

# Tema 2: Agentes Inteligentes: Conceptos fundamentales

---

---

*Authors: Vicent Botti, Carlos Carrascosa, Vicente Julián*

# Tema 1 - Índice

1.1 Definición de agente.

1.2 Entornos de agente.

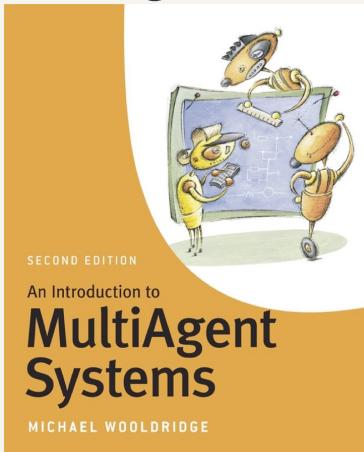
1.3 Agentes como sistemas intencionales.

1.4 Arquitecturas abstractas para agentes inteligentes.

1.5 ¿Cómo decirle a un agente lo que tiene que hacer?

# Bibliografía

## Bibliografía básica:



### Tema 2

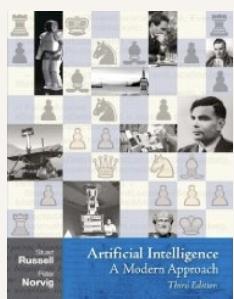
An Introduction to MultiAgent Systems – Second Edition  
by Michael Wooldrige

Published May 2009 by John Wiley & Sons

ISBN-10: 0470519460

ISBN-13: 978-0470519462

## Bibliografía complementaria:



Artificial Intelligence: A Modern Approach (Third edition)

By Stuart Russell and Peter Norvig

Published by Prentice Hall Copyright © 2010

Published Date: Dec 1, 2009

ISBN-10: 0-13-604259-7

ISBN-13: 978-0-13-604259-4

# Tema 1 - Índice

## 1.1 Definición de agente.

1.4 Entornos de agente.

1.5 Agentes como sistemas intencionales.

1.6 Arquitecturas abstractas para agentes inteligentes.

1.7 ¿Cómo decirle a un agente lo que tiene que hacer?

# 1.1 Definición de Agente



# Agentes Software

Los sistemas compuestos de múltiples agentes, comenzaron a utilizarse en la **Inteligencia Artificial Distribuida** (O'hare et al., 1996), tradicionalmente dividida en dos campos:

## *Resolución de Problemas distribuidos:*

Soluciones basadas en un cjto. de nodos cooperando en dividir y compartir.

Cada agente tiene unas tareas y conducta prefijadas.

El Sistema se centra en el comportamiento global.

## *Sistemas Multiagente:*

Agentes autónomos trabajan juntos para resolver problemas.

Cada agente tiene una información o capacidad incompleta para solucionar el problema.

Los agentes pueden decidir dinámicamente que tareas deben realizar y quien realiza cada tarea.

No hay un sistema global de control, los datos están descentralizados y la computación es asíncrona.

La investigación inicial progresó hacia la madurez. Surge la *Programación Orientada a Agentes* y los *Lenguajes de Comunicación de Agentes*

Posteriormente este concepto se extiende al resto de la Ingeniería del software, planteándose hoy en día la *Ingeniería del Software Orientada a Agente*.

# Definiciones

Agente (Diccionario RAE):

Que obra o tiene virtud de obrar

El que realiza una acción

Persona o cosa que produce un efecto

Persona que obra con poder de otra

El que actúa en representación de otro (agente artístico, comercial, inmobiliario, de seguros, de bolsa, etc)

Persona que tiene a su cargo una agencia para gestionar asuntos ajenos o prestar determinados servicios

Agentes software:

Aplicaciones informáticas con capacidad para *decidir* cómo deben actuar para alcanzar sus objetivos

Agentes inteligentes:

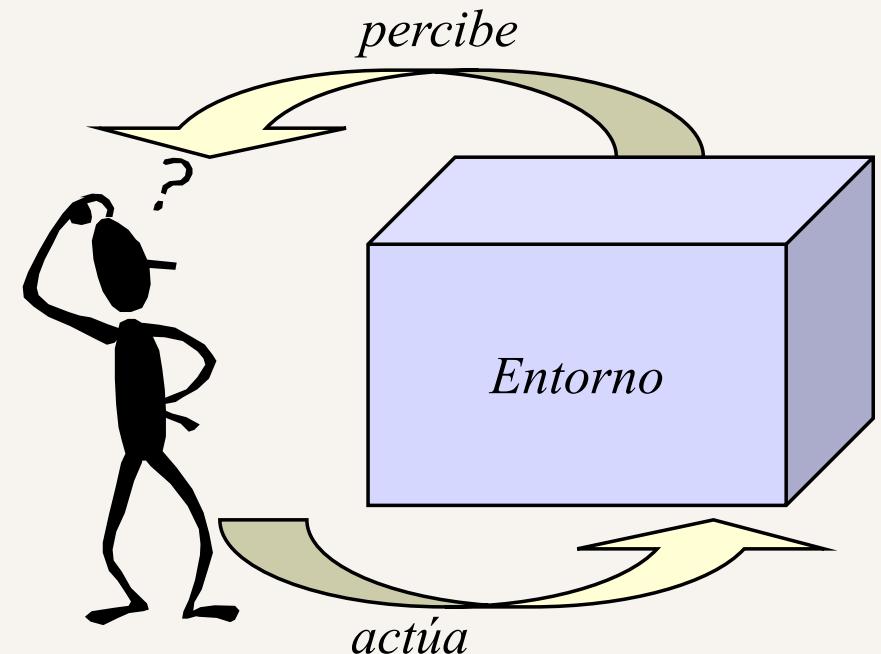
Agentes software que pueden funcionar fiablemente en un entorno rápidamente cambiante e impredecible

# ¿Pero qué es un agente?

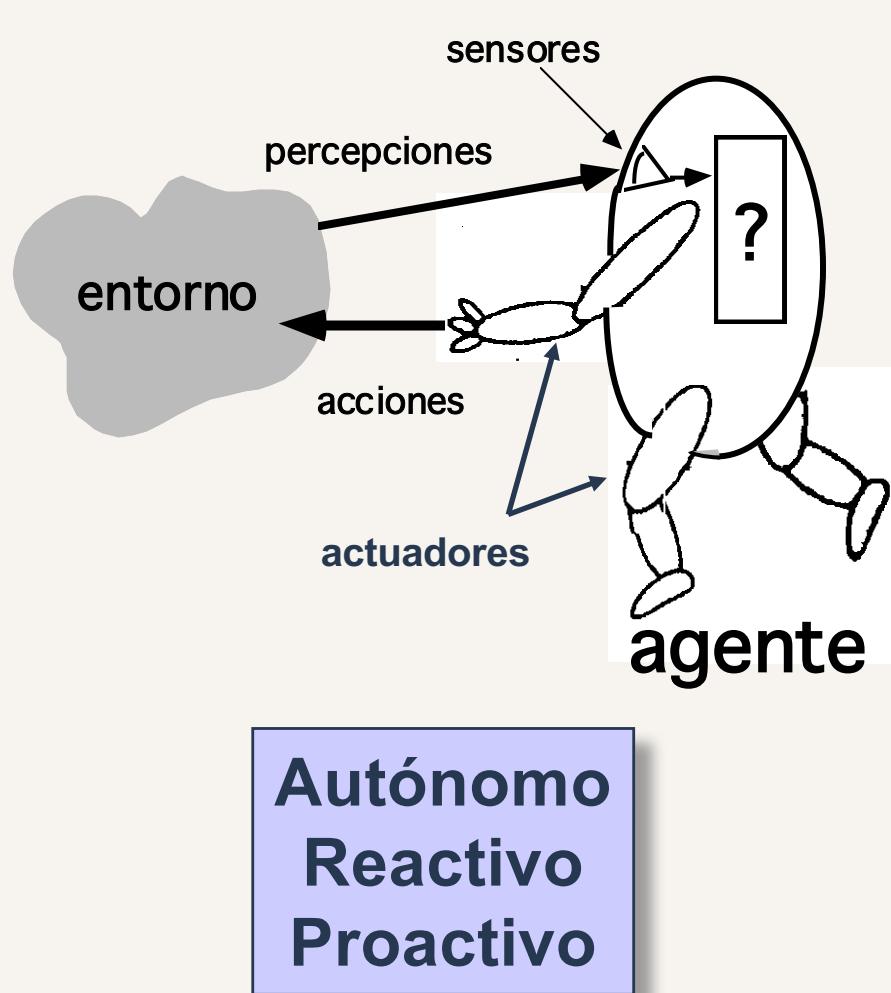
La principal característica de los agentes es que son **autónomos**: capaces de actuar independientemente.

Una primera definición (Wooldridge):  
Un agente es un sistema informático capaz de actuar autónomamente en algún entorno con el fin de alcanzar los objetivos que se le han delegado.

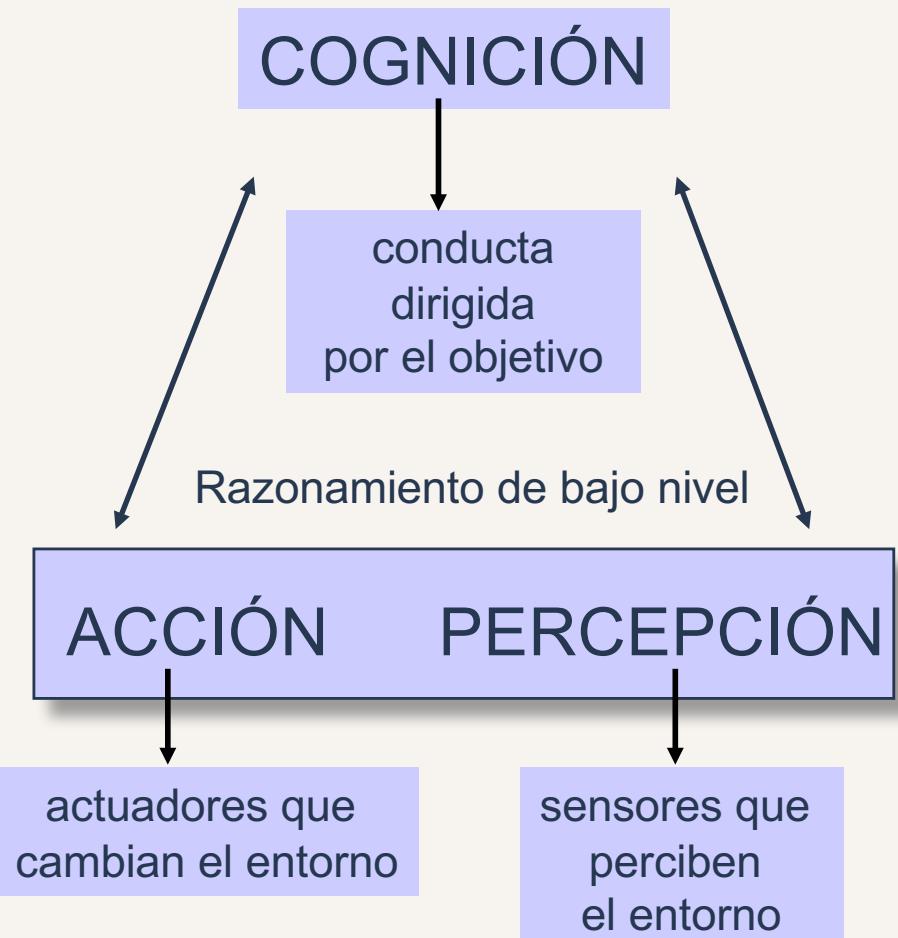
Consideramos que un agente como una entidad que está en continua interacción con su entorno:  
percibe – decide – actúa – percibe – decide - .....



# Concepto de Agente Inteligente



Razonamiento de alto nivel



# Agentes Simples (sin interés)

- Un termostato
  - el objetivo delegado es mantener la temperatura de una habitación.
  - sus acciones son calor/frio conectar/desconectar.
- Programa biff UNIX
  - el objetivo delegado es monitorizar el correo entrante y etiquetarlo.
  - sus acciones son acciones GUI.
- Estos agentes son triviales porque las decisiones que toman (sus procesos cognitivos) son triviales

# Pero ¿qué es un agente?

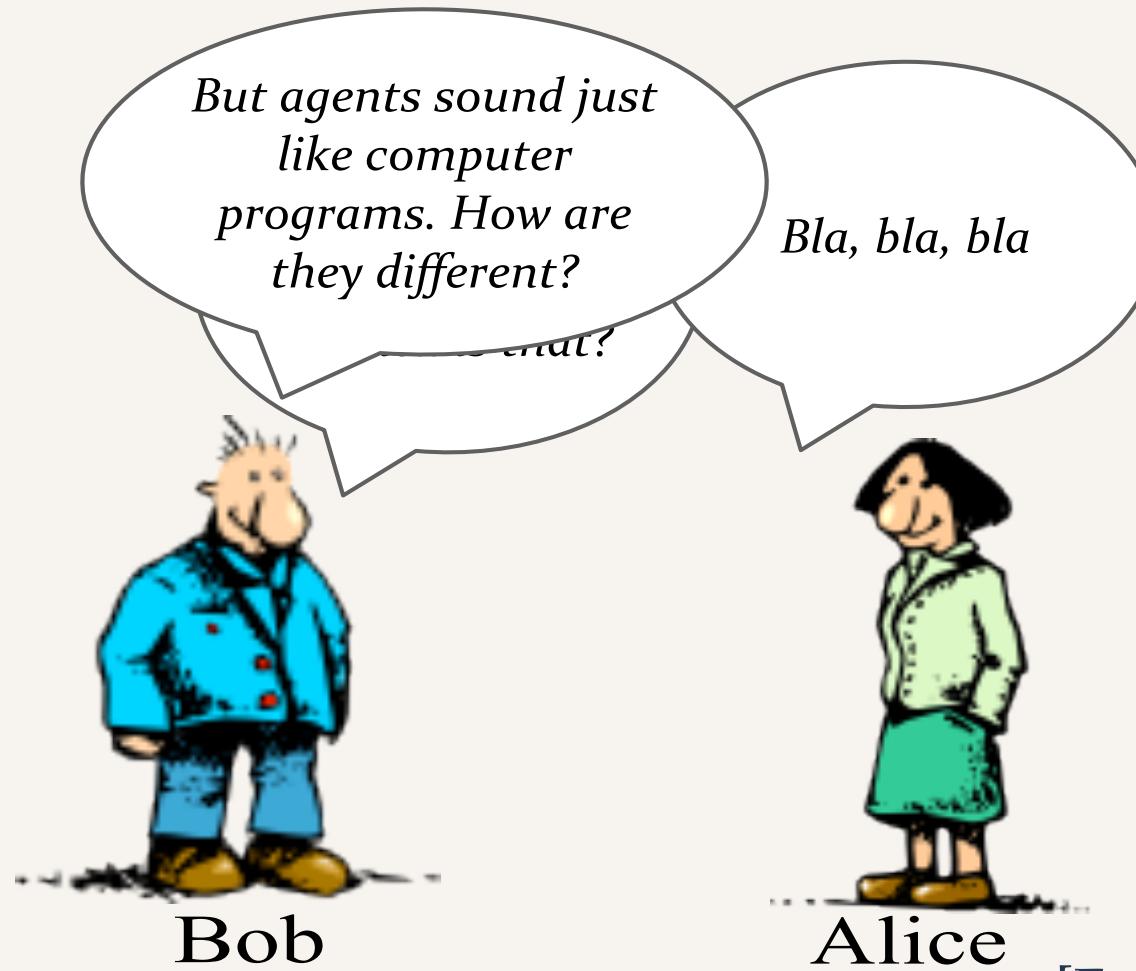
**Houston, we've got a problem!!!**

¿Proceso de larga vida (permanente)?

¿Independencia, autonomía?

¿“Inteligencia”?

# Pero ¿qué es un agente?



[Franklin&Graesser, 96]

# Teoría de Agentes

Un agente es un sistema informático, situado en algún entorno, que percibe el entorno (entradas sensibles de su entorno) y a partir de tales percepciones determina (mediante técnicas de resolución de problemas) y ejecuta acciones (de forma autónoma y flexible) que le permiten alcanzar sus objetivos y que pueden cambiar el entorno.

# Agente: Definición

❖ Wooldridge:

Cualquier proceso computacional dirigido por el objetivo capaz de interaccionar con su entorno de forma **flexible** y **robusta**



# Reactividad

- Si está garantizado que el entorno de un programa no cambia (es **estático**), un programa se puede ejecutar a ciegas.
- El mundo real no es así, no es estático, la mayoría de los entorno son **dinámicos**.
- Los programas para entornos dinámicos son difíciles de construir: los programas deben tener en cuenta la posibilidad de fallo – preguntarse a si mismos si conviene continuar su ejecución.
- Un sistema **reactivo** es aquel que mantiene una constante interacción con su entorno y responde (a tiempo para que la respuesta sea útil) a los cambios que ocurren en él.

# Proactividad /Iniciativa

- Reaccionar a un entorno es fácil (por ejemplo, reglas estimulo → respuesta).
- Pero nosotros queremos que los agentes **hagan cosas por nosotros**.
- Por ello adoptamos un **comportamiento dirigido por el objetivo**.
- La **proactividad (iniciativa)** = generar e intentar conseguir objetivos, no dirigido solamente por eventos, tomar la iniciativa.
- Reconociendo oportunidades.

# Capacidad Social (Sociabilidad)

- El mundo real es un mundo multi-agente: cuando queremos conseguir objetivos tenemos que tener en cuenta al resto de entidades (agentes) del entorno donde nos encontramos.
- En algunos casos para alcanzar un objetivo es necesario interactuar con otros ya que puede que nosotros no tengamos capacidades suficientes para lograrlo.
- Del mismo modo que sucede en entornos de computadores: un testimonio es internet.
- *La sociabilidad (capacidad social)* en agentes es la capacidad de interactuar con otros agentes (y posiblemente con humanos) mediante *cooperación, coordinación y negociación*.

***Por lo menos significa la capacidad de comunicarse.***

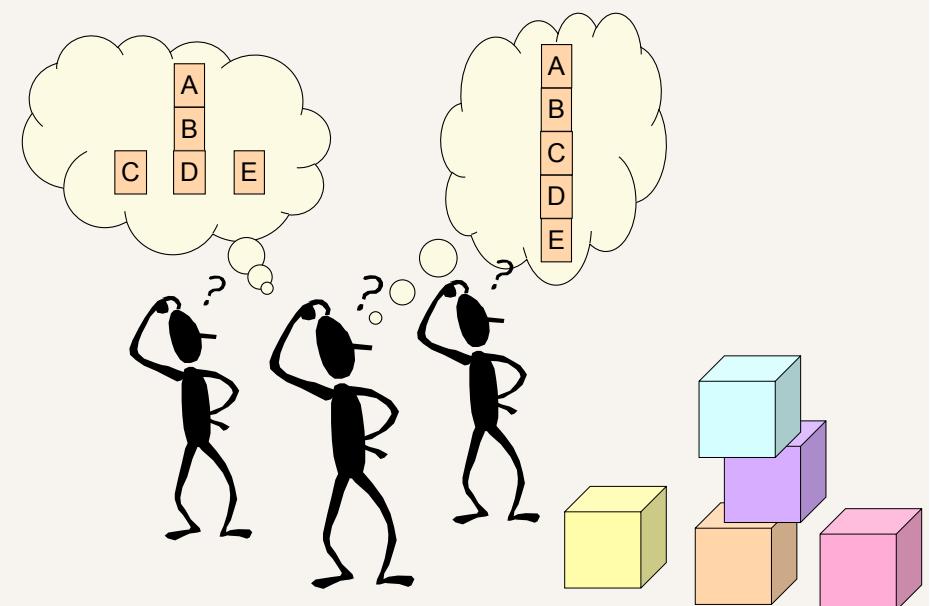
# Capacidad Social: Cooperación

- Cooperación es trabajar juntos como un equipo para conseguir un objetivo compartido.
- A menudo se requiere porque ningún agente puede conseguir el objetivo solo, o porque cooperando se obtendrá un resultado mejor (por ejemplo obtener un resultado más rápidamente).



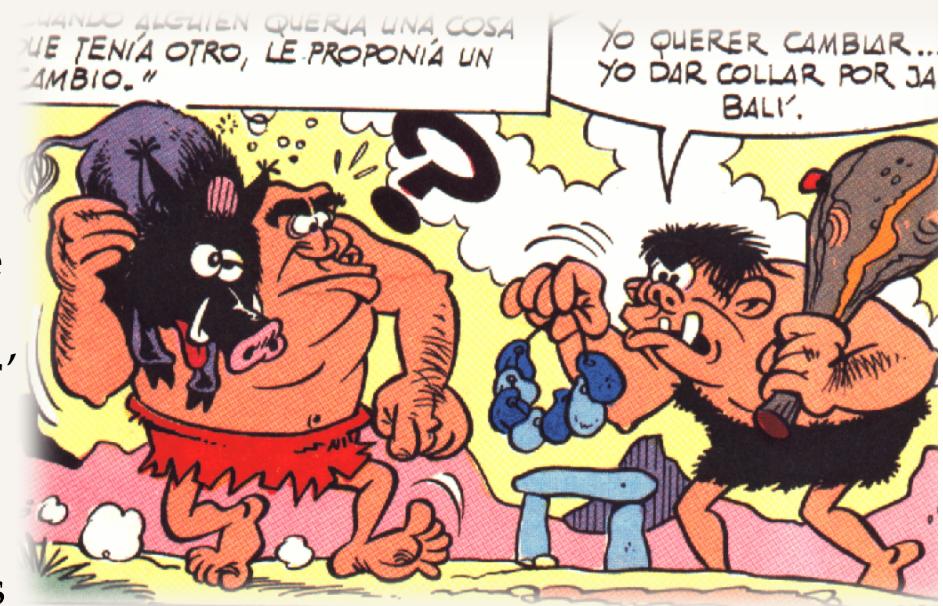
# Capacidad Social: Coordinación

- **Coordinación** es gestionar las interdependencias entre actividades..
- Por ejemplo si hay un recurso no compatible que se desea utilizar por más de una entidad (agente), necesitamos que se coordinen.



# Capacidad Social: Negociación

- La negociación es la capacidad de alcanzar acuerdos sobre temas de interés común.
- Por ejemplo: Estamos en casa delante del televisor, tu quieres ver un partido de futbol y tu pareja una película.  
Un posible trato es ver el partido esta noche y mañana una película (este acuerdo puede tener un 'precio' que una parte ha de 'pagar' a la otra).
- Normalmente la negociación implica ofertas y contraoferta , con compromisos ('pagos') asumidos por los participantes.



# Otras propiedades

## Concepto débil de agente

- Autonomía
- Proactividad
- Reactividad
- Sociabilidad

## Concepto fuerte de agente

Concepto débil +

- Movilidad
- Veracidad
- Benevolencia
- Racionalidad
- Aprendizaje / adaptación

Un agente **siempre** tiene las propiedades débiles de agencia y **puede** tener las propiedades fuertes

# Otras propiedades

**Movilidad:** habilidad de trasladarse en una red de comunicación informática.

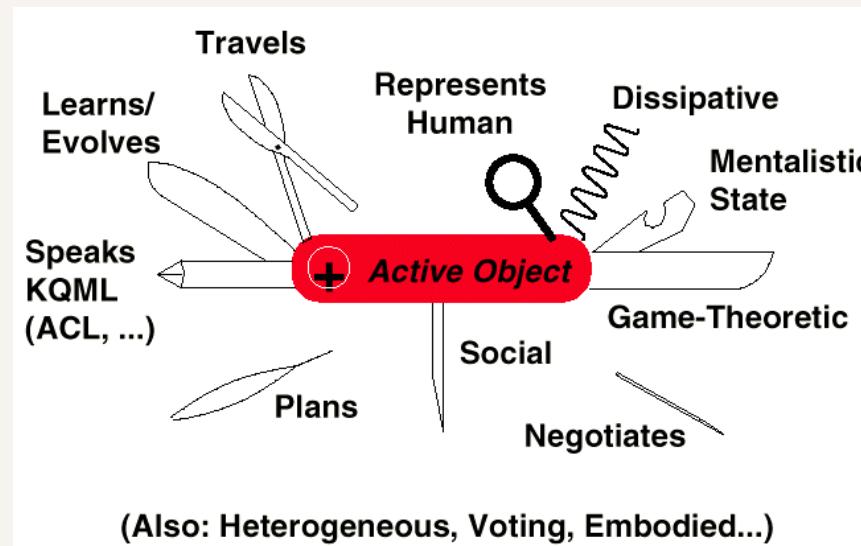
**Veracidad:** no comunica información falsa intencionadamente.

**Benevolencia:** no tiene objetivos contradictorios y siempre intenta realizar la tarea que se le solicita.

**Racionalidad:** tiene unos objetivos específicos y siempre intenta llevarlos a cabo.

# Agente: Definición

Van Parunak:



# Tema 1 - Índice

1.1 Definición de agente.

**1.2 Entornos de agente.**

1.3 Agentes como sistemas intencionales.

1.4 Arquitecturas abstractas para agentes inteligentes.

1.5 ¿Cómo decirle a un agente lo que tiene que hacer?

# I.2 Entornos de agente

- En entornos complejos:
  - Un agente no tiene control completo sobre su entorno, sólo tiene un control parcial.
  - Control parcial significa que el agente puede influir sobre el entorno con sus acciones.
  - Una acción ejecutada por un agente puede fallar o tener el efecto deseado
- Conclusión: los entornos son no deterministas y los agentes deben estar preparados para posibles fallos.
- ¿Qué propiedades tienen los entornos y como influyen las mismas en los agentes?



# Entornos de agente: propiedades

- Accesible (observable) vs inaccesible (parcialmente observable).

Un entorno observable es aquel en el que el agente puede obtener información completa, exacta y actualizada del estado del entorno (los sensores perciben todos los datos que son relevantes para la toma de decisiones).

Entornos moderadamente complejos (incluyendo por ejemplo, el mundo físico diario e internet) son inaccesibles.

Cuanto más accesible es un entorno, más fácil es construir agentes que interactúen con él.

# Entorno: propiedades

- **Determinista vs estocástico.**

Un entorno determinista es aquel en el que cualquier acción tiene un único efecto garantizado, no hay incertidumbre sobre el estado resultante de la ejecución de una acción.

El mundo físico puede a todos los efectos ser considerado como no determinista.

Los entornos estocásticos presentan grandes problemas para el diseñador de agentes.

# Entorno: propiedades

- **Episódico vs secuencial**

En un entorno episódico el desempeño / actuación de un agente depende de un número discreto de episodios, no existiendo enlaces (relación) entre el desempeño de un agente en escenarios distintos.

Cada episodio consiste en la percepción del agente y la realización de una única acción posterior. Es muy importante saber que el siguiente episodio no depende de las acciones que se realizaron anteriormente y es que en los entornos episódicos la elección de la acción en cada episodio depende sólo del episodio en sí mismo.

Los entornos episódicos son, desde el punto de vista del desarrollador de agentes, más sencillos porque el agente puede decidir que acción ejecutar basándose únicamente en el episodio actual, no necesita razonar sobre las interacciones entre el episodio actual y los episodios futuros.

# Entorno: propiedades

- **Estático vs dinámico**

Un entorno estático es aquel en el que se puede asumir que no se producen cambios excepto los provocados por la ejecución de acciones del agente.

Un entorno dinámico es aquel que tiene otros procesos que operan en él, y que por lo tanto se producen cambios que están fuera del control del agente.

El mundo físico es un entorno altamente dinámico.

# Entorno: propiedades

- **Discreto vs continuo**

Un entorno es discreto si en él hay un número fijo y finito de acciones y percepciones.

El juego del ajedrez es un ejemplo de entorno discreto, y la conducción de un taxi un ejemplo de entorno continuo (AIMA, Raussell and Norvig).

# Entornos: Propiedades

	Solitario	Backgammon	Compra por Internet	Taxi
<u>¿Observable?</u>	Sí	Sí	No	No
<u>¿Determinista?</u>	Sí	No	Parcialmente	No
<u>¿Episódico?</u>	No	No	No	No
<u>¿Estático?</u>	Sí	Semi	Semi	No
<u>¿Discreto?</u>	Sí	Sí	Sí	No
<u>¿Agente individual?</u>	Sí	No	No (excepto las subastas)	No

El tipo de entorno determina el tipo de agente.

El mundo real es (por supuesto) parcialmente observable, estocástico, secuencial, dinámico, continuo y multiagente.

# Tema 1- Índice

1.1 Definición de agente.

1.2 Entornos de agente.

**1.3 Agentes como sistemas intencionales.**

1.4 Arquitecturas abstractas para agentes inteligentes.

2.5 ¿Cómo decirle a un agente lo que tiene que hacer?

## 2.3 Agentes como Sistemas Intencionales

- Al explicarla actividad humana utilizamos declaraciones como la siguiente:

*Lola cogió su paraguas porque creía que estaba lloviendo y no quería mojarse.*
- Estas declaraciones hacen uso de una psicología popular, porque el comportamiento humano se predice y explica atribuyendo actitudes como: creer, desear, esperar, temer, ..
- Daniel Dennett acuñó el término **sistema intencional** al describir entidades "cuyo comportamiento puede ser predicho por el método de atribuir creencias, deseos y perspicacia racional".
- Un sistema intencional de primer orden tiene creencias y deseos (etc.), pero no creencias y deseos sobre creencias y deseos.
- Un sistema intencional de segundo orden es más sofisticado, tiene creencias y deseos (y sin duda otros estados intencionales) sobre creencias y deseos (y otros estados intencionales) - tanto los de los otros como de si mismo.

# Agentes como Sistemas Intencionales

- ¿Podemos utilizar la actitud intencional en máquinas?

Atribuir creencias, albedrio, intenciones, conciencia, habilidades, o deseos a una máquina es correcto cuando tal atribución expresa la misma información sobre la máquina que expresa sobre una persona.

Las teorías de la creencia, el conocimiento y el deseo se pueden construir para las máquinas de una forma más sencilla que para los seres humanos, y posteriormente aplicarse a los seres humanos.

Atribuir cualidades mentales es más sencillo para máquinas de estructura conocida, tales como termostatos y sistemas operativos, pero es más útil cuando se aplica a entidades cuya estructura no es completamente conocida (John McCarty)

# Agentes como Sistemas Intencionales

- ¿Qué podemos describir con la actitud intencional ?

Consideremos un interruptor eléctrico:

*Atribuir creencias, albedrio, intenciones, conciencia, habilidades, o deseos a una máquina es correcto cuando tal atribución expresa la misma información sobre la máquina que expresa sobre una persona.*

Accionar el interruptor es simplemente nuestra forma de comunicar nuestros deseos (Yoav Shoham)



# Agentes como Sistemas Intencionales

- La descripción de la actitud intencional es consistente,

..... no nos convence cualquier cosa, ya que esencialmente comprendemos el mecanismo suficientemente como para tener una descripción más simple y **mecanicista** (*explica los fenómenos de la naturaleza mediante leyes automáticas de causa y efecto*) de su comportamiento. (Yoav Shoham)
- Cuanto más sepamos sobre un sistema, menos necesitamos confiar en explicaciones anímicas e intencionales de su comportamiento.

# Agentes como Sistemas Intencionales

- Pero con sistemas muy complejos, una explicación mecanicista de su comportamiento puede no ser factible.
- Los sistemas informáticos son **cada vez más complejos**, por ello **necesitamos abstracciones y metáforas más potentes** para explicar su funcionamiento - especificaciones de bajo nivel son impracticables.
- La **actitud intencional** es una de tales abstracciones.

# Agentes como Sistemas Intencionales

- Los **conceptos intencionales** son, pues, **herramientas de abstracción** que nos proporciona una forma cómoda y familiar de describir, explicar y predecir el comportamiento de los sistemas complejos.
- Hagamos memoria: los desarrollos más importantes en la informática se basan en nuevas abstracciones:
  - ✓ Abstracción procedural;
  - ✓ Tipos de datos abstractos;
  - ✓ Objetos.
- Los **agentes y los agentes como sistemas intencionales**, representan una abstracción más, y cada vez más, potente.

# Agentes como Sistemas Intencionales

- Fortalezas de esta idea:

**Caracterización de Agentes**

Nos proporciona una forma familiar, no técnica, de *comprender y explicar* agentes.

**Representaciones anidadas**

Nos da la posibilidad de especificar sistemas que *incluyen representaciones de otros sistemas*.

Se acepta ampliamente que tales representaciones anidadas son esenciales para agentes que deben cooperar con otros agentes.

# Agentes: Sistemas Post-declarativos

- En la programación procedural decimos exactamente lo que un sistema debe hacer.
- En la programación declarativa, declaramos algo que queremos conseguir, damos al sistema información general sobre las relaciones entre objetos, y el mecanismo de control incorporado (por ejemplo un demostrador de teoremas dirigido por el objetivo) se encarga de deducir qué hacer;
- Con los agentes, proporcionamos una descripción de alto nivel del objetivo delegado, y dejamos que el mecanismo de control deduzca qué hacer, sabiendo que actuará según una teoría integrada de agencia racional.
- La computación como interacción (Tecnología de agentes/ sistemas multiagente y tecnologías del acuerdo).

# Tema 2- Índice

1.1 Definición de agente.

1.2 Entornos de agente.

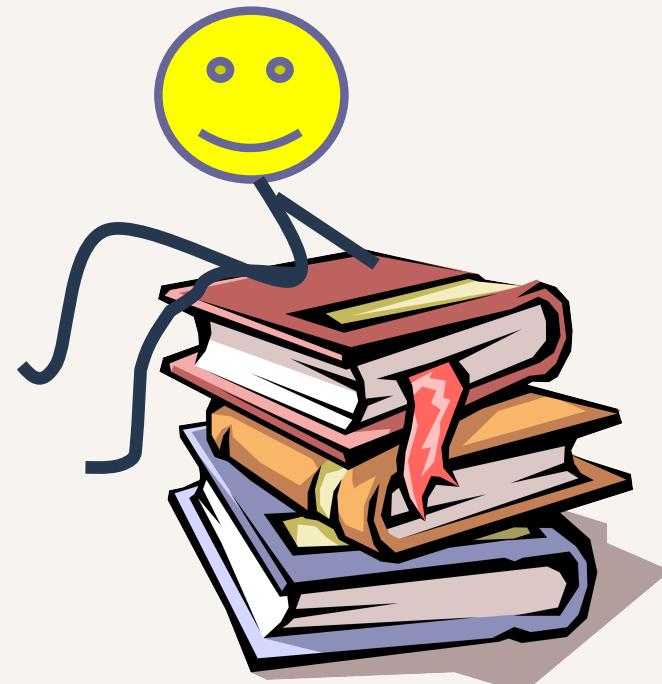
1.3 Agentes como sistemas intencionales.

**1.4 Arquitecturas abstractas para agentes inteligentes.**

1.5 ¿Cómo decirle a un agente lo que tiene que hacer?

## 2.4 Arquitecturas abstractas para agentes inteligentes: Formalización

- **Agentes**
  - **Entorno**
  - **Agentes estándar**
  - **Agentes reactivos simples**
  - **Agentes con estado**



# Agentes y entornos

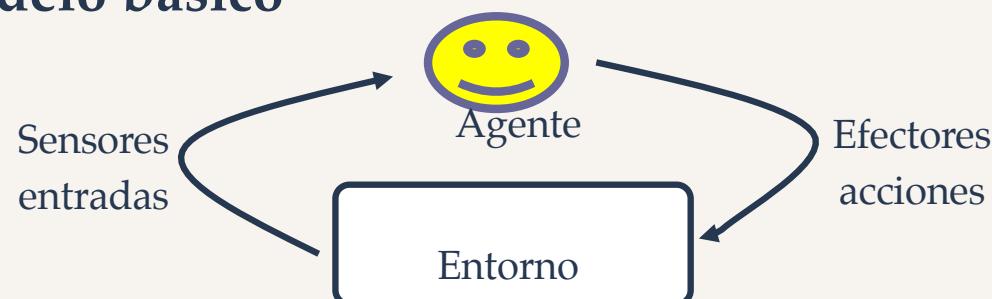
- Asumimos que el entorno puede estar en uno cualquiera de los estados de un conjunto finito de estados instantáneos discreto ( $E$ ):

$$E = \{ s_1, s_2, \dots \}$$

- Se asume que los agentes tiene disponible un repertorio (conjunto finito) de posibles acciones que transforman el estado del entorno :

$$Ac = \{ \alpha_1, \alpha_2, \dots \}$$

## Modelo básico



# Ejecución de un agente

La ejecución,  $r$ , de un agente en un entorno representa la interacción entre el agente y su entorno, es una secuencia de estados de entorno y acciones intercaladas:



donde:

$s_0$  es el estado inicial del entorno

$\alpha_n$  es la enésima acción que el agente decide realizar

$s_n$  es el enésimo estado del entorno

# Ejecución de un agente

$\mathcal{R}$  es el conjunto de todas las secuencias finitas posibles (sobre E y Ac)

$\mathcal{R}^{Ac}$  es el subconjunto de estas secuencias que finalizan con una acción

$\mathcal{R}^E$  es el subconjunto de estas secuencias que finalizan con un estado del entorno

# Entornos

- **Los entornos** pueden ser modelados mediante una función de transición de estado [Gagin et al, 1995] que representa el comportamiento del entorno:

$$\tau: \mathcal{R}^{\mathcal{A}} \rightarrow \mathcal{P}(E)$$

donde  $\mathcal{P}(E)$  es el conjunto de partes de  $E$ ;

Esta función *tiene como argumento una secuencia estado-acción ( $r$ )* que finaliza en una acción (ejecutada por el agente), y devuelve como resultado un conjunto de estados del entorno, es decir representa el efecto que las acciones de un agente tienen sobre el entorno.

Su significado es que como resultado de ejecutar la última acción de la secuencia el entorno puede encontrarse en cualquiera de los estados  $\tau(r)$ .

# Entornos

- Observar que los entornos son  
***dependientes de la historia no deterministas ( $\tau(r) \in \mathcal{P}(E)$ )***
- Si  $\tau(r)=0$  no hay posible estado sucesor de r, es decir la ejecución ha finalizado ('game over'), asumimos que eventualmente todas las ejecuciones finalizan.
- Un entorno Env es una tripleta  $\text{Env} = \langle E, s_0, \tau \rangle$  donde E es el conjunto de estados del entorno,  $s_0 \in E$  es el estado inicial y  $\tau$  es la función de transición.

# Agentes

- Necesitamos un **Modelo de Agente** que **habite** en un sistema.
- Asumimos un modelo donde el agente decide que acción ejecutar basándose en su historia (ejecución).
- Un agente puede ser visto como una función que relaciona ejecuciones (asumimos que finaliza en un estado del entorno) con acciones:

$$Ag: \mathcal{R}^E \rightarrow Ac$$

$\mathcal{R}^E$  es una ejecución que representa la historia previa

- Un agente toma la decisión de que acción ejecutar basándose en la historia del sistema de la que ha sido testigo hasta ese momento.
- Denotaremos por  $\mathcal{A}G$  al conjunto de todos los agentes.
- Asumimos que los entornos son implicitamente estocásticos, sin embargo **los agentes son deterministas**.

# Sistemas

Un sistema es un par que contiene un agente y un entorno.

Cualquier sistema tendrá asociado un conjunto de posibles ejecuciones; denotaremos el conjunto de posibles ejecuciones de un agente  $Ag$  en un entorno  $Env$  por  $\mathcal{R}(Ag, Env)$ .

Asumimos que  $\mathcal{R}(Ag, Env)$  contiene solo ejecuciones que han finalizado.

# Bucle ejecución agente

Formalmente, una secuencia

$$(s_0, \alpha_0, s_1, \alpha_1, s_2, \dots)$$

representa una ejecución de un agente  $Ag$  en un entorno  $Env = \langle E, s_0, \tau \rangle$  si:

1.  $s_0$  es el estado inicial de  $Env$
2.  $\alpha_0 = Ag(s_0)$ ; y
3. para todo  $n > 0$

$$e_n \in \tau((s_0, \alpha_0, s_1, \alpha_1, s_2, \dots, \alpha_{n-1}))$$
$$\alpha_n = Ag((s_0, \alpha_0, s_1, \alpha_1, s_2, \dots, e_n))$$

# Bucle ejecución agente

Dos agentes  $Ag_1$  y  $Ag_2$  tiene un comportamiento equivalente en un entorno  $Env$  si y solo si  $\mathcal{R}(Ag_1, Env) = \mathcal{R}(Ag_2, Env)$ .

Dos agentes  $Ag_1$  y  $Ag_2$  tiene un comportamiento equivalente si y solo si tienen un comportamiento equivalente en todos los entornos.

Hasta el momento no hemos dicho nada sobre como implementar un agente.

# Tema 1 - Índice

1.1 Definición de agente.

1.2 Entornos de agente.

1.3 Agentes como sistemas intencionales.

**1.4 Arquitecturas abstractas para agentes inteligentes.**

1.5 ¿Cómo decirle a un agente lo que tiene que hacer?

## 2.4 Arquitecturas abstractas para agentes inteligentes: Formalización

- **Agentes**
  - Entorno
  - Agentes estándar
  - **Agentes reactivos simples**
  - **Agentes con estado**



# Agente reactivo

- Un agente reactivo decide que acción ejecutar sin tener en cuenta su historia (ninguna referencia al pasado) considerando solamente el presente.
- Formalmente el comportamiento de un agente reactivo se representa mediante una función:

**acción:  $E \rightarrow A$**

- **Para cualquier agente reactivo hay un modelo de agente como el definido previamente, lo inverso no es generalmente cierto.**
- Un termostato es un agente reactivo.

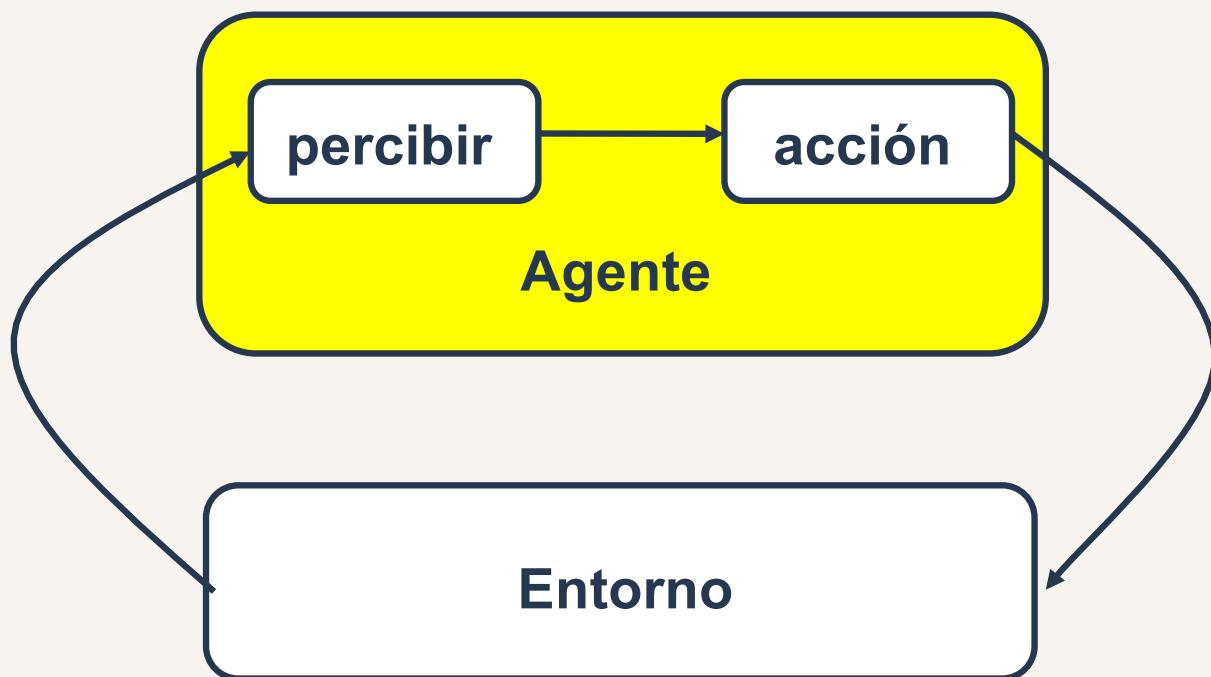
Estados del entorno = {temperatura OK, demasiado frio}

$$\text{acción(s)} = \begin{cases} \text{calentador off} & \text{si } s = \text{temperatura OK} \\ \text{calentador on} & \text{en cualquier otro caso} \end{cases}$$



# Bucle de funcionamiento: Percepción

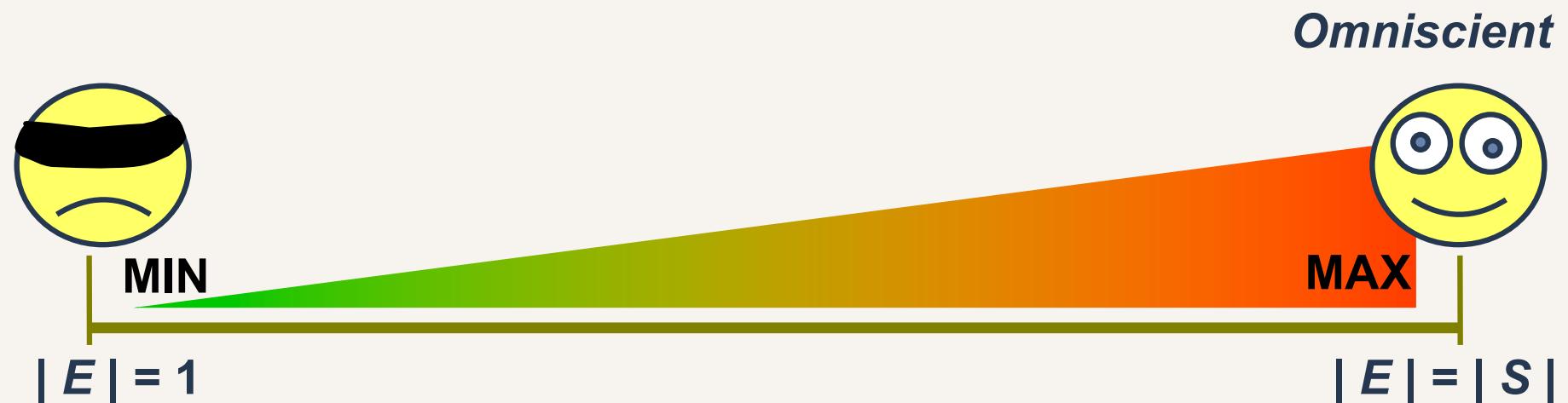
- **Bucle de funcionamiento: Funciones percibir y actuar:**



# Percepción (cont.)

- *La función percibir modela la capacidad del agente para percibir su entorno, mientras que la función actuar modela el proceso de toma de decisión del agente.*
- *La Percepción* es el resultado de la función **percibir:  $E \rightarrow P$**  donde  $P$  es un conjunto (no vacío) de **percepciones** (entradas perceptivas), que relaciona estados del entorno con percepciones.
- La decisión del agente es el resultado de la función **acción:**  
**acción:  $P^* \rightarrow A$**  que relaciona secuencias de percepciones ( $P^*$ ) con acciones.

# Capacidad de percepción



donde

$E$ : es el conjunto de los estados percibidos diferentes

Dos estados diferentes  $s_1 \in S$  y  $s_2 \in S$  ( $s_1 \neq s_2$ ) son indistinguibles  $\text{percibir}(s_1) = \text{percibir}(s_2)$

# Capacidad de Percepción (cont.)

Ejemplo:

Si

$x$  = “La temperatura de la habitación es OK”

$y$  = “No hace calor en este instante”

entonces:

$$S = \{ (x, y), (x, \neg y), (\neg x, y), (\neg x, \neg y) \}$$

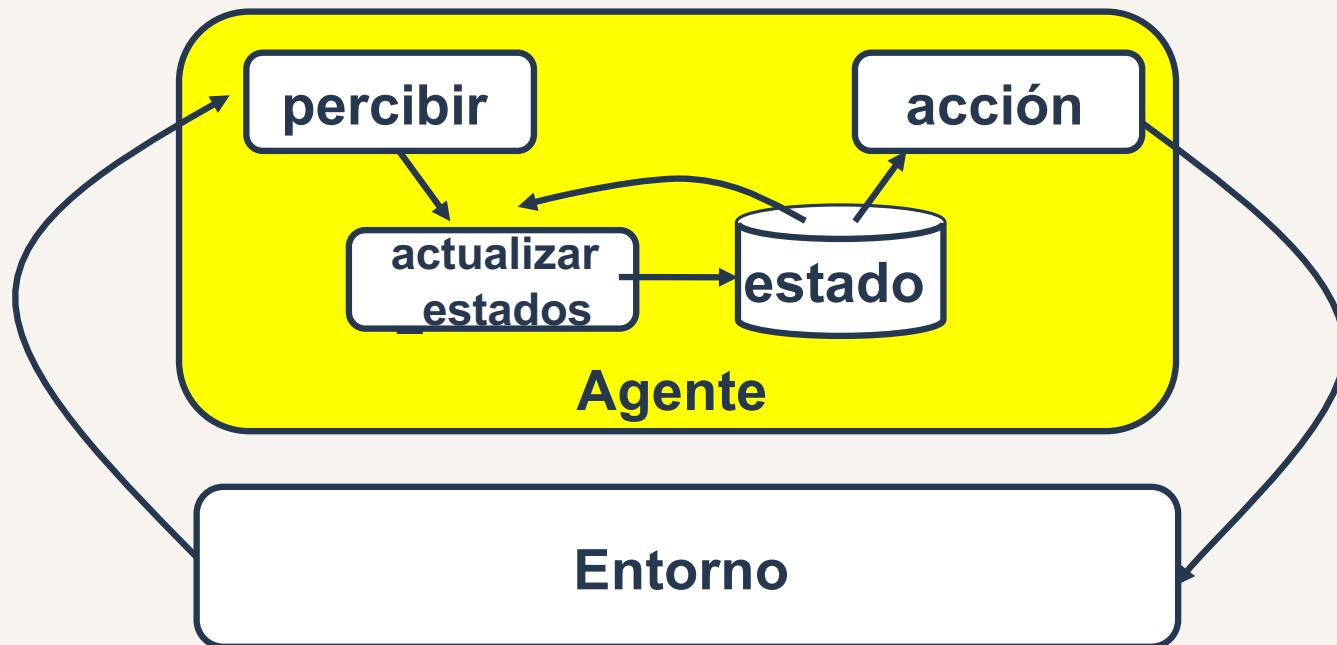
s1      s2      s3                  s4

pero para el termostato:

$$\text{percibir}(s) = \begin{cases} p_1 & \text{si } s=s_1 \text{ o } s=s_2 \\ p_2 & \text{si } s=s_3 \text{ o } s=s_4 \end{cases}$$

# Agentes basados en modelos (con estado)

- *Ahora consideramos agentes que mantienen el estado, su bucle de funcionamiento es:*



**Puntualización:** considerar agentes a este nivel de abstracción facilita su análisis, pero no nos ayuda a construirlos.

# Agentes con estado (cont.)

- Estos agentes tienen una estructura de datos interna que es utilizada para recordar información sobre la historia y estado del entorno.
- Tienen la misma función de percepción:

**percibir:  $E \rightarrow P$**

- La función acción-selección es definida ahora como mapping entre estados internos y acciones:

**acción:  $I \rightarrow Ac$**

donde:

**$I$ :** es el conjunto de todos los estados internos del agente

- Se introduce una nueva función de siguiente estado que define un estado interno a partir de un estado interno y las percepciones :

**actualizar\_estados:  $I \times P \rightarrow I$**

Esta función determinara el nuevo estado interno a partir del estado interno en que se encuentre el agente y las percepciones que perciba en su entorno.

# Búcle de control del Agente con estado

1. El agente inicia su funcionamiento en algún estado inicial  $i_0$ ,

$I_0 = \text{actualizar\_estados}(s_0, \text{percibir}(s_0)); i=1;$

2. Repetir (repeat forever)

2.1 Observar el estado del entorno  $s_i$  y generar una percepción mediante la función de percepción

$p_i = \text{percibir}(s_i)$

2.2 Actualizar el estado interno mediante la función siguiente estado

$I_i = \text{actualizar\_estados}(i_{i-1}, p_i)$

2.3 La acción seleccionada por el agente será:

$a_i = \text{acción}(I_i)$

2.4 Ejecutar la acción  $a_i$ ,  $i=i+1$

# Tema 1 - Índice

1.1 Definición de agente.

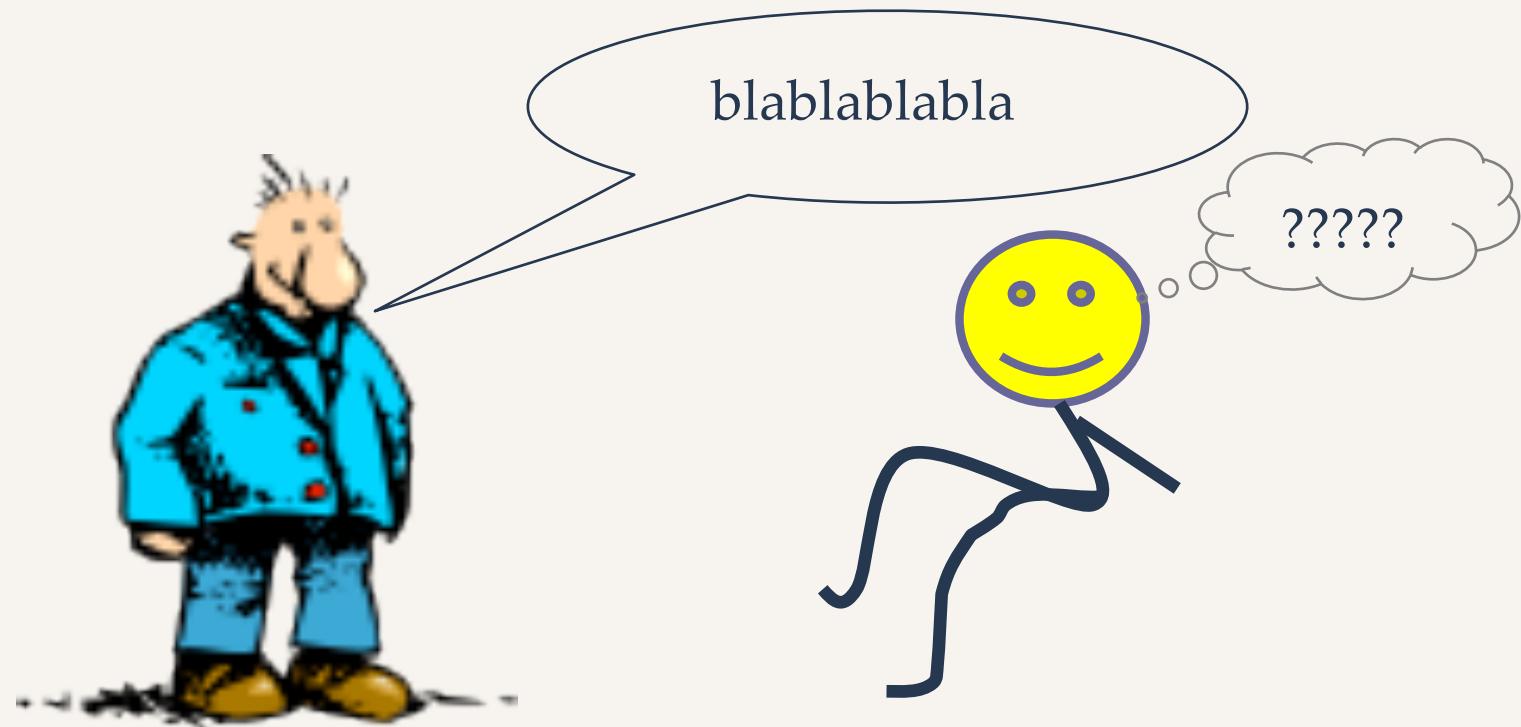
1.2 Entornos de agente.

1.3 Agentes como sistemas intencionales.

1.4 Arquitecturas abstractas para agentes inteligentes.

1.5 ¿Cómo decirle a un agente lo que tiene que hacer?

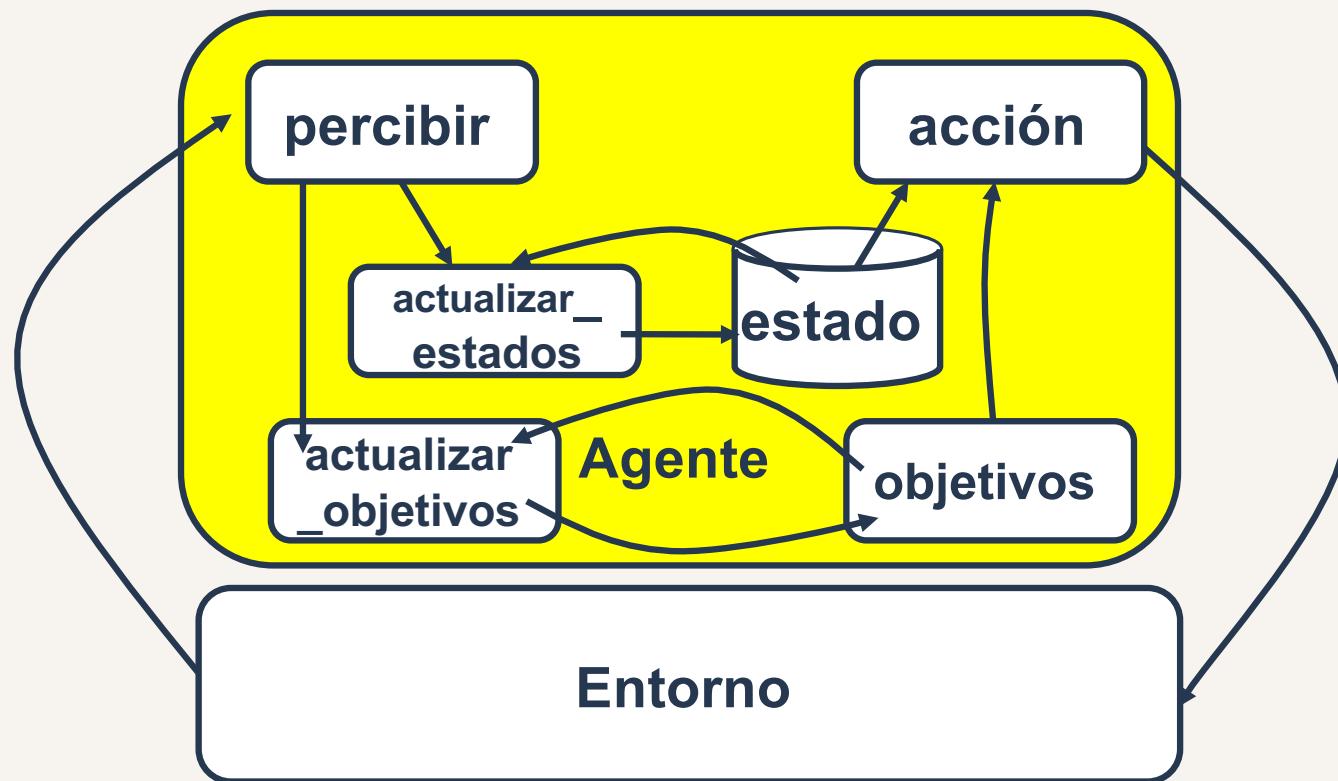
## 2.7 ¿Cómo le decimos a un agente lo que tiene que hacer?



Mediante objetivos  
Mediante funciones de utilidad

# Agentes basados en objetivos

- **Ahora consideramos agentes que mantienen el estado y que tienen objetivos que satisfacer, su bucle de funcionamiento es:**



# en objetivos (Cont.)

Estos agentes tienen la misma estructura y funciones que los agentes con estado (percibir, siguiente), se introduce un nuevo conjunto, el conjunto de objetivos  $G$  (cada uno de ellos es una descripción de un estado o conjunto de estados meta del agente –estados del entorno que el agente desea alcanzar–).

Se introduce una nueva función para actualizar los objetivos del agente a partir de las percepciones del mismo:

$$\text{actualizar\_objetivos: } G \times P \rightarrow G$$

- **Y redefinimos la función acción:**

$$\text{acción: } I \times G \rightarrow Ac$$

**La acción seleccionada por el agente dependerá del estado interno y de los objetivos que el agente querrá alcanzar.**

# Agentes basados en objetivos (Cont.)

1. El agente inicia su funcionamiento en algún estado inicial  $s_0$ ,

$I_0 = \text{actualizar\_estados}(s_0, \text{percibir}(s_0)); i=1;$

2. Repetir (repeat forever)

- 2.1 Observar el estado del entorno  $s_i$  y generar una percepción mediante la función de percepción

$p_i = \text{percibir}(s_i)$

- 2.2 Actualizar el estado interno mediante la función siguiente estado

$I_i = \text{actualizar\_estados}(I_{i-1}, p_i)$

- 2.3 Actualizar los objetivos a partir de las nuevas percepciones

$g = \text{actualizar\_objetivos}(I_i, g)$

- 2.4 La acción seleccionada por el agente será:

$a_i = \text{acción}(I_i, g)$

- 2.5 Ejecutar la acción  $a_i$ ,  $i = i + 1$

# Tema 1 - Índice

1.1 Definición de agente.

1.2 Entornos de agente.

1.3 Agentes como sistemas intencionales.

1.4 Arquitecturas abstractas para agentes inteligentes.

1.5 ¿Cómo decirle a un agente lo que tiene que hacer?

**Agentes basados en la utilidad**

# Agentes basados en la utilidad

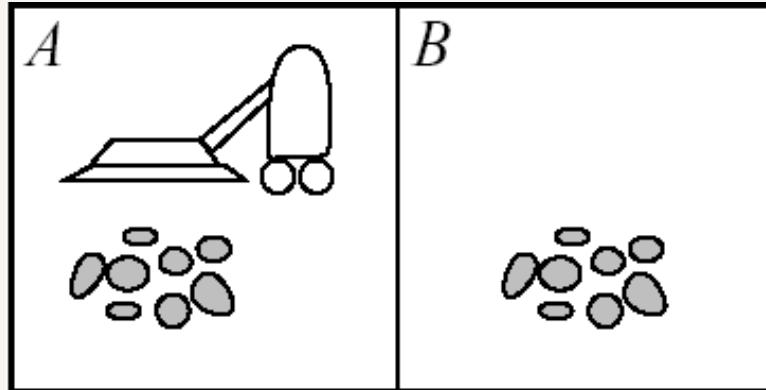
Construimos agentes para que realicen tareas por nosotros

Las tareas deben ser especificadas por nosotros ....

Queremos decirle a los agentes que tienen que hacer, pero sin decirle como lo tienen que hacer.

# El mundo de la aspiradora

(AIMA –Russell and Norvig)



Percepciones: localización y contenidos, por ejemplo, *[A, Sucio]*.

Acciones: *Izquierda, Derecha, Aspirar, NoOp*.

# El agente de la aspiradora

Secuencia de percepciones	Acción
$[A, Limpio]$	<i>Derecha</i>
$[A, Sucio]$	<i>Aspirar</i>
$[B, Limpio]$	<i>Izquierda</i>
$[B, Sucio]$	<i>Aspirar</i>
$[A, Limpio], [A, Limpio]$	<i>Derecha</i>
$[A, Limpio], [A, Sucio]$	<i>Aspirar</i>
⋮	⋮

función AGENTE-ASPIRADORA-REACTIVO([localización, estado]) devuelve una acción

```
Si estado = Sucio entonces {devolver Aspirar}
sino {si localización = A entonces {devolver Derecha}
      sino {si localización = B entonces {devolver Izquierda }}}}
```

# Agente basado en la utilidad: Racionalidad

Una única medida de rendimiento evalúa la secuencia del entorno:

- ❖ ¿Un punto por cada recuadro eliminado en un tiempo  $T$ ?
- ❖ ¿Un punto por cada recuadro limpio por período, menos un punto por movimiento?
- ❖ ¿Con penalización para  $> k$  recuadros sucios?

Un agente racional emprende aquella acción que maximice el valor esperado de la medida de rendimiento, basándose en las evidencias aportadas por la secuencia de percepciones.

Racional  $\neq$  omnisciente.

Racional  $\neq$  clarividente.

Racional  $\neq$  exitoso.

# Agente basado en la utilidad: entorno

Para diseñar un agente racional, debemos especificar el entorno de trabajo.

Por ejemplo, considere, el entorno de diseñar un taxi automático:

¿Cuál es la medida de rendimiento? Seguridad, destino, beneficios, legal, viaje confortable ...

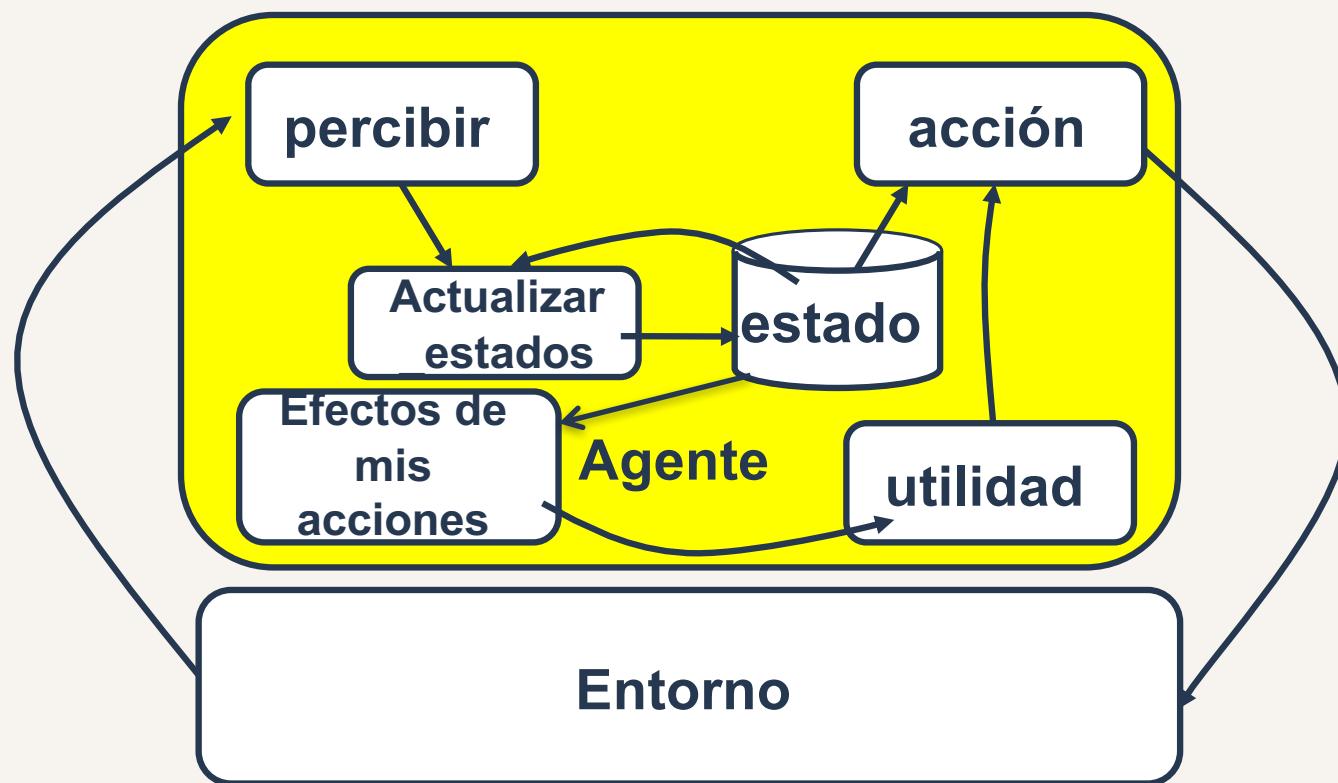
¿Cuál es el entorno? Carreteras, tráfico, peatones, el tiempo ...

¿Cuáles son los actuares? Dirección, acelerador, freno, bocina, visualizador ...

¿Cuáles son los sensores? Cámara, visualizador de la aceleración, indicador de la gasolina, sensores del motor, teclado, GPS ...

# Agente basado en la utilidad

El bucle de funcionamiento es:



# Tema 1 - Índice

1.1 Definición de agente.

1.2 Entornos de agente.

1.3 Agentes como sistemas intencionales.

1.4 Arquitecturas abstractas para agentes inteligentes.

1.5 ¿Cómo decirle a un agente lo que tiene que hacer?

## Funciones de utilidad

# Funciones de utilidad en estados

Una utilidad es un valor numérico que representa cuan ‘bueno’ es un estado, cuando mayor es la utilidad, mejor es el estado.

La tarea del agente es alcanzar estados (seleccionar acciones) que maximicen la utilidad (al agente no se le indica que tiene que hacer para alcanzar los estados).

Una especificación de tarea es una función

$$u: E \rightarrow \mathbb{R}$$

que asocia un número real a cada estado del entorno.

# Funciones de utilidad en estados

¿Pero cual es el valor de una ejecución de agente?

¿La mínima utilidad del estado de una ejecución? (pesimista)

¿La máxima utilidad del estado de una ejecución? (optimista)

¿La suma de las utilidades de los estados de una ejecución?

¿La media?

Desventajas: al asignar utilidades a estados locales, es difícil precisar una visión a largo plazo.

# Funciones de utilidad en ejecuciones

Una solución: no asignar una utilidad a estados individuales sino a las ejecuciones del agente:

$$u: \mathcal{R} \rightarrow \mathbb{R}$$

- Esta aproximación tiene inherentemente una visión a largo plazo.
- Otras variaciones, incorporar las probabilidades de los diferentes estados emergentes.
- En cada decisión del agente de qué acción elegir, elegir aquella que da lugar al estado de mayor utilidad.

# Problemas con las aproximaciones basadas en la utilidad

- ¿De donde vienen los números? (Peter Cheeseman)
- La gente no piensa en términos de utilidades, es difícil para las personas especificar las tareas en estos términos.
- Sin embargo, funciona bien en ciertos escenarios.
- ...

# Utilidad en el ‘mundo de las baldosas’ [Pollack,1990]

- Simulación de un entorno de una cuadricula dimensional sobre el que hay agentes, baldosas, obstáculos y agujeros.
- Un agente puede moverse en cuatro direcciones: arriba, abajo, izquierda o derecha, y si está localizado al lado de una baldosa puede empujarla.
- Los agujeros tienen que ser rellenados por el agente con las baldosas. Un agente suma puntos rellenando huecos con baldosas, su objetivo es llenar tantos agujeros como sea posible.
- El ‘mundo de las baldosas’ cambia con la aparición y desaparición aleatoria de agujeros.

	agujero	agujero		
		baldosa		
			baldosa	
		(:)		

		agujero		
		baldosa		
			baldosa	agujero
agujero				

# Utilidad en el ‘mundo de las baldosas’

- Definimos la función de utilidad con la siguiente ecuación:

$$u(r) = \frac{\text{número de agujeros rellenados en } r}{\text{número total de agujeros que hay en } r}$$

- Entonces:
  - Si el agente rellena todos los agujeros, utilidad = 1.
  - Si el agente no rellena ningún agujero, utilidad = 0

# Utilidad esperada

- Denotamos por  $P(r | Ag, Env)$  la probabilidad de que ocurra la ejecución  $r$  cuando el agente  $Ag$  está situado en el entorno  $Env$ . Observar que se tiene que cumplir que:
- La utilidad esperada del agente  $Ag$  en el entorno  $Env$  (dados  $P, u$ ) es:

$$\sum_{r \in R(Ag, Env)} P(r | Ag, Env) = 1 \quad (1)$$

$$EU(Ag, Env) = \sum_{r \in R(Ag, Env)} U(r)P(r | Ag, Env)$$

# Un ejemplo

- Consideremos el entorno  $Env1 = \langle E, s_0, \tau \rangle$  definido de la siguiente forma:

$$E = \{e_0, e_1, e_2, e_3, e_4, e_5\}$$

$$\tau(s_0 \xrightarrow{\alpha_0}) = \{e_1, e_2\}$$

$$\tau(s_0 \xrightarrow{\alpha_1}) = \{e_3, e_4, e_5\}$$

- Hay dos posibles agentes con respecto a este entorno:

$$Ag_1(e_0) = \alpha_0$$

$$Ag_2(e_0) = \alpha_1$$

# Un ejemplo

Las probabilidades de las diversas ejecuciones son:

$$P(e_0 \xrightarrow{\alpha_0} e_1 | Ag_1, Env_1) = 0.6$$

$$P(e_0 \xrightarrow{\alpha_0} e_2 | Ag_1, Env_1) = 0.4$$

$$P(e_0 \xrightarrow{\alpha_1} e_3 | Ag_2, Env_1) = 0.1$$

$$P(e_0 \xrightarrow{\alpha_1} e_4 | Ag_2, Env_1) = 0.2$$

$$P(e_0 \xrightarrow{\alpha_1} e_5 | Ag_2, Env_1) = 0.7$$

Asumiendo que la función de utilidad es (1):

$$u(e_0 \xrightarrow{\alpha_0} e_1) = 8$$

$$u(e_0 \xrightarrow{\alpha_0} e_2) = 11$$

$$u(e_0 \xrightarrow{\alpha_1} e_3) = 70$$

$$u(e_0 \xrightarrow{\alpha_1} e_4) = 9$$

$$u(e_0 \xrightarrow{\alpha_1} e_5) = 10$$

¿Cuáles son las utilidades esperadas de estos agentes para esta función de utilidad?

# Un ejemplo

¿Cuáles son las utilidades esperadas de estos agentes para esta función de utilidad?

$$EU(Ag_1, Env_1) = \sum_{r \in R(Ag, Env)} U(r)P(r | Ag_1, Env_1)$$

$$\begin{aligned} EU(Ag_1, Env_1) &= u(e_0 \xrightarrow{\alpha_0} e_1) \times P(e_0 \xrightarrow{\alpha_0} e_1 | Ag_1, Env_1) + u(e_0 \xrightarrow{\alpha_0} e_2) \times P(e_0 \xrightarrow{\alpha_0} e_2 | Ag_1, Env_1) = \\ &= 8 \times 0.6 + 11 \times 0.40 = 9,2 \end{aligned}$$

$$\begin{aligned} EU(Ag_2, Env_1) &= u(e_0 \xrightarrow{\alpha_1} e_3) \times P(e_0 \xrightarrow{\alpha_1} e_3 | Ag_2, Env_1) + u(e_0 \xrightarrow{\alpha_1} e_4) \times P(e_0 \xrightarrow{\alpha_1} e_4 | Ag_2, Env_1) + \\ &\quad u(e_0 \xrightarrow{\alpha_1} e_5) \times P(e_0 \xrightarrow{\alpha_1} e_5 | Ag_2, Env_1) = 70 \times 0.1 + 9 \times 0.20 + 10 \times 0.7 = 15.8 \end{aligned}$$

# Agentes óptimos

- El agente óptimo  $Ag_{opt}$  en un entorno  $Env$  es aquel que maximiza la utilidad esperada:

$$Ag_{opt} = \arg \max_{Ag \in AG} EU(Ag, Env)$$

- Por supuesto, el hecho de que un agente sea óptimo no significa que será el mejor, solo que en promedio podemos esperar que lo haga mejor

# Agentes óptimos limitados

- Hay agentes que no pueden ser implementados en algunos computadores.
- Denotamos por  $\mathcal{AG}_m$  el conjunto de los agentes que pueden ser implementados en una máquina (computador)  $m$ :

$$\mathcal{AG}_m = \{Ag \mid Ag \in \mathcal{AG} \text{ y } Ag \text{ puede ser implementado en } m\}$$

- El conjunto de agentes óptimos limitados,  $Ag_{bopt}$ , con respecto a  $m$  es entonces:

$$Ag_{bopt} = \arg \max_{Ag \in \mathcal{AG}_m} EU(Ag, Env) \quad (3)$$

# Especificaciones de predicado de tareas

- Un caso especial de asignación de utilidades a historias es asignar 0 (falso) o 1 (verdadero) a una ejecución.
- Si se asigna un 1 a una ejecución, entonces el agente *tiene éxito* en esa ejecución, en cualquier otro caso *falla*.
- Llamamos a esto especificaciones predicado de tareas.
- Denotamos las especificaciones de predicado de tarea mediante  $\psi$ :

$$\psi : \mathcal{R} \rightarrow \{0,1\}$$

# Especificaciones de predicado de tareas

- Un *entorno de tarea* es un par  $\langle Env, \psi \rangle$ , donde  $Env$  es entorno y:

$$\psi : \mathcal{R} \rightarrow \{0,1\}$$

es un predicado de ejecuciones.

- Sea  $\tau\mathcal{E}$  el conjunto de todos los entornos de tarea, un entorno de tarea especifica:
  - las propiedades del sistema en el que habitan los agentes;
  - Los criterios por los cuales se considera que un agente ha fracasado o tenido éxito.

# Especificaciones de predicado de tareas

- Denotaremos por  $\mathcal{R}_\psi(Ag, Env)$  el conjunto de todas las ejecuciones de un agente  $Ag$  en un entorno  $Env$  que satisface  $\psi$ :

$$\mathcal{R}_\psi(Ag, Env) = \{r \mid r \in \mathcal{R}(Ag, Env) \text{ y } \psi(r) = 1\}$$

- Entonces decimos que una agente  $Ag$  tiene éxito en un entorno de tarea si

$$\mathcal{R}_\psi(Ag, Env) = \mathcal{R}(Ag, Env)$$

# Probabilidad de éxito

- Sea  $P(r | Ag, Env)$  la probabilidad de que la ejecución  $r$  ocurra si el agente  $Ag$  está situado en el entorno  $Env$ .
- Entonces la probabilidad  $P(\psi | Ag, Env)$  de que  $\psi$  sea satisfecho por el agente  $Ag$  en el entorno  $Env$  será:

$$P(\psi | Ag, Env) = \sum_{r \in \mathcal{R}_\psi(Ag, Env)} P(r | Ag, Env)$$

# Tareas de mantenimiento y de logro

- Los dos tipos de tareas más comunes son las de logro y mantenimiento:
  1. *Las tareas de logro* son las de la forma ‘conseguir el estado de sucesos  $\psi$ ’
  1. *Las tareas de mantenimiento* son de la forma ‘mantener el estado de sucesos  $\psi$ ’

# Tareas de mantenimiento y de logro

- Una tarea de logro es especificada por un conjunto  $G$  de estados ‘buenos’ o ‘objetivo’:  $G \subseteq E$

El agente tiene éxito si se garantiza que alcanza al menos uno de estos estados (no nos importa cual, todos son considerados igual de buenos)

- Un objetivo de mantenimiento es especificado por un conjunto  $B$  de estados ‘malos’:  $B \subseteq E$

El agente tiene éxito en un entorno determinado si consigue evitar todos los estados de  $B$ , es decir, si nunca ejecuta una acción cuyo resultado sea cualquier estado que ocurra en  $B$ .

# Síntesis de agentes

- La síntesis de agentes es una programación automática: el objetivo es tener una programa que tome un entorno de tarea y, a partir de este entorno de tarea, automáticamente genere un agente que tenga éxito en este entorno:

$$syn : \mathcal{TE} \rightarrow (\mathcal{AG} \cup \{\perp\})$$

(considerar  $\perp$  como el nulo de JAVA)

# Síntesis de agentes: solidez y completitud

- El algoritmo de síntesis es:
  - ✓ *solido* si cada vez que genera un agente este tiene éxito en el entorno de tarea que se le ha pasado como entrada, y
  - ✓ *completo* si está garantizado que genera un agente, siempre que exista un agente que tendrá éxito en el entorno de tarea que se le ha pasado como entrada.

# Síntesis de agentes: solidez y completitud

El algoritmo de síntesis es sólido si se satisface la siguiente condición:

$$syn(\langle Env, \psi \rangle) = Ag \Rightarrow \mathcal{R}(Ag, Env) = \mathcal{R}_\psi(Ag, Env)$$

y completo si:

$$\exists Ag \in \mathcal{AG} / \mathcal{R}(Ag, Env) = \mathcal{R}_\psi(Ag, Env) \Rightarrow syn(\langle Env, \psi \rangle) \neq \perp$$