Slovak University of Technology
Faculty of informatics and information Technology
Ilkovičova 2, 842 16 Bratislava 4

Artificial Intelligence

Assignment 4
Production System

Dopredný produkčný systém

Sultan Numyalai

Instructor: Ing. Ivan Kapustík
Study Field: INFO3
Year: 2. Bc
Academic Year: 2017/2018

Semster:Summer

Contents

1 Assignment

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 činnost i 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 Algorithm and Solution

The program contains two structures for the two different types which makes the data structure of the problem, FACTS , RULE.
FACTS has two parts name and pointer to the next fact
RULE has three parts , name, action and pointer to the next rule
1. reading facts from the file to the FACTS structure

2. Reading rules from the file to the RULE structure

3. we make the data-structure of the program in which in each iteration it writes the facts it finds out during that specified iteration

— 4.Then we come through all the rules for creating new facts using cycles, calling expand function, which creates new facts from the given rules based on the given facts and Expand contains a cycle which searches for the facts that are correspondent with the x^{th} rule.

a. For each fact the program inters to another cycle which is responsible to change the un-common variables like ?X with the name from the given facts using "exchange" function.

This function renames all the variable with that name which is the first string in the condition line. during these changes , the indexes which shows the characters of facts and rules shifts with the size of the name. if there's no name but just character which makes the sentence then just shifts about one character

- Then we find out whether the index through which we iterated the characters of the facts is at the end of the facts array (it means that this part is in the brackets of the condition and exchanged by the normal name Condition: if variables are to the number of names in the fact)

then we have two case scenarios:

1: if the index of the condition is at the first position of the next condition which is not performed still which we find out by the '()' control. Then we call the function Expand recursively.

2. if the index of the condition is not at the first position of the non-performed condition, then we insert to the performed fact of the condition of the action which is then returned.

- Else, we control the '<' character. Which starts the special condition. Then we check whether the arguments are identical. If yes: return the non-changed fact from Expand. Else: return action.
- If there's no special condition it means we have to search for next fact. Because we didn't find a fact correspondent with the rule (there's another sentence)
- If all the facts are searched, returns a fact which we got from Expand function (un-changed)

b. Then we call the function insertAction, which inserts the action to the series of actions or so called technically called linked list with a duplicate control in the next step so there is no duplicate action for a fact. This way we find all the action from the facts

5. for the series of actions we have to control whether the actual action has already been performed on a fact. (is it in facts or not). Which the action performs. If it is already there then there is no need to perform so we just delete it.

6. Controlling whether the set of actions which we have to perform is empty.

a. if it's not empty , call's the function apply which in case of insertion makes a string of fact from the string of action and return's it to the set of facts and in case of deletion delete's it from the set of facs.

b. if it's empty there's nothing to write or perform so the program ends.

7. Print's all the possible facts

Rules:

DruhyRodic1:
AK ((?X je rodic ?Y)(manzelia ?X ?Z))
POTOM ((pridaj ?Z je rodic ?Y))
DruhyRodic2:
AK ((?X je rodic ?Y)(manzelia ?Z ?X))
POTOM ((pridaj ?Z je rodic ?Y))
Otec:
AK ((?X je rodic ?Y)(muz ?X))
POTOM ((pridaj ?X je otec ?Y))
Matka:
AK ((?X je rodic ?Y)(zena ?X))
POTOM ((pridaj ?X je matka ?Y))
Surodenci:
AK ((?X je rodic ?Y)(?X je rodic ?Z)(<> ?Y ?Z))
POTOM ((pridaj ?Y a ?Z su surodenci))
Brat:
AK ((?Y a ?Z su surodenci)(muz ?Y))
POTOM ((pridaj ?Y je brat ?Z))
Stryko:
AK ((?Y je brat ?Z)(?Z je rodic ?X))
POTOM ((pridaj ?Y je stryko ?X)(sprava ?X ma stryka))
Test_mazania:
AK ((?Y je stryko ?X)(zena ?X))
POTOM ((vymaz zena ?X))

Initial Working Memory:

(Peter je rodic Jano)
(Peter je rodic Vlado)
(manzelia Peter Eva)
(Vlado je rodic Maria)
(Vlado je rodic Viera)
(muz Peter)
(muz Jano)
(muz Vlado)
(zena Maria)
(zena Viera)
(zena Eva)

3 Conclusion

“A production system (or production rule system) is a computer program typically used to provide some form of artificial intelligence, which consists primarily of a set of rules about behavior. These rules, termed productions, are a basic representation found useful in automated planning, expert systems and action selection. A production system provides the mechanism necessary to execute productions in order to achieve some goal for the system.”

The program is written in procedural programming language C, using 14 small and large functions and two structures FACTS and RULE. Linked list’s are used as data structures for the algorithm.

Two files at the very start of the program are used as the database and are inserted to the linked list. Which are then used for making more possible facts using the rules. And Modus ponens.

The source code is commented for better understanding

4 References

1. https://www.aaai.org/Papers/AAAI/1987/AAAI87-008.pdf
2. Newell A, Production Systems: Models of Control Structures, in *Visual* Information *Processing*, William G. Chase (ed.), pp 463-526, Academic Press, 1973.