subject domain.

Вступ

Одна з актуальних проблем штучного інтелекту – проблема задоволення обмежень (CSP – constraint satisfaction problem) [1], яка має ряд застосувань: прогнозування, розподілення ресурсів, планування, розміщення ресурсів та ін. Наукові пошуки в області CSP базуються на класичних задачах штучного інтелекту, мовах програмування штучного інтелекту, абстрактних обчисленнях, теоріях логіки. В першому наближенні обмеження можна розглядати як прості логічні відношення між кількома невідомими чи змінними які набувають значення з своїх доменів. Обмеження таким чином розділяє можливі значення, які змінні можуть набувати. Вони представляють деяку часткову інформацію про змінні. Формалізовано проблема задоволення обмежень може бути представлена кортежем $(X, D, Constr)$ [2], яка складається з множини змінних $X = \{x_i\}, i = \overline{1.n}$, скінченних множин D_i їх можливих значень (доменів), $D = \{D_i\}, i = \overline{1,n}$, і множини обмежень які обмежують значення, що змінні можуть одночасно приймати $Constr = \{Constr_i\}, i = \overline{1,n}$. Розв'язком CSP є набір значень з відповідних доменів для кожної змінної, таких щоб задовольнялося кожне накладене обмеження. Розрізняють пошук тільки одного розв'язку; всіх розв'язків; оптимального розв'язку[3]. Обмеження мають ряд властивостей: можливість описувати часткову інформацію; обмеження не є напрямленими; обмеження мають декларативний характер, тобто можуть описувати відношення без опису обчислювальних процедур; обмеження адитивні, тобто порядок їх виклику не є суттєвим; спільні обмеження можуть використовуватися для розділених змінних.

Проблему задоволення обмежень (над скінченними доменами) типово розв'язують використовуючи алгоритми пошуку. Найбільш часто вживані алгоритми – це форми *бектрекінгу* (backtracking), *поширення обмежень* (constraint propagation), та *локального пошуку* (local search). Як відомо, *бектрекінг* – це рекурсивний алгоритм, що підтримує часткове присвоєння змінних. На кожному кроці, вибраній змінній присвоюються всі можливі значення, які вона може отримати. Для кожного отриманого значення використовується послідовність часткового присвоєння з накладеними обмеженнями які верифікуються через рекурсивний виклик. Коли всі значення будуть верифіковані, задається рекурсивне повторення. В базових імплементаціях бектрекінгу для CSP, виконуюча послідовність означається як процес задоволення всіх обмежень для ініціалізованих змінних. Для концепції CSP існує кілька імплементацій: *backmarking* – реалізація, що дозволяє покращити ефективність перевірки послідовності, *backjumping* – зберегти частину множини пошуку в вигляді розділених змінних, *look-ahead* – використовувати ефект передбачення вибору змінних чи їх послідовностей і використовується при виконанні задоволення обмежень у випадку розбиття загальної CSP задачі на ряд підзадач. Концепція *поширення обмежень* (constraint propagation) – це методи, що використовуються, для модифікації проблеми задоволення обмеження у вигляді локальної послідовності, умови якої пов'язані з послідовністю групи змінних чи обмежень. Технологія поширення обмежень має застосування у вигляді кількох кроків. На першому кроці початкова задача перетворюється на еквівалентну, яка є простішою для вирішення. На другому кроці виконується задоволення обмежень або доводиться неможливість задоволення системи обмежень. Найбільш вживаними формами локальної послідовності для поширення обмежень є: дугові послідовності, гіпер-дугові послідовності, шляхові послідовності[3]. Найбільш оптимальний метод поширення обмеження – алгоритм AC-3[2]. З другого боку методи локального пошуку що використовуються є алгоритмами неповного задоволення обмежень. Тобто вони, можуть, знайти вирішення проблеми, а можуть видати помилку навіть у випадку коли проблема є такою, що не може бути розв'язана. Суть їхнього функціонування близька до ітераційного методу,

тобто поступова зміна початкових значень змінних з метою задоволення накладених обмежень. На кожному кроці велика кількість змінних з модифікованими значеннями приводить до збільшення числа обмежень, що були задоволені даним присвоєнням. В практичних імплементаціях локальний пошук працює ефективно, коли зміни вносяться не тільки базовим алгоритмом пошуку а ще й деякими альтернативними алгоритмами. Тобто реально діючі алгоритми є гібридними за своєю структурою [1,2]. Правила керування обмеженнями [3] спочатку означалися як формалізм для специфікації розв'язувачів обмежень. Пізніше відбулося їхнє вбудовування в логічне програмування. Виділяються два типи правил керування обмеженнями. Правила першого типу задаються згідно умови, що множина обмежень є еквівалентною до деякої іншої множини. Правила другого типу задаються згідно умови, що з однієї множини обмежень слідує інша множина. В теорії constraint satisfaction problem (CLP), що підтримує правила виведення з обмеженнями можемо використати ці правила для специфікування модифікації множини обмежень, і можливого додавання обмежень до цієї множини.

Отже, виконаний аналіз літературних джерел показує актуальність проблеми інтелектуалізації інформаційно-пошукових задач шляхом ефективного керування множиною обмежень. Проте невирішеними залишається ряд проблем ефективного керування множиною накладених обмежень засобами абстрактних логічних програм з обмеженнями та їх модифікацій.

Мета роботи – введення формально-логічної процедури для ефективного керування множиною обмежень накладених на задану пошукову задачу, та їх ефективне використання з точки зору концепції constraint logic programming.

З точки зору теорії абстрактної імплементації проблема задоволення обмеження в обмеженому домені є NP – повною проблемою [2]. Тобто можемо застосовувати контексти обмежень і формулювати результуюче дерево рішень. Деякі дослідження показують існуюче співвідношення проблеми задоволення обмеження і абстрактних теорій побудованих на скінченних моделях. З другого боку, семантика CLP[4] може бути сформульована в термінах віртуального інтерпретатора, що підтримує пари $\langle Goal, ConstrSet \rangle$ під час виконання програми. Перший елемент (Goal – ціль) називається поточною ціллю, другий елемент називається множиною обмежень, що накладається на процес задоволення цілі. Поточна ціль містить літерали, які інтерпретатор намагається довести і також може містити обмеження, які він намагається задовольнити. Множина обмежень містить всі обмеження, які інтерпретатор розглядає як такі, що можуть бути задоволені на даний момент часу. Після вибору початкової цілі множина обмежень є порожньою. Інтерпретатор починає свою роботу з видалення першого елемента поточної цілі і його аналізу. В результаті даного аналізу можемо одержати успішне або неуспішне завершення програми. Даний аналіз може включати в себе також рекурсивні виклики і додавання нових літералів до поточної цілі і нових обмежень до множини обмежень. Інтерпретатор починає працювати в зворотному напрямі якщо генерується переривання про хибне завершення. Успішне завершення генерується якщо поточна ціль є порожньою і множина обмежень є такою, що задовольняється. Після того, як видалили літерал з заголовку цілі, виконується перевірка чи це є обмеження чи літерал. Якщо це є обмеження то воно додається до множини обмежень. Якщо це літерал то вибирається твердження, заголовок якого має той самий предикат що й вибраний літерал. Це твердження перезаписується через заміну змінних новими змінними. Результат цього процесу називається новим варіантом твердження. Тіло нового варіанта твердження розміщається в заголовку цілі.

Під час виконання цієї операції виконується ряд контрольних перевірок, зокрема множина обмежень перевіряється на консистентність кожного разу коли додається нове обмеження. В загальному випадку перевірка множини обмежень є така, що вона не може бути задоволена, має виконуватися алгоритмом бектрекінгу, хоча така перевірка, на кожному кроці не є ефективною для загального процесу. З цієї причини використовується аналізатор часткового задоволення. На практиці можливість задоволення множини обмежень перевіряється на основі методів, що спрощують множину обмежень, тобто виконують її переписування в еквівалентну форму, але яка є простішою для розв'язання. Ці методи можуть, але не завжди, доводити не можливість задоволення множини обмежень. Вважається, що інтерпретатор довів істинність цілі, якщо поточна ціль порожня і множина обмежень є така, що не містить фактів які можуть бути інтерпретовані як не можливість задоволення цілі.

Наслідком виконання описаної процедури є поточна множина або спрощена поточна множина обмежень. Дана множина може включати обмеження, вигляду $X_{D_i}^{C_i}$ (де $i = \overline{1,n}$, C_i – обмеження накладені на змінні X_i з домену D_i), що змушують змінні приймати певні значення, але можуть включати також обмеження $X_{D^i}^{C^j}$, що виконують прив'язування змінних до певного діапазону без присвоєння їм певного значення.

Формально семантика CLP позначається в термінах послідовних доведень [4]. Для цього використовується представлення пар ціль/множина обмежень $\langle Goal, ConstrSet \rangle$ і такий перехід у процесі доведення позначається як $\langle Goal, ConstrSet \rangle << \langle Goal', ConstrSet' \rangle$. Така пара описує можливі переходи з стану $\langle Goal, ConstrSet \rangle$ в $\langle Goal', ConstrSet' \rangle$. Згідно [3, 4] перехід можливий в трьох випадках:

- елемент $Goal$ – обмеження $\tilde{N}onstr$, тоді $Goal' = Goal \setminus \{Constr\}$, $ConstrSet' = ConstrSet \cup \{Constr\}$; тобто обмеження може бути перенесено з цілі до множини обмежень;

- елемент $Goal$ – літерал $L(t_1,...,t_n)$ тоді існує твердження, що може бути модифіковано використанням нових змінних, таких як $L(t_1',...,t_n') << B$, $Goal$ є $Goal'$ з $L(t_1,...,t_n)$ і замінюються вони таким чином $t_1 = t_1,...,t_n = t_n, B$ і $ConstrSet' = ConstrSet$; іншими словами літерал може бути замінений тілом нового варіанта твердження, що має ті самі предикати в заголовку через додавання тіла нового варіанта і відповідності термів у цілі;
- $ConstrSet$ і $ConstrSet'$ – еквівалентні згідно певної семантики обмежень.

Отже, послідовність доведення розглядається як послідовність пар $\langle Goal, ConstrSet\rangle$, ціль $Goal$ може бути верифікована, якщо існує таке доведення з $\langle Goal, 0\rangle$ в $\langle 0, ConstrSet\rangle$ з деякої множини обмежень $ConstrSet$, що може бути задоволена. Ця семантика формалізує подальший розвиток ідеї інтерпретації, як явного вибору літералів з цілі, для обробки тверджень, для заміни літералів. Іншими словами, ціль є доведеною в певній семантиці, якщо існує послідовність виборів літералів і тверджень серед множини можливих інших тверджень, що ведуть до порожньої цілі, множини обмежень, що можуть бути задоволені. Фактичний процес інтерпретації елементів цілі слідує порядку LIFO: тобто елементи додаються до заголовка цілі і обробляються з заголовка цілі. Вони відповідно вибирають також твердження з другого правила відповідно до порядку в якому вони записані і переписують множину обмежень коли виконується її верифікація. Іншим можливим варіантом інтерпретації є заміна множини обмежень на еквівалентну. Така заміна обмежується тільки певними специфічними методами як, наприклад поширенням обмежень. Семантика CLP – параметрична, не тільки з точки зору вигляду обмежень, що використовуються, але також з точки зору методів перезапису множини обмежень. Специфічний метод, що використовується на практиці дозволяє замінити множину обмежень такою, яка простіша для задоволення. Якщо множина обмежень така, що не може бути задоволена, то таке спрощення дозволить виявити таку можливість в деяких випадках, про те не у всіх. Результат оцінювання цілі CLP програми означається через доведення цілі. В даному випадку можна стверджувати, що існують деякі доведення з початкової пари до деякої пари, для якої ціль є порожньою. Множина обмежень для другої пари розглядається як наслідок такого оцінювання. Така ситуація існує тому, що множина обмежень містить всі обмеження, для яких було припущено, що вони є такі, що можуть бути задоволені для досягнення поставленої цілі. Іншими словами, ціль доведена для всіх змінних оцінювання, що задовольняють дані обмеження.

Кожна CLP програма, як логічна програма, що складається з правил, може бути модифікована з метою підвищення ефективності. Першим правилом у такій програмі є процедура, що задає опис літералів, воно має бути поміщено після певної множини обмежень для помічених літералів, що згенеровані в множині обмежень. Тобто, $\Pi(X) << labeling(X), X_{D_i}^{C_i}$ є еквівалентною до $\Pi(X) << X_{D_i}^{C_i}, labeling(X)$, де $labeling()$ – це процес маркування. Пошук виконується коли інтерпретатор зустрічає маркуючий літерал у множині обмежень, що не містить обмеження $X_{D_i}^{C_i}$. Як результат можемо одержати генерацію розв'язків, таких, що задовольняють $X_{D_j}^{C_j}$ які пізніше будуть встановлені як такі, що не задовольняють дане обмеження. З іншого боку, в другому формулюванні пошук здійснюється тільки тоді коли обмеження знаходиться в початковій множині обмежень. Як результат, алгоритм пошуку буде повертати тільки ті рішення, що зменшують область пошуку.

Друга модифікація яка збільшує ефективність є розміщення обмежень та літералів у тілі тверджень. Тоді $\Pi_1(X) << \Pi_2(X), X_{D_i}^{C_i}$ є в принципі еквівалентна до $\Pi_1(X) << X_{D_i}^{C_i}, \Pi_2(X)$. Проте перший запис вимагає певних обчислень. Наприклад, якщо множина обмежень містить обмеження $X_{D_i}^{C_i}$, тоді інтерпретатор рекурсивно оцінить $\Pi_2(X)$ в обох випадках. Приклад правил: $\Pi_1(X) \Leftrightarrow \Pi_2(X) \,|\, \Pi_3(X)$; $\Pi_1(X) \Rightarrow \Pi_2(X) \,|\, \Pi_3(X)$.

Третім способом модифікації є додавання залишкових обмежень. Якщо програміст знає, що рішення програми задовольняє певні обмеження, то вони можуть включати обмеження, що спричиняє незв'язність множини обмежень найшвидшим способом.

Практичний досвід імплементації інформаційних систем для нафтогазової предметної області [5,6] та досвід практичної експлуатації інформаційних інтелектуальних систем побудованих на основі концепції абстрактних логічних програм показує, що користувач стикається з проблемою накладання та задоволення обмежень навіть у випадку найпростіших систем даного класу, зокрема експертних систем: COLLECTOR, PLAST, PRYPLUV; які розроблені на кафедрі програмного забезпечення автоматизованих систем Івано-Франківського національного технічного університету нафти і газу (www.nung.edu.ua, www.pz.nung.edu.ua) під керівництвом доктора технічних наук, професора, В.М. Юрчишина, та використовуються в профільних підприємствах галузі. Тому зручним є виконання представлення множини обмежень у вигляді баз знань ($K_B^{ConstrSet}$). Згідно представлення логічних програм у вигляді правил, що модифікуються[7] враховуючи послідовності $\langle Goal, ConstrSet\rangle$, запишемо:

$$K_B^{ConstrSet^{In}}(o) \overset{\langle Goal, ConstrSet\rangle}{<<} K_B^{ConstrSet^{In}}(o_1),...,K_B^{ConstrSet^{In}}(o_l), K_B^{ConstrSet^{Out}}(p_1),...,K_B^{ConstrSet^{Out}}(p_m), \quad (1)$$

$$K_B^{ConstrSet^{Out}}(o) \overset{\langle Goal, ConstrSet\rangle}{<<} K_B^{ConstrSet^{In}}(o_1),...,K_B^{ConstrSet^{In}}(o_l), K_B^{ConstrSet^{Out}}(p_1),...,K_B^{ConstrSet^{Out}}(p_m), \quad (2)$$

де $o, o_i, p_i \in O, <<$ – дескриптор модифікації. Розглядатимемо базу знань як набір інформаційних сутностей атомарних предикатів з деякого скінченого інформаційного простору O [8, 9]. Всі зміни, що відбуватимуться в базі знань, будемо розглядати, як наслідок модифікаційних предикатних запитів, що генеруються ІС відповідно до вказівок користувача. Основою самих запитів є набір модифікаційних предикатних правил. Основна ідея такого запису правил у тому, що $K_B^{ConstrSet^{In}}(o)$ означає, що деяке обмеження o має бути включено в базу знань обмежень $K_B^{ConstrSet}$, а $K_B^{ConstrSet^{Out}}(o)$ означає, що o – має бути виключено з бази знань обмежнь.

Введемо поняття необхідної модифікації, яка буде описувати всі операції $K_B^{ConstrSet^{In}}()$ і $K_B^{ConstrSet^{Out}}()$, що спричиняються модифікаційним запитом для вихідної бази знань обмежень $K_B^{ConstrSet}$. Мета такої модифікації – ефективне керування множиною обмежень, щоб отримати таке наповнення бази знань яке може бути задоволено найпростішим (оптимальним) способом.

Означення 1. Нехай $Q_M^{ConstrSet}$ – модифікаційний предикатний запит. Позначимо $\lambda_{nm}^{ConstrSet}(Q_M^{ConstrSet})$ необхідні модифікації для $K_B^{ConstrSet}$. $\lambda_{nm}^{ConstrSet}(Q_M^{ConstrSet})$ фактично є найменшою моделлю для $Q_M^{ConstrSet}$, якщо $\lambda_{nm}^{ConstrSet}(Q_M^{ConstrSet})$ розглядати, як Хорн-програму $Q_M^{ConstrSet}$ побудовану з незалежних атомарних предикатів вигляду $K_B^{ConstrSet^{In}}(o)$ і $K_B^{ConstrSet^{Out}}(o)$. Нехай, наприклад, модифікаційний предикатний запит $Q_M^{ConstrSet}$ має вигляд

$$\{K_B^{ConstrSet^{In}}(o) \overset{\langle Goal, ConstrSet\rangle}{<<} K_B^{ConstSet^{Out}}(p) \overset{\langle Goal, ConstrSet\rangle}{<<} K_B^{ConstrSet^{In}}(o), K_B^{ConstrSet^{Out}}(r) \overset{\langle Goal, ConstrSet\rangle}{<<} K_B^{ConstrSet^{Out}}(o)\}$$, тоді

$$\lambda_{nm}^{ConstrSet}(Q_M^{CosntrSet}) = \{K_B^{ConstrSet^{In}}(o), K_B^{ConstrSet^{Out}}(p)\}.$$

В результаті виконання модифікаційних предикатних запитів база знань обмежень $K_B^{ConstrSet}$ змінюється, але статус деяких елементів залишається незмінним. При створенні набору модифікаційних правил ми власне зазначаємо тільки опис того, чи слід змінити певні елементи. Якщо для певного елемента немає правила модифікації, то відповідно, згідно принципу інерції, він не буде змінюватися в ході виконання модифікацій. Тобто, як і показує досвід практичної експлуатації систем COLLECTOR, PLAST, PRUPLUV на певних етапах роботи даних систем користувач має справу з інертними обмеженнями. Проблема усунення інертних обмежень з бази знань обмежень буде однією з наступних тем досліджень даного напряму, що виконуються на кафедрі.

Отже, множину всіх модифікаційних літералів, що описують елементи бази знань $K_B^{ConstrSet}$, статус яких не змінюється в процесі переходу від бази знань $K_B^{ConstrSet^1}$ до бази знань $K_B^{ConstrSet^2}$, $Q_M^{ConstrSet}$ і $K_B^{ConstrSet^1} \overset{\langle Goal, ConstrSet\rangle}{<<} K_B^{ConstrSet^2}$ можна розглядати множину інерції для $K_B^{ConstrSet^1}$, $K_B^{ConstrSet^2}$ і означати, як:

$$O_i(K_B^{ConstrSet^1}, K_B^{ConstrSet^2}) = \{K_B^{ConstrSet^{In}}(o) : o \in K_B^{ConstrSet^1} \cap K_B^{ConstrSet^2}\} \cup \{K_B^{ConstrSet^{Out}} : o \notin K_B^{ConstrSet^1} \cup K_B^{ConstrSet^2}\}.$$

Означення 2. Залишком для $Q_M^{ConstrSet}$ стосовно баз знань обмежень $(K_B^{ConstrSet^1}, K_B^{ConstrSet^2})$ $Q_M^{K_B^{ConstrSet^1}, K_B^{ConstrSet^2}}$ будемо називати новий модифікаційний запит, який отримується із $Q_M^{ConstrSet}$ шляхом видалення із тіла кожного модифікаційного правила в $Q_M^{ConstrSet}$ всіх модифікаційних літералів, які належать також і $O_i(K_B^{ConstrSet^1}, K_B^{ConstrSet^2})$.

Означення 3. Базу знань $K_B^{ConstrSet^2}$ будемо називати $Q_M^{ConstrSet}$ – модифікацією для $K_B^{ConstrSet^1}$, якщо множина необхідних модифікацій для $Q_M^{K_B^{ConstrSet^1}, K_B^{ConstrSet^2}}$ є когерентною і якщо

$$K_B^{ConstrSet^2} = K_B^{ConstrSet^1} \circ \lambda_{nm}^{ConstrSet}(Q_M^{K_B^{ConstrSet^1}, K_B^{ConstrSet^2}}).$$

Розглянемо наступний приклад.

Приклад 1. Припустимо, нам слід будувати процедуру логічного висновку для інтелектуальної системи на основі баз даних і знань нафтогазової предметної області PLAST виходячи із чотирьох характеристик нафтогазоносної породи:

$$ConstrSet = \{пористість, насиченість, проникність, коефіцієнт\_опору\}.$$

Причому, виходячи із досвіду експертів за опрацюванням даних характеристик породи, будемо мати справу із певним набором додаткових обмежень. А саме:

Constr[1]: Характеристики “пористості” і “насиченості” – найбільш продуктивна для виконання процедури логічного висновку тому одна із них має обов’язково бути присутньою у вихідній базі знань $K_B^{ConstrSet^{Initial}}$.

Constr[2]: Використання характеристики “проникність” – неефективна без одночасного опрацювання характеристики “коефіцієнт_опору”. Це так звані парні характеристики і якщо в $K_B^{ConstrSet^{Initial}}$ не буде включено характеристику “коефіцієнт_опору”, то не слід включати і характеристику “проникність”.

Constr[3]: Спільне використання характеристик “коефіцієнт_опору” і “проникність” не є ефективним, оскільки вони мають різну фізичну природу і методики їх вимірювання базуються на різних принципах.

Constr[4]: Неефективне також спільне використання характеристик “насиченість”, “коефіцієнт_опору” відповідно до причин, викладених у пункті 3.

Згідно перелічених обмежень як початкового складу бази знань приймемо

$$K_B^{ConstrSet^{Initial}} = \{пористість, проникність\}.$$

Прагнемо сформувати базу знань обмежень $K_B^{ConstrSet}$, яка б задовольняла всім чотирьом накладеним обмеженням. Таким чином у термінах вищевведених означень отримуємо $K_B^{ConstrSet^1} = \{пористість, проникність\}$. Побудуємо модифікаційний предикатний запит $Q_M^{ConstrSet}$:

$$\{K_B^{ConstrSet^{In}}(íàñè÷åí³ñò\ \ ü) \overset{\langle Goal, ConstrSet\rangle}{<<} K_B^{ConstrSet^{Out}}(ïîðèñò³ñòü\ \ ),$$

$$K_B^{ConstrSet^{In}}(ïîðèñò³ñòü\ \ ) \overset{\langle Goal, ConstrSet\rangle}{<<} K_B^{ConstrSet^{Out}}(íàñè÷åí³ñò\ \ ü),$$

$$K_B^{ConstrSet^{In}}(êîåô³ö³ºíò\ \ \_îïîðó\ \ ) \overset{\langle Goal, ConstrSet\rangle}{<<} K_B^{ConstrSet^{In}}(ïðîíèêí³ñò\ \ ü),$$

$$K_B^{ConstrSet^{Out}}(ïðîíèêí³ñò\ \ ü) \overset{\langle Goal, ConstrSet\rangle}{<<} K_B^{ConstrSet^{Out}}(êîåô³ö³ºíò\ \ \_îïîðó\ \ ),$$

$$K_B^{ConstrSet^{Out}}(ïîðèñò³ñòü\ \ ) \overset{\langle Goal, ConstrSet\rangle}{<<} K_B^{ConstrSet^{In}}(êîåô³ö³ºíò\ \ \_îïîðó\ \ ),$$

$$K_B^{ConstrSet^{Out}}(êîåô³ö³ºíò\ \ \_îïîðó\ \ ) \overset{\langle Goal, ConstrSet\rangle}{<<} K_B^{ConstrSet^{In}}(íàñè÷åí³ñò\ \ ü)\}$$

Покажемо, що $K_B^{ConstrSet^2} = \{пористість\}$ є $Q_M^{ConstrSet}$ – модифікацією для $K_B^{ConstrSet^1}$. Виходимо з того, що

$$O = \{пористість, насиченість, проникність, коефіцієнт\_опору\}.$$

Тоді

$$O_i(K_B^{ConstrSet^1}, K_B^{ConstrSet^2}) = \{K_B^{ConstrSet^{In}}(пористість), K_B^{ConstrSet^{Out}}(насиченість), K_B^{ConstrSet^{Out}}(коефіцієнт\_опору)\},$$

$$Q_M^{K_B^{ConstrSet^1}, K_B^{ConstrSet^2}} = \{K_B^{ConstrSet^{In}}(насиченість), K_B^{ConstrSet^{Out}}(пористість), K_B^{ConstrSet^{In}}(пористість)$$

$$\overset{\langle Goal, ConstrSet\rangle}{<<} K_B^{ConstrSet^{In}}(êîåô³ö³ºíò\ \ \_îïîðó\ \ ) \overset{\langle Goal, ConstrSet\rangle}{<<} K_B^{ConstrSet^{In}}(ïðîíèêí³ñò\ \ ü), K_B^{ConstrSet^{Out}}(ïðîíèêí³ñò\ \ ü)$$

$$\overset{\langle Goal, ConstrSet\rangle}{<<} K_B^{ConstrSet^{Out}}(ïîðèñò³ñòü\ \ ) \overset{\langle Goal, ConstrSet\rangle}{<<} K_B^{ConstrSet^{In}}(êîåô³ö³ºíò\ \ \_îïîðó\ \ ),$$

$$K_B^{ConstrSet^{Out}}(êîåô³ö³ºíò\ \ \_îïîðó\ \ ) \overset{\langle Goal, ConstrSet\rangle}{<<} K_B^{ConstrSet^{In}}(íàñè÷åí³ñò\ \ ü)\}.$$

Звідки

$$\lambda_{nm}^{ConstrSet}(Q_M^{K_B^{ConstrSet^1}, K_B^{ConstrSet^2}}) = \{K_B^{ConstrSet^{In}}(пористість), K_B^{ConstrSet^{Out}}(проникність)\}.$$

Збережена властивість когерентності

$$K_B^{ConstrSet^2} = K_B^{ConstrSet^1} \circ \lambda_{nm}^{ConstrSet}(Q_M^{K_B^{ConstrSet^1}, K_B^{ConstrSet^2}}).$$

Так показали, що $K_B^{ConstrSet^2}$ є $Q_M^{ConstrSet}$ – модифікація для $K_B^{ConstrSet^1}$.

Тепер покажемо, що $K_B^{Q_M^{ConstrSet}}$ є $Q_M^{ConstrSet}$ – модифікацією для $K_B^{ConstrSet^{Initial}}$. Припустимо, що правила з модифікаційного запиту $Q_M^{ConstrSet}$ є такими, що не є всі одночасно хибними. Тоді $E_M^{ConstrSet}$ є найменшою моделлю для $(\Delta_R^{Initial})^{E_M^{ConstrSet}} \cup (< (\Delta_R^{In})^{E_M^{ConstrSet}} >) \cup Q_M^{ConstrSet}$. Для того, щоб обчислити залишок $Q_M^{ConstrSet}\Big|_{K_B^{ConstrSet^{Initial}}, K_B^{Q_M^{ConstrSet}}}$ для $Q_M^{ConstrSet}$ маємо видалити з тіла кожного модифікаційного правила всі атомарні предикати $K_B^{ConstrSet^{In}}(o)$, такі, що $o \in K_B^{ConstrSet^{Initial}} \cap K_B^{Q_B^{ConstrSet}}$, і всі атомарні предикати $K_B^{ConstrSet^{Out}}(o)$, такі, що $o \in K_B'^{ConstrSet^{Initial}} \cap K_B'^{Q_B^{ConstrSet}}$. Це будуть ті атомарні предикати, що є істинними в $E_M^{ConstrSet}$, завдяки множині $< (\Delta_R^{In})^{E_M^{ConstrSet}} >$. Тому $E_M^{ConstrSet}$ залишається найменшою моделлю модифікованого запиту

$$(\Delta_R^{Initial})^{E_M^{ConstrSet}} \cup (< (\Delta_R^{In})^{E_M^{ConstrSet}} >) \cup Q_M^{ConstrSet}\Big|_{K_B^{ConstrSet^{Initial}}, K_B^{Q_B^{ConstrSet}}}.$$

Необхідна модифікація $\lambda_{nm}^{ConstrSet}(Q_M^{ConstrSet}\Big|_{K_B^{ConstrSet^{Initial}}, K_B^{Q_B^{ConstrSet}}})$ – найменша модель для $Q_M^{ConstrSet}\Big|_{K_B^{ConstrSet^{Initial}}, K_B^{Q_B^{ConstrSet}}}$. Розглянемо дві множини:

$$O_{In} = \{o : K_B^{ConstrSet^{In}}(o) \in \lambda_{nm}^{ConstrSet}(Q_M^{ConstrSet}\Big|_{K_B^{ConstrSet^{Initial}}, K_B^{Q_B^{ConstrSet}}})\},$$

$$O_{Out} = \{o : K_B^{ConstrSet^{Out}}(o) \in \lambda_{nm}^{ConstrSet}(Q_M^{ConstrSet}\Big|_{K_B^{ConstrSet^{Initial}}, K_B^{Q_B^{ConstrSet}}})\}.$$

Тоді запит $(\Delta_R^{Initial})^{E_M^{ConstrSet}} \cup (<(\Delta_R^{In})^{E_M^{ConstrSet}}>) \cup Q_M^{ConstrSet}\Big|_{K_B^{ConstrSet^{Initial}}, K_B^{Q_B^{ConstrSet}}}$ складається із трьох незалежних частин, причому залишок $Q_M^{ConstrSet}\Big|_{K_B^{ConstrSet^{Initial}}, K_B^{Q_B^{ConstrSet}}}$ більше не містить атомарних предикатів, які б входили в інші дві частини. Множина $E_M^{ConstrSet}$ буде складатися з:

$$\{K_B^{ConstrSet_{\in}^{In}}(o) : o \in K_B^{ConstrSet^{Initial}}\} \cup \{K_B^{ConstrSet_{\in}^{Out}}(o) : o \in K_B'^{ConstrSet^{Initial}}\},$$

$$\{K_B^{ConstrSet_{\in}^{In}}(o) : o \in K_B^{Q_B^{ConstrSet}} \cap K_B^{ConstrSet^{Initial}}\} \cup \{K_B^{ConstrSet_{\in}^{Out}}(o) : o \in K_B'^{Q_B^{ConstrSet}} \cap K_B'^{ConstrSet^{Initial}}\},$$

$$\{K_B^{ConstrSet_{\in}^{In}}(o) : o \in O_{In}\} \cup \{K_B^{ConstrSet_{\in}^{Out}}(o) : o \in O_{Out}\}$$

на даній основі, робимо висновок, що:

$$K_B^{Q_B^{ConstrSet}} = \{o : K_B^{ConstrSet^{In}}(o) \in E_M^{ConstrSet}\} = (K_B^{Q_B^{ConstrSet}} \cap K_B^{ConstrSet^{Initial}}) \cup O_{In},$$

$$K_B'^{Q_B^{ConstrSet}} = O \setminus K_B^{Q_B^{ConstrSet}} = \{o : K_B^{ConstrSet^{Out}}(o) \in E_M^{ConstrSet}\} = (K_B'^{Q_B^{ConstrSet}} \cap K_B'^{ConstrSet^{Initial}}) \cup O_{Out}.$$

Отже множина необхідних модифікацій $\lambda_{nm}^{ConstrSet}(Q_M^{ConstrSet}\Big|_{K_B^{ConstrSet^{Initial}}, K_B^{Q_B^{ConstrSet}}})$ є когерентною. Також $K_B^{Q_M^{ConstrSet}} = (K_B^{ConstrSet^{Initial}} \cup O_{In}) \setminus O_{Out}$. Звідки слідує, що $K_B^{Q_M^{ConstrSet}}$ є $Q_M^{ConstrSet}$ – модифікацією для початкової множини обмежень $K_B^{ConstrSet^{Initial}}$.

Висновки

Отже, в даному дослідженні введено формально-логічне обґрунтування процедури ефективного керування множиною обмежень на прикладі задачі нафтогазової предметної області. Представлено аналіз проблеми застосування CSP-концепцій для виконання інтелектуального аналізу даних засобами технології Constraint Logic Programming для інформаційно-пошукової задачі з чотирма накладеними обмеженнями.

Подальші дослідження будуть напрямлені на імплементацію отриманих формально-логічних результатів на основі шаблонів JCL (Java Constraint Library).

1. http://www.bracil.net/edward/FCS.html
2. http://en.wikipedia.org/wiki/Constraint_satisfaction
3. http://kti.mff.cuni.cz/~bartak/constraints/systems.html
4. http://en.wikipedia.org/wiki/Constraint_logic_programming
5. *Шекета В.І.* Інформаційна система для прогнозування нафтогазоносних покладів // Дис. канд. техн. наук: 05.13.06 / Херсон. держ. техн. ун-т. – Херсон, 1999. – 130 с.
6. *Юрчишин В.М.* Наукові основи застосування інформаційних технологій при управлінні процесами розробки нафтогазових родовищ //Дис. д-ра техн. наук: 05.15.06 / Івано-Франків. нац. техн. ун-т нафти і газу. – Івано-Франківськ, 2006. – 353 с.
7. *Шекета В.І.* Модифікаційні предикатні запити // Проблеми програмування. – 2004. – № 2-3. – С. 339 – 43.
8. *Шекета В.І.* Використання необхідної модифікації, як інструменту трансформації баз знань нафтогазової предметної області // Наук. пр. Донецьк. нац.–техн. ун. – Випуск 70. Сер.: "Інформатика, кібернетика та обчислювальна техніка" – Донецьк: ДонНТУ, 2003. – С. 39–46.
9. *Шекета В.І., Бестильний М.Я.,Храбатин Р.І.* Інтерпретація предикатних запитів на основі премоноїдної дедукції в семантичній стратегії і системі обмежень // Проблеми програмування. – 2006. – № 2-3. – С. 436 – 444.