Artificial Intelligence: A Modern Approach (2nd Edition)

Chapter 2, Problem 7E | 2 Bookmarks | Show all steps: ON

Problem

Implement a performance-measuring environment simulator for the vacuum-cleaner world depicted in Figure 2.2 and specified on page 38. Your implementation should be modular so that the sensors, actuators, and environment characteristics (size, shape, dirt placement, etc.) can be changed easily. (*Note:* for some choices of programming language and operating system there are already implementations in the online code repository.)

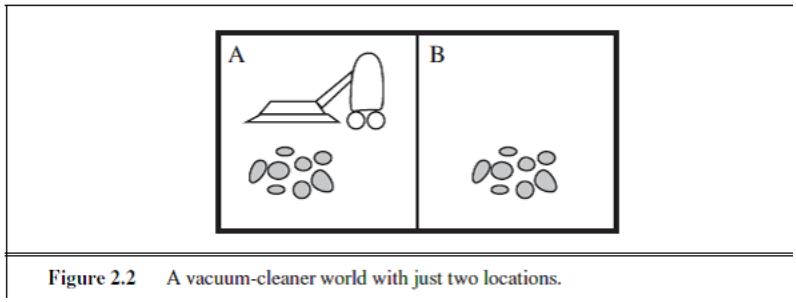

Figure 2.2 A vacuum-cleaner world with just two locations.

Step-by-step solution

Step 1 of 3

In the case of performance measure of a vacuum cleaner, test the amount of dirt cleaned, the amount of time taken to clean, the amount of electricity consumed, and the amount of noise generated. The environment is the agent's design. It has three characteristics – size, shape, and dirt placement. It is easy to change dirt placement and choose the size of the brush, experiment by changing the shape, size, and dirt placement.

Comment

Step 2 of 3

The vacuum world-cleaning up the dirt in grid: Implementation in Lisp

;;; -*- Mode: Lisp; Syntax: Common-Lisp; -*-

;;;; **The Vacuum World: cleaning up dirt in a grid**

```
(defstructure (dirt (:include object (name "*")
(size 0.001))))
(defstructure (vacuum-world (:include grid-environment
(size (@ 7 7))
(aspec '(random-vacuum-agent))
(cspec '((at all (P 0.25 dirt))))))
```

```
(defmethod performance-measure ((env vacuum-world) agent)
"100 points for each piece of dirt vacuumed up, -1 point for each step taken, and -1000 points if the agent does not return home."
(- (* 100 (count-if #'dirt-p (object-contents (agent-body agent))))
(environment-step env)
(if (equal (object-loc (agent-body agent))
(Grid-environment-start env))
0
1000)))
(defmethod get-percept ((env vacuum-world) agent)
"Percept is a three-element sequence: bump, dirt and home."
(let ((loc (object-loc (agent-body agent))))
(list (if (object-bump (agent-body agent)) 'bump)
(if (find-object-if #'dirt-p loc env) 'dirt)
(if (equal loc (grid-environment-start env)) 'home))))
(defmethod legal-actions ((env vacuum-world))
'(suck forward turn shut-off))
;;;; Actions (other than the basic grid actions of forward and turn)
(defmethod suck ((env vacuum-world) agent-body)
(let ((dirt (find-object-if #'dirt-p (object-loc agent-
body) env)))
(when dirt(place-in-container dirt agent-body env))))
(defmethod shut-off ((env environment) agent-body)
(declare-ignore env)setf (object-alive? agent-body) nil))
```

Comment

Step 3 of 3

Variables used:

- dirt
- vacuum-world

Functions:

- performance-measure
- get-percept
- legal-actions

Actions:

- suck method
- shut-off method

Comment

Was this solution helpful? 1 0

Chapter 2, Problem 10E

Consider a modified version of the vacuum environment in Exercise 2.8, in which the geography of the environment—its extent...

See solution

Chapter 2, Solution 5E

591-2-5E SA Code: 7872 SR Code: 6747 (a) Robot soccer player: The...

See solution

See more problems in subjects you study