Slovenská technická univerzita
Fakulta informatiky a informačných technológií
Ilkovičova 3, 812 19 Bratislava

Dokumentácia k zadaniu z UI:

Dopredný produkčný systém

Autor: **Peter Markuš**
Študijný program: INFO-4
Akademický rok: 2016/2017

Obsah

1 Zadanie úlohy

Úlohou je vytvoriť jednoduchý dopredný produkčný systém, s prípadnými rozšíreniami, napríklad o kladenie otázok používateľovi alebo vyhodnocovanie matematických výrazov.

Produkčný systém patrí medzi znalostné systémy, teda medzi systémy, ktoré so svojimi údajmi narábajú ako so znalosťami. Znalosti vyjadrujú nielen informácie o nejakom objekte, ale aj súvislosti medzi objektami, vlastnosti zvolených problémov a spôsoby hľadania ich riešenia. Znalostný systém je teda v najjednoduchšom prípade dvojica - program, ktorý dokáže všeobecne manipulovať so znalosťami a báza znalostí, ktorá opisuje problém a vzťahy, ktoré tam platia. Znalosti majú definovanú nejakú štruktúru a spôsob narábania s touto štruktúrou - to sa nazýva formalizmus reprezentácie znalostí. Program vie pracovať s týmto formalizmom, ale nesmie byť závislý od toho, aké znalosti spracováva, inak by to už nebol systém, kde riešenie úlohy je dané použitými údajmi.

Produkčný systém na základe odvodzovacieho pravidla *modus ponens* (pravidlo odlúčenia) odvodzuje zo známych faktov a produkčných pravidiel nové fakty. Ak systém nemá dostatok vstupných údajov, môže klásť používateľovi otázky.
Produkčný systém ako program nepozná konkrétne pravidlá ani fakty! Pozná len formalizmus, v tomto prípade štruktúru pravidiel a faktov a spôsob ich spracovania. Pozná akcie (pridaj, vymaž, ...), ktoré sa môžu vykonávať, lebo tie patria do opisu formalizmu.

Táto úloha sa v tomto tvare nemôže riešiť v jazyku PROLOG, pretože PROLOG už má vstavaný mechanizmus na odvodzovanie znalostí a výsledný program by neriešil úlohu, len vhodne načítal znalosti. Ak má niekto záujem riešiť to v Jazyku prolog, dostane od cvičiaceho rozšírenú verziu zadania.

K funkčnému programu je potrebné pripojiť aj dokumentáciu s opisom konkrétneho riešenia (reprezentácia znalostí, algoritmus, špecifické vlastnosti) a zhodnotením činnosti vytvoreného systému. Systém musí správne pracovať aspoň nad jednoduchou bázou znalostí (ekvivalentnou s prvou uvedenou), bázu znalostí si **musí** systém vedieť načítať zo súboru. Je vhodné si vytvoriť aj vlastné bázy znalostí a odovzdať spolu so zdrojovým kódom.

2 Opis konkrétneho riešenia

Znalosti sa reprezentujú a spracúvajú prostredníctvom štruktúry FACTS, ktorý obsahuje string vety faktu a odkaz na nasledujúci fakt. Ďalšou štruktúrou je RULE, ktorý obsahuje 3 časti stringov, meno podmienky, samotná podmienka, akcia a odkaz na nasledujúce pravidlo.

1. Načítavanie faktov zo súboru do štruktúry FACTS
2. Načítavanie pravidiel zo súboru do štruktúry RULE
3. Vytvoríme hlávnú slučku programu, pri ktorej každá iterácia je jeden výpiše všetky aktuálne nájdené fakty
4. V nej p**rechádzame všetky pravidlá** na vytvorenie nových faktov prostredníctvom cyklu volajúci funkciu expand, ktorý z daných pravidiel vytvorí nové fakty na základe daných faktov
    a. Expand obsahuje cyklus, ktorý **hľadá fakty** korešpondujúce s pravidlom
        i. Pre každý fakt sa vchádza do ďalšieho cyklu, ktorý má na starosti premenovať premenné (napr. ?X) menom z daných faktov prostredníctvom funkcie exchange. Táto funkcia premenuváva všetky premenné s takým menom, ktorý sa ako prvý vyskytuje v riadku podmienky. Keď sa spracuje, indexy (prostreníctvom ktorých sa prechádzajú znaky faktu a pravidla) sa posunú o veľkosť toho mena. Ak tam niesú mená, ale len spoločné znaky (ktoré tvoria vetu) tak sa posúvajú len o ten jeden znak.
        ii. Následne zistíme, či sa index (prostredníctvom ktorých sa prechádzali znaky faktu) faktu nachádza na konci pola znakov faktu (to znamená, že časť v zátvorkách podmienky je celý premenovaný menom.. [premenných je vždy rovný počtu mien vo fakte]), ak áno, nastávajú 2 možné prípady:
            1. Ak sa index podmienky nachádza na začiatku ďalšej nespracovanej podmienky (prostredníctvom kontroly znaku ‘(‘)), rekurzívne zavoláme funkciu Expand
            2. Ak sa index podmienky nenachádza na začiatku ďalšej nespracovanej podmienky, vložíme do spracovávaného faktu akciu podmienky, ktorú následne vrátime.
        iii. Ak nie, nastane kontrola so znakom ‘<’, čím začína špeciálna podmienka. Pozreme sa, či sú argumenty zhodné. Ak áno, vrátime taký fakt, ktorý sme dostali do funkcie Expand, teda nezmenený. Ak nie, vrátime akciu.
        iv. Ak tam nezačína špeciálna podmienka, znamená to, že potrebujeme prehľadať nasledujúci fakt, pretože sme nenašli korešpondujúci fakt s pravidlom (obsahuje inú vetu)
        v. Ak sme prehľadali všetky fakty, vrátime taký fakt, ktorý sme dostali do funkcie Expand, teda nezmenený

   b. Ďalej sa zavolá funkcia insertAction, ktorá pridá akciu do radu akcií, ktoré sa budú vykonávať (neduplicitne pomocou nasledujúceho kroku), takto nájdeme všetky akcie z aktuálnych faktov
5. Pri rade vykonávaných akcií je treba skontrolovať, či sme už danú akciu nevykonali (resp. sme ju už mali vo faktoch), teda či existuje v množine faktov skutočnosť, ktorú vykoná akcia. Ak tam existuje, nemá zmysel ju vykonávať, a tak ju zmažeme.
6. Skontrolujeme, či množina akcií ktorú je treba vykonať, je prázdna.
   a. Ak nieje, zavolá sa funkcia apply, ktorá zo stringu akcie v prípade pridania vytvorí string faktu a vráti ju do množiny faktov a v prípade vymazania vymaže fakt z množiny faktov
   b. Ak je, nemá čo nové vypisovať a tak program skončil
7. Vypíšu sa všetky zaktualizované fakty

3 Zhodnotenie

Implementácia bola zrealizovaná v programovacom jazyku C++. Je potrebné použiť kompilátor GNU GCC, prípadne je možné otvoriť projekt prostredníctvom CodeBlocks kompilátora.
Kód je intenzívne komentovaný pre ľahšie pochopenie.