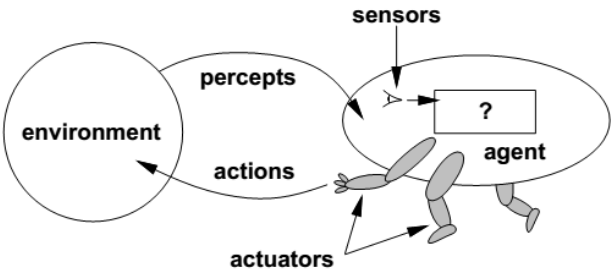
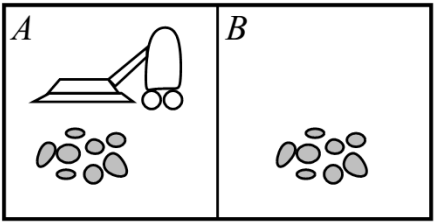

Inteligencia Artificial

Agentes y ambientes



La función de agente es una función matemática que mapea una secuencia de percepciones con una acción.

El programa de agente es una implementación de la función de agente que se ejecuta en una arquitectura dada.



Percepts: location and contents, e.g., $[A, Dirty]$

Actions: $Left$, $Right$, $Suck$, $NoOp$

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

Racionalidad

Medida de performance: indica el grado de éxito de un agente. Se mide en base a los estados del ambiente.

La racionalidad depende de:

- la medida de performance
- conocimiento previo
- acciones que el agente puede realizar
- la secuencia de percepciones

Definición de agente racional

Para cada posible secuencia de percepciones, un agente racional debe elegir una acción que se espera maximice la medida de rendimiento, dada la evidencia provista por la secuencia de percepciones y cualquier conocimiento previo que el agente posea.

Omnisciencia, aprendizaje y autonomía.

- **Omniciencia:** un agente omniciente **sabe** cual va a ser el resultado de sus acciones. **Omniciencia <> Racionalidad**
- **Aprendizaje:** es la capacidad de generar nuevo conocimiento a partir de las percepciones. Se espera que un agente racional sea capaz de aprender.
- **Autonomía:** es la capacidad del agente de no depender solamente de conocimientos previos.

Naturaleza de los ambientes

PEAS: Performance, Environment, Actuators, Sensors

Sirve para definir el ambiente donde el agente va situarse.

Agent Type	Perfonnance Measure	Environment	Actuators	Sensors
Taxi driver	Safe, fast, legal, comfortable trip, maximize profits	Roads, other traffic, pedestrians, customers	Steering, accelerator, bmke, signal, horn, display	Cameras, sonar, speedometer, GPS, odometer, accelerometer, engine sensors, keyboard

Figure 2.4 PEAS description of the task environment for an automated taxi.

Clasificaciones.

- Completamente observable vs parcialmente observable
- Un único agente vs multiagente
- Determinístico vs estocástico
- Episodico vs secuencial
- Estático vs dinámico vs semidinámico
- Discreto vs continuo
- Conocido vs desconocido

Estructura de los agentes

agente = arquitectura + programa

El trabajo de la IA es implementar **programas de agente** que implementen la función de agente.

Los programas de agente **toman una percepción y devuelven una acción**.

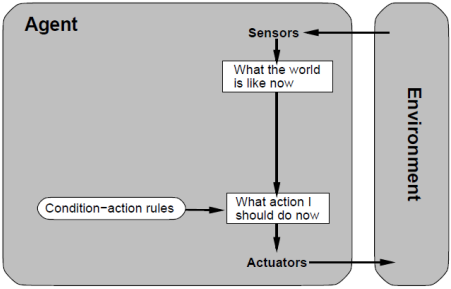
```
function TABLE-DRIVEN-AGENT(percept) returns an action
  persistent: percepts, a sequence, initially empty
              table, a table of actions, indexed by percept sequences, initially fully specified

  append percept to the end of percepts
  action ← LOOKUP(percepts, table)
  return action
```

Figure 2.3 The TABLE-DRIVEN-AGENT program is invoked for each new percept and returns an action each time. It retains the complete percept sequence in memory.

El desafío: generar agentes a partir de pequeños programas

Agente reflejo simple

- El más sencillo
- Toman decisiones en base a la percepción actual
- Programas escritos en base a reglas condición-acción (if-then)



```
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
```

Figure 2.6 A simple reflex agent. It acts according to a rule whose condition matches the current state, as defined by the percept.

```
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
```

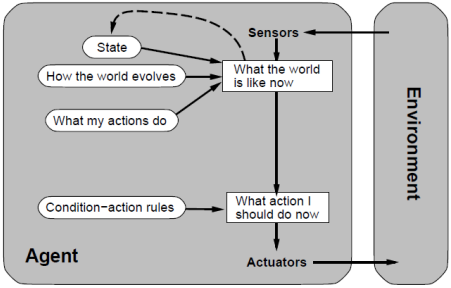
Figure 2.6 A simple reflex agent. It acts according to a rule whose condition matches the current state, as defined by the percept.

```
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
```

Figure 2.4 The agent program for a simple reflex agent in the two-state vacuum environment. This program implements the agent function tabulated in Figure ??.

Agente reflejo basados en modelos

- mantienen un estado interno que depende de la historia de percepciones
- 2 tipos de conocimientos para actualizar el **modelo**: cómo funciona el mundo y cómo afectan las propias acciones

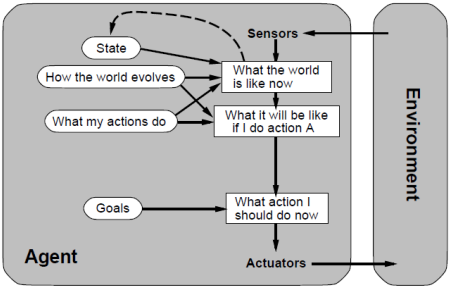


```
function MODEL-BASED-REFLEX-AGENT(percept) returns an action
  persistent: state, the agent's current conception of the world state
              model, a description of how the next state depends on current state and action
              rules, a set of condition–action rules
              action,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

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

Figure 2.8 A model-based reflex agent. It keeps track of the current state of the world, using an internal model. It then chooses an action in the same way as the reflex agent.

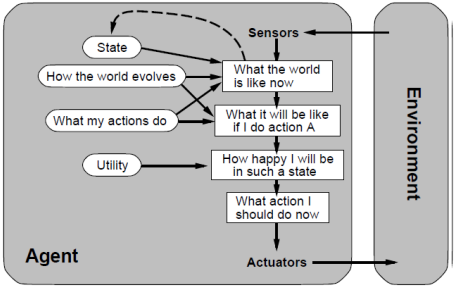
Agente basados en objetivos

- Los distintos estados se clasifican en meta o no meta
- La representación del objetivo se explicita
- Son más flexibles

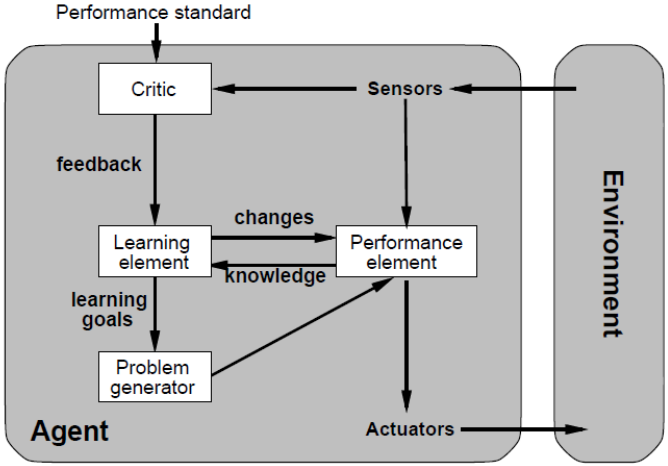


Agente basados en utilidades

- Se asigna una utilidad a cada estado
- La función de utilidad es una internalización de la medida de performance
- Un agente racional maximiza la **utilidad esperada**



Agentes que aprenden



Bibliografía y enlaces útiles.

- Russell S., Norvig P.: Artificial Intelligence: A modern Approach. Third Edition