Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение
Высшего образования
Факультет Программной Инженерии и Компьютерной Техники

Модуль 1 по СИИ

Группа: P3316
Выполнил:
Сиразетдинов А.Н.
Проверил:
Авдюшина А.Е

Г. Санкт-Петербург
2024

Оглавление

Введение.....3
Prolog.....4
Protégé.....8
Программа.....9

Введение

Модуль включает в себя выполнение трех заданий:

1) Требуется создать базу знаний в языке программирования Prolog и реализовать набор запросов, используя эту базу знаний. Задача направлена на развитие навыков работы с фактами, предикатами, и правилами в логическом программировании.
2) Целью этой лабораторной работы является знакомство со средой разработки онтологий Protege и перевод базы знаний, созданной в предыдущей лабораторной работе в онтологическую форму в Protege.
3) Целью этой лабораторной работы является разработка программы (рекомендательной системы), которая будет использовать базу знаний или онтологию для предоставления рекомендаций на основе введенных пользователем данных. (Knowledge-based support system)

Prolog

Разработаны правила факты и запросы:

```
% Факты описывают объекты (настольные игры) и их характеристики
настольная_игра(шахматы).
настольная_игра(монополия).
настольная_игра(каркассон).
настольная_игра(колонизаторы).
настольная_игра(домино).
настольная_игра(мафия).
настольная_игра(покер).
настольная_игра(рикен).
настольная_игра(дженга).
настольная_игра(скрабл).
настольная_игра(алиас).
настольная_игра(шашки).
настольная_игра(алхимики).
настольная_игра(чапаев).
настольная_игра(лото).
настольная_игра(элиас).
настольная_игра(карты).
настольная_игра(настольный_футбол).

% Факты о том, к какому типу относится игра
стратегическая(шахматы).
стратегическая(шашки).
стратегическая(каркассон).
стратегическая(колонизаторы).
стратегическая(алхимики).
стратегическая(рикен).
экономическая(монополия).
экономическая(колонизаторы).
карточная(покер).
карточная(мафия).
карточная(карты).
```

психологическая(мафия).

психологическая(покер).

психологическая(крокодил).

интерактивная(дженга).

интерактивная(домино).

интерактивная(скрабл).

интерактивная(чапаев).

интерактивная(лото).

интерактивная(элиас).

интерактивная(настольный_футбол).

% Категория возрастной группы

детская(домино).

детская(лото).

детская(дженга).

детская(чапаев).

детская(лото).

детская(настольный_футбол).

подростковая(карты).

подростковая(шахматы).

подростковая(шашки).

подростковая(мафия).

подростковая(колонизаторы).

подростковая(элиас).

подростковая(монополия).

подростковая(скрабл).

взрослая(покер).

взрослая(каркассон).

взрослая(рикен).

взрослая(алхимики).

% Факты о количестве игроков и возрасте

минимальное_количество_игроков(шахматы, 2).

минимальное_количество_игроков(монополия, 2).

минимальное_количество_игроков(каркассон, 2).

минимальное_количество_игроков(колонизаторы, 3).

минимальное_количество_игроков(мафия, 6).

минимальное_количество_игроков(домино, 3).

минимальное_количество_игроков(рикен, 3).

минимальное_количество_игроков(покер, 2).

минимальное_количество_игроков(элиас, 4).

минимальное_количество_игроков(шашки, 2).

минимальное_количество_игроков(алхимики, 3).

минимальное_количество_игроков(чапаев, 2).

минимальное_количество_игроков(лото, 3).

минимальное_количество_игроков(карты, 2).

минимальное_количество_игроков(дженга, 2).

минимальное_количество_игроков(скрабл, 2).

минимальное_количество_игроков(настольный_футбол, 2).

% Факты о продолжительности игры в минутах

продолжительность(шахматы, 40).

продолжительность(монополия, 120).

продолжительность(каркассон, 45).

продолжительность(колонизаторы, 90).

продолжительность(домино, 30).

продолжительность(мафия, 40).

продолжительность(рикен, 40).

продолжительность(покер, 30).

продолжительность(дженга, 20).

продолжительность(алиас, 10).

продолжительность(шашки, 15).

продолжительность(алхимики, 20).

продолжительность(чапаев, 10).

продолжительность(лото, 10).

продолжительность(карты, 10).

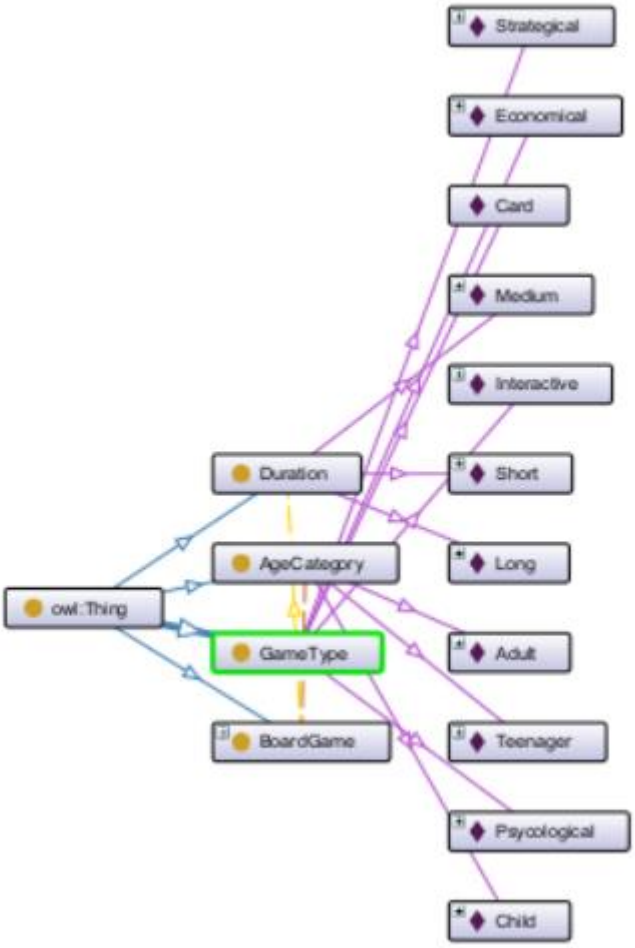
продолжительность(крокодил, 40).

продолжительность(настольный_футбол, 20).

```
% Правило 1: Игра подходит для детей, если она относится к детской категории
подходит_для_детей(Игра) :- детская(Игра).

% Правило 2: Игра длиннее 60 минут считается долгой
долгая_игра(Игра) :- продолжительность(Игра, Время), Время > 60.

% Правило 3: Игра подходит для двух игроков, если минимальное количество игроков — 2
подходит_для_двух(Игра) :- минимальное_количество_игроков(Игра, 2).

% Правило 4: Игра подходит нескучная если она интерактивная и занимает меньше 15 минут
нескучная(Игра) :- интерактивная(Игра), продолжительность(Игра, Время), Время < 15.

% Правило 5: Игра является психологической, если она относится к психологическим играм
психологическая_игра(Игра) :- психологическая(Игра).

% Запросы

% Запрос 1: стратегические игры
% стратегическая(Игра).

% Запрос 2: узнать продолжительность мафии
% продолжительность(мафия, Время).

% Запрос 3: найти стратегические или карточные игры
% стратегическая(Игра) ; карточная(Игра).

% Запрос 4: найти детские игры с продолжительностью до 30 минут
% подходит_для_детей(Игра), продолжительность(Игра, Время), Время =< 30.
% подходит_для_детей(Игра), \+ долгая_игра(Игра)

% Запрос 5: игры для двух подростков
% минимальное_количество_игроков(Игра, 2), возрастная_категория(Игра, подростковая).

% Запрос 6: нескучные детские интерактивные игры
% нескучная(Игра), интерактивная(Игра), детская(Игра)
```

Protégé

Разработаны классы и отношения



Программа

Разработана программа на языке Clojure

```
(ns my-clojure-java-project.core
  (:gen-class)
  (:import (connector Connector)))

(defmacro try-times
  "Retries expr for times times,
  then throws exception or returns evaluated value of expr"
  [times & expr]
  `(loop [err# (dec ~times)]
     (let [[result# no-retry#] (try [(do ~@expr) true]
                                    (catch Exception e#
                                      (when (zero? err#)
                                        (throw e#))
                                      [nil false]))]
       (if no-retry#
         result#
         (recur (dec err#))))))

(defn ask-user-age []
  (loop []
    (try-times 3 (prn "Сколько вам лет? (введите число)") (Integer/parseInt (read-line)))))

(defn ask-age-rule []
  (let [age (ask-user-age)]
    (cond
      (< age 12) "детская"
      (and (>= age 12) (< age 18)) "подростковая"
      :else "взрослая"
      )))

(def game-types #{:стратегическая
                  :экономическая
```

```
            :карточная
            :психологическая
            :интерактивная})

(defn ask-game-type []
  (println "Выберите тип игры который был бы вам более интересен")
  (doseq [type game-types]
    (println (name type)))
  (let [input (keyword (read-line))]
    (if (contains? game-types input)
      (name input)
      (do
        (prn "Такого нет в списке!")
        (recur)))))

(defn ask-player-count []
  (try-times 3 (prn "Какое у вас количество игроков? (введите число)") (Integer/parseInt (read-line))))

(defn ask-player-time []
  (try-times 3 (prn "Какая продолжительность игры (примерная) вас бы устроила?")
          (Integer/parseInt (read-line))))



(defn create-query-statement []
  (let [age-rule (ask-age-rule)
        game-type (ask-game-type)
        player-count (ask-player-count)]
    ;player-time (ask-player-time)]
    (str
      age-rule "(Игра), "
      game-type "(Игра), "
      "минимальное_количество_игроков(Игра, Количество), Количество =< " player-count ". "
      ;"продолжительность(Игра, Время), Время >= " (int (* 0.5 player-time))
      ;", Время =< " (int (* 1.5 player-time)) "."
      )))
```

```
;подростковая(Игра),
; стратегическая(Игра),
; минимальное_количество_игроков(Игра, Количество),
; Количество >= 2,
; продолжительность(Игра, Время),
; Время >= 10, Время =< 60. (15, 40)


(defn print-result [result]
  (let [name (.getTerm result "Игра")]
    (prn (str "Игра: " (.toString name)
              )))
  )



(defn recommendation-system []
  (let [result (Connector/executeQuery (create-query-statement))]
    (let [hasAnything (.next result)]
      (cond hasAnything
            (do
              (prn "По вашему запросу нашлись игры: ")
              (print-result result)
              (while (.next result)
                (print-result result)
                ))
            :else (prn "По вашему запросу ничего не нашлось(")))))

(defn -main
  [& args]
  (recommendation-system))
```