Name: Pratik Rajendra Deshpande Dr. Sanda Harabagiu

Student ID: 2021495785 (Prd190001) 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)

The assumed environment of the kitty is the **environment with the 6 rooms, and some human agent to interact with the kitty.** The rooms are numbered 1-6 respectively, all of them connected to each other and the kitty starting in room one.

The **percepts** of the kitty are **pet, bump, see, clap**, (Assuming the sensors that capture, touch, vision to some aspects and sounds)

The **actions** of the kitty are **walk, pounce, purr, sleep, meow, move head**

The Utility Function: Loved > Excited > Happy > Sad > Angry > Sleepy & Battery Health that's Low or Not Low.

1. **Atomic Representation:** As the states in the Atomic representation are a black box the kitty will have a state called "location", which changes as per the kitty walks. Every time she walks she changes location. She also has an internal happiness state to track utility.

2. **Factored Representation:** As this state allows us to have a vector of values thus leading us to have some detailed information, the kitty would store information like, "currentLocation", "batteryHealth","timeElapsedSinceLastPercept", "selectedMode".

3. **Structured Representation:** This state allows us to have a detailed relational view of the world, thus in this mode, kitty's states internal representation would have an object for the human hand that claps, and has location and the kittys internal object itself that would have indication of her own happiness, and location, and battery. Key difference being Human Hand separated from the kitty object.

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

1. **Atomic Representation:**

function ATOMIC-KITTY(percept) **returns** action

static: state(location, happiness), a description of the current world state rules, a set of condition-action rules, models also includes how utility is affected.

action, the most recent action, initially none

state ←UPDATE-STATE(state,percept,action)

rule ← RULE-MATCH(state, rules)

estimatedWorldState <- EXPECTED-STATE(state, rule)

estimatedUtility <- MAX-EXPECTED-UTILITY (estimatedWorldState, state)

action ← estimatedUtility.maximizingAction

return action

State Update:

state ←UPDATE-STATE(previous-state, percept, action)

Happiness +1 ←UPDATE-KITTY[Happines,Clap]
Happiness +2 ←UPDATE-KITTY[Happines,Pet]
Angry ←UPDATE-KITTY[Happiness,Bump]

Position-1 ←UPDATE-KITTY[Position,Walk]
Position ← UPDATE-KITTY[Position,Purr|Meow|Blink|Pounce]

Condition Action Rules:
IF percept==clap & happiness ==sleepy then action = [meow,blink]
IF percept == bump & happiness < angry then action = [pounce,purr]
IF percept == [clap,clap] & happiness > Sad then action = [walk]

2. **Factored Representation:**

function FACTORED-KITTY(percept) **returns** action

static: state(location, batteryhealth, happiness, previousAction, timeElapsedSinceLastPercept,selectedMode), a description of the current world state rules, a set of condition-action rules, a description of how the utility is affected by actions and percepts.

action, the most recent action, initially none

state ←UPDATE-STATE(state,percept,action)

rule ← RULE-MATCH(state, rules)

estimatedWorldState <- EXPECTED-STATE(state, rule)

estimatedUtility <- EXPECTED-UTILITY (estimatedWorldState, state)

action ← estimatedUtility.maximizingAction

return action

State Update:

state ←UPDATE-STATE(previous-state, percept, action)

Happiness +1 ←UPDATE-KITTY[Happines,Clap]

Happiness +2 ←UPDATE-KITTY[Happines,Pet]

Sleep <- UPDATE-KITTY[Happiness, BatteryHealth.low]

Angry ←UPDATE-KITTY[Happiness,Bump]

SelectedMode <- UPDATE-KITTY[autonomous, clap clap]

SelectedMode <-UPDATE-KITTY[cuddle, pet]

batteryHealth – 1 <- UPDATE-KITTY[batteryHealth, Purr|Blink|Pounce]

batteryHealth – 2 <- UPDATE-KITTY[batteryHealth, Walk]

timeElapsedSinceLastPercept <- RESET-TIME[timeElapsedSinceLastPercept, percept]

currentLocation + 1 ←UPDATE-KITTY[Position,Walk]

currentLocation ← UPDATE-KITTY[Position,Purr|Meow|Blink|Pounce]

Condition Action Rules:

If batteryHealth > Low:

IF selectedMode == Autonomous:

IF percept==clap & happiness ==sleepy then action = [meow or blink]

IF percept == bump & happiness < angry then action = [pounce or purr]

IF percept == [clap,clap, pet] then [pounce or purr or walk or meow or blink]

IF percept == pet then blink then purr

IF percept == [pet pet] then sleep

IF selectedMode == Cuddle:

IF percept not bump then [blink, purr, meow]

ELSE: Pounce

Else:

Sleep

3. **Structured Representation:**

function STRUCTURED-KITTY(percept) **returns** action

static: state(location, batteryhealth, happiness, previousAction, timeElapsedSinceLastPercept,selectedMode), a description of the current world state rules, a set of condition-action rules, a description of how the utility is affected by actions and percepts.

action, the most recent action, initially none

state ←UPDATE-STATE(state,percept,action)

rule ← RULE-MATCH(state, rules)

estimatedWorldState <- EXPECTED-STATE(state, rule)

estimatedUtility <- EXPECTED-UTILITY (estimatedWorldState, state)

action ← estimatedUtility.maximizingAction

return action

State Update:
Kitty = {
batteryHealth – 1 <- Update-Kitty[batteryHealth, action]
location +1 <- Update-Kitty[location, walk]
location <- Update-Kitty[location, purr|sleep|meow|pounce]
distanceFromHand = Update-Kitty[HumanHand.currentLocation, location]
happiness +1 <- Update-Kitty[Happiness, HumanHand.currentPercept.Clap]
happiness+2<-Update-Kitty[Happiness,HumanHand.currentPercept.Pet]

}

HumanHand = {

currentLocation <- Update-Human-Hand[currentLocation,percept]
currentPercept <- Update-Human-Hand[currentPercept, percept]

}

Condition Action Rules:

If batteryHealth > Low:

IF HumanHand.currentPercept == Clap & Kitty.happiness > Sleep Then [Purr,Meow]

IF Kitty.distancefromHand > 1 then Walk

IF Human.currentLocation == Kitty.Location & Human.currentPercept == Clap Then [Sleep|Meow,Blink]

Else:

Sleep