МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

«Тверской государственный технический университет»

(ТвГТУ)

Кафедра «Программного обеспечения»

**Отчёт по лабораторной работе №4**

по дисциплине “Системы искусственного интеллекта”

Выполнил: студент группы

ПИН-17.06

Завгороднев Е.Ю

Проверил:

Мальков А. А.

Тверь 2021

# Задача

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.

1. Построить полное дерево поиска для данной задачи.

2. Разработать, используя среду CLIPS, программу решения данной головоломки. Программа должна быть построена по модульному принципу и состоять из трех модулей:

· основного (MAIN);

· контроля ограничений (CONSTRAINTS);

· вывода решения (SOLUTION).

Для объявления модуля используется конструкция defmodule, в которой указываются экпортируемые в другие модули или экспортируемые из других модулей конструкции. Например модуль MAIN экспортирует шаблон status:

(defmodule MAIN

(export deftemplate status))

2.1. Модуль MAIN должен содержать:

· объявление шаблона состояния status;

· определение факта исходного состояния – initial-positions;

· определение факта отношения между берегами – opposites;

· определение правил генерации пути, соответствующих четырем операторам в пространстве состояний.

Имена всех конструкций модуля MAIN должны начинаться с префикса MAIN::. Например:

(deftemplate MAIN::status

. . .

)

2.2. Модуль контроля ограничений CONSTRAINTS должен импортировать из модуля MAIN шаблон status:

(defmodule CONSTRAINTS

(import MAIN deftemplate status))

и содержать:

· два правила для распознавания запрещенных ситуаций fox-eats-goat и goat-eats-cabbage;

· правило для распознавания зацикливания пути – circular-path;

Имена всех конструкций модуля CONSTRAINTS должны начинаться с префикса CONSTRAINTS::. Например:

(defrule CONSTRAINTS::goat-eats-cabbage

. . .

)

У всех правил модуля CONSTRAINTS должно быть установлено свойство автофокусировки. Это делается так:

(defrule CONSTRAINTS::fox-eats-goat

(declare (auto-focus TRUE))

. . .

Если свойство автофокусировки правила установлено, то всякий раз при активации правила автоматически выполняется команда фокусировки на модуле, в котором определено данное правило.

2.3. Модуль вывода решения SOLUTION также должен импортировать из модуля MAIN шаблон status:

(defmodule SOLUTION

(import MAIN deftemplate status))

и содержать:

· объявление шаблона факта-решения moves;

· правило распознавания целевого состояния goal-test;

· правило построения пути-решения – build-solution;

· правило вывода решения на экран – print-solution.

Имена всех конструкций модуля SOLUTION должны начинаться с префикса SOLUTION::. Например:

(defrule SOLUTION::print-solution

. . .

)

У правила распознавания целевого состояния должно быть установлено свойство автофокусировки:

(defrule SOLUTION:: goal-test

(declare (auto-focus TRUE))

. . .

3. Выполните программу в пошаговом режиме, проанализируйте и объясните ход поиска решения. В отчете необходимо привести трассу поиска решения.

# Листинг

(defmodule MAIN ; основной модуль

(export deftemplate ?ALL))

(deftemplate MAIN::node ; определяем набор фактов

(slot farmer-loc (type SYMBOL) (allowed-symbols shore-1 shore-2)) ; фермер и слоты для его местоположения

(slot fox-loc (type SYMBOL) (allowed-symbols shore-1 shore-2)) ; лис и слоты для его местоположения

(slot goat-loc (type SYMBOL) (allowed-symbols shore-1 shore-2)) ; коза и слоты для его местоположения

(slot cab-loc (type SYMBOL) (allowed-symbols shore-1 shore-2)) ; капуста и слоты для его местоположения

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

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

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

(deffacts MAIN::init ; инициализация изначальных позиций

(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 ; перемещение с одного берега на другой одного фермера

?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 ; перемещение фермера с лисой

?nd<-(node

(s-depth ?num) ; фиксация адреса текущей вершины и ее глубины

(fox-loc ?ps) ; лиса на том же берегу, что и крестьянина

(farmer-loc ?ps)) ; фиксация текущего местонахождения фермера

(opposite-of ?ps ?ns) ; связывание значения противоположного берега

=>

(duplicate ?nd ; создать новую вершину дублированием

(s-depth(+ 1 ?num)) ; установить ее глубину инкрементом текущей

(farmer-loc ?ns) ; установить новое местонахождение фермера

(fox-loc ?ns) ; установить новое местонахождение лисы

(parent ?nd) ; установить в качестве родительской вершины текущую

(last-move fox))) ; установить тип последнего перемещения

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

(defrule MAIN::move-goat ; перемещение фермера с козой

?nd<-(node

(s-depth ?num) ; фиксация адреса текущей вершины и ее глубины

(goat-loc ?ps) ; коза на том же берегу, что и крестьянина

(farmer-loc ?ps)) ; фиксация текущего местонахождения фермера

(opposite-of ?ps ?ns) ; связывание значения противоположного берега

=>

(duplicate ?nd ; создать новую вершину дублированием

(s-depth(+ 1 ?num)) ; установить ее глубину инкрементом текущей

(farmer-loc ?ns) ; установить новое местонахождение фермера

(goat-loc ?ns) ; установить новое местонахождение козы

(parent ?nd) ; установить в качестве родительской вершины текущую

(last-move goat))) ; установить тип последнего перемещения

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

(defrule MAIN::move-cab ; перемещение фермера с капустой

?nd<-(node

(s-depth ?num) ; фиксация адреса текущей вершины и ее глубины

(cab-loc ?ps) ; капуста на том же берегу, что и крестьянина

(farmer-loc ?ps)) ; фиксация текущего местонахождения фермера

(opposite-of ?ps ?ns) ; связывание значения противоположного берега

=>

(duplicate ?nd ; создать новую вершину дублированием

(s-depth(+ 1 ?num)) ; установить ее глубину инкрементом текущей

(farmer-loc ?ns) ; установить новое местонахождение фермера

(cab-loc ?ns) ; установить новое местонахождение капусты

(parent ?nd) ; установить в качестве родительской вершины текущую

(last-move cab))) ; установить тип последнего перемещения

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

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

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

(defmodule CONSTRAINS

(import MAIN deftemplate ?ALL))

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

(defrule CONSTRAINS::goat-eaten

(declare (auto-focus TRUE))

?nd<-(node

(farmer-loc ?s1) ; фиксируется адрес вершины и положение фермера

(fox-loc ?s2&~?s1) ; лиса и фермер на разных берегах

(goat-loc ?s2)) ; коза на том же береге , что и лиса

=>

(retract ?nd)) ; удалить вершину

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

(defrule CONSTRAINS::cab-eaten

(declare (auto-focus TRUE))

?nd<-(node

(farmer-loc ?s1) ; фиксируется адрес вершины и положение фермера

(cab-loc ?s2&~?s1) ; капуста и фермер на разных берегах

(goat-loc ?s2)) ; коза на том же береге , что и капуста

=>

(retract ?nd)) ; удалить вершину

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

; Распознавание ситуации зацикливания процесса поиска, т.е. повторного ;попадания в уже пройденное состояние. Для этого новое состояние должно ;сравниваться с ранее достигнутыми. Если имеется состояние с меньшей ;глубиной и точно таким же местоположением всех персонажей, то новая ;вершина должна удаляться.

(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



