Министерство образования и науки Российской Федерации

Севастопольский государственный университет

Кафедра ИС

Отчет

По дисциплине: “Методы и системы искусственного интеллекта”

Лабораторная работа №4

“ЕЯ доступ к базе данных на основе алгоритма

сопоставления с образцом”

Выполнил:

ст.гр. ИС/б-17-2

Долженко И.А.

Проверил:

Забаштанский А.К.

Севастополь

2020

1 ЦЕЛЬ РАБОТЫ

Исследование алгоритма сопоставления с образцом и особенностей его применения для формирования запросов к базам данных, а также для организации доступа к базам данных на ограниченном подмножестве естественного языка.

2 ПОСТАНОВКА ЗАДАЧИ

Для базы данных, созданной в лабораторной работе 3, необходимо написать на языке Лисп интерфейс, который позволяет выполнять ЕЯ-запросы с помощью алгоритма сопоставления с образцом. Ответ на запрос должен также представляться на естественном языке в виде списка слов предложения. Кроме запроса, заданного по варианту задания, предусмотреть 5-6 различных дополнительных запросов.

3 ХОД РАБОТЫ

Код программы:

(defun main ()

(query '(Загрузить базу))

(format t "Добавить маршрут из Москва в Краснодар с номером 418~%")

(query '(Добавить маршрут из "Москва" в "Краснодар" с номером "418"))

(format t "Добавить маршрут из Краснодар в Симферополь с номером 8~%~%")

(query '(Добавить маршрут из "Краснодар" в "Симферополь" с номером "8"))

(format t "Какие маршруты идут в город Симферополь~%")

(selectQuery '(Какие маршруты идут в город "Симферополь"))

(format t "Найти маршрут номер 8~%")

(selectQuery '(Найти маршрут номер "8"))

(format t "Какой маршрут идет в Краснодар~%")

(selectQuery '(Какой маршрут идет в "Краснодар"))

(format t "Показать все маршруты")

(selectQuery '(Показать все маршруты))

(format t "Изменить маршруты с конечной станцией Симферополь на Омск~%")

(query '(Изменить маршруты с конечной станцией "Симферополь" на "Омск"))

(format t "Изменить начальный маршрут у 1 номера на город Красноярск~%~%")

(query '(Изменить начальный маршрут у "1" номера на город "Красноярск"))

(format t "Показать все маршруты")

(selectQuery '(Показать все маршруты))

)

(defvar \*db\* nil)

(defun insert (start end number)

(push (list :start start :end end :number number) \*db\*)

)

(defun savef (filename)

(with-open-file (out filename :direction :output :if-exists :supersede)

(with-standard-io-syntax

(print \*db\* out)

)

)

)

(defun loadf (filename)

(with-open-file (in filename)

(with-standard-io-syntax

(setf \*db\* (read in))

)

)

)

(defun select\* ()

(format t "~%")

(format t "~%~{~{~a:~a~%~}~%~}" \*db\*)

)

(defun where(&key start end number)

#'(lambda (row)

(and

(if start (equal (getf row :start) start) t)

(if end (equal (getf row :end) end) t)

(if number (equal (getf row :number) number) t)

)

)

)

(defun update (where-func &key start end number)

(setf \*db\*

(mapcar

#'(lambda (row)

(when (funcall where-func row)

(if start (setf (getf row :start) start))

(if end (setf (getf row :end) end))

(if number (setf (getf row :number) number))

)

row

)

\*db\*

)

)

)

(defun selectByNumber (\_number)

(if (eq nil (setf rows (remove-if-not #'(lambda (row) (equal (getf row :number) \_number)) \*db\*)))

(print "Такого маршрута нет!")

(print rows)

)

)

(defun match (p d)

(cond

;; правило 1

((and (null p) (null d)) t)

;; правило 2

((and (null d)

(eq (car p) '$)

(null (cdr p))) t)

;; один из списков исчерпан

((or (null p) (null d)) nil)

;; правило 3 и правило 4

((or (equal (car p) '?)

(equal (car p) (car d)))

(match (cdr p) (cdr d)))

;; правило 5 и 6

((eq (car p) '$)

(cond ((match (cdr p) d) t)

((match p (cdr d)) t)))

;; правило 7 - сопоставление списков, включающих подсписки

((and (not (atom (car p)))

(not (atom (car d)))

(match (car p) (car d)))

(match (cdr p) (cdr d)) )

;; правило 8 – подстановка значения в переменную

((and (atom (car p))

(eq (car-letter (car p)) #\?)

(match (cdr p)(cdr d)))

(set (cdr-name (car p)) (car d)) t)

;; правило 9 - подстановка сегмента значений в переменную

((and (atom (car p))

(eq (car-letter (car p)) #\$))

(cond ((match (cdr p)(cdr d))

(set (cdr-name (car p)) (list (car d)))

t)

((match p (cdr d))

(set (cdr-name (car p))

(cons (car d)(eval (cdr-name (car p)))))

t)))

;; правило 10 - обработка пакета ограничений, если в пакете есть «?»

((and (not(atom (car p)))

(eq (caar p) 'restrict)

(eq (cadar p) '?)

(and-to-list

(mapcar #'(lambda (pred)

(funcall pred (car d))) (cddar p))))

(match (cdr p)(cdr d)))

;; правило 11 - обработка пакета ограничений, если в пакете есть «?V»

;; например: (match '((restrict ?V integerp evenp) b c) '(36 b c))

((and (not (atom (car p)))

(not (atom d))

(eq (caar p) 'restrict)

(eq (car-letter (cadar p)) #\?)

(and-to-list

(mapcar #'(lambda (pred)

(funcall pred (car d))) (cddar p)))

(match (cdr p)(cdr d)))

(set (cdr-name (cadar p)) (car d))

t)

))

;;;; Вспомогательные функции

;; выделение первой литеры из имени

(defun car-letter (x) (if (not (numberp x)) (car (coerce (string x) 'list))))

;; возвращает имя без первой

(defun cdr-name (x)

(intern (coerce (cdr (coerce (string x) 'list)) 'string))

)

;; проверяет, все ли элементы списка lis имеют значение T

(defun and-to-list (lis) ;lis - список логических значений

(let ((res t))

(dolist (temp lis res)

(setq res (and res temp)))

)

)

(defun get-matches (p database)

(remove-if-not #'(lambda (record) (match p record)) database)

)

(defun query (q)

(cond

((match `($ загрузить $) q)

(loadf "database.txt"))

((match `($ сохранить $) q)

(savef "database.txt"))

((match `(Добавить $ из ?start в $ ?end $ с номером $ ?number) q)

(insert start end number))

((match `($ в город $ ?end) q)

(setf temp (get-matches `($ :end ,end $) \*db\*))

(if (null temp) "Маршрутов туда нет" temp))

((match `($ номер $ ?number) q)

(setf temp (get-matches `($ :number ,number $) \*db\*))

(if (null temp) "Маршрутов с таким номером нет" temp))

((match `($ идет в $ ?end) q)

(setf temp (get-matches `($ :end ,end $) \*db\*))

(if (null temp) "Отсутствуют маршруты с таким пунктом назначения" temp))

((match `($ все $) q)

(select\*))

((match `(Изменить $ конечной станцией ?end на ?endNew) q)

(update (where :end end) :end endNew))

((match `(Изменить начальный маршрут у ?number номера на город ?start) q)

(update (where :number number) :start start))

)

)

(defun selectQuery (q)

(setf temp (query q))

(if (listp temp)

(format t "~%~{~{~a:~a~%~}~%~}" temp)

(format t "~%" temp)

)

)

Результат работы программы:

Содержимое файла до выполнения запросов – ((:START "Москва" :END "Симферополь" :NUMBER "99") (:START "Евпатория" :END "Санкт-Петербург" :NUMBER "2") (:START "Севастополь" :END "Москва" :NUMBER "1"))

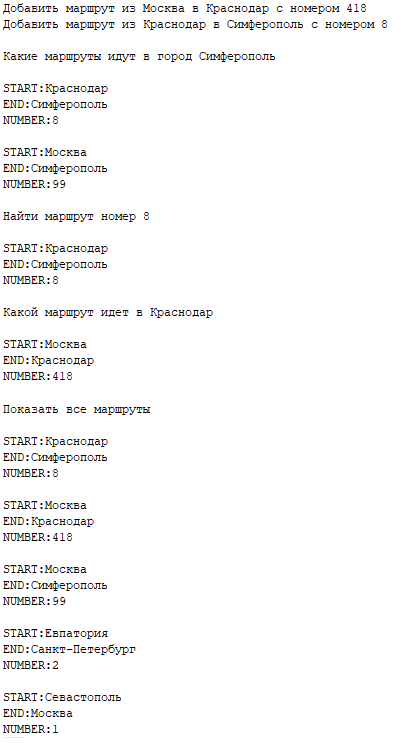


Рисунок 1 – Результат выполнения программы. Часть 1

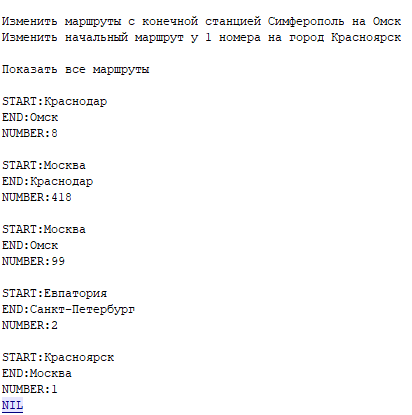


Рисунок 2 – Результат выполнения программы. Часть 2

ВЫВОДЫ

В ходе выполнения лабораторной работы был исследован алгоритм сопоставления с образцом и особенности его применения для формирования запросов к базам данных, а также для организации доступа к базам данных на ограниченном подмножестве естественного языка.