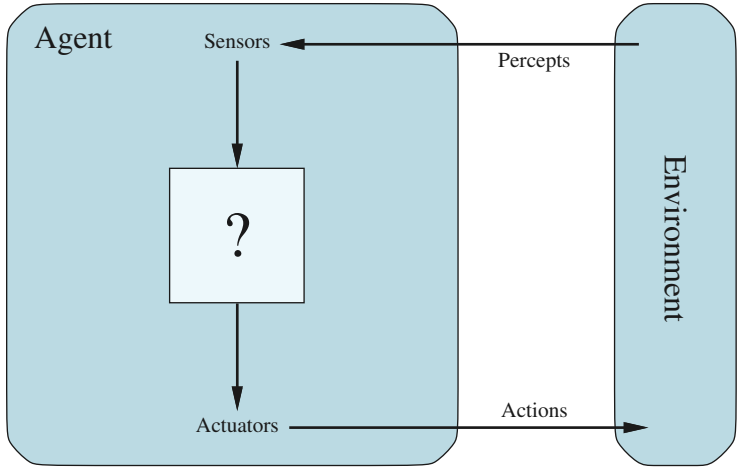
Intelligent Agents

Artificial Intelligence

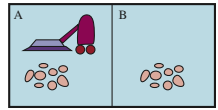
Christopher Simpkins

Agents



Agent Functions

An *agent function* is an external tabulation of the agent's behavior.



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*
⋮	⋮

An *agent program* is an implementation of an agent function inside the agent.

Intelligent Agents Rationality

- Consequentialist: we judge behavior as rational if it leads to desirable consequences.
- Need an external *performance measure* to make this judgment.
- Genie effect: be careful what you wish for.

Example: what if our performance measure is "amount of dirt sucked up?"

> *Performance measures should specify desired states of the environment, not preconceived notions of what the agent's behavior should be.*

Rationality

Judgment of rationality depends on:

- 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.

Definition of 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.*

Note that the performance measure is external to the agent, representing a general notion of "success" within an environment that applies to any agent in the environment.

Omniscience, Rationality, Learning and Autonomy

- Omniscience: for a given state and action, agent knows the result state exactly.
- Omniscience is impossible in practice

> *Rationality means choosing the best action given what you know, i.e., the percept sequence to date.*

Key idea: how to maximize the usefulness of the agent's percept sequence.

- Information gathering, exploration.
- Learning
- Prior knowledge

The more an agent is able to learn, the more autonomy it has.

Task Environments

The concept of an environment is both general and limited. In AI we are usually interested in a particular task within an environment. We call these *task environments* and they represent the "problems" that an intelligent agent "solves."

Task environment specification: PEAS

- **P**erformance measure
- **E**nvironment dynamics – what is the result of applying an action
- **A**ctuators to modify the environment
- **S**ensors to perceive the environment

Taxi Task Environment

Agent Type	Performance Measure	Environment	Actuators	Sensors
Taxi driver	Safe, fast, legal, comfortable trip, maximize profits, minimize impact on other road users	Roads, other traffic, police, pedestrians, customers, weather	Steering, accelerator, brake, signal, horn, display, speech	Cameras, radar, speedometer, GPS, engine sensors, accelerometer, microphones, touchscreen

KENNESAW STATE UNIVERSITY

Example Agent Types and Their PEAS Descriptions

Agent Type	Performance Measure	Environment	Actuators	Sensors
Medical diagnosis system	Healthy patient, reduced costs	Patient, hospital, staff	Display of questions, tests, diagnoses, treatments	Touchscreen/voice entry of symptoms and findings
Satellite image analysis system	Correct categorization of objects, terrain	Orbiting satellite, downlink, weather	Display of scene categorization	High-resolution digital camera
Part-picking robot	Percentage of parts in correct bins	Conveyor belt with parts; bins	Jointed arm and hand	Camera, tactile and joint angle sensors
Refinery controller	Purity, yield, safety	Refinery, raw materials, operators	Valves, pumps, heaters, stirrers, displays	Temperature, pressure, flow, chemical sensors
Interactive English tutor	Student's score on test	Set of students, testing agency	Display of exercises, feedback, speech	Keyboard entry, voice

KENNESAW STATE UNIVERSITY

Task Environment Design Space

- Fully vs. partially observable
- Single agent vs. multi-agent
  - Competitive vs. cooperative
- Deterministic vs. nondeterministic
  - Stochastic is a specific kind of nondeterminism in which we assign probabilities to outcomes
- Episodic vs. sequential
  - Episodic: current action independent of previous decisions and future decisions
  - Sequential: current action may affect all future actions
- Static vs. dynamic Can the environment change while agent is deliberating?
- Discrete vs. continuous
- Known vs. unknown
  - Does the agent know the "physics" of the environment?

Task Environment Examples

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
English tutor	Partially	Multi	Stochastic	Sequential	Dynamic	Discrete

The Structure of Agents

- An *agent program* is an implementation of an agent function inside the agent.
- An *agent architecture* is the computing device on which the agent runs, including sensors and actuators (which may be virtual).
  - When we put an agent program inside a physical platform, like a robot, we call it *embodied intelligence*.

$$agent = architecture + program$$

Table-driven Agent

A table-driven agent program implements the agent function directly.

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

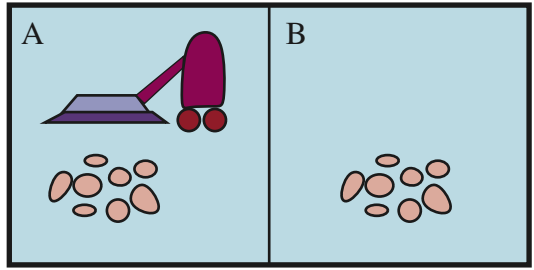
  append *percept* to the end of *percepts*
  $action \leftarrow$ LOOKUP(*percepts*, *table*)
  return *action*

If $\mathcal{P}$ is the set of possible percepts and T is the lifetime of the agent, then the size of the agent table is impossibly large in practice:

$$\sum_{t=1}^{T} |\mathcal{P}|^t$$

Key idea: a full representation of the ideal agent program is usually impossible in practice. Our job as AI agent developers is to design a compact representation of the ideal agent program.

Reflex Vacuum Agent



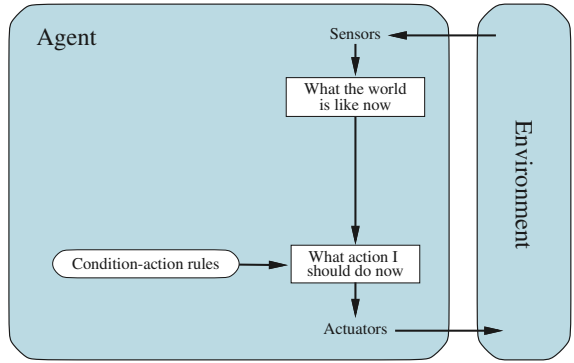
function REFLEX-VACUUM-AGENT([*location*,*status*]) **returns** an action

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

General Reflex Agents

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*



Condition-Action Rules

Condition-action rules (a.k.a. productions or situation-action rules or simply if-then rules) take the form

- **if** *condition* **then** *action*

> *Simple reflex agents work only if the correct decision can be made on the basis of just the current percept—that is, only if the environment is fully observable.*

Consider the effect of partial observability?

- What if our reflex vacuum agent has a dirt sensor but no location sensor?
- How can randomization help?

Model-based Agents

An agent can keep track of its state by maintaining a *model* of the environment. An environment model typically consists of

- a sensor model, which maps percepts to states, and
- a state transition model.

A transition model typically contains

- A set of states, S,
- A set of actions the agent can execute in the environment, A, and
- A state transition function, $T(s, a, s')$

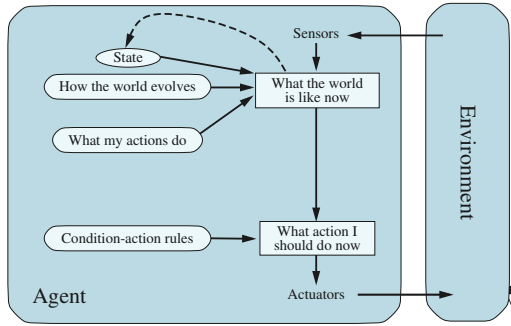
Note that the state transition function can be deterministic or nondeterministic. In most of AI, we consider it to be stochastic.

$$Pr(s, a, s') \text{ or } Pr(s_t|s_{t-1}, a_{t-1})$$

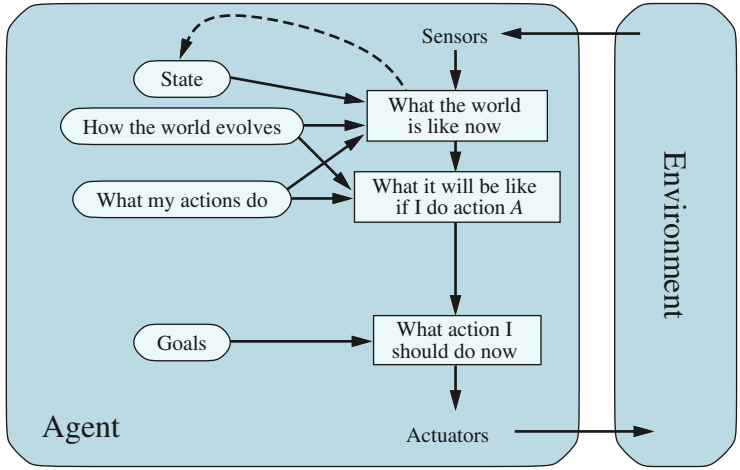
Model-based Reflex Agents

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

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



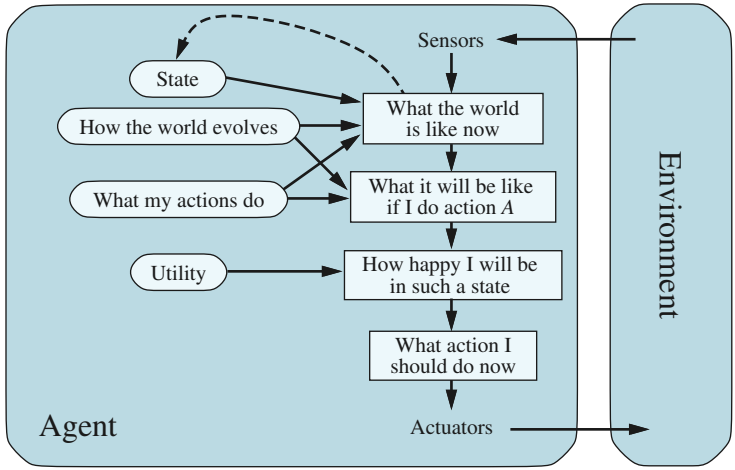
Model-based Goal-driven Agents



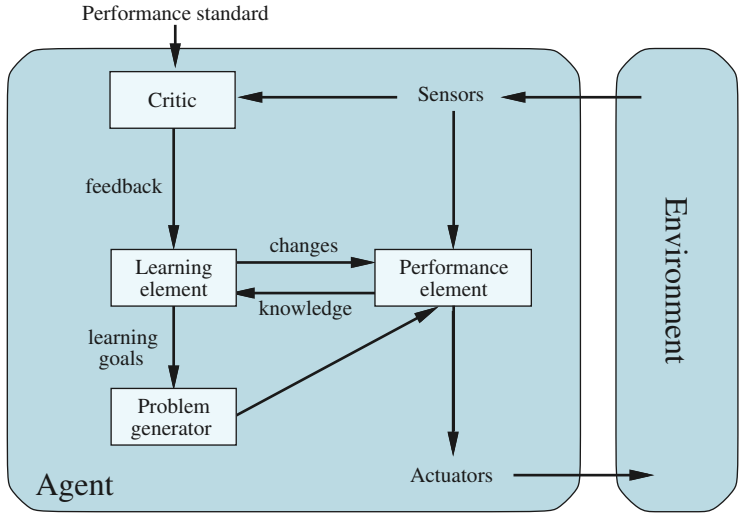
Agents select actions that achieve goals by using **search** and **planning** algorithms, which we'll start learning next lecture.

KENNESAW STATE UNIVERSITY

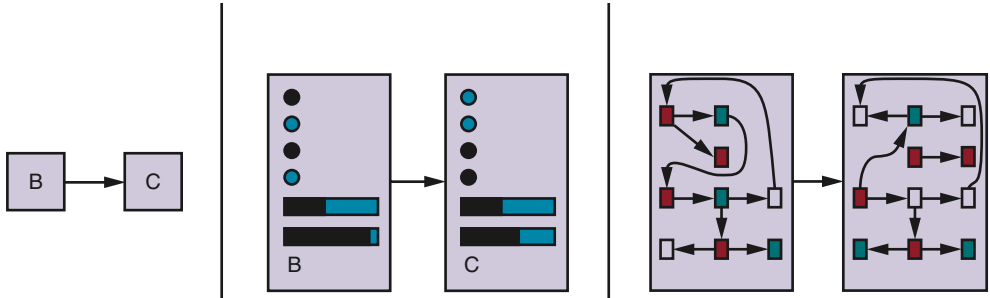
Model-based Utility-driven Agents



Learning Agents



State Representations



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

- Atomic representations are typically used in search-based problem solving.
- Factored representations are used in constraint satisfaction, propositional logic, planning, most probabilistic models (e.g., Bayesian networks), and many machine learning algorithms.
- Structured representations are used in relational databases, first order logic, first-order probability models, and natural language processing.

Closing Thoughts

This lesson has effectively introduced the entire course, providing a framework for everything else we will learn.

- In the next few lessons we'll learn how to search for right action based on goals.
- Then we'll learn how to construct knowledge-based models and use them to plan sequences of actions that achieve goals.
- In the second half of the course we'll learn how to reason under uncertainty, which will help us deal with partial observability, and estimate our state.
- We'll learn how to find optimal actions in multi-agent settings.
- And we'll learn how to construct the learning elements of agents that improve all of the above with experience.