Міністерство освіти і науки України

Тернопільський національний економічний університет

Факультет комп’ютерних інформаційних технологій

Кафедра КІ

Лабораторна робота №5 з дисципліни «Дослідження комп’ютерних систем штучного інтелекту»

Виконав:

Студент групи КСМм-51

Поважний Я.М.

Перевірив :

Вербовий С.О.

Тернопіль-2013

Тема: Представлення знань у мові Пролог

Мета: Вивчити можливості засобів мови програмування Пролог для представлення знань різними способами.

1.Теоретичні відомості

**Правила виводу**

Правила виводу (породжуючі правила, продукції), є системою представлення знань. Вони являють собою умовні вирази в різних формах.

**якщо** виконуються умова **то** висновок;

**якщо** ситуація **то** дія;

**якщо** виконуються умови 1 і умова 2 **то** умова 3 не виконується.

Правила виводу дозволяють надати пояснення послідовності міркувань та надати пояснення передумов конкретної дії.

Наприклад:

% Якщо індикатор “Power” світиться, то мереж. комутатор отримує живлення

if powerled\_on then switch\_is\_plugged\_in.

Першим кроком є формування бази знань. Якщо дано правило то спершу потрібно виділити об’єкти і їх властиіості, або якщо дано ситуацію то виділити стани. Наприклад

fact(driver\_isnt\_working\_properly)

Наступним кроком після формування бази знань потрібно сформувати процедури виводу висновку.

Зворотним логічним виводом називається такий стиль формування міркувань в ході його ланцюжок правил простежується у зворотному напрямку, від гіпотези (switch\_is\_plugged\_in) до спостережуваних фактів (powerled\_on.). Таку процедура відповідає власному, вбудованому стилю формування міркувань мови Prolog. Можна визначити правила бази знань як правила Prolog. При такій організації роботи база знань синтаксично не відрізняється від іншої програми.

Для реалізації правил використовуємо операторний запис мови Prolog. Створюємо логічні конструкції "if", "then", "and" й "or" як оператори, оголошені відповідним чином, як показано нижче. Спостережувані факти будуть вводитися як завдання для експертної системи за допомогою процедури fact.

% Простий інтерпретатор правил зворотного логічного виводу

% Оголошення операторів

:- op( 800, fx, if) .

:- op( 700, xfx, then).

:- op( 300, xfy, or).

:- op( 200, xfy, and).

% Спостережувані факти

fact(hall\_wet).

fact(bathroom\_dry).

fact(window\_closed).

% Правила

if hall\_wet and kitchen\_dry

then leak\_in\_bathroom.

if hall\_wet and bathroom\_dry

then problem\_in\_kitchen.

if window\_closed or no\_rain

then no\_water\_from\_outside.

% Інтерпретатор правил зворотного логічного виводу

% можна визначити як процедуру is\_true( Р)

is\_true( P):-

fact( P).

is\_true( P):-

if Condition then P, % Підходяще правило, умова якого є істинною

is\_true( Condition). %

is\_true( P1 and P2) :-

is\_true( P1) , is\_true( P2).

is\_true( P1 or P2) :-

is\_true( P1) ; is\_true( P2).

Тепер цей інтерпретатор можна викликати на виконання за допомогою такого питання:

?- is\_true( leak\_in\_kitchen).

yes

**Фрейми**

У поданні знань у вигляді фреймів факти асоціюються з об'єктами. У цьому контексті під "об'єктом" мається на увазі конкретний фізичний об'єкт або більш абстрактне поняття, таке як клас об'єктів або ситуація.

Фрейм— це структура даних, компоненти якої називаються *слотами.* Слоти позначаються іменами й застосовуються для розміщення в них інформації різного роду. Тому в слотах можна знайти елементарні значення, посилання на інші фрейми або процедури, за допомогою яких може бути обчислене значення слота. Слот може також залишатися незаповненим. Незаповнені слоти можуть заповнюватися із застосуванням логічного виводу. Широко застосовуваним принципом логічного виводу є наслідування. Якщо фрейм являє собою клас об'єктів (таких як альбатроси), а інший фрейм - суперклас цього класу (такий як птаха), то фрейм класу може успадковувати значення від фрейму суперкласу.

processor( socket\_type, socket).

processor( internal\_clock, internal).

processor( core\_amount, num).

processor( bus\_frequency, freq).

amdathlon( a\_kind\_of, processor).

amdathlon( family, amd\_athlon\_II\_X3).

amdathlon( socket\_type, socket\_am3).

amdathlon( internal\_clock, 3300).

amdathlon( core\_amount, 3).

amdathlon( bus\_frequency, 4000).

amdathlon( cache, kb1536).

amdfx( a\_kind\_of, processor).

amdfx( family, amd\_fxseries).

amdfx( socket, am3).

amdfx( internal\_clock, 3733).

amdfx( core\_amount, 4).

amdfx( bus\_frequency, 1800).

amdfx( cache, mb4).

value( Frame, Slot, Value) :-

Query =.. [Frame, Slot, Value],

call( Query), !.

value( Frame, Slot, Value) :-

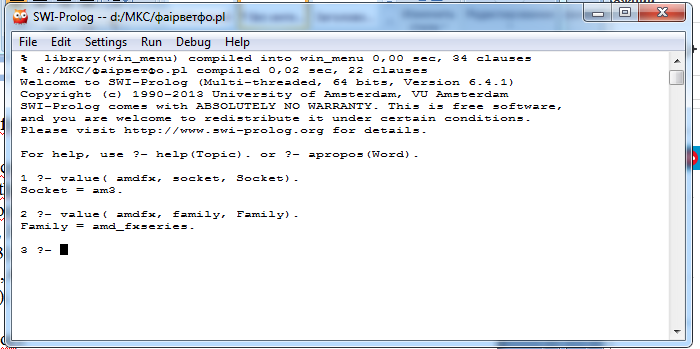
parent( Frame, ParentFrame),

value( ParentFrame, Slot, Value).

parent( Frame, ParentFrame) :-

Query =.. [ Frame, a\_kind\_of, ParentFrame],

call( Query).



Висновки.

Вивчив можливості засобів мови програмування Пролог для представлення знань різними способами.