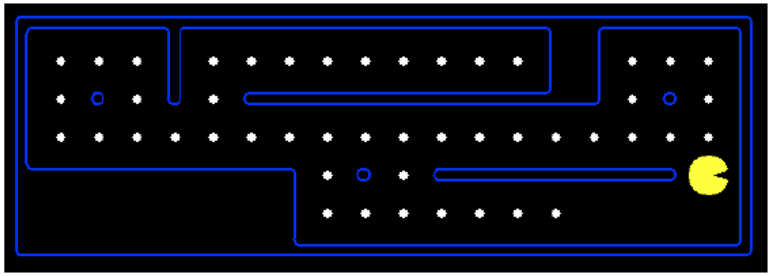
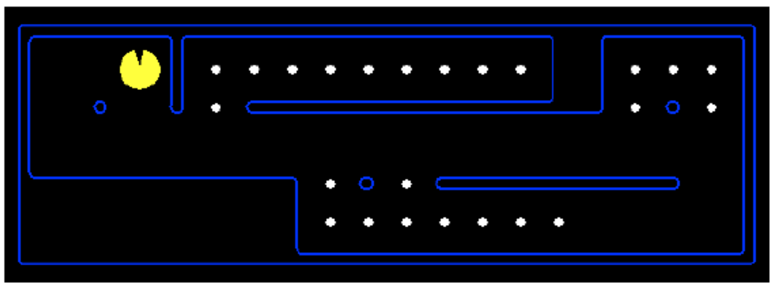
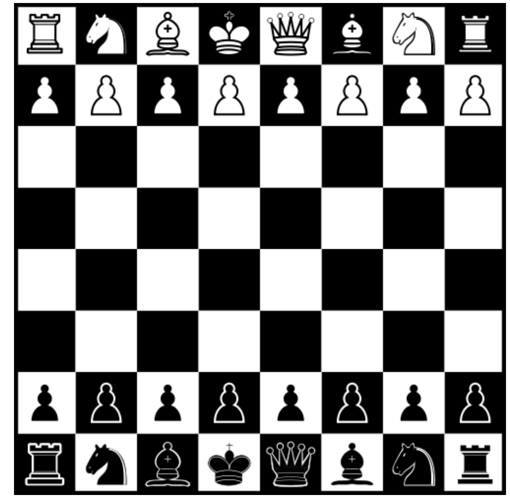
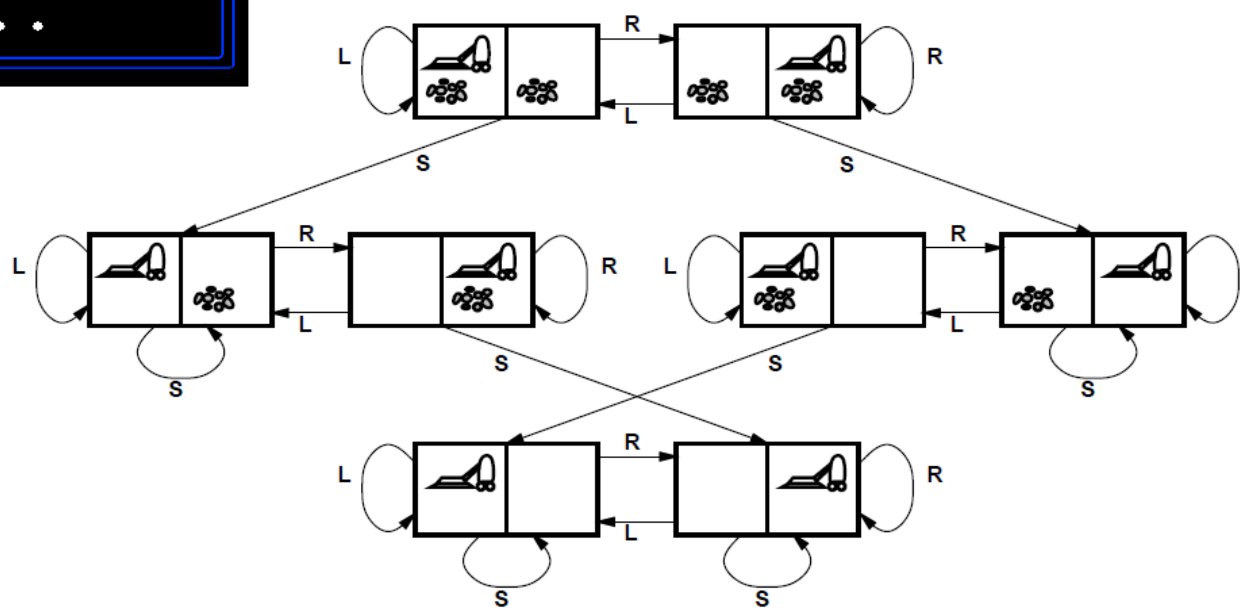
Agents and State Spaces





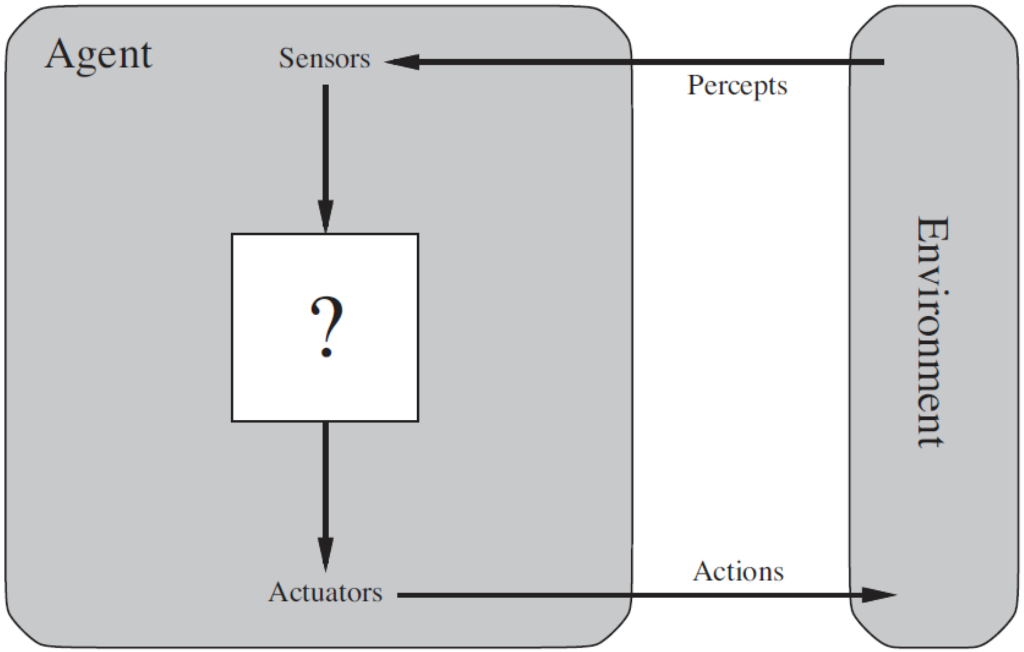




Overview

- Agents and environments
- Rationality
- Agent types
- Specifying the task environment
  - Performance measure
  - Environment
  - Actuators
  - Sensors
- Search problems
- State spaces

Agents and environment

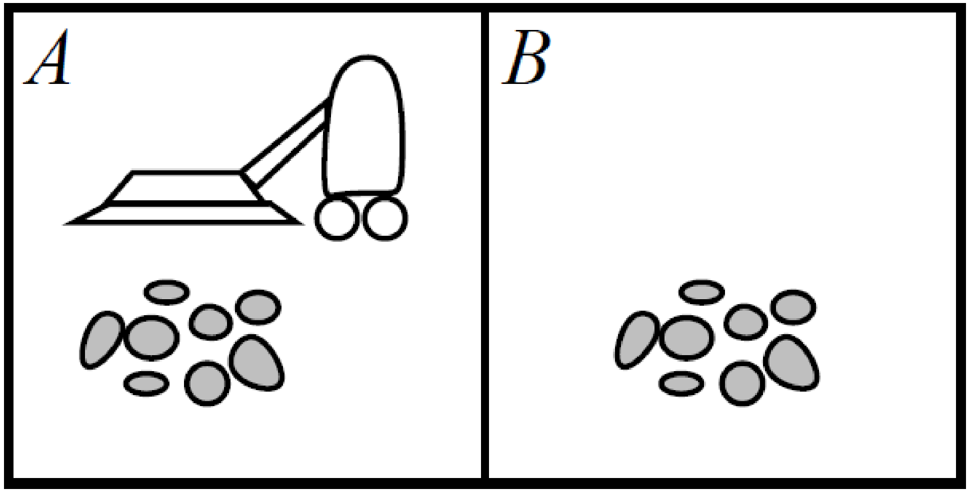


Agents: human, robots, bots, thermostats, etc.

Agent function: maps from percept history to action

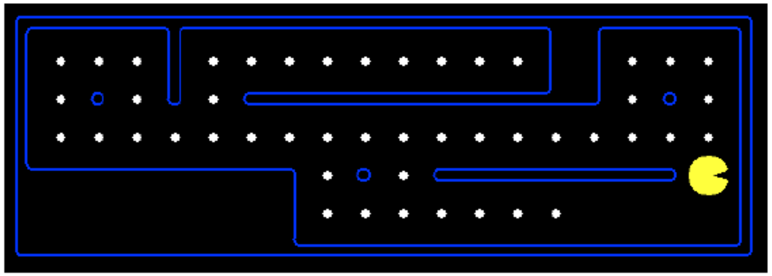
Agent program: runs on the physical system

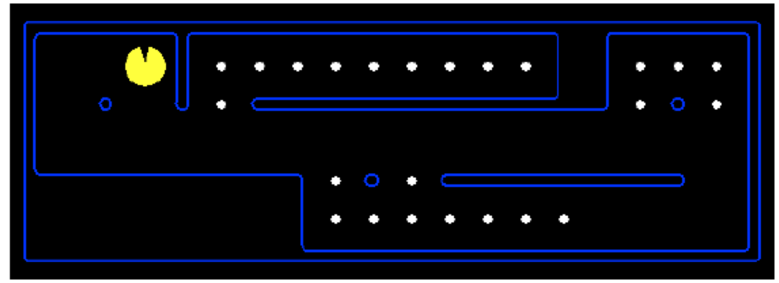
Vacuum cleaner world



- Percepts:
  - location
  - contents
    - e.g. [A, Dirty]
- Actions:
  - {Left, Right, Suck, NoOp}

Pacman's goal: eat all the dots





- Percepts:
  - location of Pacman
  - location of dots
  - location of walls
- Actions:
  - {`Left`, `Right`, `Up`, `Down`}

Rationality

- We want to design *rational* agents
  - Rational ≠ level-headed, practical
- We use rational in a particular way:
  - Rational: maximally pursuing pre-defined goals
  - Rationality is only concerned with what decisions are made
    - Not the thought process behind them
    - Not whether the outcome is successful or not
  - Goals are expressed in terms of some fixed performance measure evaluating the environment sequence:
    - One point per square cleaned up in time T?
    - One point per clean square per time step, minus one per move?
    - Penalize for > k dirty squares
  - Being rational means maximizing your expected utility

Target Tracking Agents

Percepts:

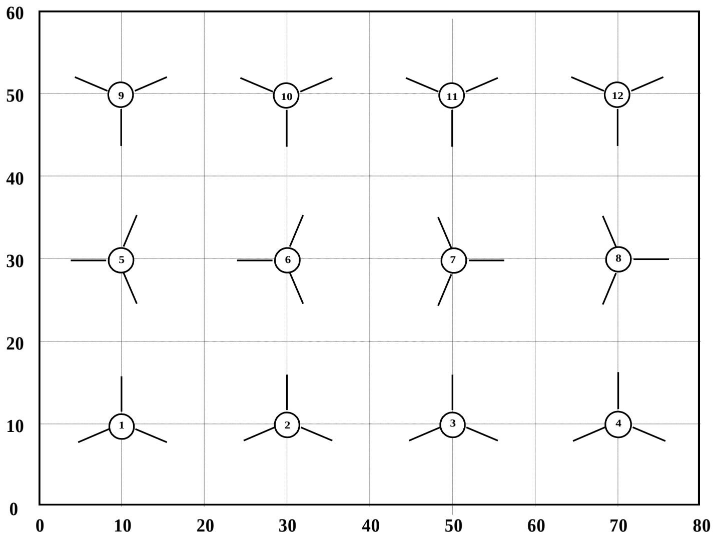
- My radar's current location
- Which radar sector is on
- Radar signal detected
- Communication from other agents

Actions:

{Turn on sector, Track,
Send Request, Negotiate}

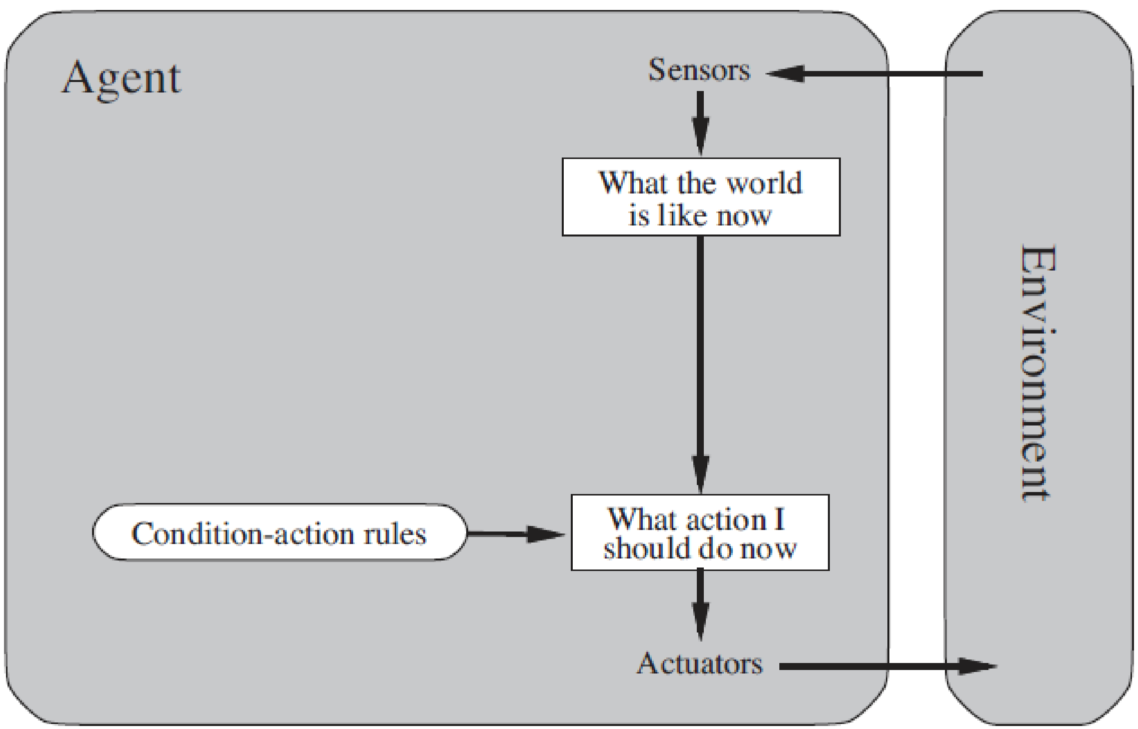
Performance Evaluation Criteria:

- Planned Measurements per Target
- Three or More Measurements in a Two Second Window per Target
- Balanced Measurements Across Multiple Targets
- Total Number of Measurements Taken
- Average Tracking Error



Agent types: Reflex agents

- Simple reflex agents:
  - Choose action based on current percept
  - Do not consider future consequences of actions
  - Consider how the world *IS*



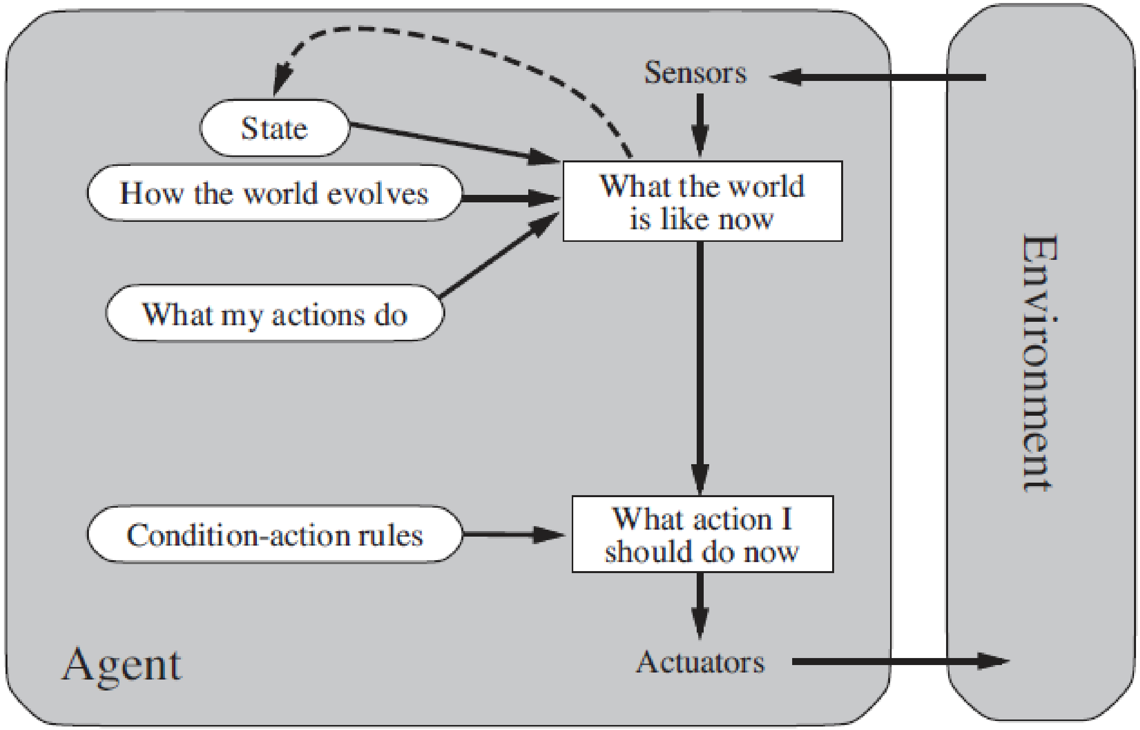
Reflex agent example

```
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
```

Percept sequence	Action
$[A, Clean]$	$Right$
$[A, Dirty]$	$Suck$
$[B, Clean]$	$Left$
$[B, Dirty]$	$Suck$
$[A, Clean]$, $[A, Clean]$	$Right$
$[A, Clean]$, $[A, Dirty]$	$Suck$
⋮	⋮

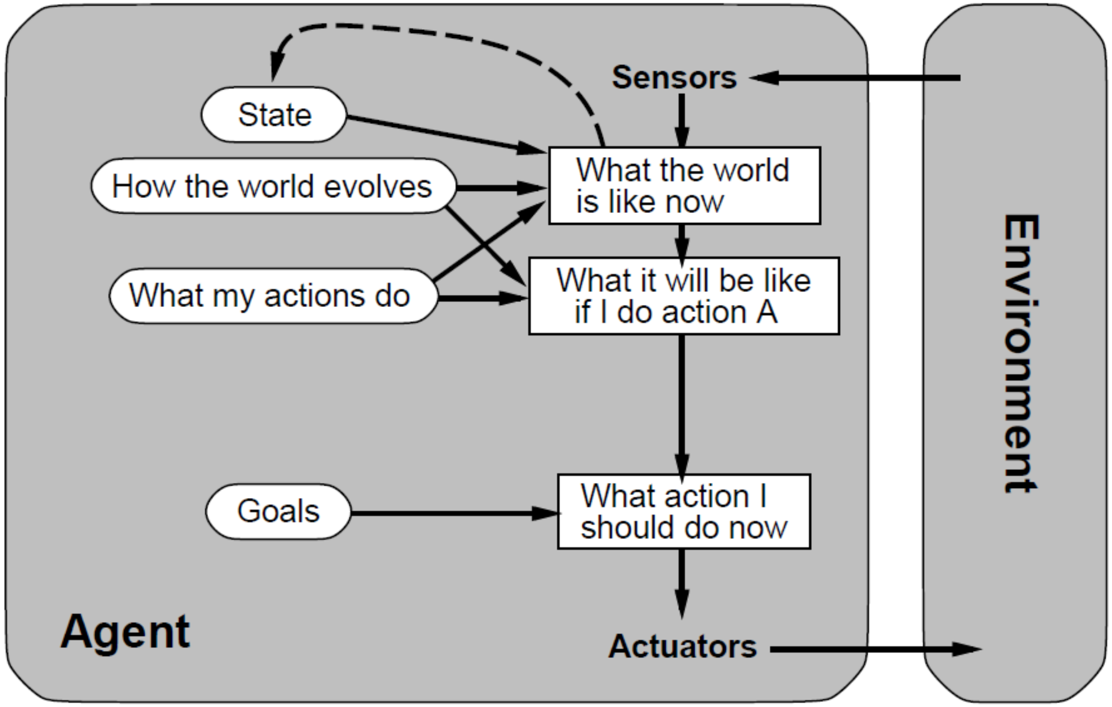
Agent types: Model-based reflex agent

- Model-based reflex agents:
  - Choose action based on current and past percepts:
    - Tracks some sort of *internal state*
  - Consider how the world *IS* or *WAS*



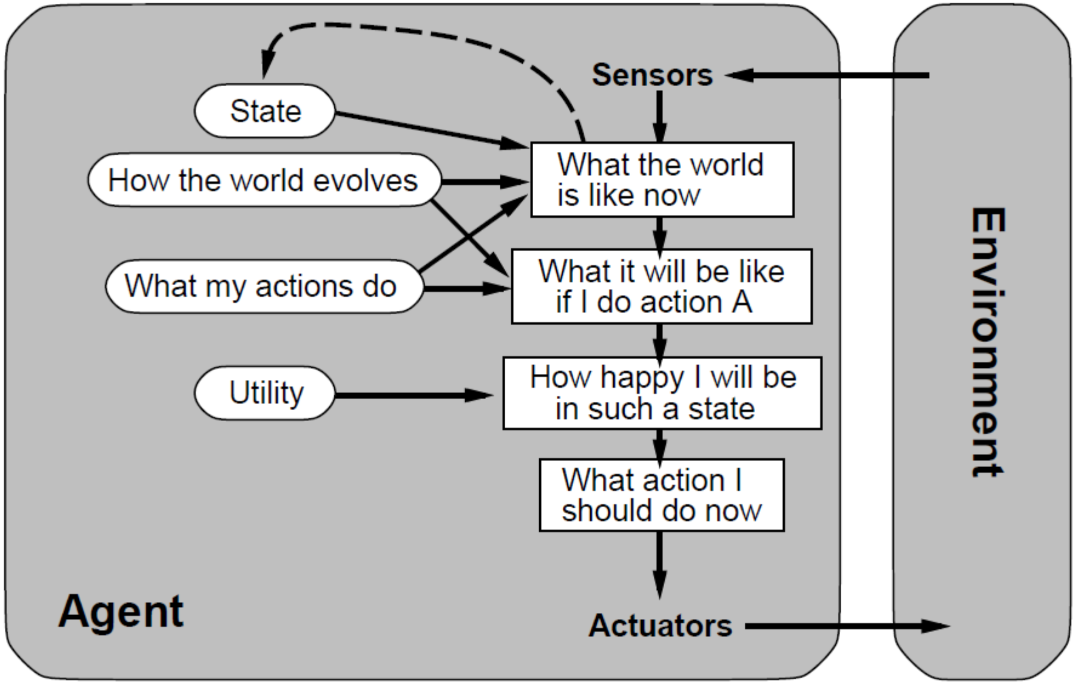
Agent types: Goal-based agents

- Goal-based agents:
  - Track current and past percepts (same as model-based reflex agent)
  - *Goal* information describing desirable situations
  - Considers the future:
    - "What will happen if I do such-and-such?"
    - "What will make me happy?"



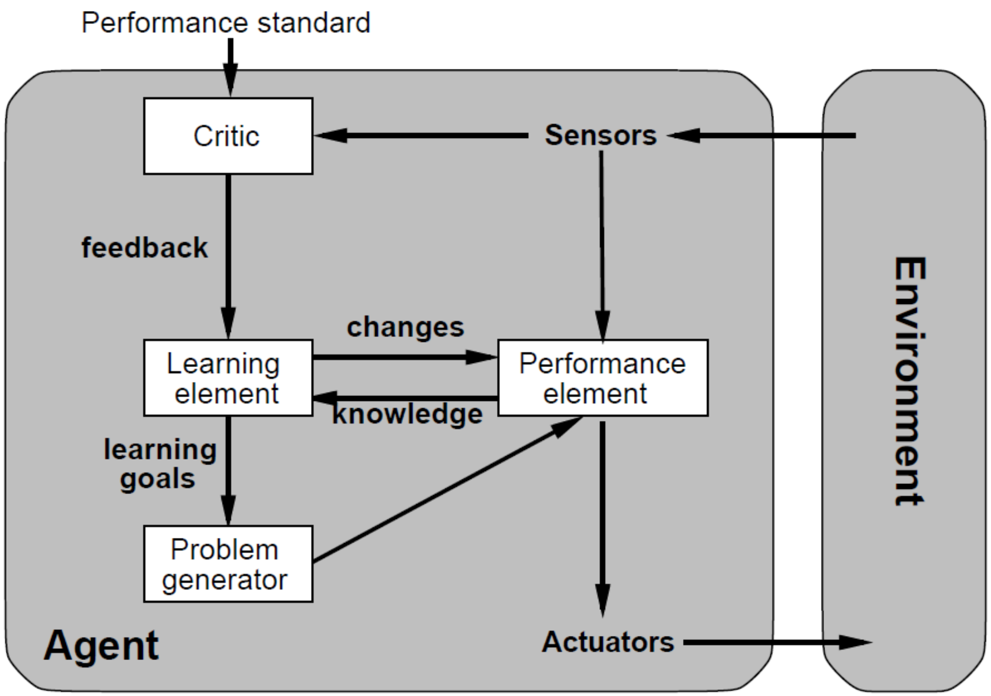
Agent types: Utility-based agents

- Utility-based agents:
  - Many actions may achieve a goal
    - But some are quicker, safer, more reliable, cheaper, etc.
  - Maximize your "happiness" = *utility*
  - Requires a utility function



Agents that learn

- Learning agents:
  - *Critic:* determines how agent is doing and how to modify performance element to do better
  - *Learning element:* makes improvements
  - *Performance element:* selects external actions
  - *Problem generator:* seeks out informative new experiences



The "PEAS" task environment

- Performance measure
  - What we value when solving the problem
  - e.g. trip time, cost, dots eaten, dirt collected
- Environment
  - Dimensions categorizing the environment the agent is operating within
- Actuators
  - e.g. accelerator, steering, brakes, video display, audio speakers
- Sensors
  - e.g. video cameras, sonar, laser range finders

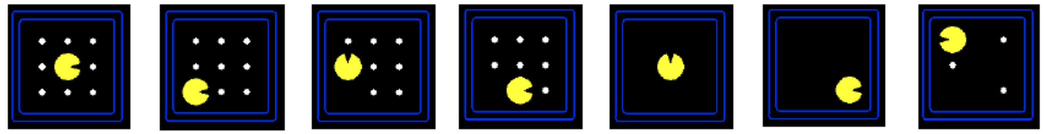
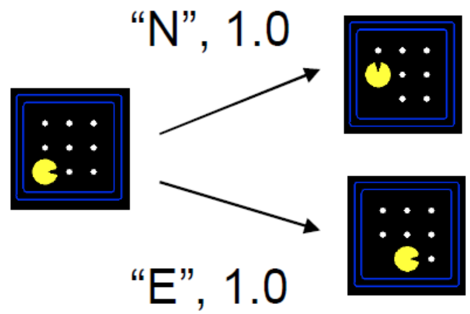
7 task environment dimensions

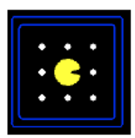
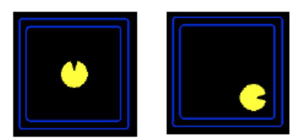
- Fully observable vs. partially observable
  - e.g. vacuum senses dirt everywhere = fully observable
  - e.g. vacuum senses dirt only at current location = partially
- Single agent vs. multiagent
  - e.g. solving a crossword = single agent
  - e.g. playing chess = multiagent
- Deterministic vs. stochastic
  - Is the next state completely determined by current state and action executed by agent?
- Episodic vs. sequential
  - Does the next episode depend on previous actions?
  - e.g. spotting defective parts on assembly line = episodic
  - e.g. playing chess is sequential

7 task environment dimensions

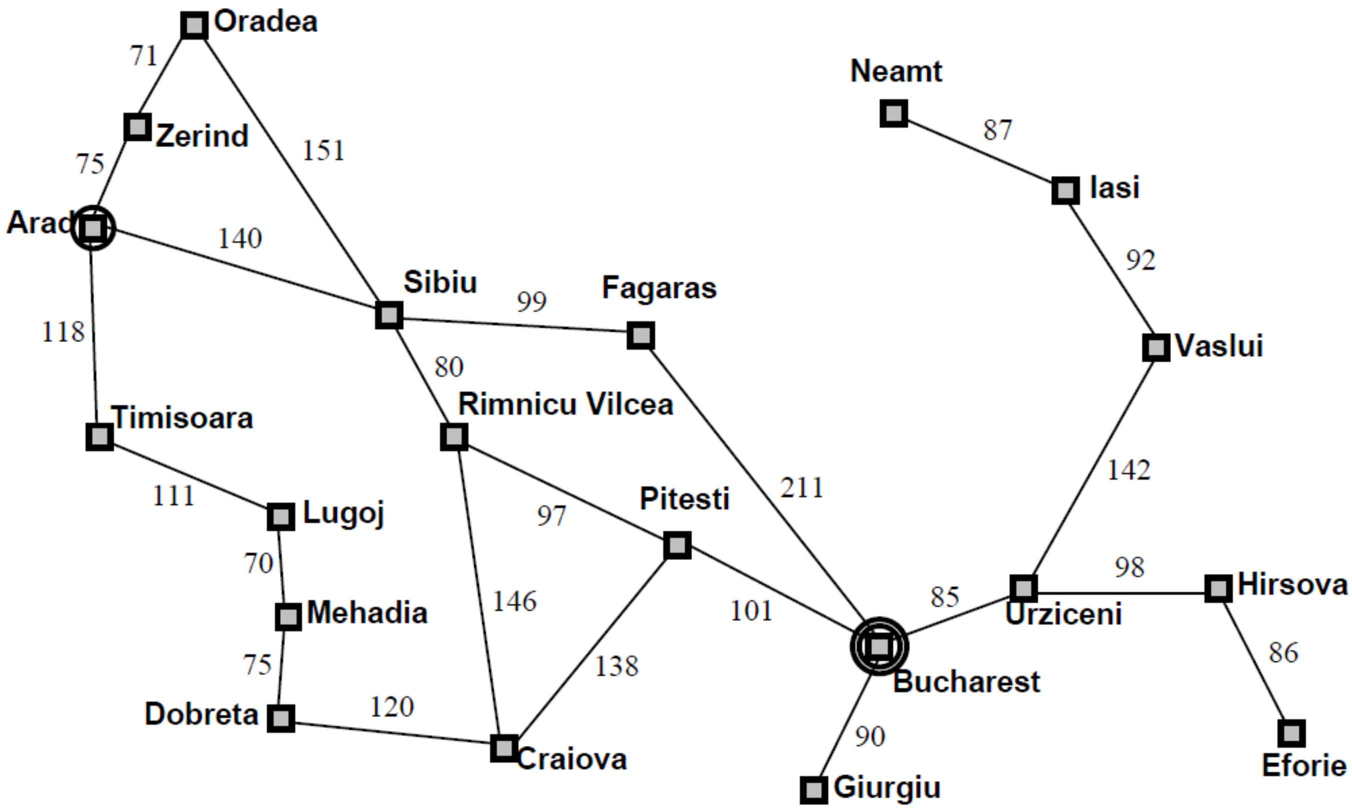
- Static vs. dynamic
  - Can things change while we're trying to make a decision?
  - e.g. crossword puzzle = static
  - e.g. taxi driving = dynamic
- Discrete vs. continuous
  - Does the environment state/percepts/actions/time take on a discrete set of values or do they vary continuously?
  - e.g. chess = discrete
  - e.g. taxi driving = continuous
- Known vs. unknown
  - Agent's knowledge about the rules of the environment
  - e.g. playing solitaire = known
  - e.g. a new video game with lots of buttons = unknown

Search problems

- A search problem consists of:
  - State space 
  - Successor function (with actions, costs) 

  - Start state 
  - Goal test (e.g. all dots eaten) 
- A solution is a sequence of actions (a plan) transforming start state to a state satisfying goal test

Example: Romania



State space: Cities

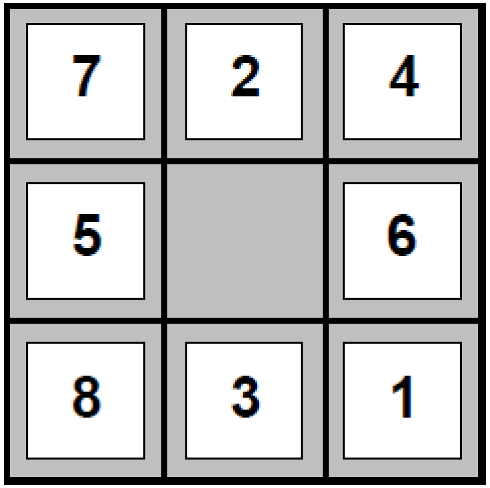
Successor function: Adjacent cities with cost = distance

Start state: Arad

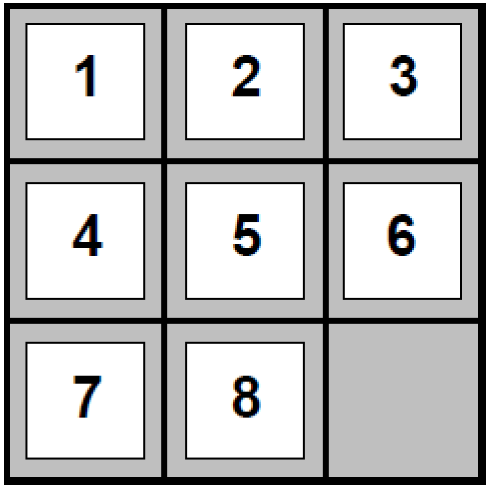
Goal test: Is state == Bucharest?

Solution? Sequence of roads from Arad to Bucharest

Example: 8-puzzle



Start State



Goal State

State space: Location of each of the eight tiles

Successor function: States resulting from any slide, cost = 1

Start state: Any state can be start state

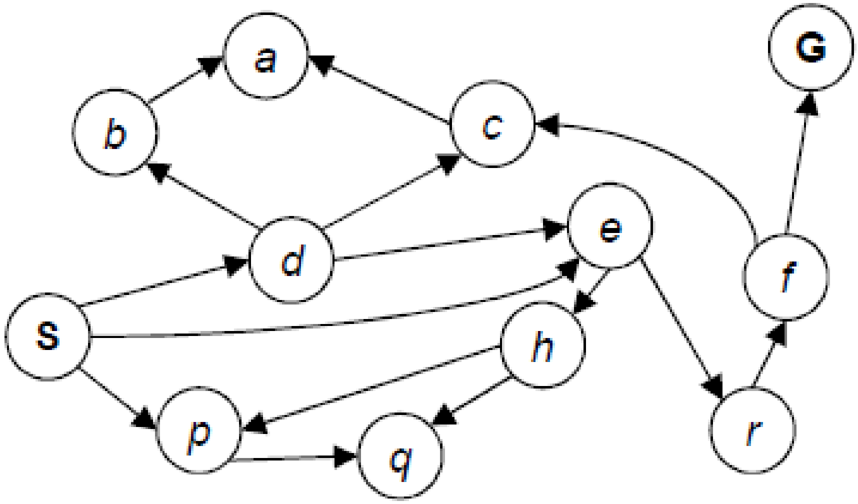
Goal test: Is state == given goal state

Solution? Sequence of tile slides to get to goal

Note: optimal solution of n-puzzle is NP-hard

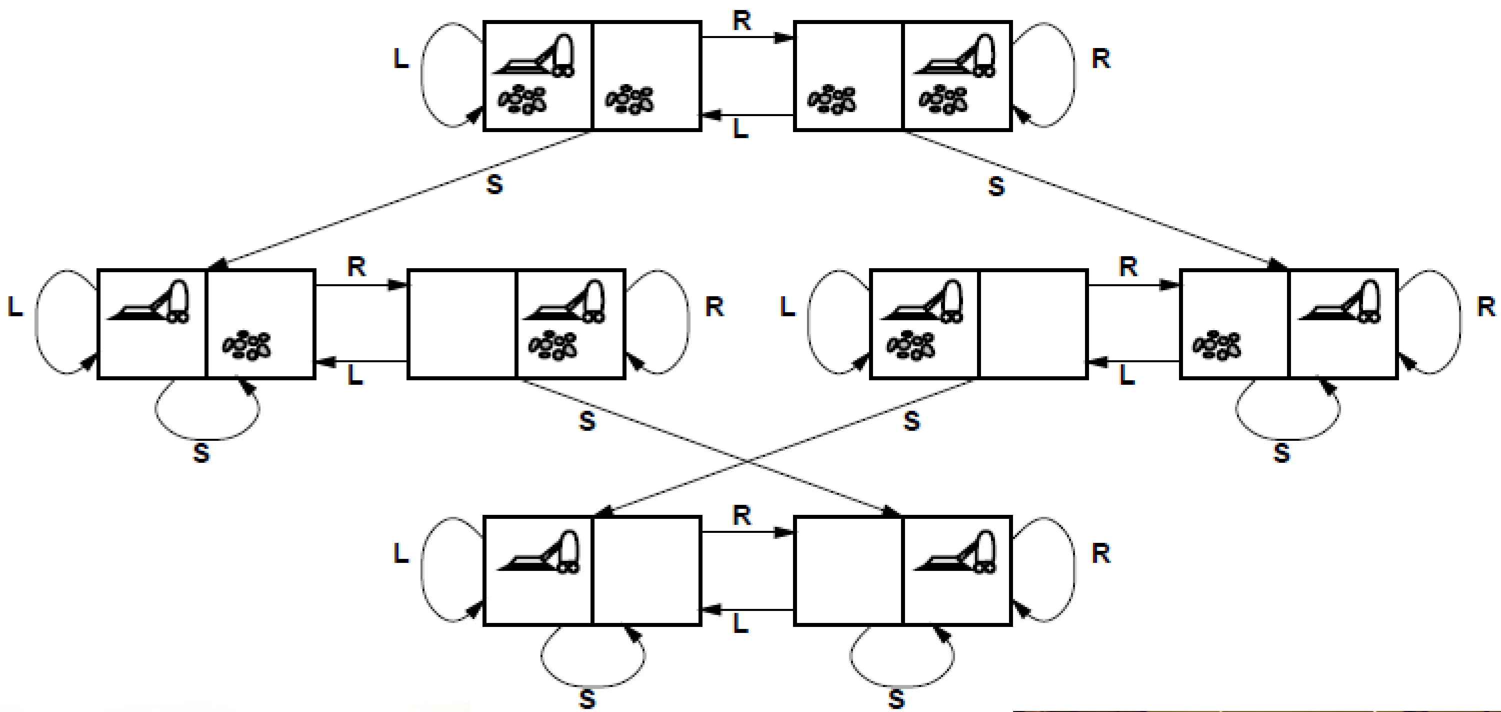
State space graph

- State space graph
  - A directed graph
  - Nodes = states
  - Edges = actions (successor function)
  - For every search problem, there's a corresponding state space graph
  - We can rarely build this graph in memory



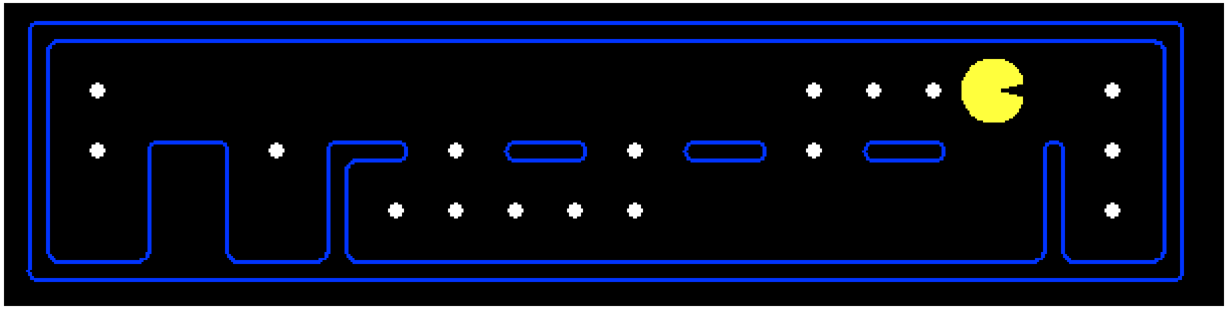
Search graph for a tiny search problem.

State space graph: vacuum world



What's in a state space?

The world state specifies every last detail of the environment

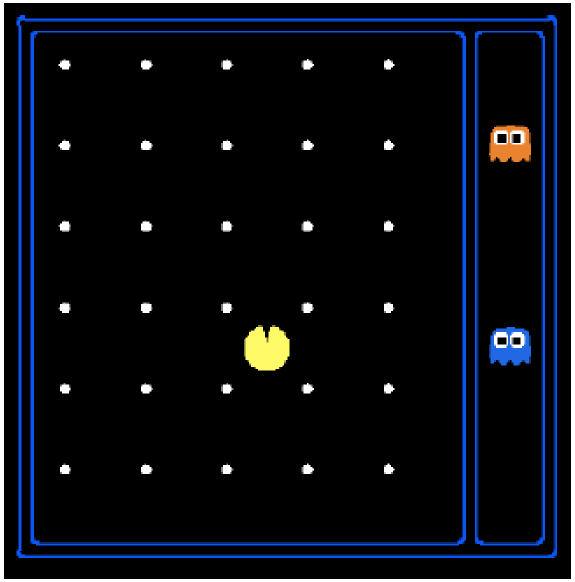


A search state keeps only the details needed (abstraction)

- Problem: Pathing
  - States: (x, y)
  - Actions: NSEW
  - Successor: update location only
  - Goal test: (x,y) = END
- Problem: Eat all dots
  - States: (x, y), dot booleans
  - Actions: NSEW
  - Successor: update location and possibly dot boolean
  - Goal test: dots all false

State space sizes?

- World state:
  - Agent positions: 120 = 10 * 12
  - Food count: 30 = 5 * 6
  - Ghost positions: 12
  - Agent facing: 4 = NSEW



- How many?
  - World states: $120 * 2^{30} * 12^{2} * 4$ = big
  - States for pathing: 120
  - States for eat all dots: $120 * 2^{30}$ = 128,849,018,880

Summary

- Agent: Something that perceives and acts in an environment
- Performance measure: Evaluates the behavior of an agent
- Rational agent: Maximize expected value of performance measure
- Agent types:
  - Simple reflex agents = respond directly to percepts
  - Model-based reflex agents = internal state based on current + past
  - Goal-based agents = act to achieve some goal
  - Utility-based agents = maximize expected "happiness"
  - All agents can improve performance through learning
- Search problems:
  - Components: state space, successor function, start state, goal state
  - Find sequence from start to goal through the state space graph