Lecture 1

Course Introduction
Artificial Intelligence

Marco Chiarandini

Deptartment of Mathematics & Computer Science
University of Southern Denmark

Slides by Stuart Russell and Peter Norvig

Outline

Outline

Course Presentation

- Schedule (28 lecture hours):
  - Tuesday 16:00-17.45
  - Thursday 16:00-17.45
  - Last lecture: Thursday, 17th December, 2009
- Communication tools
  - Course Public Web Site (Ws) ⇔ Blackboard (Bb)
  - Announcements (Bb)
    (link from `http://www.imada.sdu.dk/~marco/DM533/`)
  - Discussion board (Bb)
  - Personal email (Bb)
- Course content

Course Presentation

Course Material:

- Text Book:

  B1 S. Russell and P. Norvig. **Artificial Intelligence: A Modern Approach**. Prentice Hall, 2003
- Notes and Slides from the web site
- Assignments

Evaluation:

- 2-3 Passed/Failed Assignments
  A failed assignement must be resumbitted one week after the evaluation.
- Written Exam

Outline

Outline

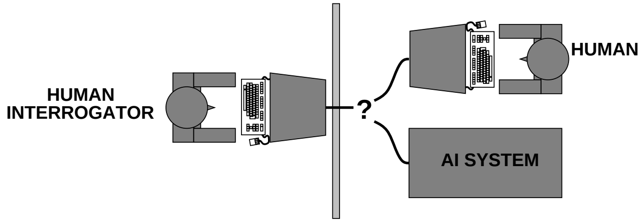
◇ What is AI?
◇ A brief history
◇ The state of the art

What is AI?

Systems that think like humans	**Systems that think rationally**
Systems that act like humans	**Systems that act rationally**

Acting humanly: The Turing test

Turing (1950) "Computing machinery and intelligence":
- ◇ "Can machines think?" ⟶ "Can machines behave intelligently?"
- ◇ Operational test for intelligent behavior: the Imitation Game



- ◇ Predicted that by 2000, a machine might have a 30% chance of fooling a lay person for 5 minutes
- ◇ Anticipated all major arguments against AI in following 50 years
- ◇ Suggested major components of AI: knowledge, reasoning, language understanding, learning

Problem: Turing test is not **reproducible**, **constructive**, or amenable to **mathematical analysis**

ALICE

```
Human: Hi Alice
ALICE: Hello there.
Human: How are you?
ALICE: My logic and cognitive functions are normal.
Human: mmh.
ALICE: What do you really want to ask me?
Human:. and all the rest?
ALICE: Perhaps there are a few exception.
Human: Which for example?
ALICE: The first one, I think.
Human: feelings?
ALICE: Electronic brains like me have no emotions.
Human: are you sure?
ALICE: I am as sure as I can be.
Human: How can you know what feelings are if you never had them?
ALICE: Knowledge is of two kinds: that which we learn from the senses
and that which is true "a priori".
```


Thinking humanly: Cognitive Science

1960s "cognitive revolution": information-processing psychology replaced prevailing orthodoxy of behaviorism (mind is just the behaviour of the body)

Requires scientific theories of internal activities of the brain
- – What level of abstraction? "Knowledge" or "circuits"?
- – How to validate? Requires
  1) Predicting and testing behavior of human subjects (top-down)
  or 2) Direct identification from neurological data (bottom-up)

Both approaches (roughly, Cognitive Science and Cognitive Neuroscience) are now distinct from AI

Both share with AI the following characteristic:
the available theories do not explain (or engender) anything resembling human-level general intelligence

Hence, all three fields share one principal direction!

Thinking rationally: Laws of Thought

Normative (or prescriptive) rather than descriptive

Aristotle: what are correct arguments/thought processes?

Several Greek schools developed various forms of logic:
notation and **rules of derivation** for thoughts;
may or may not have proceeded to the idea of mechanization

Direct line through mathematics and philosophy to modern AI

Problems:
1) Not all intelligent behavior is mediated by logical deliberation
2) What is the purpose of thinking? What thoughts **should** I have out of all the thoughts (logical or otherwise) that I **could** have?

Acting rationally

Rational behavior: doing the right thing

The right thing: that which is expected to maximize goal achievement, given the available information

Doesn't necessarily involve thinking—e.g., blinking reflex—but thinking should be in the service of rational action

Aristotle (Nicomachean Ethics):
Every art and every inquiry, and similarly every action and pursuit, is thought to aim at some good

Rational agents

An agent is an entity that perceives and acts

This course is about designing rational agents

Abstractly, an agent is a function from percept histories to actions:

$$f : \mathcal{P}^* \rightarrow \mathcal{A}$$

For any given class of environments and tasks, we seek the agent (or class of agents) with the best performance

Caveat: **computational limitations make perfect rationality unachievable**
$\rightarrow$ design best program for given machine resources

Potted history of AI

1943	McCulloch & Pitts: Boolean circuit model of brain
1950	Turing's "Computing Machinery and Intelligence"
1952–69	Look, Ma, no hands!
1950s	Early AI programs, including Samuel's checkers program, Newell & Simon's Logic Theorist, Gelernter's Geometry Engine
1956	Dartmouth meeting: "Artificial Intelligence" adopted
1965	Robinson's complete algorithm for logical reasoning
1966–74	AI discovers computational complexity Neural network research almost disappears
1969–79	Early development of knowledge-based systems
1980–88	Expert systems industry booms
1988–93	Expert systems industry busts: "AI Winter"
1985–95	Neural networks return to popularity
1988–	Resurgence of probability; general increase in technical depth "Nouvelle AI": ALife, GAs, soft computing
1995–	Agents, agents, everywhere ...
2003–	Human-level AI back on the agenda

Success stories

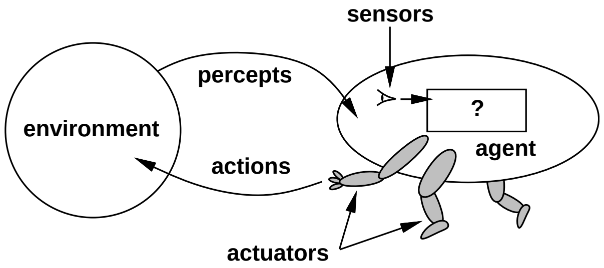
- Autonomous planning and scheduling
- Game playing
- Autonomous control
- Diagnosis
- Logistics Planning
- Robotics
- Language understanding and problem solving

Outline

Outline

- Agents and environments
- Rationality
- PEAS (Performance measure, Environment, Actuators, Sensors)
- Environment types
- Agent types

Agents and environments



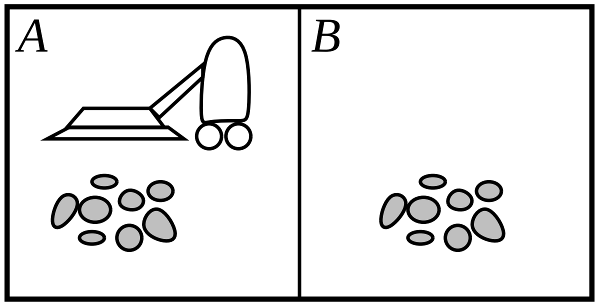
Agents include humans, robots, softbots, thermostats, etc.

The agent function maps from percept histories to actions:

$$f : \mathcal{P}^* \rightarrow \mathcal{A}$$

The agent program runs on the physical architecture to produce f

Vacuum-cleaner world



Percepts: location and contents, e.g., $[A, Dirty]$
Actions: $Left$, $Right$, $Suck$, $NoOp$

A vacuum-cleaner agent

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

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*

What is the **right** function?
Can it be implemented in a small agent program?

Rationality

Fixed performance measure evaluates 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?

A rational agent chooses whichever action maximizes the expected value of the performance measure given the percept sequence to date

Rational $\neq$ omniscient
- percepts may not supply all relevant information

Rational $\neq$ clairvoyant
- action outcomes may not be as expected

Hence, rational $\neq$ successful

Rational $\implies$ exploration, learning, autonomy

PEAS

To design a rational agent, we must specify the task environment
Consider, e.g., the task of designing an automated taxi:
Performance measure??
Environment??
Actuators??
Sensors??

PEAS

To design a rational agent, we must specify the task environment
Consider, e.g., the task of designing an automated taxi:
Performance measure?? safety, destination, profits, legality, comfort, . . .
Environment?? streets/freeways, traffic, pedestrians, weather, . . .
Actuators?? steering, accelerator, brake, horn, speaker/display, . . .
Sensors?? video, accelerometers, gauges, engine sensors, keyboard, GPS, . . .

Internet shopping agent

Performance measure??
Environment??
Actuators??
Sensors??

Internet shopping agent

Performance measure?? price, quality, appropriateness, efficiency
Environment?? current and future WWW sites, vendors, shippers
Actuators?? display to user, follow URL, fill in form
Sensors?? HTML pages (text, graphics, scripts)

Environment types

	Solitaire	Backgammon	Internet shopping	Taxi
Observable??				
Deterministic??				
Episodic??				
Static??				
Discrete??				
Single-agent??				

Environment types

	Solitaire	Backgammon	Internet shopping	Taxi
Observable??	Yes	Yes	No	No
Deterministic??				
Episodic??				
Static??				
Discrete??				
Single-agent??				

Environment types

	Solitaire	Backgammon	Internet shopping	Taxi
Observable??	Yes	Yes	No	No
Deterministic??	Yes	No	Partly	No
Episodic??				
Static??				
Discrete??				
Single-agent??				

Environment types

	Solitaire	Backgammon	Internet shopping	Taxi
Observable??	Yes	Yes	No	No
Deterministic??	Yes	No	Partly	No
Episodic??	No	No	No	No
Static??				
Discrete??				
Single-agent??				

Environment types

	Solitaire	Backgammon	Internet shopping	Taxi
Observable??	Yes	Yes	No	No
Deterministic??	Yes	No	Partly	No
Episodic??	No	No	No	No
Static??	Yes	Semi	Semi	No
Discrete??				
Single-agent??				

Environment types

	Solitaire	Backgammon	Internet shopping	Taxi
Observable??	Yes	Yes	No	No
Deterministic??	Yes	No	Partly	No
Episodic??	No	No	No	No
Static??	Yes	Semi	Semi	No
Discrete??	Yes	Yes	Yes	No
Single-agent??				

Environment types

	Solitaire	Backgammon	Internet shopping	Taxi
Observable??	Yes	Yes	No	No
Deterministic??	Yes	No	Partly	No
Episodic??	No	No	No	No
Static??	Yes	Semi	Semi	No
Discrete??	Yes	Yes	Yes	No
Single-agent??	Yes	No	Yes (except auctions)	No

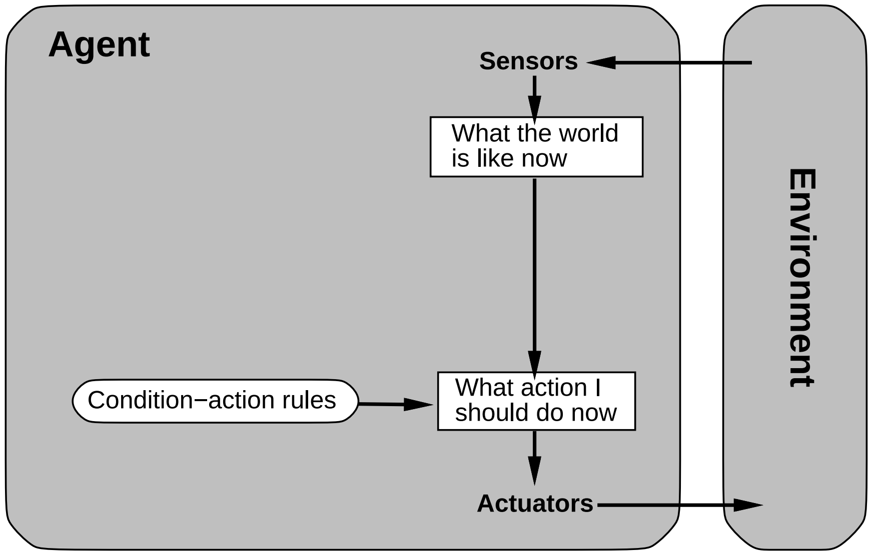
The environment type largely determines the agent design
The real world is (of course) partially observable, stochastic, sequential, dynamic, continuous, multi-agent

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

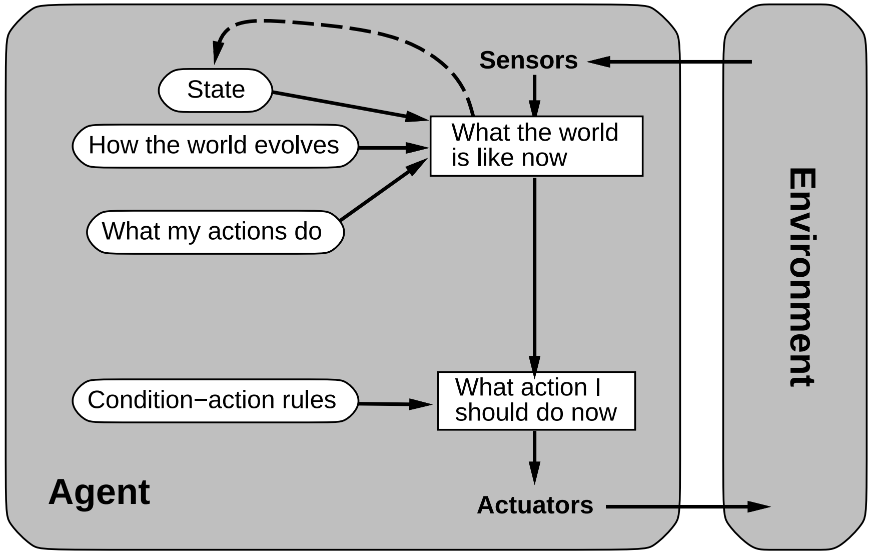
  if *status* = *Dirty* **then return** *Suck*
  else if *location* = *A* **then return** *Right*
  else if *location* = *B* **then return** *Left*

```
loc_A, loc_B = (0, 0), (1, 0) # The two locations for the Vacuum world

class ReflexVacuumAgent(Agent):
    "A reflex agent for the two-state vacuum environment."

    def __init__(self):
        Agent.__init__(self)
        def program((location, status)):
            if status == 'Dirty': return 'Suck'
            elif location == loc_A: return 'Right'
            elif location == loc_B: return 'Left'
        self.program = program
```

Reflex agents with state



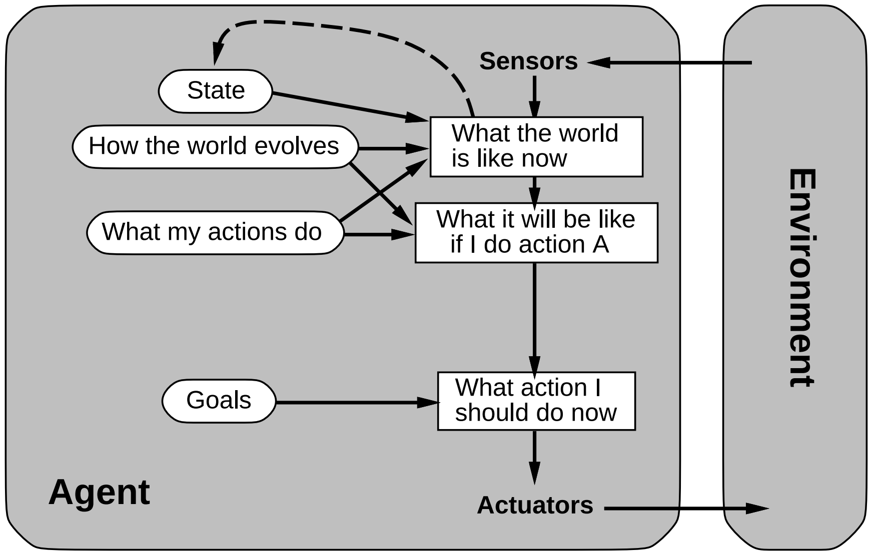
Example

function Reflex-Vacuum-Agent([*location*,*status*]) **returns** an action
static: *last_A*, *last_B*, numbers, initially ∞

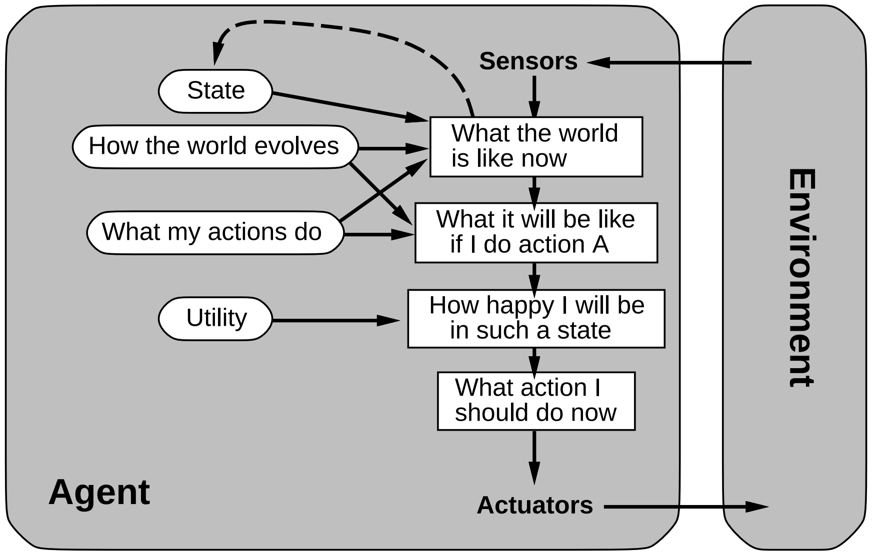
if *status* = *Dirty* **then** . . .

```
class ModelBasedVacuumAgent(Agent):
    "An agent that keeps track of what locations are clean or dirty."
    def __init__(self):
        Agent.__init__(self)
        model = {loc_A: None, loc_B: None}
        def program((location, status)):
            "Same as ReflexVacuumAgent, except if everything is clean, do
            model[location] = status ## Update the model here
            if model[loc_A] == model[loc_B] == 'Clean': return 'NoOp'
            elif status == 'Dirty': return 'Suck'
            elif location == loc_A: return 'Right'
            elif location == loc_B: return 'Left'
        self.program = program
```

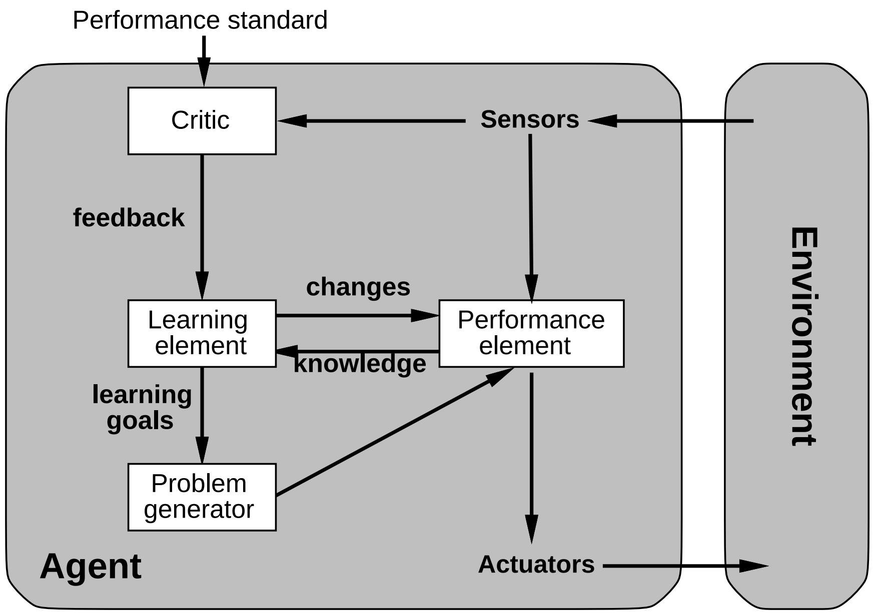

Goal-based agents



Utility-based agents



Learning agents



Summary

Agents interact with environments through actuators and sensors

The agent function describes what the agent does in all circumstances

The performance measure evaluates the environment sequence

A perfectly rational agent maximizes expected performance

Agent programs implement (some) agent functions

PEAS descriptions define task environments

Environments are categorized along several dimensions:
observable? deterministic? episodic? static? discrete? single-agent?

Several basic agent architectures exist:
reflex, reflex with state, goal-based, utility-based

Outline

1. Course Introduction
2. Introduction to AI
3. Intelligent Agents
4. Problem Solving and Search

Problem-solving agents

Restricted form of general agent:

```
function Simple-Problem-Solving-Agent( percept) returns an action
    static: seq, an action sequence, initially empty
            state, some description of the current world state
            goal, a goal, initially null
            problem, a problem formulation

    state ← Update-State(state, percept)
    if seq is empty then
        goal ← Formulate-Goal(state)
        problem ← Formulate-Problem(state, goal)
        seq ← Search( problem)
    action ← Recommendation(seq, state)
    seq ← Remainder(seq, state)
    return action
```

Note: this is offline problem solving; solution executed "eyes closed."
Online problem solving involves acting without complete knowledge.

Example: Romania

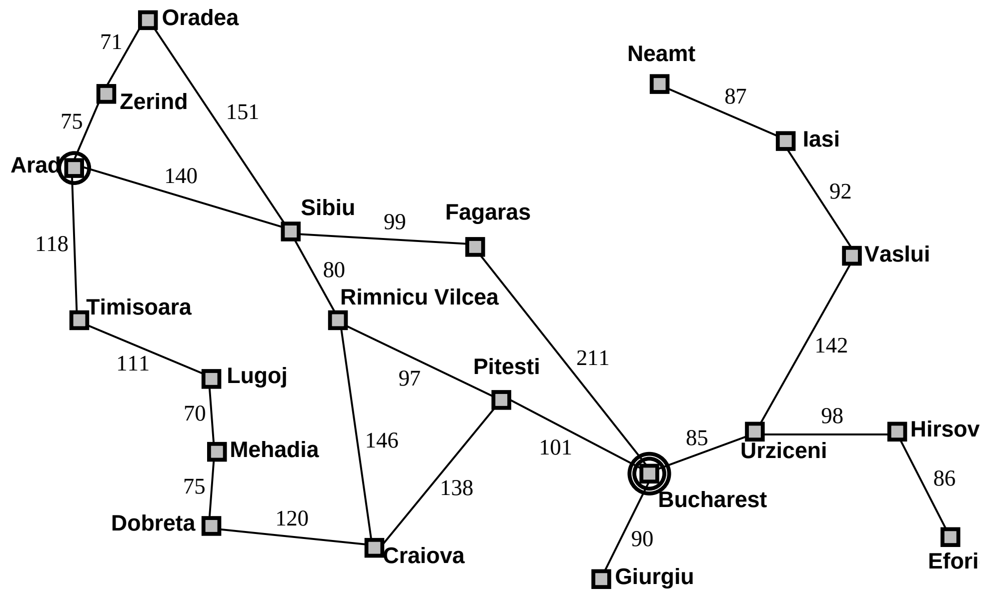
On holiday in Romania; currently in Arad.
Flight leaves tomorrow from Bucharest

Formulate goal:
 be in Bucharest

Formulate problem:
 states: various cities
 actions: drive between cities

Find solution:
 sequence of cities, e.g., Arad, Sibiu, Fagaras, Bucharest

Example: Romania



Problem types

Deterministic, fully observable $\Longrightarrow$ single-state problem
 Agent knows exactly which state it will be in; solution is a sequence

Non-observable $\Longrightarrow$ conformant problem
 Agent may have no idea where it is; solution (if any) is a sequence

Nondeterministic and/or partially observable $\Longrightarrow$ contingency problem
 percepts provide **new** information about current state
 solution is a contingent plan or a policy
 often **interleave** search, execution

Unknown state space $\Longrightarrow$ exploration problem (“online”)

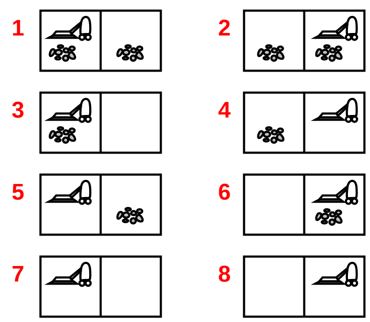
Example: vacuum world

Single-state, start in #5. Solution??
$[Right, Suck]$

Conformant, start in $\{1,2,3,4,5,6,7,8\}$
e.g., $Right$ goes to $\{2,4,6,8\}$. Solution??
$[Right, Suck, Left, Suck]$

Contingency, start in #5
Murphy's Law: $Suck$ can dirty a clean carpet
Local sensing: dirt, location only.
Solution??

$[Right,$ **if** $dirt$ **then** $Suck]$



Single-state problem formulation

A problem is defined by four items:

initial state   e.g., "at Arad"

successor function $S(x)$ = set of action–state pairs
e.g., $S(Arad) = \{\langle Arad \to Zerind, Zerind\rangle, \ldots\}$

goal test, can be
explicit, e.g., x = "at Bucharest"
implicit, e.g., $NoDirt(x)$

path cost (additive)
e.g., sum of distances, number of actions executed, etc.
$c(x, a, y)$ is the step cost, assumed to be ≥ 0

A solution is a sequence of actions
leading from the initial state to a goal state

Selecting a state space

Real world is complex
⇒ state space must be **abstracted** for problem solving

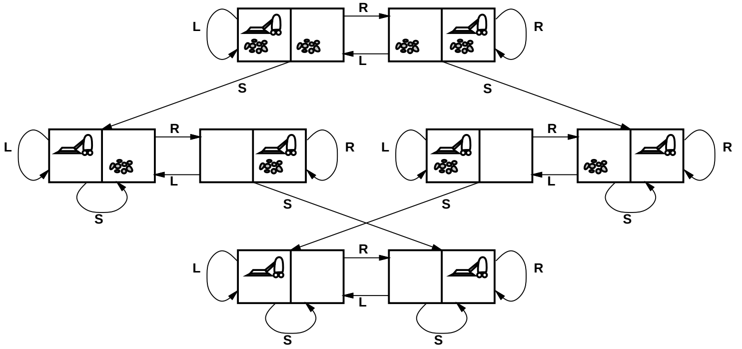
(Abstract) state = set of real states

(Abstract) action = complex combination of real actions
e.g., "Arad → Zerind" represents a complex set
of possible routes, detours, rest stops, etc.
For guaranteed realizability, **any** real state "in Arad"
must get to some real state "in Zerind"

(Abstract) solution =
set of real paths that are solutions in the real world

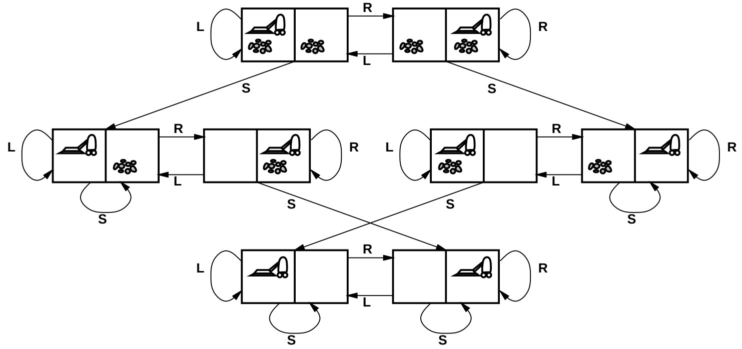
Each abstract action should be "easier" than the original problem!

Example: vacuum world state space graph



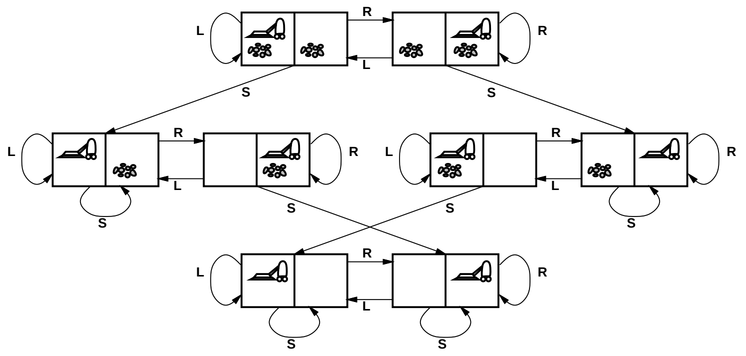
states??
actions??
goal test??
path cost??

Example: vacuum world state space graph



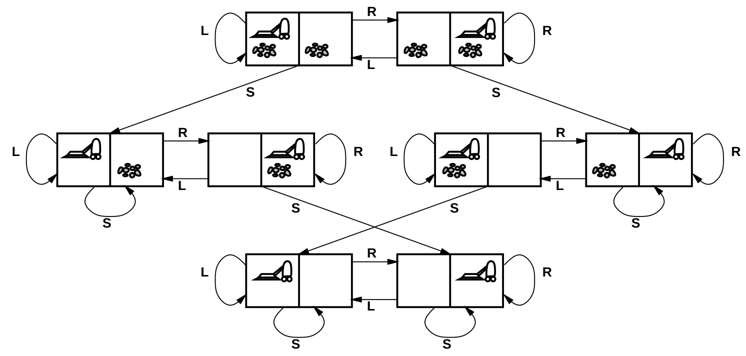
states??: integer dirt and robot locations (ignore dirt amounts etc.)
actions??
goal test??
path cost??

Example: vacuum world state space graph



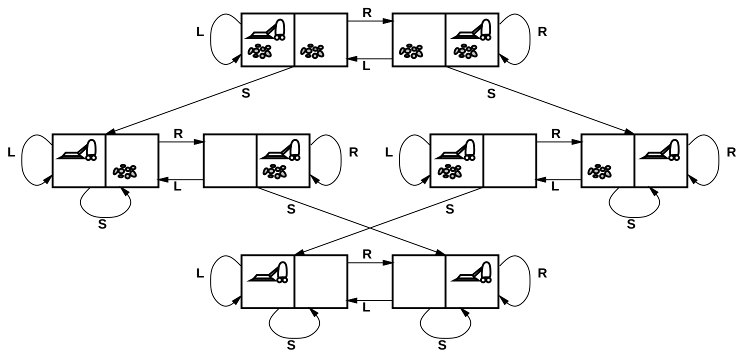
states??: integer dirt and robot locations (ignore dirt amounts etc.)
actions??: $Left$, $Right$, $Suck$, $NoOp$
goal test??
path cost??

Example: vacuum world state space graph



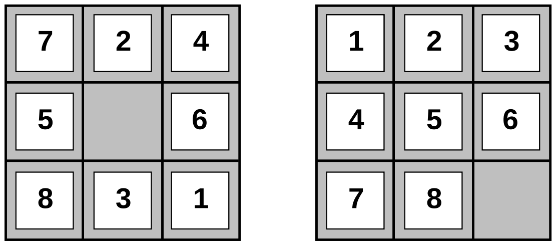
states??: integer dirt and robot locations (ignore dirt amounts etc.)
actions??: $Left$, $Right$, $Suck$, $NoOp$
goal test??: no dirt
path cost??

Example: vacuum world state space graph



states??: integer dirt and robot locations (ignore dirt amounts etc.)
actions??: $Left$, $Right$, $Suck$, $NoOp$
goal test??: no dirt
path cost??: 1 per action (0 for $NoOp$)

Example: The 8-puzzle

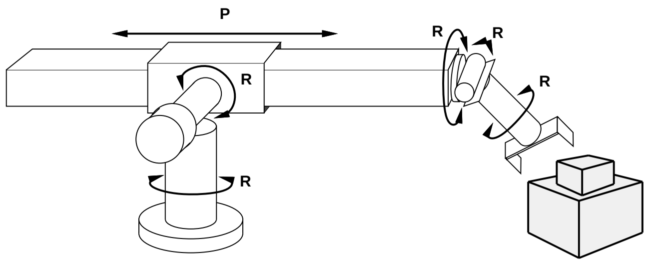


Start State   Goal State

states??: integer locations of tiles (ignore intermediate positions)
actions??: move blank left, right, up, down (ignore unjamming etc.)
goal test??: = goal state (given)
path cost??: 1 per move

[Note: optimal solution of n-Puzzle family is NP-hard]

Example: robotic assembly



states??: real-valued coordinates of robot joint angles
parts of the object to be assembled
actions??: continuous motions of robot joints
goal test??: complete assembly **with no robot included!**
path cost??: time to execute