МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

«Тверской государственный технический университет»

(ТвГТУ)

Кафедра «Программного обеспечения»

**Отчёт по лабораторной работе №6**

по дисциплине “Системы искусственного интеллекта”

Выполнил: студент группы

ПИН-17.06

Завгороднев Е.Ю

Проверил:

Мальков А. А.

Тверь 2021

# Задача

Задача построения башни из блоков (кубиков) заключается в последовательном выборе из неупорядоченной кучи блоков и постановки их друг на друга. Процесс решения задачи представляет собой чередование двух фаз: выбора блока из кучи и установки его в башню. При этом больший по размерам блок не может ставиться на меньший по размерам и, следовательно, на каждом шаге решения задачи необходимо выбирать из кучи самый большой блок.

Каждый блок характеризуется размером, цветом и положением, поэтому для его описания можно использовать следующий шаблон:

(deftemplate block

(slot size (type INTEGER))

(slot place (type SYMBOL))

(slot color (type SYMBOL))

)

Каждый блок может находиться либо в куче (heap), либо в руке (hand), когда он выбран из кучи, либо в башне (tower).

Результатом решения задачи должно быть взаимное положение блоков в башне, т. е. при выводе результата необходимо указать порядок блоков в башне по цвету. С этой целью для блоков, находящихся в башне, необходимо определить отношение, указывающее, какой блок (upper) на каком (lower) стоит:

(deftemplate on

(slot upper (type SYMBOL))

(slot lower (type SYMBOL))

(slot place (type SYMBOL)(default heap))

)

Если блок ставится в основание башни, слоту place присваивается значение tower, а слоты upper и lower остаются неопределенными. Если блок ставится на другой блок, слотам upper и lower должны присваиваться значения цветов соответствующих блоков.

Поскольку процесс решения задачи представляет собой чередование фаз поиска в куче самого большого блока (find) и его установки в башню (build), необходимо иметь факт, определяющий текущую задачу. Его можно определить с помощью следующего шаблона:

(deftemplate goal

(slot current-task (type SYMBOL))

)

Порядок выполнения работы

1. Разработать программу на языке CLIPS, реализующую описанные выше правила, в соответствии с вариантом задания указанным в таблице (см. ниже).

Исходное состояние должно описываться конструкцией deffacts, содержащей четыре факта, соответствующих блокам в куче.

2. Вывод результатов на экран должен обеспечивать пошаговое отображение процесса построения башни.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 8 | белый/16 | красный/5 | коричневый/15 | синий/10 |

# Листинг

(defglobal ?\*in-heap\* = 4)

(deftemplate block

(slot color (type SYMBOL))

(slot size (type INTEGER))

(slot place (type SYMBOL))

)

(deftemplate goal

(slot found (type SYMBOL))

)

(deftemplate on-block

(slot color (type SYMBOL))

(slot up-block (type SYMBOL))

(slot down-block (type SYMBOL))

)

(deftemplate task

(slot current (type SYMBOL)(allowed-symbols find build))

)

(deffacts init

(block

(color red)

(size 5)

(place heap)

)

(block

(color brown)

(size 15)

(place heap)

)

(block

(color blue)

(size 10)

(place heap)

)

(block

(color white)

(size 16)

(place heap)

)

)

(defrule init-r

(initial-fact)

=>

(assert

(task

(current find)

)

)

)

(defrule find-biggest

?tf<-(task

(current find)

)

(test (> ?\*in-heap\* 0))

?pbl<-(block

(size ?sz0)

(place heap)

)

(not (exists (block (place heap)(size ?sz1&:(> ?sz1 ?sz0))) ) )

=>

(modify ?pbl

(place hand)

)

(bind ?\*in-heap\* (- ?\*in-heap\* 1))

(modify ?tf

(current build)

)

)

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

(defrule build-first

?tf<-( task

(current build)

)

?fbl<-(block

(place hand)

(color ?cl)

)

(not (exists (on-block (up-block undefined))))

=>

(assert (on-block

(color ?cl)

(down-block undefined)

(up-block undefined)

)

)

(modify ?tf

(current find)

)

(modify ?fbl

(place tower)

)

)

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

(defrule build-next

?tf<-( task

(current build)

)

?fbl<-(block

(place hand)

(color ?cl)

)

?onf<-(on-block

(up-block undefined)

(color ?cl-old)

)

=>

(assert (on-block

(color ?cl)

(down-block ?cl-old)

(up-block undefined)

)

)

(modify ?tf

(current find)

)

(modify ?fbl

(place tower)

)

(modify ?onf

(up-block ?cl)

)

)

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

(defrule goal-test

(not (exists (block (place heap))))

=>

(printout t "Built! Print from top to bottom" crlf)

(assert (goal (found done)))

)

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

(defrule print-tower

(exists (goal (found done)))

?bl<-(on-block

(color ?cl)

(up-block undefined)

(down-block ?cl-down)

)

?bl-under<-(on-block

(color ?cl-down)

)

=>

(printout t "Block: "?cl crlf)

(retract ?bl)

(modify ?bl-under

(up-block undefined)

)

)

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

(defrule print-last

?bl<-(on-block

(up-block undefined)

(down-block undefined)

(color ?cl)

)

=>

(printout t "Block: "?cl crlf)

(retract ?bl)

)

# Результат

