Search, Autograder, Reflex

Adapted from slides kindly shared by Stuart Russell

Announcements

Project P2 Multi-Agent Pac-Man coming later this week.

Office hours for me on Thursday 2:30-3:30 and Friday 1:30-2:30, ECST 121
And after class

Outline

- ♢ Search project and autograder
- ♢ Reflex Agents
- ♢ Project P2 Multi-Agent Pac-Man

Appreciations

Your hard work!!

You all giving me feedback

You all sharing insights on Piazza

Exercise! Walking!

Wise sayings. "Fear is excitement, without the breath"

P1 Contest Winners

P1 Search contest winners and demos:

Second place: cost of 350 in 0.51 sec: Michael Hutchison!

P1 Contest Winners 2

First place: cost of 280 in 10 sec: Eliot Glairon!

P1 Scores

Preliminary results are in for P1.

Statistics, not including the bonus or extra credit points:

Out of 20 points:

Median : 16.0

Mean : 14.1

Std. dev : 5.6

Graph Search Algorithms

Effect of different graph search algorithms....

The 3rd edition of the text has a slightly different algorithm for graphSearch than the 2nd edition

The version from the 2nd edition is simpler, and is the one I presented in the first lecture on search (08-29)

See also the Piazza question "DFS in Berkeley Handout #1"

See the Piazza "General Resources" section for a pdf of all the algorithms in the 3rd edition

Graph search (2nd Edition of AIMA)

```
function GRAPH-SEARCH( problem, fringe) returns a solution, or failure
    closed ← an empty set
    fringe ← INSERT(MAKE-NODE(INITIAL-STATE[problem]), fringe)
    loop do
        if fringe is empty then return failure
        node ← REMOVE-FRONT(fringe)
        if GOAL-TEST(problem, STATE[node]) then return node
        if STATE[node] is not in closed then
            add STATE[node] to closed
            fringe ← INSERTALL(EXPAND(node, problem), fringe)
    end
```

Nodes expanded and autograder

Affects Q2, breadth first search. Autograder was saying:

q2: *** Wrong number of nodes expanded for breadthFirstSearch: 275 vs 269

I'm fixing that so both are accepted

Autograder

The autograder certainly isn't perfect

Others that use it love it

Great at checking a wide variety of common issues with how the code runs

But we're learning more about some places where it can be improved, and/or manual intervention is needed

Grading changes

Change to how Breadth-First-Search is graded, so that you get full credit for using the pseudocode from the 3rd edition

We've seen some examples where the autograder "nodes expanded" result is significantly different from what running it from the command line produces. If this happened to you and affected your score, let us know.

If you got a low autograder score on the search project you have some options to make up some of the points you missed

♢ Meet with me, go over code and scores, see if they make sense, adjust if warranted

♢ If necessary, improve code and resubmit

Other insights gained

Will be posting tips on working more effectively with python, pacman

Speeding up pacman runs

Running pacman remotely, e.g. on CSEL via ssh

Nodes expanded inquiry

Time to brainstorm about nodes expanded

Highly important - this is the costly thing about search!

Talk about implications of

- ♢ Different graphSearch algorithms
- ♢ Different state representations
- ♢ ???

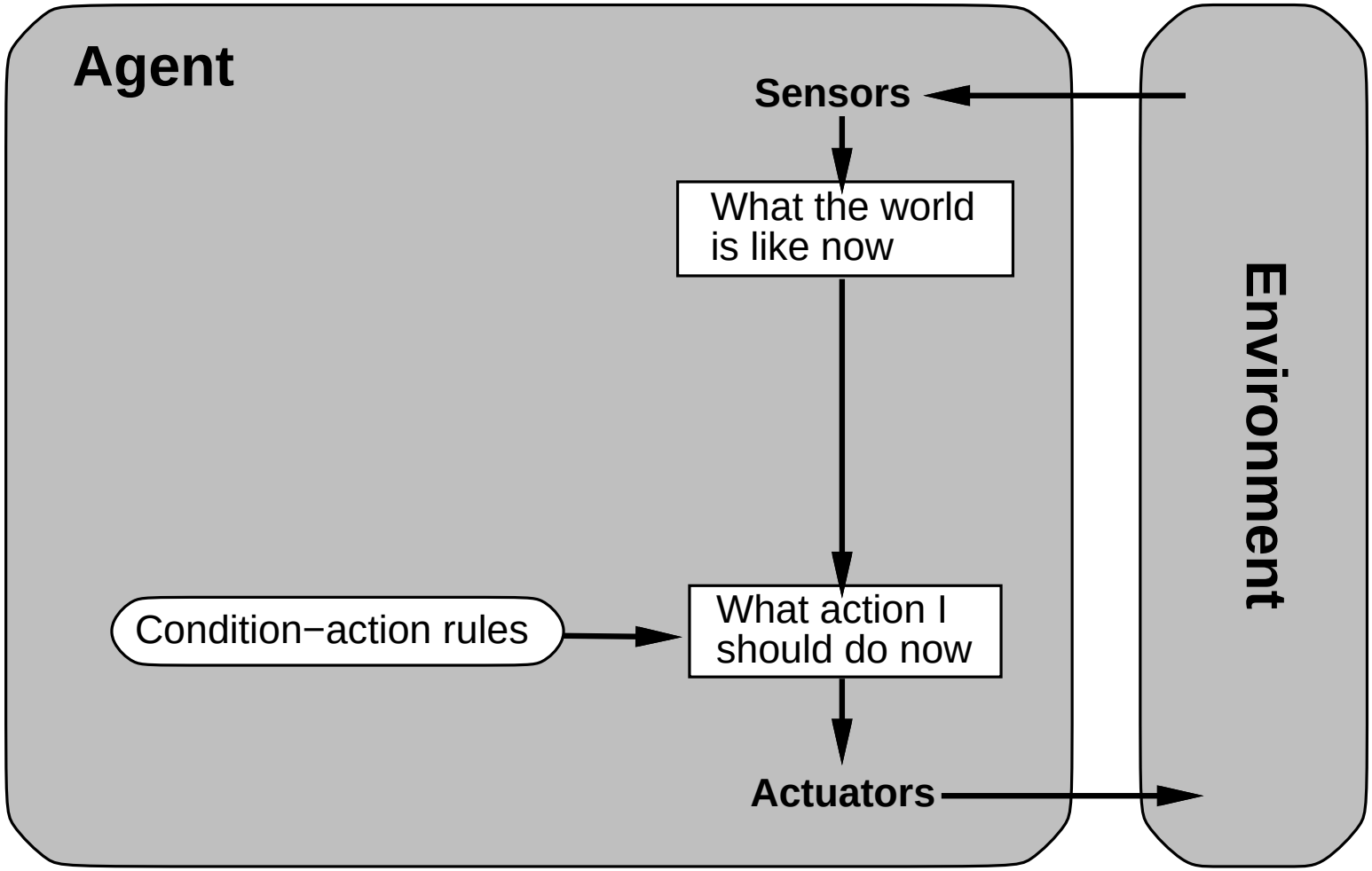
Agent types

Four basic types in order of increasing generality:

- simple reflex agents
- reflex agents with state
- goal-based agents
- utility-based agents

All these can be turned into learning agents

Simple reflex agents



Example

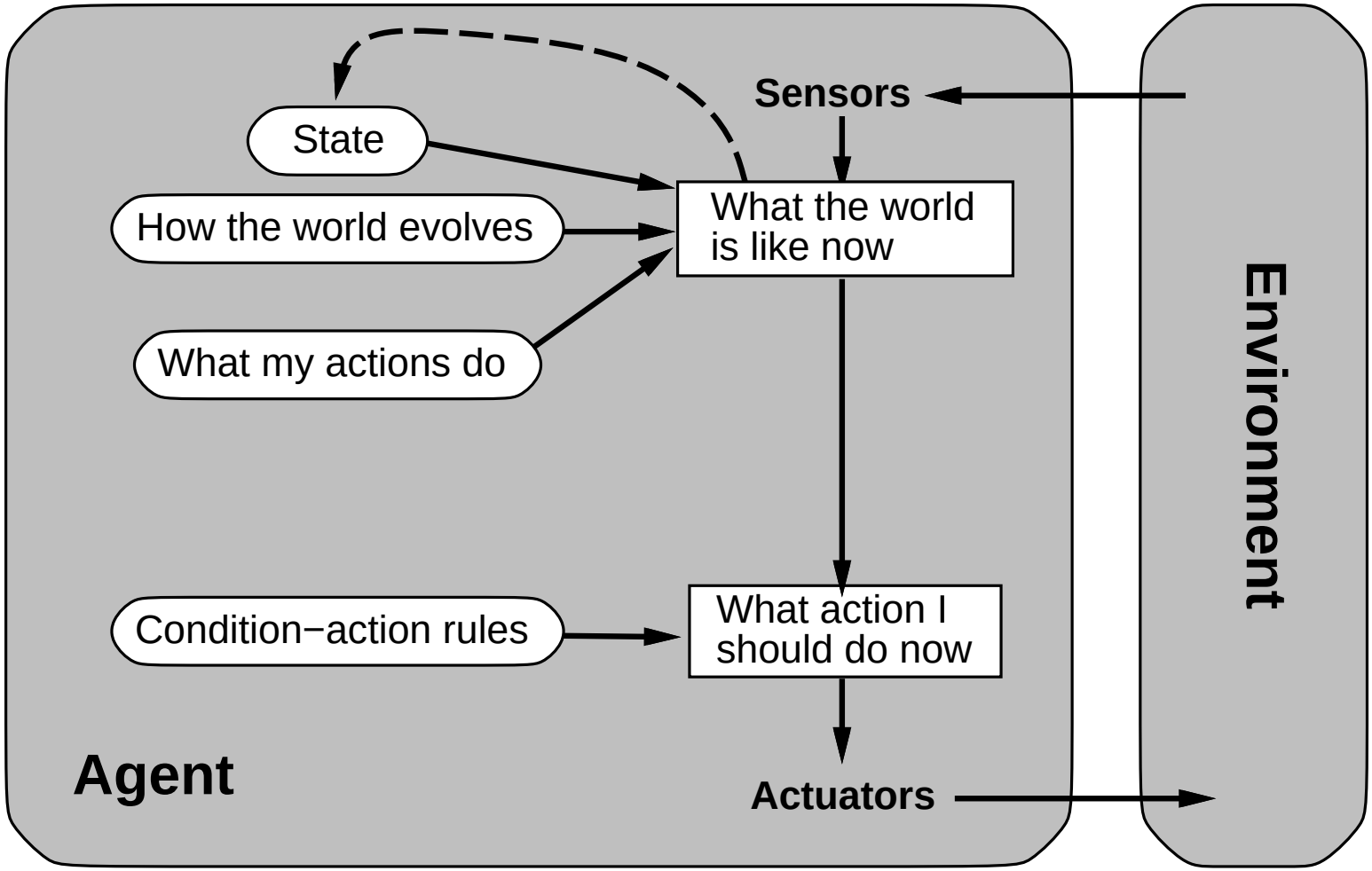
function REFLEX-VACUUM-AGENT([*location*,*status*]) **returns** an action

  if *status* = *Dirty* **then return** *Suck*
  else if *location* = *A* **then return** *Right*
  else if *location* = *B* **then return** *Left*

```
(setq joe (make-agent :name 'joe :body (make-agent-body)
                      :program (make-reflex-vacuum-agent-program))

(defun make-reflex-vacuum-agent-program ()
  #'(lambda (percept)
      (let ((location (first percept)) (status (second percept)))
        (cond ((eq status 'dirty) 'Suck)
              ((eq location 'A) 'Right)
              ((eq location 'B) 'Left)))))
```

Reflex agents with state



Example

function REFLEX-VACUUM-AGENT([*location*,*status*]) **returns** an action
static: *last_A*, *last_B*, numbers, initially ∞

if *status* = *Dirty* **then** ...

```
(defun make-reflex-vacuum-agent-with-state-program ()
  (let ((last-A infinity) (last-B infinity))
  #'(lambda (percept)
      (let ((location (first percept)) (status (second percept)))
        (incf last-A) (incf last-B)
        (cond
         ((eq status 'dirty)
          (if (eq location 'A) (setq last-A 0) (setq last-B 0))
          'Suck)
         ((eq location 'A) (if (> last-B 3) 'Right 'NoOp))
         ((eq location 'B) (if (> last-A 3) 'Left 'NoOp)))))))
```

MultiAgent Project

Question: put MultiAgent Search out with 4-spaces per indent?

Walk thru multiagent problem