|  |
| --- |
| Wydział Matematyki i Nauk Informacyjnych |
| Sprawozdanie |
| Identyfikacja chmur |
|  |
| **Łukasz Dragan** |
| **2016-01-10** |

# Kod programu

Kod programu dzieli się na trzy pliki:

1. main.lisp: plik startowy zarządzający interfejsem użytkownika i wywołujący odpowiednie funkcje używane do rozpoznawania chmur i operacji na bazie.

(load "lib.lisp")

(setq data (list))

(load "db.lisp")

(setq q "")

(defun BODY ()

(setq tree (make-tree Nil data))

(princ tree)

(print (test tree))

(print "q to quit")

(print "n to new rule")

(print "a to ask again")

(setq q (read))

(cond ((string-equal q "N")

(print "new rule:")

(save\_rule)

(setq data (list))

(load "db.lisp")

)

)

)

(loop while (not (string-equal q "Q"))

do(BODY)

)

1. db.lisp: dynamiczna baza wiedzy. Plik zawiera reguły, wg których system, zadając pytania, rozpoznaje chmurę.

(rule CUMULONIMBUS => (PADA\_DESZCZ) (ZASLANIA\_CALE\_NIEBO) )

(rule NIMBOSTRATUS => (PADA\_DESZCZ ZASLANIA\_CALE\_NIEBO) () )

(rule CUMULUS => (JEST\_NISKO) (PADA\_DESZCZ ZASLANIA\_CALE\_NIEBO) )

(rule STRATUS => (JEST\_NISKO ZASLANIA\_CALE\_NIEBO JEST\_ZWARTA) (PADA\_DESZCZ) )

(rule STRATOCUMULUS => (JEST\_NISKO ZASLANIA\_CALE\_NIEBO) (PADA\_DESZCZ JEST\_ZWARTA))

(rule ALTOSTRATUS => (JEST\_GLADKA POKRYWA\_WIEKSZOSC\_NIEBA) (PADA\_DESZCZ JEST\_NISKO))

(rule CIRROSTRATUS => (JEST\_GLADKA JEST\_ZWARTA) (PADA\_DESZCZ JEST\_NISKO POKRYWA\_WIEKSZOSC\_NIEBA))

(rule CIRRUS => (JEST\_GLADKA) (PADA\_DESZCZ JEST\_NISKO POKRYWA\_WIEKSZOSC\_NIEBA JEST\_ZWARTA))

(rule ALTOCUMULUS => (POKRYWA\_WIEKSZOSC\_NIEBA) (PADA\_DESZCZ JEST\_NISKO JEST\_GLADKA))

1. lib.lisp: zestaw funkcji i makr odpowiedzialnych za interakcję z bazą wiedzy i wnioskowanie.

(defmacro rule (\_name => \_listT \_listN)

`(setq data (append data (list(list ',\_name ',\_listT ',\_listN))))

)

(defmacro many-rules (&rest rules)

(cond ((null (rest rules))

`(rule ,@(first rules)))

(T `(many-rules ,@(rest rules)))))

(defun node-name (node) (car node))

(defun node-pos (node) (car (cdr node)))

(defun node-neg (node) (car (cdr (cdr node))))

(defun contains (\_elem \_list)

(cond

((null \_list) Nil)

((null (car \_list)) Nil)

((string-equal (car \_list) \_elem) T)

(T (contains \_elem (cdr \_list)))

)

)

(defun remove-str (\_elem \_list)

(cond

((null \_list) Nil)

((null (car \_list)) Nil)

((string-equal (car \_list) \_elem) (remove-str \_elem (cdr \_list)))

(T (append (list (car \_list)) (remove-str \_elem (cdr \_list))))

)

)

(defun add\_elem (\_node \_name \_pos \_neg)

(cond

((null \_node)

(cond

((and (null (car \_pos)) (null (car \_neg))) (list Nil \_name))

(T (cond

((null (car \_pos)) (list (car \_neg) Nil (add\_elem Nil \_name \_pos (remove-str (car \_neg) \_neg))))

(T (list (car \_pos) (add\_elem Nil \_name (remove-str (car \_pos) \_pos) \_neg) Nil))

)

)

)

)

((null (car \_node)) \_node)

((null (contains (car \_node) \_pos))

(list (node-name \_node) (node-pos \_node) (add\_elem (node-neg \_node) \_name \_pos (remove-str (car \_node) \_neg ))))

((null (contains (car \_node) \_neg))

(list (node-name \_node) (add\_elem (node-pos \_node) \_name (remove-str (car \_node) \_pos ) \_neg) (node-neg \_node)))

(T \_node)

)

)

(defun make-tree (\_tree \_data)

(cond ((null \_data) \_tree)

(T (make-tree (add\_elem \_tree (node-name (car \_data)) (node-pos (car \_data)) (node-neg (car \_data)))(cdr \_data)))

)

)

(defun test (node)

(cond

((null node) Nil)

((null (car node)) (car (cdr node)))

(T (print (car node)) (setq ans (read))

(cond

((string-equal "Y" ans) (test (car (cdr node))) )

(T (test (car (cdr (cdr node)))))

))

)

)

(defun save\_rule ()

(setq suffix (read))

(print suffix)

(with-open-file

(str "db.lisp" :direction :output :if-exists :append)

(format str "~%~a~%" suffix))

)

Opis funkcji i makr:

* 1. node-name pobiera nazwę właściwości zapisanej w węźle drzewa decyzyjnego
  2. node-pos pobiera węzeł, do którego prowadzi pozytywna wartość właściwości węzła
  3. node-neg pobiera węzeł, do którego prowadzi negatywna wartość właściwości węzła
  4. contains sprawdza, czy lista stringów zawiera dany string
  5. remove-str usuwa daną wartość z listy stringów
  6. add\_elem dodaje do drzewa nowy element na podstawie reguły pobranej z bazy wiedzy
  7. make-tree tworzy drzewo decyzyjne na podstawie bazy wiedzy
  8. test wykonuje przejście po zbudowanym wcześniej drzewie na podstawie odpowiedzi użytkownika, czy właściwość odpowiadająca węzłowi jest prawdziwa lub nie
  9. save\_rule zapisuje podaną regułę w bazie wiedzy
  10. rule dodaje informację o zjawisku do listy reguł, która może zostać przekształcona do drzewa
  11. many-rules zapisuje wiele reguł na raz

# Funkcjonalności i przykładowe wywołania

W skład wchodzą dwie funkcjonalności:

1. Rozpoznanie chmury.

Następuje zadanie przez system serii pytań niezbędnych do rozpoznania chmury. Możliwe odpowiedzi na pytania to: „y.” lub „n.”. Jeżeli w bazie wiedzy znajduje się chmura opisana odpowiedziami na zadane pytanie, system wypisze nazwę chmury. W przeciwnym razie system wypisze NIL

1. Rozszerzanie bazy wiedzy.

Za pomocą wyboru ‘n’ można rozszerzać bazę wiedzy programu. System oczekuje wtedy na podanie reguł zapisanych na dwa możliwe sposoby: za pomocą makra RULE lub makra MANY-RULES. Rekurencyjne makro MANY-RULES pozwala określić wiele reguł na raz. Do pliku db.lisp zostaje dopisana odpowiednia reguła. Pierwszy argument to nazwa zjawiska. Drugi: zestaw faktów które muszą zajść, aby zaistniało dane zjawisko. Trzeci: zestaw faktów, które nie mogą zająć, aby zjawisko zaistniało.

Przykładowe wywołanie:

(PADA\_DESZCZ (ZASLANIA\_CALE\_NIEBO (NIL NIMBOSTRATUS) (NIL CUMULONIMBUS))

(JEST\_NISKO (ZASLANIA\_CALE\_NIEBO (JEST\_ZWARTA (NIL STRATUS) (NIL STRATOCUMULUS)) (NIL CUMULUS))

(JEST\_GLADKA

(POKRYWA\_WIEKSZOSC\_NIEBA (NIL ALTOSTRATUS) (JEST\_ZWARTA (NIL CIRROSTRATUS) (NIL CIRRUS)))

(POKRYWA\_WIEKSZOSC\_NIEBA (NIL ALTOCUMULUS) NIL))))

;System wypisał zbudowane drzewo decyzyjne

PADA\_DESZCZ Y

ZASLANIA\_CALE\_NIEBO y

NIMBOSTRATUS

;System stwierdził, że szukaną chmurą jest nimbostratus

"q to quit"

"n to new rule"

"a to ask again" n

;Wybrano opcję rozszerzenia bazy

"new rule:" (RULE CIRROCUMULUS => NIL (PADA\_DESZCZ JEST\_NISKO JEST\_GLADKA POKRYWA\_WIEKSZOSC\_NIEBA))

(RULE CIRROCUMULUS => NIL (PADA\_DESZCZ JEST\_NISKO JEST\_GLADKA POKRYWA\_WIEKSZOSC\_NIEBA))

(PADA\_DESZCZ (ZASLANIA\_CALE\_NIEBO (NIL NIMBOSTRATUS) (NIL CUMULONIMBUS))

(JEST\_NISKO (ZASLANIA\_CALE\_NIEBO (JEST\_ZWARTA (NIL STRATUS) (NIL STRATOCUMULUS)) (NIL CUMULUS))

(JEST\_GLADKA

(POKRYWA\_WIEKSZOSC\_NIEBA (NIL ALTOSTRATUS) (JEST\_ZWARTA (NIL CIRROSTRATUS) (NIL CIRRUS)))

(POKRYWA\_WIEKSZOSC\_NIEBA (NIL ALTOCUMULUS) (NIL CIRROCUMULUS)))))

;System wypisał nowe drzewo decyzyjne

PADA\_DESZCZ N

JEST\_NISKO N

JEST\_GLADKA N

POKRYWA\_WIEKSZOSC\_NIEBA N

CIRROCUMULUS

;System stwierdził, że szukaną chmurą jest nimbostratus

"q to quit"

"n to new rule"

"a to ask again" q