**2.4**

(1)PEAS描述

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 活动 | Performance  Measure | Environment | Actuators | Sensors |
| 足球运动 | 进球数、成功防守次数、抢球数、断球数、任意球数 | 球、队友、对方球员 | 马达、机械腿、机械臂 | 摄像头、距离传感器、加速减速装置 |
| 探索Titan的地下海洋 | 耗电量、探索范围、安全性 | 海洋生物、岩石、水流方向、水流速度、海底压力 | 马达、发动机、灯泡、转向装置 | 摄像头、距离传感器、加速减速装置、GPS、压力传感器 |
| 在互联网上购买AI旧书 | 书的完好性、性价比、快递时效、发货周期、货物描述是否相符 | 店铺信誉、书的种类、店铺评价表 | 与店铺交流、购买付款 | 语义分析 |
| 打一场网球比赛 | 得分 | 球的位置、速度、方向、对方位置 | 机械臂、机械腿、转向装置 | 摄像头、距离传感器 |
| 对着墙壁练网球 | 接球数、连续接球数 | 球的位置、速度、方向 | 机械臂、机械腿、转向装置 | 摄像头、距离传感器 |
| 完成一次跳高 | 跳的高度、安全性 | 起跳点、助跑道路状况、落点 | 腿、全身肌肉 | 眼睛(视觉)、腿脚膝盖压力缓冲 |
| 织一件毛衣 | 毛衣大小、舒适度、美观程度 | 针织顺序、毛线颜色 | 机械臂 | 摄像头，压力传感器 |
| 在一次拍卖中对一个物品投标 | 投标次数、加价金额、最终付款价格与预期之比 | 竞拍对手人数、单次加价价格 | 加价、竞拍 | 竞拍价格分析 |

(2)性质分析

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 活动 | 可观测 | 单与多Agent | 确定与  随机 | 片段与  延续 | 静态与  动态 | 离散与  连续 | 已知与  未知 |
| 足球运动 | 局部可观 | 多 | 随机 | 延续 | 动态 | 足球运动是连续的 | 未知 |
| 探索Titan的地下海洋 | 局部可观 | 单 | 随机 | 延续 | 动态 | 探索过程是连续的 | 未知 |
| 在互联网上购买AI旧书 | 局部可观 | 单 | 确定 | 延续 | 静态 | 买书分析过程是离散的 | 未知 |
| 打一场网球比赛 | 局部可观 | 多 | 随机 | 延续 | 动态 | 打球时是连续的 | 未知 |
| 对着墙壁练网球 | 局部可观 | 单 | 随机 | 延续 | 动态 | 练球是连续的 | 未知 |
| 完成一次跳高 | 全局可观 | 单 | 随机 | 延续 | 静态 | 跳高过程是连续的 | 已知 |
| 织一件毛衣 | 全局可观 | 单 | 确定 | 延续 | 静态 | 织毛衣的动作是连续的 | 已知 |
| 在一次拍卖中对一个物品投标 | 局部可观 | 多 | 随机 | 延续 | 动态 | 投标竞标是离散的 | 未知 |

**2.6 Agent函数与程序区别**

a.

是的。例如，也可以通过来实现，两种表达式程序描述不一样，但是他们的输出是一样的，即Agent函数一样。

b.

是的。比如围棋的agent函数，这超过了计算机运行存储能力。

c.

可以。对于给定的体系结构，可以通过编写不同的agent程序来实现不同的agent函数。

d.

种

e.

不一定。当环境不发生改变时，机器运行速度不会影响agent函数；若环境是变化的，agent函数有可能会改变。

**2.8**

环境特征包括，loc和dis。Loc表示当前env是哪一个，用0、1进行区分。Dis表示当前env是否有垃圾，0表示没有垃圾，1表示有垃圾。代码描述如下：

**class** ENV():  
 **def** \_\_init\_\_(self,loc,dis):  
 self.loc = loc*#环境的位置* self.dis = dis*#环境是否有垃圾，0代表无垃圾，1代表有垃圾* **def** getloc(self):  
 **return** self.loc  
 **def** getdis(self):  
 **return** self.dis

定义agent类。类内包括执行器、传感器。传感器用于将env的dis传递给agent。执行器定义了agent的活动过程。首先初试化agent的位置(location、home)，如果当前位置有垃圾，清扫完成后转移至另一个位置；如果当前位置没有垃圾，转移至另一个位置。完成一个块的清扫任务可以得到奖励(reward)。转移过程需要消耗cost\_run的电量，清扫过程需要消耗cost\_clean的电量。当完成两个位置的清扫任务后，需要回到最初的位置，此时表示完成了一个仿真时长。代码描述如下：

**class** agent():  
 **def** \_\_init\_\_(self,reword,cost\_clean,cost\_run,home):  
 self.reword=reword*#正确判断的奖赏* self.cost\_clean=cost\_clean*#清扫的电量消耗* self.cost\_run=cost\_run*#移动的电量消耗* self.home=home*#初始位置* self.location=home*#起始位置在家* self.perf=0.0  
 **def** print\_perf(self):  
 print(**"性能："**,self.perf)  
 **def** sensor(self,env):  
 **if** env :  
 **return** 1  
 **else** :  
 **return** 0  
 *# env代表输入(0无垃圾，1有垃圾)，0代表没有垃圾，1代表有垃圾* **def** action(self,enva,envb):  
 **if** (self.location == enva.loc):  
 **if**(self.sensor(enva.dis)==1):  
 self.perf = self.perf + self.cost\_clean*#当前有垃圾，需要耗电clean* self.location = envb.loc  
 self.perf = self.perf + self.cost\_run + self.reword*#转移到另一个env，耗电run，并得到奖赏* **if** (self.sensor(envb.dis) == 1):  
 self.perf = self.perf + self.cost\_clean  
 self.perf = self.perf + self.reword  
 **else** :  
 **if**(self.sensor(envb.dis)==1):  
 self.perf = self.perf + self.cost\_clean  
 self.location = enva.loc  
 self.perf = self.perf + self.cost\_run + self.reword  
 **if** (self.sensor(enva.dis) == 1):  
 self.perf = self.perf + self.cost\_clean  
 self.perf = self.perf + self.reword  
 **if** (self.location != self.home):  
 self.location = self.home  
 self.perf = self.perf + self.cost\_run*#最终需要返回到home，在下一个仿真步判断需不需要清扫此时的home*

2.9

吸尘器的初始位置具有两种可能，两个打扫位置的状态一共有四种组合，一共需要八种情况进行仿真。如下表所示：

|  |  |  |
| --- | --- | --- |
| 吸尘器初始位置 | 方块A状态 | 方块B状态 |
| A | 无垃圾 | 无垃圾 |
| A | 无垃圾 | 有垃圾 |
| A | 有垃圾 | 无垃圾 |
| A | 有垃圾 | 有垃圾 |
| B | 无垃圾 | 无垃圾 |
| B | 无垃圾 | 有垃圾 |
| B | 有垃圾 | 无垃圾 |
| B | 有垃圾 | 有垃圾 |

定义移位耗电1，清扫耗电2，打扫干净奖励10。仿真结果如下所示

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 次序 | 1 | 2 | 3 | 4 |
|  | 18 | 16 | 16 | 14 |
| 次序 | 5 | 6 | 7 | 8 |
|  | 18 | 16 | 16 | 14 |

平均性能为：16.0

单时间仿真代码见附录

附录

**class** ENV():  
 **def** \_\_init\_\_(self, loc, dis):  
 self.loc = loc *# 环境的位置* self.dis = dis *# 环境是否有垃圾，0代表无垃圾，1代表有垃圾* **def** getloc(self):  
 **return** self.loc  
  
 **def** getdis(self):  
 **return** self.dis  
  
  
**class** agent():  
 **def** \_\_init\_\_(self, reword, cost\_clean, cost\_run, home):  
 self.reword = reword *# 正确判断的奖赏* self.cost\_clean = cost\_clean *# 清扫的电量消耗* self.cost\_run = cost\_run *# 移动的电量消耗* self.home = home *# 初始位置* self.location = home *# 起始位置在家* self.perf = 0.0  
  
 **def** print\_perf(self):  
 print(**"性能："**, self.perf)  
  
 **def** sensor(self, env):  
 **if** env:  
 **return** 1  
 **else**:  
 **return** 0  
  
 *# env代表输入(0无垃圾，1有垃圾)，0代表没有垃圾，1代表有垃圾* **def** action(self, enva, envb):  
 **if** (self.location == enva.loc):  
 **if** (self.sensor(enva.dis) == 1):  
 self.perf = self.perf + self.cost\_clean *# 当前有垃圾，需要耗电clean* self.location = envb.loc  
 self.perf = self.perf + self.cost\_run + self.reword *# 转移到另一个env，耗电run，并得到奖赏* **if** (self.sensor(envb.dis) == 1):  
 self.perf = self.perf + self.cost\_clean  
 self.perf = self.perf + self.reword  
 **else**:  
 **if** (self.sensor(envb.dis) == 1):  
 self.perf = self.perf + self.cost\_clean  
 self.location = enva.loc  
 self.perf = self.perf + self.cost\_run + self.reword  
 **if** (self.sensor(enva.dis) == 1):  
 self.perf = self.perf + self.cost\_clean  
 self.perf = self.perf + self.reword  
 **if** (self.location != self.home):  
 self.location = self.home  
 self.perf = self.perf + self.cost\_run *# 最终需要返回到home，在下一个仿真步判断需不需要清扫此时的home*env00 = ENV(0, 0)  
env01 = ENV(0, 1)  
env10 = ENV(1, 0)  
env11 = ENV(1, 1)  
a = agent(10, -2, -1, 0)  
a.action(env00, env10)  
a.print\_perf()  
a = agent(10, -2, -1, 0)  
a.action(env00, env11)  
a.print\_perf()  
a = agent(10, -2, -1, 0)  
a.action(env01, env10)  
a.print\_perf()  
a = agent(10, -2, -1, 0)  
a.action(env01, env11)  
a.print\_perf()  
a = agent(10, -2, -1, 1)  
a.action(env00, env10)  
a.print\_perf()  
a = agent(10, -2, -1, 1)  
a.action(env00, env11)  
a.print\_perf()  
a = agent(10, -2, -1, 1)  
a.action(env01, env10)  
a.print\_perf()  
a = agent(10, -2, -1, 1)  
a.action(env01, env11)  
a.print\_perf()