МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

«Тверской государственный технический университет»

(ТвГТУ)

Кафедра «Программного обеспечения»

**Отчёт по лабораторной работе №4**

**Вариант 1**

по дисциплине “Системы искусственного интеллекта”

Выполнил: студент группы

ПИН-17.06

Иванов Р. В.

Проверил:

Мальков А. А.

Тверь 2021

**Цель работы** – реализация в среде CLIPS задачи поиска в пространстве состояний и анализ ее решения.

**Постановка задачи:**

Одной из классических задач ИИ, рассматриваемых при построении и анализе алгоритмов поиска, является известная головоломка о фермере, которому необходимо переправить на другой берег реки лису, козу и капусту. Он располагает двуместной лодкой, т.е. может перевозить только по одному объекту. При этом нельзя оставлять на берегу лису с козой и козу с капустой, т. к. в этом случае первая из них съест вторую.

**Листинг:**

(defmodule MAIN

(export deftemplate ?ALL))

(deftemplate MAIN::node

(slot farmer-loc (type SYMBOL) (allowed-symbols shore-1 shore-2))

(slot fox-loc (type SYMBOL) (allowed-symbols shore-1 shore-2))

(slot goat-loc (type SYMBOL) (allowed-symbols shore-1 shore-2))

(slot cab-loc (type SYMBOL) (allowed-symbols shore-1 shore-2))

(slot s-depth (type INTEGER) (range 1 ?VARIABLE))

(slot parent (type FACT-ADDRESS SYMBOL) (allowed-symbols no-parent))

(slot last-move (type SYMBOL) (allowed-symbols alone goat fox cab no-move)))

(deffacts MAIN::init

(node

(s-depth 1)

(parent no-parent)

(farmer-loc shore-1)

(fox-loc shore-1)

(goat-loc shore-1)

(last-move no-move)))

(deffacts opposites

(opposite-of shore-1 shore-2)

(opposite-of shore-2 shore-1))

;---------------------------------------------------;

(defrule MAIN::move-alone

?nd<-(node

(s-depth ?num)

(farmer-loc ?ps))

(opposite-of ?ps ?ns)

=>

(duplicate ?nd

(s-depth(+ 1 ?num))

(farmer-loc ?ns)

(parent ?nd)

(last-move alone)))

;---------------------------------------------------;

(defrule MAIN::move-fox

?nd<-(node

(s-depth ?num)

(fox-loc ?ps)

(farmer-loc ?ps))

(opposite-of ?ps ?ns)

=>

(duplicate ?nd

(s-depth(+ 1 ?num))

(farmer-loc ?ns)

(fox-loc ?ns)

(parent ?nd)

(last-move fox)))

;---------------------------------------------------;

(defrule MAIN::move-goat

?nd<-(node

(s-depth ?num)

(goat-loc ?ps)

(farmer-loc ?ps))

(opposite-of ?ps ?ns)

=>

(duplicate ?nd

(s-depth(+ 1 ?num))

(farmer-loc ?ns)

(goat-loc ?ns)

(parent ?nd)

(last-move goat)))

;---------------------------------------------------;

(defrule MAIN::move-cab

?nd<-(node

(s-depth ?num)

(cab-loc ?ps)

(farmer-loc ?ps))

(opposite-of ?ps ?ns)

=>

(duplicate ?nd

(s-depth(+ 1 ?num))

(farmer-loc ?ns)

(cab-loc ?ns)

(parent ?nd)

(last-move cab)))

;---------------------------------------------------;

;---------------------------------------------------;

;---------------------------------------------------;

(defmodule CONSTRAINS

(import MAIN deftemplate ?ALL))

;---------------------------------------------------;

(defrule CONSTRAINS::goat-eaten

(declare (auto-focus TRUE))

?nd<-(node

(farmer-loc ?s1)

(fox-loc ?s2&~?s1)

(goat-loc ?s2))

=>

(retract ?nd))

;---------------------------------------------------;

(defrule CONSTRAINS::cab-eaten

(declare (auto-focus TRUE))

?nd<-(node

(farmer-loc ?s1)

(cab-loc ?s2&~?s1)

(goat-loc ?s2))

=>

(retract ?nd))

;---------------------------------------------------;

(defrule CONSTRAINS::loop

(declare (auto-focus TRUE))

(node

(farmer-loc ?fl)

(fox-loc ?fol)

(goat-loc ?gl)

(cab-loc ?cl)

(s-depth ?d))

?nd<-(node

(farmer-loc ?fl)

(fox-loc ?fol)

(goat-loc ?gl)

(cab-loc ?cl)

(s-depth ?d2&:(< ?d ?d2)))

=>

(retract ?nd))

;---------------------------------------------------;

;---------------------------------------------------;

;---------------------------------------------------;

(defmodule SOLUTION

(import MAIN deftemplate node))

(deftemplate SOLUTION::moves

(slot id (type FACT-ADDRESS SYMBOL) (allowed-symbols no-parent))

(multislot moves-list (type SYMBOL) (allowed-symbols no-move alone fox goat cabbage)))

;---------------------------------------------------;

(defrule SOLUTION::goal-test

(declare (auto-focus TRUE))

?nd <- (node (parent ?parent)

(farmer-loc shore-2)

(fox-loc shore-2)

(goat-loc shore-2)

(cab-loc shore-2)

(last-move ?move))

=>

(printout t "Solution found (see facts window)"))

;(assert (moves (id ?parent) (moves-list ?move))))

;---------------------------------------------------;

**Результат выполнения программы:**

