Fundamentals of Artificial Intelligence
Chapter 02: **Intelligent Agents**

Roberto Sebastiani

DISI, Università di Trento, Italy – roberto.sebastiani@unitn.it

https://disi.unitn.it/rseba/DIDATTICA/fai_2025/

Teaching assistant:

Paolo Morettin, paolo.morettin@unitn.it, https://paolomorettin.github.io/

M.S. Course "Artificial Intelligence Systems", academic year 2024-2025

Last update: Wednesday 10th September, 2025, 16:20

Outline

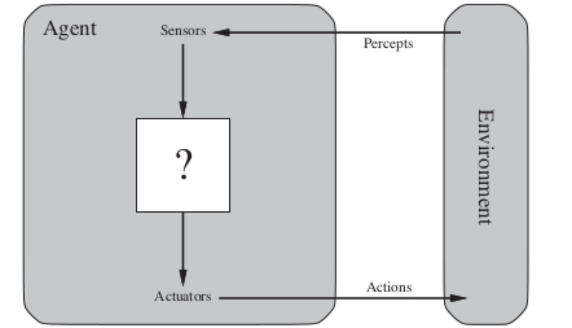
Outline

Agents and Environments

Agents

An agent is any entity that can be viewed as:

- perceiving its environment through sensors, and
- acting upon that environment through actuators.



(© S. Russell & P. Norwig, AIMA)

Agents and Environments [cont.]

Agents

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

- human:
  - perceives: with eyes, ears, nose, hands, ...,
  - acts: with voice, hands, arms, legs, ...
- robot:
  - perceives: with video-cameras, infra-red sensors, radar, ...
  - acts: with wheels, motors, ...
- softbot:
  - perceives: receiving keystrokes, files, network packets, ...
  - acts: displaying on the screen, writing files, sending network packets
- thermostat:
  - perceives: with heat sensor
  - acts: electric impulses to valves, devices, ...

Key concepts

Percept and Percept sequences

- percept: the collection of agent's perceptual inputs at any given instant
- percept sequence: the complete history of everything the agent has ever perceived

An agent's choice of action at any given instant

- can depend on the entire percept sequence observed to date
- does not depend on anything it hasn't perceived

Remark

An agent can perceive its own actions, but not always it effects.

Key concepts [cont.]

Agent function

An agent's behavior is described by the agent function $f : P^* \longmapsto A$ which maps any given percept sequence into an action.

- ideally, can be seen as a table [*percept sequence*, *action*]

Agent program

Internally, the agent function for an artificial agent is implemented by an agent program.

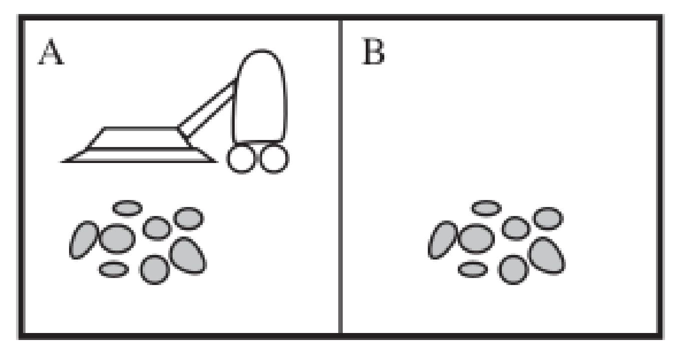
Note: Agent function vs. agent program

- The agent function is an abstract mathematical description
  - possibly-infinite description
- The agent program is a concrete implementation of the agent function
  - finite description
  - runs on the physical architecture to produce the agent function f

Example

A very-simple vacuum cleaner

- Environment: squares A and B
- Percepts: location ($\{A, B\}$) and content ($\{Dirty, Clean\}$)
  - e.g. [A, Dirty]
- Actions: $\{left, right, suck, no\_op\}$



(© S. Russell & P. Norwig, AIMA)

Example [cont.]

A simple agent function

If the current square is dirty, then suck; otherwise, move to the other square.

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

Note: this agent function depends only on the last percept, not on the whole percept sequence.

Example [cont.]

Corresponding agent program

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*

Outline

Rational Agents

Main question

What is a rational agent?

- Intuition: a rational agent is one that “does the right thing”
  - i.e., every entry in the agent function-table is filled out correctly
- Problem: What is the right thing?
- Approximated answer: the most “succesfull” thing:
  - In a given environment, according to the percept sequence it receives, an agent generates a sequence of actions, ...
  - ... causing the environment to go through a sequence of states.
  - If such sequence is desirable, then the agent has performed well.

$\Longrightarrow$ We need a performance measure to evaluate any sequence of environment states

Rational Agents [cont.]

What is rational at any given time depends on four ingredients:

- The performance measure that defines the criterion of success
- The agent's prior knowledge of the environment
- The actions that the agent can perform
- The agent's percept sequence to date (from sensors)

Definition of a rational agent

For each possible percept sequence, a rational agent should select an action that is expected to maximize its performance measure, given the evidence provided by the percept sequence and whatever built-in knowledge the agent has.

Rational Agents: Example

The simple vacuum-cleaner agent

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** $Left$

(© S. Russell & P. Norwig, AIMA)

Under the following assumptions:

- Performance measure:
  one point for each clean square at each time step, over 1000 time steps
- Environment knowledge:
  - "geography" known a priori,
  - dirt distribution and agent initial location unknown
  - [clean squares cannot become dirty again]
- Perception: self location, presence of dirt
- Actions: Left, Right, Suck

Is the above-described agent rational? $\Longrightarrow$ Yes! (provided the given performence measure)

Beware: if a penalty for each move is given, the agent behaves poorly
$\Longrightarrow$ better agent: do nothing once it is sure all the squares are clean

Rationality vs. Omniscience vs. Perfection

Remark

- Rationality $\neq$ Omniscience!
  - An omniscient agent knows for sure the outcome of its actions
    $\Longrightarrow$ omniscience impossible in reality
  - A rational agent may only know "up to a reasonable confidence"
    (e.g., when crossing a road, what if something falling from a plane flattens you? if so, would you be considered irrational?)
- Rational behaviour is not perfect behaviour!
  - perfection maximizes actual performance
  - (given uncertainty) rationality maximizes expected performance

Information Gathering, Learning, Autonomy

Rationality requires other important features

- Information gathering/exploration:
  - the rational choice depends only on the percept sequence to date
    $\Longrightarrow$ actions needed in order to modify future percepts
  - Ex: look both ways before crossing a busy road
- Learning:
  - agent's prior knowledge of the environment incomplete
    $\Longrightarrow$ learning from percept sequences improves & augments it
  - Ex: a baby learns from trial&errors the right movements to walk
- Being Autonomous:
  - prior knowledge may be partial/incorrect or evolving

  $\Longrightarrow$ learn to compensate for partial or incorrect prior knowledge
  - Ex: a child learns how to climb a tree

Information gathering, learning, autonomy play an essential role in AI.

Outline

Task Environments

PEAS Description of Task Environments

- To design a rational agent we must specify its task environment
  - i.e. the “problems” to which rational agents are the “solutions”
- Task environment described in terms of four elements (“PEAS”):
  - Performance measure
  - Environment
  - Actuators
  - Sensors

Simple Example: Simple Vacuum Cleaner

- Performance measure: 1 point per clean square per time step
- Environment: squares A and B, possibly dirty
- Actuators: move left/right, suck
- Sensors: self location, presence of dirt

Task Environments [cont.]

Complex Example: Autonomous Taxi

- Performance measure: safety, destination, profits, comfort, ...
- Environment: streets/freeways, other traffic, pedestrians, ...
- Actuators: steering, accelerator, brake, horn, speaker/display, ...
- Sensors: video, sonar, speedometer, engine sensors, GPS, ...

Remark

Some goals to be measured may conflict!

- e.g. profits vs. safety, profits vs. comfort, ...
  $\Longrightarrow$ tradeoffs are required

Task Environments: Examples

Agent Type	Performance Measure	Environment	Actuators	Sensors
Medical diagnosis system	Healthy patient, reduced costs	Patient, hospital, staff	Display of questions, tests, diagnoses, treatments, referrals	Keyboard entry of symptoms, findings, patient's answers
Satellite image analysis system	Correct image categorization	Downlink from orbiting satellite	Display of scene categorization	Color pixel arrays
Part-picking robot	Percentage of parts in correct bins	Conveyor belt with parts; bins	Jointed arm and hand	Camera, joint angle sensors
Refinery controller	Purity, yield, safety	Refinery, operators	Valves, pumps, heaters, displays	Temperature, pressure, chemical sensors
Interactive English tutor	Student's score on test	Set of students, testing agency	Display of exercises, suggestions, corrections	Keyboard entry

Outline

Properties of Task Environments

Task environments can be categorized along six dimensions:

- Single-agent vs. multi-agent
- Fully observable vs. partially observable
- Deterministic vs. stochastic
- Episodic vs. sequential
- Static vs. dynamic
- Discrete vs. continuous

Properties of Task Environments [cont.]

Single-agent vs. multi-agent

- A task environment is multi-agent iff contains other agents who are also maximizing some performance measure that depends on the current agent's actions
  - latest condition essential
  - distinction between single- and multi-agent sometimes subtle
- Two important cases
  - competitive multi-agent environment:
    other agents' goals conflict with, or even oppose to, the agent's goals
    - Ex: chess, war scenarios, taxi driving (compete for parking lot), ...
  - cooperative multi-agent environment:
    other agents' goals coincide in full, or in part, with the agent's goals
    - Ex: ants' nest, factory, taxi driving (avoid collisions), ...
- Different design problems for multi-agent wrt. single-agent
  - competitive: randomized behaviour often rational (unpredictable)
  - collaborative: communication with other agents often rational

Properties of Task Environments [cont.]

Fully observable vs. partially observable

- A task environment is fully observable iff the sensors detect the complete state of the environment

  $\Longrightarrow$ no need to maintain internal state to keep track of the environment
- A task environment is effectively fully observable iff the sensors detect all aspects of the state of the environments that are relevant to the choice of action
  - ”relevant” depends on the performance measure
- A task environment may be partially observable (Ex: Taxi driving):
  - noisy and inaccurate sensors
  - parts of the state are not accessible for sensors
- A task environment might be even unobservable (no sensors)
  - e.g. fully-deterministic actions

Properties of Task Environments [cont.]

Deterministic vs. stochastic

- A task environment is deterministic iff its next state is completely determined by its current state and by the action of the agent. (Ex: a crossword puzzle).
- If not so:
  - A t.e. is stochastic if uncertainty about outcomes is quantified in terms of probabilities (Ex: dice, poker game, component failure,...)
  - A t.e. is nondeterministic iff actions are characterized by their possible outcomes, but no probabilities are attached to them

In a multi-agent environment we ignore uncertainty that arises from the actions of other agents (Ex: chess is deterministic even though each agent is unable to predict the actions of the others).

A partially observable environment could appear to be stochastic.
$\Longrightarrow$ for practical purposes, when it is impossible to keep track of all the unobserved aspects, they must be treated as stochastic.
(Ex: Taxi driving)

Properties of Task Environments [cont.]

Episodic vs. sequential

- In an episodic task environment
  - the agent's experience is divided into atomic episodes
  - in each episode the agent receives a percept and then performs a single action

  $\Longrightarrow$ episodes do not depend on the actions taken in previous episodes, and they do not influence future episodes
  - Ex: an agent that has to spot defective parts on an assembly line,
- In sequential environments the current decision could affect future decisions
  $\Longrightarrow$ actions can have long-term consequences
  - Ex: chess, taxi driving, ...
- Episodic environments are much simpler than sequential ones
  - No need to think ahead!

Properties of Task Environments [cont.]

Static vs. dynamic

- The task environment is dynamic iff it can change while the agent is choosing an action, static otherwise

  $\Longrightarrow$ agent needs keep looking at the world while deciding an action
  - Ex: crossword puzzles are static, taxi driving is dynamic
- The t.e. is semidynamic if the environment itself does not change with time, but the agent's performance score does
  - Ex: chess with a clock
- Static environments are easier to deal wrt. [semi]dynamic ones

Properties of Task Environments [cont.]

Discrete vs. continuous

- The state of the environment, the way time is handled, and agents percepts & actions can be discrete or continuous
  - Ex: Crossword puzzles: discrete state, time, percepts & actions
  - Ex: Taxi driving: continuous state, time, percepts & actions
  - ...

Properties of Task Environments [cont.]

Note

- The simplest environment is fully observable, single-agent, deterministic, episodic, static and discrete.
  - Ex: simple vacuum cleaner
- Most real-world situations are partially observable, multi-agent, stochastic, sequential, dynamic, and continuous.
  - Ex: taxi driving

Properties of Task Environments [cont.]

Example properties of task Environments

Task Environment	Observable	Agents	Deterministic	Episodic	Static	Discrete
Crossword puzzle	Fully	Single	Deterministic	Sequential	Static	Discrete
Chess with a clock	Fully	Multi	Deterministic	Sequential	Semi	Discrete
Poker	Partially	Multi	Stochastic	Sequential	Static	Discrete
Backgammon	Fully	Multi	Stochastic	Sequential	Static	Discrete
Taxi driving	Partially	Multi	Stochastic	Sequential	Dynamic	Continuous
Medical diagnosis	Partially	Single	Stochastic	Sequential	Dynamic	Continuous
Image analysis	Fully	Single	Deterministic	Episodic	Semi	Continuous
Part-picking robot	Partially	Single	Stochastic	Episodic	Dynamic	Continuous
Refinery controller	Partially	Single	Stochastic	Sequential	Dynamic	Continuous
Interactive English tutor	Partially	Multi	Stochastic	Sequential	Dynamic	Discrete

Properties of the Agent's State of Knowledge

Known vs. unknown

- Describes the agent's (or designer's) state of knowledge about the "laws of physics" of the environment
  - if the environment is known, then the outcomes (or outcome probabilities if stochastic) for all actions are given.
  - if the environment is unknown, then the agent will have to learn how it works in order to make good decisions
- Orthogonal wrt. task-environment properties

Known $\neq$ Fully observable

- a known environment can be partially observable
  (Ex: a solitaire card games)
- an unknown environment can be fully observable
  (Ex: a game I don't know the rules of)

Outline

Agents

Agent = Architecture + Program

- AI Job: design an agent program implementing the agent function
- The agent program runs on some computing device with physical sensors and actuators: the agent architecture
- All agents have the same skeleton:
  - Input: current percepts
  - Output: action
  - Program: manipulates input to produce output

Remark

- the agent function takes the entire percept history as input
- the agent program takes only the current percept as input

$\Longrightarrow$ if the actions need to depend on the entire percept sequence, then the agent will have to remember the percepts

A trivial Agent Program

The Table-Driven Agent

- The table represents explicitly the agent function
  - Ex: the simple vacuum cleaner

function TABLE-DRIVEN-AGENT(*percept*) **returns** an action
  persistent: *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*

Blow-up in table size $\Longrightarrow$ doomed to failure

Agent Types

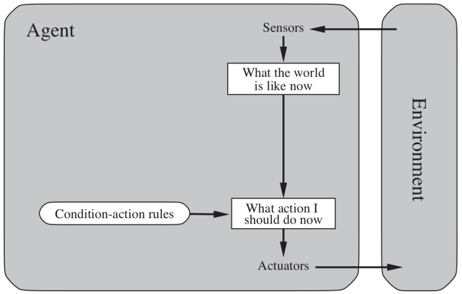
Four basic kinds of agent programs

- Simple-reflex agents
- Model-based reflex agents
- Goal-based agents
- Utility-based agents

All these can be turned into learning agents.

Agent Types: Simple-reflex agent

- Select action on the basis of the current percept only
  - Ex: the simple vacuum-agent
- Implemented through condition-action rules
  - Ex: "**if** car-in-front-is-braking **then** initiate-braking"
  - can be implemented, e.g., in a Boolean circuit
- Large reduction in possible percept/action situations due to ignoring the percept history



(© S. Russell & P. Norwig, AIMA)

Agent Types: Simple-reflex agent [cont.]

Simple-reflex agent program

function SIMPLE-REFLEX-AGENT(*percept*) **returns** an action
  persistent: *rules*, a set of condition–action rules

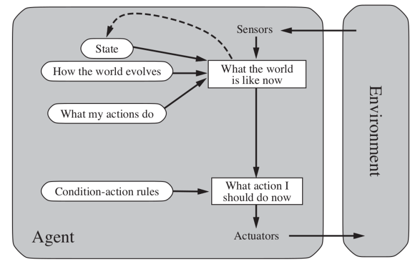
  state ← INTERPRET-INPUT(*percept*)
  rule ← RULE-MATCH(*state*, *rules*)
  action ← *rule*.ACTION
  return *action*

- very simple
- may work only if the environment is fully observable
  - errors, deadlocks or infinite loops may occur otherwise

$\Longrightarrow$ limited applicability

Agent Types: Model-based Reflex Agent

- Idea: To tackle partially-observable environments, keeps track of the part of the world it can't see now
  - maintain internal state depending on the percept history
  - reflects at least some of the unobserved aspects of current state
- To update internal state the agent needs a model of the world:
  - how the world evolves independently of the agent
    - Ex: an overtaking car will soon be closer behind than it was before
  - how the agent's own actions affect the world
    - Ex: turn the steering wheel clockwise $\Longrightarrow$ the car turns to the right



(© S. Russell & P. Norwig, AIMA)

Agent Types: Model-based Reflex Agent [cont.]

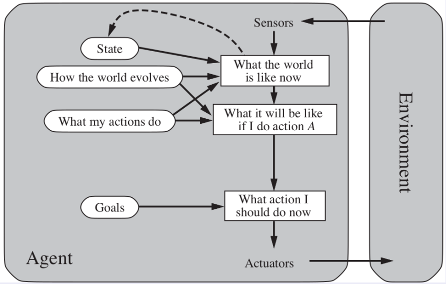
Model-based Agent program

function MODEL-BASED-REFLEX-AGENT(*percept*) **returns** an action
  persistent: *state*, the agent's current conception of the world state
    model, a description of how the next state depends on current state and action
    rules, a set of condition–action rules
    action, the most recent action, initially none

  state ← UPDATE-STATE(*state*, *action*, *percept*, *model*)
  rule ← RULE-MATCH(*state*, *rules*)
  action ← *rule*.ACTION
  return *action*

Agent Types: Model-based Goal-based agent

- The agent needs goal information describing desirable situation
  - Ex: destination for a Taxi driver
- Idea: combine goal with the model to choose actions
- Difficult if long action sequences are required to reach the goal
  $\Longrightarrow$ Typically investigated in search and planning research.
- Major difference: future is taken into account
  - rules are simple condition-action pairs, do not target a goal



(© S. Russell & P. Norwig, AIMA)

Agent Types: Model-based Goal-based Agent [cont.]

Goal-based Agents

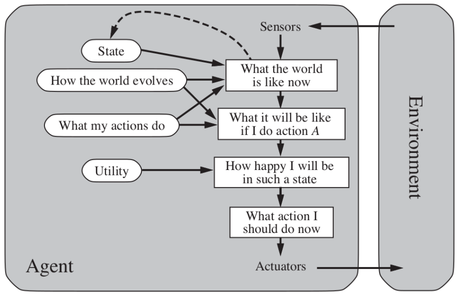
- more flexible:
  - the knowledge that supports its decisions is represented explicitly
  - such knowledge can be modified

    $\Longrightarrow$ all of the relevant behaviors to be altered to suit the new conditions
    - Ex: If it rains, the agent can update its knowledge of how effectively its brakes operate
  - the goal can be modified/updated $\Longrightarrow$ modify its behaviour
    - no need to rewrite all rules from scratch
- more complicate to implement
- may require expensive computation (search, planning)

Agent Types: Utility-based agent

- Goals alone often not enough to generate high-quality behaviors
  - Certain goals can be reached in different ways, of different quality
  - Ex: some routes are quicker, safer, or cheaper than others
- Idea: Add utility function(s) to drive the choice of actions
  - maps a (sequence of) state(s) onto a real number
    $\Longrightarrow$ actions are chosen which maximize the utility function
  - under uncertainty, maximize the expected utility function

$\Longrightarrow$ utility function = internalization of performance measure



(© S. Russell & P. Norwig, AIMA)

Agent Types: Utility-based Agent [cont.]

Utility-based Agents

- advantages wrt. goal-based:
  - with conflicting goals, utility specifies and appropriate tradeoff
  - with several goals none of which can be achieved with certainty, utility selects proper tradeoff between importance of goals and likelihood of success
- still complicate to implement
- require sophisticated perception, reasoning, and learning
- may require expensive computation

Agent Types: Learning

Problem

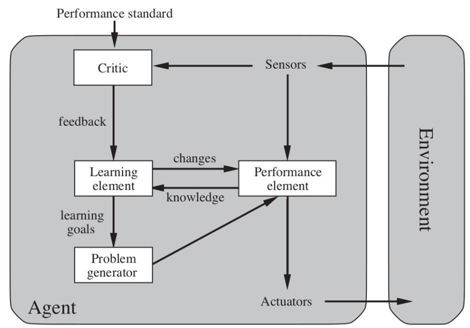
- Previous agent programs describe methods for selecting actions
  - How are these agent programs programmed?
  - Programming by hand inefficient and ineffective!
  - Solution: build learning machines and then teach them (rather than instruct them)
  - Advantage: robustness of the agent program toward initially-unknown environments

Agent Types: Learning

Learning Agent Types: components

Performance element: selects actions based on percepts

- Corresponds to the previous agent programs

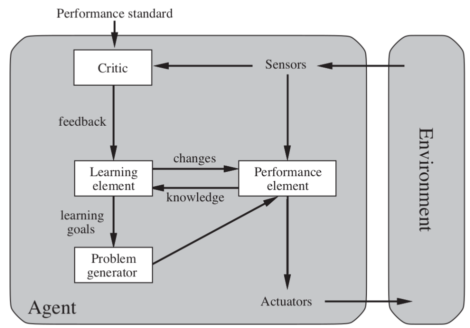


(© S. Russell & P. Norwig, AIMA)

Agent Types: Learning

Learning Agent Types: components

Critic tells how the agent is doing wrt. performance standard



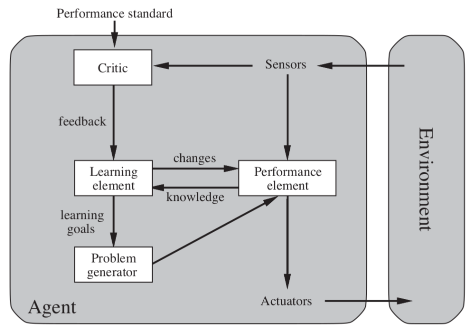
(© S. Russell & P. Norwig, AIMA)

Agent Types: Learning

Learning Agent Types: components

Learning element: introduces improvements

- uses feedback from the critic on how the agent is doing
- determines improvements for the performance element



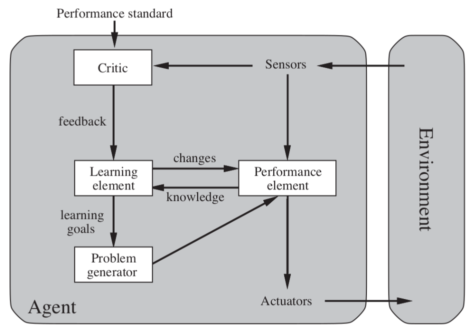
(© S. Russell & P. Norwig, AIMA)

Agent Types: Learning

Learning Agent Types: components

Problem generator: suggests actions that will lead to new and informative experiences

- forces exploration of new stimulating scenarios



(© S. Russell & P. Norwig, AIMA)

Learning Agent Types: Example

Taxi Driving

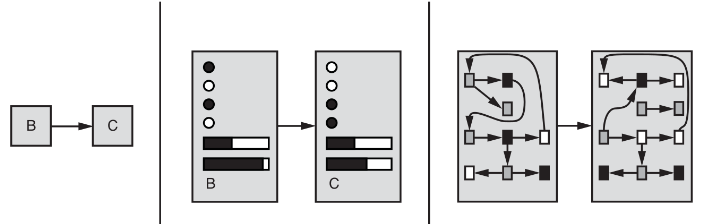
- After the taxi makes a quick left turn across three lanes, the critic observes the shocking language used by other drivers.
- From this experience, the learning element formulates a rule saying this was a bad action.
- The performance element is modified by adding the new rule.
- The problem generator might identify certain areas of behavior in need of improvement, and suggest trying out the brakes on different road surfaces under different conditions.

Outline

Representations

Representations of states and transitions

- Three ways to represent states and transitions between them:
  - atomic: a state is a black box with no internal structure
  - factored: a state consists of a vector of attribute values
  - structured: a state includes objects, each of which may have attributes of its own as well as relationships to other objects
- increasing expressive power and computational complexity
- reality represented at different levels of abstraction



(a) Atomic (b) Factored (b) Structured

Representations [cont.]

Atomic Representations

- each state of the world is indivisible
  - no internal structure
- state: one among a collection of discrete state values
  - Ex: find driving routes: $\{$*Trento*, *Rovereto*, *Verona*, ...$\}$

  $\Longrightarrow$ only property: be identical to or different from another state
- very high level of abstraction

  $\Longrightarrow$ lots of details ignored
- The algorithms underlying
  - search and game-playing
  - hidden Markov models
  - Markov decision processes

  all work with atomic representations (or treat it as such)

Representations [cont.]

Factored Representation

- Each state represented in terms of a vector of attribute values
  - Ex: $\langle zone, \{dirty, clean\}\rangle$, $\langle town, speed\rangle$
- State: combination of attribute values
  - Ex: $\langle A, dirty\rangle$, $\langle Trento, 40kmh\rangle$
- Distinct states may share the values of some attribute
  - Ex: $\langle Trento, 40kmh\rangle$ and $\langle Trento, 47kmh\rangle$
  - identical iff all attribute have the same values
    $\Longrightarrow$ must differ for at least one value to be different
- Can represent uncertainty (e.g., ignorance about the amount of gas in the tank represented by leaving that attribute blank)
- Lower level of abstraction $\Longrightarrow$ less details ignored
- Many areas of AI based on factored representations
  - constraint satisfaction and propositional logic
  - planning
  - Bayesian networks
  - (most of) machine learning

Representations [cont.]

Structured Representation

- States represents in terms of objects and relations over them
  - Ex $\forall x.(Men(x) \rightarrow Mortal(x))$, $Woman(Maria)$, $Mother \equiv Woman \cap \exists hasChild.Person$
- Lowest level of abstraction $\Longrightarrow$ can represent reality in details
- Many areas of Ai based on factored representations
  - relational databases
  - first-order logic
  - first-order probability models
  - knowledge-based learning
  - natural language understanding