Name SHANKARANARAYANAN KALLIDAIKURUCHI RAMAKRISHNAN Dr. Sanda Harabagiu

Student ID SXK190109 Fall 2020

CS 6364

Quiz 1 (Take-Home)

Issued: August 24 2020

Due: September 2 2020 before Midnight (on eLearning)

Read section 2.4.7 from the Textbook and represent the Kitty world as:

1. Atomic representation.
2. Factored representation.
3. Structured representation.

A. Describe each of the representations (5 points for each representation)

1. **Atomic representation**

In Atomic representation, each state of the world is indivisible. Atomic representation, the states do not have any internal structure. In the kitty robot for instance, Considering the position state.
Position state – [1,2,3,4,5,6]

We can either have the correct state/incorrect state we are looking for. We can only compare the atomic state to the goal. So, we have to perform "Search" until we reach the goal. For example, let current state be 6 and goal state be 3. Agent has to perform all actions with memory to reach goal state.

2. **Factored representation.**

Factoring representation splits up each state into a fixed set of variables which have a real/Boolean value.

In kitty robot,

a. In position state we can have attribute as GPS location.

b. In happiness state we can have Boolean attribute based on percept

c. Common attributes can be battery level of the robot

This allows us to ask other questions to identify goal state from current state. For instance, using the position state with GPS location attribute. We can identify how far away the robot is from the goal state using difference in GPS.

3. Structured representation.

In Structured representation, we can represent the world with objects and relations between each object. We can add logic and conditionals to the agent to represent relations.
In kitty robot, we can represent all objects that can be found in the real world such as obstacles and humans that can intervene with the robot. For example,

a. In case of position states, if kitty encounters obstacles, it bumps and walks. If kitty encounters humans, it purrs and walks.
b. In case of happiness states, if kitty encounters obstacles, it becomes sad. If kitty encounters humans, it becomes curious. If it encounters both, it becomes happy. If it does not encounter anything it is very happy etc.

B. Write the pseudo-code of a utility-based agent for each of these representations (5 points for each representation)

Utility based agent consists of a utility which determines how "happy" we are with the goal. Using utility, we can disambiguate between conflicting goals.

In kitty robot, we can modify the pseudo-code given in lecture for utility-based agents.

1. Atomic Representation

Assuming we are working with kitty robot position state and we require the robot to eventually reach the goal state by moving forward or backward. We can have utility function has number of moves it takes for kitty robot to reach goal state.

```
function UTILITY-AGENT-ATOMIC(percept)
        returns action
        static: state, a description of the current position state
                goal, desired goal state

        state ←UPDATE-STATE(state,percept)
        action ← ATOMIC-ACTION(state, goal)  // Stops if goal is reached
        best-action ← UPDATE-MOVES(state, action, goal) // Calculate whether to walk left or right to
reach goal and returns best action to perform
```

return best-action

2. Factored Representation

In factored representation we can have GPS location on kitty robot which can identify the position state of the robot. Similar, to atomic representation, we can use number of moves to reach goal state for measuring performance.

function UTILITY-AGENT-FACTORED(percept)
    returns action
    static: state, a description of the current position state
        goal, desired goal state
    state ←UPDATE-STATE(state,percept)
    action ← GPS-ACTION(state, goal) // Calculates action based on GPS location
    best-action ← UPDATE-MOVES(state, action, goal)
return action

3. Structured Representation

In structured representation, we can use conditions and first order logic to determine actions with objects in the environment. We can include complex utility function such as battery efficiency and minimum number of moves to reach goal state.

function UTILITY-AGENT-STRUCTURED(percept)
    returns action
    static: state, a description of the current position state
        goal, desired goal state
        rules, a set of condition-action rules action, the most recent action, initially none
    state ←UPDATE-STATE(state,percept)
    rule ← RULE-MATCH(state, rules, goal)
    action ← RULE-ACTION[rule]
    best-action ← UTILITY-MEASURE(state, possible-actions, goal)
return best-action

Example rules:
If state == goal then
    Return “STOP”
If percept == Pet then
    If state.object == Human then
        Return “MEOW”
    Else “PURR”
If percept == Bump then
    If state.object == Obstacle then
        Return “WALK”
    Else “BLINK”