Agents

Practice

TABLE-DRIVEN-AGENT

- **Table** contains all possible *percepts* that can occur.
- Each step appends current *percept* to list of *percepts*.
- **LOOKUP** current *percepts* in *table.*

```
function TABLE-DRIVEN-AGENT( percept ) returns an action
    static: percepts, a sequence, initially empty
            table, a table of actions, indexed by percept sequences,
                initially fully specified

    append percept to the end of percepts
    action = LOOKUP( percepts, table)
    return action
```

```
def TABLE_DRIVEN_AGENT(percept):
# Determine action based on table and percepts
percepts.append(percept)
# Append percept
action = LOOKUP(percepts, table)
# Lookup appropriate action for percepts return action
```

- Refer to table_driven_agent.png

Try…

Not to be submitted

1. Run the module (using run()).
2. The *percepts* should now be: [('A', 'Clean'), ('A', 'Dirty'), ('B', 'Clean')].
   - The table contains all possible percept sequences to match with the percept history.
   - Enter:

   ```
   print (TABLE_DRIVEN_AGENT((B, 'Clean')))
   percepts
   ```

   - Explain the results.
3. How many table entries would be required if only the *current* percept was used to select an action rather than the percept history?
4. How many table entries are required for an agent lifetime of *T* steps?

REFLEX-VACUUM-AGENT

- Only responds to current percept (location and status) ignoring percept history.
- Uses *condition-action* rules rather than table.
  - **if** *condition* **then return** *action*
  - **if** *status = Dirty* **then return** *Suck*
- **Sensors () -** Function to sense current location and status of environment (i.e. *location* of agent and *status* of square).
- **Actuators (action) -** Function to affect current environment location by some action (i.e. *Suck, Left, Right, NoOp*).

```
function REFLEX-VACUUM-AGENT( [location, status] )
    returns  an action
      if status = Dirty then return Suck
      else if location = A then return Right
      else if location = B then return Left
```

```
def REFLEX_VACUUM_AGENT((location, status)):
# Determine action
    if status == 'Dirty': return 'Suck'
    elif location == A: return 'Right'
    elif location == B: return 'Left'
```

- Refer to reflex_vacuum_agent.png

Try…

Not to be submitted

1. Run the module.
2. Enter: *run(10)*
3. Should bogus actions be able to corrupt the environment? Change the REFLEX_VACUUM_AGENT to return bogus actions, such as Left when should go Right, etc. Run the agent. Do the Actuators allow bogus actions?

SIMPLE-REFLEX-AGENT

```
function SIMPLE-REFLEX-AGENT( percept ) returns  an action
    static: rules, a sequence, a set of condition-action rules

    state = INTERPRET-INPUT( percept )
    rule  = RULE-MATCH( state, rules )
    action = RULE-ACTION[ rule ]
     return action
```

```
def SIMPLE_REFLEX_AGENT(percept): # Determine action
    state = INTERPRET_INPUT(percept)
    rule = RULE_MATCH(state, rules)
    action = RULE_ACTION[rule]
    return action
```

Condition-action

- **rules** = { (A,'Dirty'):1, (B,'Dirty'):1, (A,'Clean'):2, (B,'Clean'):3, (A, B, 'Clean'):4 }

  Defines *rule* for each *condition,* such as: condition == (A,'Dirty') uses rule 1.

- **RULE_ACTION** = { 1:'Suck', 2:'Right', 3:'Left', 4:'NoOp' }

  Defines *action* for each *rule,* such as: rule 1 produces action 'Suck'

- Refer to simple_reflex_agent.png

Not to be submitted

1. Run the module.
2. Enter: *run(10)*
3. Change the SIMPLE_REFLEX_AGENT *condition-action* rules to return bogus actions, such as Left when should go Right, or Crash, etc. Rerun the agent. Do the Actuators allow bogus actions?

REFLEX-AGENT-WITH-STATE

Reflex agent only responded to current percepts, no history or knowledge.

Model-based reflex agents

- Maintain internal state that depends upon percept history.
- Agent has a *model* of how the world works.
- The model requires two types of information to update internal:
  - How environment evolves independent of the agent (e.g. Clean square stays clean)
  - How agent's actions affect the environment (e.g. Suck cleans square)

```
**function** REFLEX-AGENT-WITH-STATE( *percept )* **returns** an action
    **static:** *state,* a description of the current world state
            *rules,* a sequence, a set of condition-action rules
            *action*, the most recent action, initially none
    *state* = UPDATE-STATE( *state, action, percept )*
    *rule* = RULE-MATCH( *state, rules )*
    *action* = RULE-ACTION*[ rule ]*
    **return** *action*
```

```
def REFLEX_AGENT_WITH_STATE(percept): global state, action
   state = UPDATE_STATE(state, action, percept)
   rule = RULE_MATCH(state, rules)
   action = RULE_ACTION[ rule ]
   return action
```

Model - Used to update history.

- History initially empty:

  model = {A: None, B: None}

- Model only used to change state when A == B == 'Clean'

  if model[A] == model[B] == 'Clean' : state = (A, B, 'Clean')

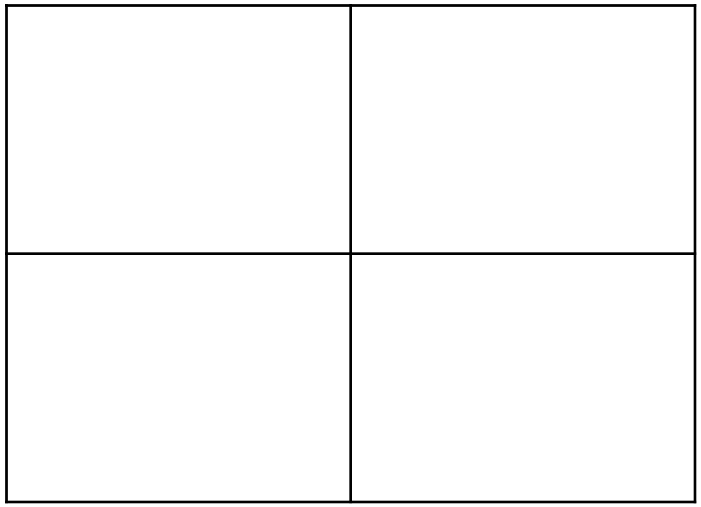
- Refer to reflex_agent_with_state.png

Homework 1– REFLEX-VACUUM-AGENT.

To be submitted

Extend the REFLEX-VACUUM-AGENT program to have 4 locations (4 squares):

- The agent should only sense and act on the square where it is located.
- Allow any starting square.
- Use run (20) to test and display results.



Homework 2– REFLEX-VACUUM-AGENT-WITH-STATE.

To be submitted

Extend the REFLEX-AGENT-WITH-STATE program to have 4 locations (4 squares):

- The agent should only sense and act on the square where it is located.
- Allow any starting square.
- Use run (20) to test and display results.