# 1 AGENCIA

*Agencia e IA*

La **Inteligencia Artificial** (IA) tiene como objeto de estudio a las entidades inteligentes y su comportamiento; pero a diferencia de la filosofía, la psicología, las neurociencias, y demás disciplinas que comparten este objeto de estudio, su meta no tiene que ver únicamente con la **comprensión** de tales entidades, sino con su **construcción**. La construcción de **agentes racionales** constituye la idea central del curiosamente llamado enfoque moderno de la IA[1], propuesto por Russell y Norvig [161]. De hecho, como se muestra en la Figura 1.1, Varela [181] ubica a la IA como parte de las Ciencias Cognitivas resaltando esta asimetría: su interés, único en el área, por la síntesis de entidades inteligentes.

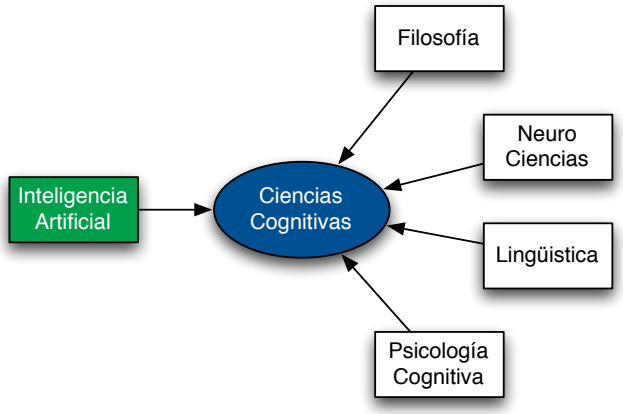


**Figura 1.1:** La IA como parte de las Ciencias Cognitivas, según Varela [181], resalta por su interés en la síntesis de entidades inteligentes y no solo en su característico análisis presente en otras disciplinas.

*Organización del capítulo*

En este capítulo se abordarán de manera informal, una serie de conceptos que conforman un vocabulario básico de la programación de agentes inteligentes. La Sección 1.1 presenta una definición de agente racional con elementos computacionales, propia del área de los **Sistemas Multiagentes** (SMA). La Sección 1.2 introduce una posible caracterización del **comportamiento flexible y autónomo** que los agentes racionales deben observar, con especial énfasis en la autonomía. La Sección 1.3 introduce el concepto de **medio ambiente** y la relación con los agentes racionales en él situados. La Sección 1.4 presenta una **arquitectura abstracta** de agente y su adaptación a diferentes tipos de agentes reportados en la literatura. Finalmente, la Sección 1.6 propone una serie de lecturas suplementarias y ejercicios. Es de

1 AIMA (*Artificial Intelligence: A Modern Approach*) es el libro de texto más usado en el mundo sobre IA. Se tiene registro de su uso en más de 1200 universidades de más de 100 paises. A diferencia de estas notas, AIMA es un libro genérico de introducción a la IA. La página web del libro se encuentra en: `http://aima.cs.berkeley.edu/index.html`

esperar que la aproximación informal a estos conceptos facilite este primer encuentro, así como su posterior formalización y estudio a profundidad.

## 1.1 AGENTES COMPUTACIONALES

*Agencia y filosofía*

*Agencia y derecho*

Históricamente, fuera de la IA, el término agente ha sido usado con dos acepciones. Primero, a partir de Aristóteles [5] y hasta nuestros días, los **filósofos** usan el término agente para referirse a una entidad que actúa con un **propósito** dentro de un contexto social. Segundo, la noción **legal** de agente, como la persona que actúa en beneficio de otra con un propósito específico, bajo la **delegación** limitada de autoridad y responsabilidad, estaba ya presente en el derecho Romano y ha sido ampliamente utilizada en economía [132].

*Agencia y computación*

En el contexto de la computación, Wooldridge [186], argumenta que el concepto de agente se consolida como una solución a una demanda propia de los **entornos computacionales actuales**, caracterizados por su: ubicuidad, interconexión, inteligencia, delegación y disponibilidad a amplios sectores de la población (Figura 1.2). Esto es, en entornos donde tenemos una gran variedad de usuarios, con una diversidad de dispositivos de cómputo distribuidos en nuestro entorno e interconectados, los agentes inteligentes emergen como la herramienta para **delegar** adecuadamente nuestro trabajo y abordar esta problemática desde una perspectiva más familiar para usuarios especializandos, no especializados, programadores y diseñadores.

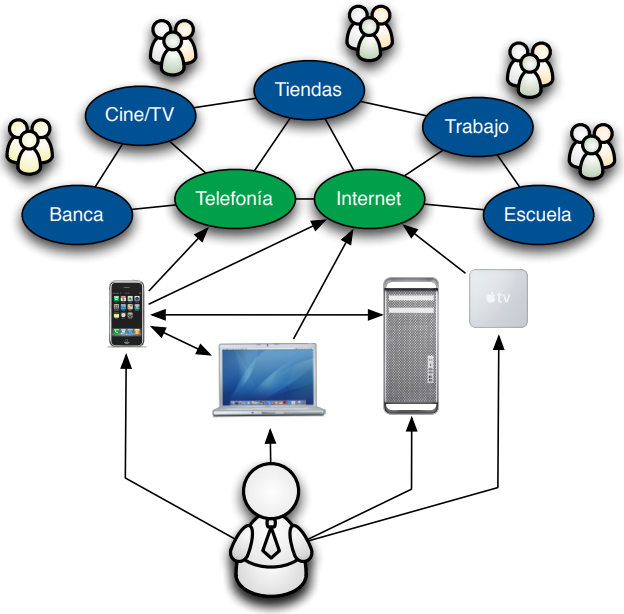


**Figura 1.2:** Entornos computacionales ubicuos, distribuidos, e interconectados, de amplia disponibilidad, demandan la delegación de tareas en agentes inteligentes. Adaptado de Wooldridge [186].

Pero, **¿Qué es un agente inteligente?** Como veremos, la respuesta no es sencilla, ni unívoca. Franklin y Graesser [66] argumentan que todas las definiciones del término agente en el contexto de la IA, se basan en alguna de las

dos acepciones históricas mencionadas al inicar esta sección. Una definición consensual de agente [187, 161] en este contexto, puede ser:

*Definición consensual*

> Un **agente** es un sistema computacional capaz de actuar de manera **autónoma** para satisfacer sus objetivos y **metas**, mientras se encuentra **situado** persistentemente en su medio ambiente.

*Ventajas de la definición*

Esta definición provee una **abstracción** del concepto de agente basada en su presencia e interacción persistente con el medio ambiente. Como se puede ver en la Figura 1.3, las **acciones** del agente modifican el medio ambiente; mientras que los cambios en el medio ambiente son **percibidos** por el agente. Eventualmente, las acciones del agente lo llevan a la consecución de sus objetivos. Russell y Subramanian [162] encuentran que tal abstracción presenta tres ventajas al permitirnos:

1. Concentrarnos en las **facultades cognitivas** de los agentes, en función del servicio que prestan para computar **cómo hacer lo correcto**.
2. Considerar diferentes **tipos de agente**, incluyendo aquellos que no se supone tengan tales facultades cognitivas.
3. Considerar diferentes **especificaciones** sobre los subsistemas que componen los agentes.

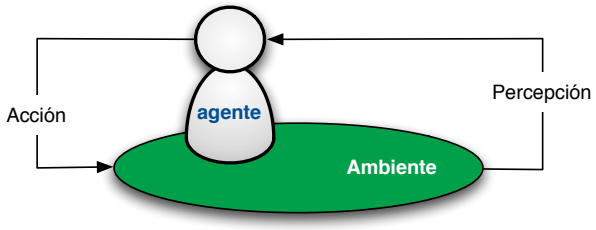


**Figura 1.3:** Abstracción de un agente a partir de su interacción persistente con el medio ambiente, para la consecución de sus metas.

*Problemas de la definición*

Sin embargo, la definición consensual de agente es demasiado **genérica** si queremos hablar de agentes inteligentes. Consideren el programa `xbiff` [2], descrito en el Ejemplo 1.1, que difícilmente puede ser percibido como un agente inteligente, pero se ajusta a la definición consensual del término.

**Ejemplo 1.1.** *`xbiff` es una pequeña interfaz gráfica, para un sistema de notificaciones de correo electrónico en UNIX. Cuando un mensaje llega al buzón del usuario, `xbiff` despliega el ícono que se muestra en la ventana superior izquierda (bandera arriba) de la Figura 1.4.*

Observen que esta pequeña aplicación se las arregla para identificar a su usuario, encontrar su buzón de correo electrónico en la red, buscar mensajes nuevos y comunicar al usuario la presencia de éstos. Es decir, `xbiff` es un sistema de cómputo que actúa persistentemente en su entorno UNIX,

2 Una descripción completa se puede consultar en `http://www.x.org/archive/X11R7.6/doc/man/man1/xbiff.1.xhtml`

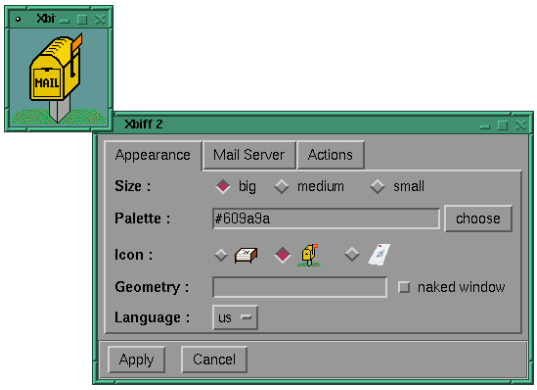


**Figura 1.4:** La aplicación `xbiff` para notificaciones de correo electrónico en UNIX.

percibiendo la llegada de mensajes nuevos para satisfacer su objetivo: tenernos al tanto de ello. Además, las tres ventajas de la abstracción de agente aplican aquí: Puede que `xbiff` no parezca muy listo, pero **hace lo correcto** y su funcionamiento es claro, independientemente de la especificación e implementación subyacentes.

*Medida de desempeño*

¿Cómo sabemos que un agente hace lo correcto? El término **medida de desempeño** se refiere al criterio usado para determinar el éxito de un agente.

**Ejemplo 1.2.** *La medida de desempeño para `xbiff` podría ser:*

$$md(R) = \frac{notificaciones(R)}{mensajesRecibidos(R)} \tag{1.1}$$

*esto es, la medida de desempeño md de una corrida R del programa será uno, si todos los mensajes recibidos en la corrida han sido notificados; si el agente deja pasar mensajes recibidos sin notificar, su desempeño tenderá a cero.*

*Racionalidad y desempeño*

Las medidas de desempeño, como criterio para establecer la correctez de un comportamiento, nos provee de una definición funcional de racionalidad. Se dice que un agente es **racional** si hace lo correcto, es decir, si maximiza su medida de desempeño. Esto reduce el problema de establecer cuando un agente es racional, al de definir **cómo y cuando evaluar** el desempeño de un agente.

Es preferible manejar una medida de desempeño objetiva, impuesta por alguna forma de autoridad. Esto es, nosotros como observadores estableceremos para cada agente, un estándar de lo que significa ser exitoso en un ambiente dado, y usaremos ese estándar para medir el desempeño del agente. Bajo ninguna circunstancia, un agente podrá manipular [3] tal estándar. El ejercicio 1.1 propone definir una medida de desempeño para un caso menos trivial que `xbiff`: Un concurso de robots limpiadores.

*Evaluación de la racionalidad*

Desde esta perspectiva, la racionalidad se define en relación con el **éxito esperado**, dadas las capacidades y oportunidades presentes. Esto es, no podemos exigir a un agente que tome en cuenta lo que no puede percibir, o haga lo que sus actuadores no pueden hacer. Por lo tanto, para **evaluar** la racionalidad de un agente, además de la medida de desempeño, deben considerarse otros factores, que incluyen:

3 Un agente que puede manipular la medida de desempeño, suele llegar a razonamientos del tipo –¡Oh! Si yo ni quería terminar este curso; y actuar en consecuencia.

- La **secuencia de percepciones** del agente, esto es, todo lo que el agente haya percibido hasta el momento de la evaluación.
- El **conocimiento** del agente sobre el medio ambiente en el que está situado y el problema a resolver.
- La **competencia** del agente, entendida aquí como su capacidad, en términos del repertorio de acciones que éste puede llevar a cabo en su medio ambiente.

*Agente ideal*

Un **agente racional ideal** es aquel que para toda secuencia de percepciones posible, selecciona y ejecuta una acción que se espera maximice la medida de desempeño, con base en la información que proveen su percepción y conocimiento sobre el ambiente.

*Función de comportamiento*

Observen que podríamos describir a un agente por medio de una tabla o una **función de comportamiento**, donde se establecen las acciones que el agente toma en respuesta a cada posible secuencia de percepciones. Por lo tanto, una **función ideal** describe a un agente ideal y define su diseño. El Ejemplo 1.3 muestra la función ideal de `xbiff`.

**Ejemplo 1.3.** *En el caso de `xbiff` la función ideal es muy sencilla. Si `biff` avisa que llegó un mensaje nuevo, desplegar el ícono de mensaje nuevo; en cualquier otro caso, desplegar el ícono de mensajes leídos:*

$$selAcc(Per \leftarrow check()) = \begin{cases} set() & Si\ Per = true \\ unset() & En\ cualquier\ otro\ caso \end{cases} \tag{1.2}$$

*donde: $check()$ es una acción de `xbiff` que regresa true si hay un mensaje nuevo en el buzón; set() es la acción que sube la bandera en la interfaz; y unset() la baja.*

*Selección de acción*

Y es aquí donde los problemas de `xbiff` para ser calificado como un agente inteligente comienzan. ¿Porqué si esta aplicación cumple con la definición consensual de agente y tiene una función ideal, difícilmente se le ve como un agente inteligente? Primero, un programa con una función de comportamiento tan simple, no suele percibirse como un agente inteligente. Por un lado, el número de secuencias de percepción suele ser mucho mayor que $Per \in \{true, false\}$. De hecho, un problema central en la programación de agentes es la búsqueda de mecanismos que aproximen las funciones de comportamiento que no pueden definirse por extensión; Segundo, el repertorio de acciones de un agente suele ser mayor que $\{check(), set(), unset()\}$, por lo que una función de **selección de acción** es necesaria –En términos de Maes [121], un mecanismo para resolver el problema de decidir ¿Cómo hacer lo correcto?

Podemos decir que, dada su simplicidad, el comportamiento de `xbiff`, y otros agentes similares, difícilmente puede verse como flexible y autónomo; requisito para calificar a un agente de inteligente. Veamos a continuación la caracterización de este tipo de comportamiento.

## 1.2 COMPORTAMIENTO FLEXIBLE Y AUTÓNOMO

Independientemente de la implementación usada para construir a `xbiff`, no resulta natural identificar a los daemons de UNIX como agentes y menos

aún como agentes inteligentes. Foner [65] argumenta que para ser percibido como inteligente, un agente debe exhibir cierto tipo de comportamiento, caracterizado más tarde por Wooldridge y Jennings [187], como **comportamiento flexible y autónomo**. Este tipo comportamiento se caracteriza por su:

*Flexibilidad y autonomía*

- **Reactividad.** Los agentes inteligentes deben ser capaces de percibir su medio ambiente y responder a tiempo a los cambios en él, a través de sus acciones.
- **Iniciativa.** Los agentes inteligentes deben exhibir un comportamiento orientado por sus metas, tomando la iniciativa para satisfacer sus objetivos de diseño y posiblemente decidiendo que objetivo atender.
- **Sociabilidad.** Los agentes inteligentes deben ser capaces de interactuar con otros agentes, posiblemente tan complejos como los seres humanos, con miras a la satisfacción de sus objetivos. Esto incluye la capacidad de comunicarse, negociar y alcanzar acuerdos.

Aún considerando estas restricciones, y asumiendo lo caricaturesco del caso, sigue siendo difícil descartar la agencia de `xbiff`. Es necesaria una caracterización más detallada de autonomía, como la presentada por Covrigaru y Lindsay [44]. Su *desiderata*, que incluye algunos de los aspectos ya mencionados, expresa que un agente se percibe como **autónomo** en la medida en que:

*Autonomía en detalle*

1. Su comportamiento está orientado por sus **metas** y es capaz de **seleccionar** que meta va a procesar a cada instante.
2. Su existencia es **persistente**, se da en un período relativamente mayor al necesario para satisfacer sus metas.
3. Es lo suficientemente **robusto** como para seguir siendo viable a pesar de los cambios en el ambiente.
4. Puede interactuar con su ambiente en una modalidad de **procesador de información**.
5. Es capaz de exhibir una **variedad de respuestas**, incluyendo movimientos de adaptación fluidos; y su atención a los estímulos es **selectiva**.
6. Sus funciones, acciones o decisiones, **no están totalmente gobernadas** por un agente externo.
7. Una vez en operación, el agente no necesita ser **reprogramado** por un agente externo.

De entre todos estos puntos, el sexto requiere una defensa rápida puesto que la dimensión social de los agentes será abordada hasta el Capítulo 3 de este texto [4]. En este punto se puede adivinar una tensión entre autonomía y sociabilidad ¿Para qué necesita un agente autónomo socializar con otros

*Autonomía y sociabilidad*

4 Para los impacientes, Castelfranchi [32] presenta una excelente introducción a las interacciones sociales entre agentes racionales.

agentes? La filosofía política liberal parece resolver esta cuestión: es aceptado que los agentes solo pueden llegar a ser autónomos si se da un conjunto de condiciones necesarias para ello. Dentro de estas condiciones, Rawls [156] habla de un **contexto de elección**, una pluralidad de bienes primarios que proveen los medios necesarios para que el agente tenga éxito en la satisfacción de sus intenciones. Tal pluralidad es posible únicamente, si el agente tiene una relación cercana con su **ambiente social y cultural**. No sólo ambos conceptos no están reñidos, sino que no puede haber autonomía sin sociabilidad. En el contexto de la IA, es Newell [135] quien ofrece argumentos similares –Los humanos deben vivir autónomamente en una comunidad social... Si se nos colocan fuera de nuestras comunidades, devenimos ineptos y disfuncionales de diversas maneras.

Covrigaru y Lindsay argumentan además que ser autónomo, depende no sólo de la habilidad para seleccionar metas u objetivos de entre un conjunto de ellos, ni de la habilidad de formularse nuevas metas, sino de tener el tipo adecuado de metas. Los agentes artificiales son usualmente diseñados para llevar a cabo tareas por nosotros, de forma que debemos comunicarles que es lo que esperamos que hagan. En un sistema computacional tradicional esto se reduce a escribir el programa adecuado y ejecutarlo. Un agente puede ser instruido sobre que hacer usando un programa, con la ventaja colateral de que su comportamiento estará libre de incertidumbre. Pero programar un agente de esta forma, atenta contra su autonomía, teniendo como efecto colateral la incapacidad del agente para enfrentar situaciones imprevistas mientras ejecuta su programa. Las **metas** y las funciones de **utilidad** son dos maneras de indicarle a un agente lo que hacer, sin decirle cómo hacerlo. Abundaremos más sobre ambas en la Sección 1.5.

*Metas, utilidad y autonomía*

Con los elementos discutidos hasta ahora, es posible definir a un **agente racional** como aquel que exhibe un comportamiento flexible y autónomo, mientras está situado en un sistema de información, por ej., un sistema operativo o el web. Esto constituye lo que Wooldridge y Jennings [187] llaman la **noción débil** de agente, en referencia indirecta a una **noción fuerte** de agencia, que además de las propiedades mencionadas hasta ahora, utiliza términos que hacen referencia a estados que solemos aplicar exclusivamente a los seres humanos: creencias, deseos, intenciones (**Modelo BDI**, por sus siglas en inglés). La noción fuerte de agencia es central en este curso y de ella nos ocuparemos en detalle en el capítulo 2. La noción débil de agencia contiene los elementos necesarios para abordar otro tema de interés más inmediato, la situación de los agentes en su medio ambiente.

*Agencia débil y fuerte*

*Modelo BDI*

## 1.3 MEDIO AMBIENTE

Por medio ambiente, entendemos el espacio donde un agente, o un grupo de ellos, se encuentra **situado**. Brooks [26] argumenta que el medio ambiente por excelencia es el mundo real, y en su propuesta todo agente toma una forma robótica. Por el contrario, Etzioni [58], considera que no es necesario que los agentes tengan implementaciones robóticas porque los ambientes virtuales, como los sistemas operativos y el web, son igualmente válidos que el mundo real.

*Real o virtual*

En este texto asumimos una tercera vía propuesta por Ricci [139, 158], donde la interacción de los agentes con el medio ambiente se concibe como una interfaz, una capa de servicios compuesta por objetos reactivos, más no autónomos ni proactivos, llamados **artefactos** . Como es de esperar, los agentes pueden usar estos artefactos para **sensar** y **actuar** en el medio ambiente; además pueden **conectar** unos con otros para obtener funcionalidades más complejas. Agentes y artefactos pueden organizarse en **espacios de trabajo**, para reflejar la geografía del medio ambiente y proveer una noción de localidad ó **sitio**. Esto constituye un meta-modelo que puede aproximar ambientes tanto reales, como virtuales, tal y como se muestra en la figura 1.5. El modelo de Agentes y Artefactos será estudiado en detalle en el capítulo 7.

*Artefactos*

*Espacio de trabajo*

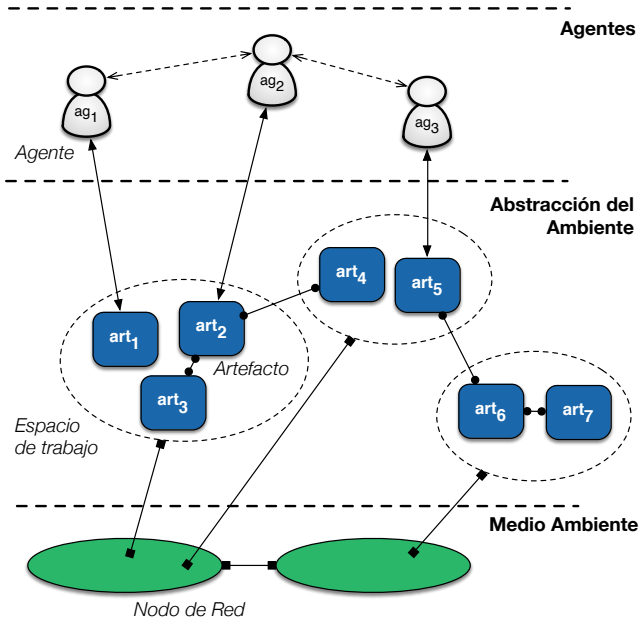


**Figura 1.5:** Medio ambiente, agentes y artefactos. Estos últimos proveen una interfaz entre los agentes y su medio ambiente.

El concepto de espacio de trabajo se asemeja al de **entorno de trabajo**, propuesto por Russell y Norvig [161] para especificar tipos de agente con base en la medida de desempeño utilizada, el ambiente donde el agente está situado, los actuadores del agente y sus sensores. Tal especificación suele conocerse como **PEAS** por sus siglas en inglés (*Performance, Environment, Actuators, Sensors*). Consideren el siguiente ejemplo de un agente taxista:

*Entorno de trabajo*

*Descripción PEAS*

**Ejemplo 1.4.** *Un agente taxista tiene la siguiente especificación PEAS:*

- ***Medida de desempeño.*** *Un taxista artificial debería: llegar al destino correcto; minimizar el consumo de gasolina; minimizar el tiempo y costo de la carrera; minimizar las infracciones de tránsito y las molestias ocasionadas a otros conductores; maximizar la seguridad y comodidad del pasajero; maximizar las ganancias. Observen que algunos componentes del desempeño son contradictorios y el agente deberá contender con tales contradicciones.*

- ***Ambiente.*** *La calle, o mejor aún las posibles calles donde conducirá. Esto incluye avenidas, callejones, caminos rurales, autopistas donde suele haber otros autos, peatones, perros, vacas, hoyos; y eventualmente nieve. O un simulador.*

*Evidentemente, entre más restringido sea el ambiente, más sencillo es el diseño del agente.*

- ***Actuadores.*** *Incluye el control de auto: acelerador, freno, cambio de velocidad, puertas, ventanas, limpia brisas, etc.; Y alguna forma de comunicarse con los clientes y otros autos, por ejemplo, una pantalla donde despegar mensajes, una bocina para hablar, etc.*

- ***Sensores.*** *Video cámaras controlables para ver el camino; sensores infra rojos o sonares para medir distancias; velocímetro; acelerómetro; sensores electrónicos de motor, gasolina, sistema eléctrico; GPS; Y posiblemente micrófono, pantalla táctil y/o teclado para comunicarse con el usuario.*

En el caso de los espacios de trabajo, sensores y actuadores están **encapsulados** en los artefactos. Los artefactos emergen al agrupar actuadores y sensores según convenga. Esto tiene una ventaja inherente: se puede usar cualquier metodología de programación para abordar estos componentes de cara al ambiente, por ejemplo una metodología orientada a objetos bajo Java, con su riqueza de librerías y extensibilidad; mientras que de cara al agente, los artefactos proveen un conjunto de operaciones y percepciones que los agentes usan para actualizar sus representaciones del medio ambiente y actuar en éste. El ejemplo 1.5 ilustra el caso con un artefacto tanque de gasolina. La descripción PEAS puede verse con una fase previa al diseño de artefactos y espacios de trabajo. De hecho, una descripción PEAS adecuada, es indispensable para el diseño de agentes siguiendo prácticamente cualquier metodología de **Ingeniería de Software Orientada Agentes** (AOSE, por sus siglas en inglés). El Cuadro 1.1 muestra otros ejemplos de descripciones PEAS para diversos agentes.

*AOSE*

**Ejemplo 1.5.** *El artefacto tanque de gasolina provee al agente la percepción del nivel de gasolina del auto; el agente puede efectuar las operaciones de abrir y cerrar el tanque. El auto sería modelado así, como un conjunto de artefactos que pueden interactuar con el agente y entre ellos. La percepción del nivel de gasolina puede ser usado por el agente para actualizar su representación del medio ambiente, por ejemplo con creencias como* $nivel(gasolina, bajo)$*; mientras que el computo del nivel de gasolina pasa por un programa que contienda con el sensor físico del auto. De igual forma, el agente contiende con las operaciones* $abrir(tanqueGas)$ *y* $cerrar(tanqueGas)$*; mientras que la implementación detallada de estas operaciones está en el artefacto tanque de gasolina.*

Ahora bien, los elementos que componen la descripción PEAS nos permiten establecer diferentes **propiedades** de los entornos de trabajo:

*Propiedades del entorno de trabajo*

- **Ambiente y sensores.** Si los sensores de un agente le permiten percibir el estado completo del ambiente en cualquier momento, decimos que el entorno de trabajo es **observable**. Los juegos formales, por ejemplo: crucigramas, ajedrez, backgammon, etc. constituyen entornos de trabajo observables. Un entorno de trabajo es **eficazmente observable** si los sensores del agente detectan todos los aspectos relevantes para decidir que acción debe llevarse a cabo. Relevancia aquí, depende de la definición de la medida de desempeño. Las líneas de montaje donde

*Observable*

*Eficazmente observable*

| Agente | Desempeño | Ambiente | Actuadores | Sensores |
|---|---|---|---|---|
| diagnóstico médico | salud paciente, minimizar costos y demandas | paciente, hospital, persona, tratamientos | preguntas, pruebas, diagnósticos, | lecturas, reportes, respuestas |
| análisis imágenes de satélite | clasificación correcta | satélite, control | despliegue imagen | color, intensidad |
| brazo robótico | porcentaje de piezas colocadas | línea de producción | codo, mano | cámara, ángulo de articulación |

**Cuadro 1.1:** Agentes y su descripción PEAS, adaptado de Russell y Norvig [161].

se situan brazos robóticos suelen ser entornos de trabajo efectivamente observables. Un entorno de trabajo puede ser **parcialmente observable** debido a la imprecisión y el ruido en los sensores; o bien porque algunos aspectos del ambiente caen fuera del rango de lectura de los sensores. El poker, los ambientes donde navegan los robots o conducimos nuestros autos suelen ser parcialmente observables. Los entornos de trabajo observables son los más convenientes, ya que un agente no necesita mantener el historial de estados del ambiente para ser efectivo en ellos. Si el agente no tiene sensores, se dice que el entorno de trabajo es **inobservable**.

*Parcialmente observable*

*Inobservable*

- **Número de agentes y sus interacciones.** Si un agente resuelve solo su tarea, ya sea porque es el único agente en el ambiente, o porque no interactua con los demás agentes, decimos que el entorno de trabajo es **monoagente** . En caso contrario, decimos que se trata de un entorno de trabajo **multiagente**. A pesar de que esta categorización de los ambientes parece obvia, esconde una cuestión en extremo relevante: ¿Qué entidades en el sistema son considerados por el agente como agentes? Como hemos visto, el concepto de artefacto nos permite concebir entidades en el ambiente que no son agentes y el comportamiento autónomo y flexible es la clave para distinguir ambas entidades. Sin embargo, la situación puede ser ambigua. Por ejemplo, en el caso del agente taxista,¿Debe el agente concebir los otros autos como agentes o como artefactos? La clave aquí está en el concepto de concepto de **interacción**. Si el agente ve al otro auto como un objeto que sigue las leyes de la física mientras se mueve en el mismo espacio, posiblemente se le deba modelar como un artefacto. Sin embargo, si el otro auto se concibe como una entidad autónoma y flexible que me puede obstaculizar o ayudar a circular más rápido, definitivamente debe ser modelado como un agente. De forma que puede haber entornos de trabajo **competitivos**, como el ajedrez, o los otros autos buscando el mejor carril; o **colaborativos** cuando por ejemplo, otro auto me deja pasar, o un grupo de agentes busca dar el mejor diagnóstico posible. Los entornos de trabajo multiagente introducen aspectos que no se consideran en el caso monoagente, por ejemplo: comunicación, negociación, colaboración, etc.

*Monoagente*
*Multiagente*

*Competitivo*
*Colaborativo*

- **Ambiente y actuadores.** Si el próximo estado del ambiente depende exclusivamente de su estado actual y de la acción que ejecuta el agente, se dice que el ambiente es **determinista**. Evidentemente, todos los juegos formales son deterministas. Si otros factores influyen en el próximo estado del ambiente, éste es **estocástico**. Si el ambiente es parcialmente observable, entonces parecerá ser estocástico. Esto es particularmente cierto en el caso de ambientes complejos, donde es difícil dar seguimiento a los aspectos del ambiente. Si el entorno de trabajo es estocástico o parcialmente observable, decimos que es **incierto**. El término estocástico refuerza aquí el hecho de que en estos entornos, la incertidumbre se cuantifica en términos de probabilidades. Un entorno de trabajo se dice **no determinista** si los actuadores se caracterizan en términos de sus posibles resultados, pero ninguna probabilidad es asociada a estos. El carácter estocástico o no determinista del entorno de trabajo captura dos nociones importantes:

  *Determinista*
  *Estocástico*
  *Incierto*
  *No determinista*

  1. El hecho de que los agentes tienen una esfera de influencia limitada, es decir, en el mejor de los casos tienen un control parcial de su ambiente;
  2. y el hecho de que las acciones de un agente puede fallar y no lograr el resultado deseado por el agente.

  Si el entorno de trabajo es determinista, excepto por las acciones de otros agentes, como en los juegos con contrincante, se dice que es **estratégico**. Es más sencillo construir agentes en entornos de trabajo deterministas.

  *Estratégico*

- **Episódico o secuencial.** Si el desempeño del agente puede evaluarse en rondas, se dice que el ambiente es **episódico**. Las acciones se evalúan en cada episodio o ronda, esto es, la calidad de la acción en los episodios subsecuentes no depende de las acciones ocurridas en episodios previos. Por ejemplo, el detector de basura en las botellas de una cervecería es episódico: la decisión de si una botella esta sucia o no, no depende de los casos anteriores. Dada la persistencia temporal de los agentes, estos tienen que hacer continuamente decisiones locales que tienen consecuencias globales. Los ambientes episódicos reducen el impacto de estas consecuencias, y por lo tanto es más fácil construir agentes en ambientes episódicos. Si el ambiente no es episódico, se dice que es **secuencial**.

  *Episódico*
  *Secuencial*

- **Estático o Dinámico.** Si el ambiente puede cambiar mientras el agente se encuentra deliberando, se dice que es **dinámico**; de otra forma, se dice que es **estático**. Si el ambiente no cambia con el paso del tiempo, pero si lo hace con las acciones del agente, se dice que el ambiente es **semi-dinámico**. Los ambientes dinámicos tienen dos consecuencias importantes:

  *Dinámico*
  *Estático*
  *Semi-dinámico*

  1. Un agente debe percibir continuamente, porque aún si no ha ejecutado ninguna acción entre los tiempos $t_0$ y $t_1$, el agente no puede asumir que el estado del ambiente sea el mismo en $t_0$ que en $t_1$;

2. Otros procesos en el ambiente pueden interferir con las acciones del agente, incluyendo las acciones de otros agentes.

Es más sencillo diseñar agentes en ambientes estáticos.

- **Discreto o Continuo.** Si hay un número limitado de posibles estados del ambiente, distintos y claramente definidos, se dice que el ambiente es **discreto**; de otra forma se dice que es **continuo**. Esta propiedad puede aplicarse al estado del ambiente; a la forma en que se registra el tiempo y; a las percepciones y acciones de los agentes. Es más fácil construir agentes en ambientes discretos, porque las computadoras también son sistemas discretos y aunque es posible simular sistemas continuos con el grado de precisión deseado, una parte de la información disponible se pierde al hacer esta aproximación. Por lo tanto, la información que manejan los agentes discretos en ambientes continuos es inherentemente aproximada.

*Discreto*
*Continuo*

Esta categorización sugiere que es posible encontrar diferentes clases de entornos de trabajo. Russell y Norvig [161] presentan algunos ejemplos de ambientes bien estudiados en Inteligencia Artificial y sus propiedades (ver Cuadro 1.2). Cada ambiente, o clase de ambientes, requiere de alguna forma agentes diferentes para que estos tengan éxito. La clase más compleja de ambientes corresponde a aquellos que son inaccesibles, no episódicos, dinámicos, continuos y multi-agente, lo que desgraciadamente corresponde a nuestro ambiente cotidiano.

| Ambiente | Observable | Determinista | Episódico | Estático | Discreto | SMA |
|---|---|---|---|---|---|---|
| Crucigrama | si | si | no | si | si | mono |
| Ajedrez con reloj | si | estratégico | no | semi | si | multi |
| Backgammon | si | estocástico | no | si | si | multi |
| Poker | parcial | estocástico | no | si | si | multi |
| Tutor inglés | parcial | estocástico | no | no | si | multi |
| Brazo robótico | efectivo | estocástico | si | no | no | mono |
| Control refinería | parcial | estocástico | no | no | no | mono |
| Robot navegador | parcial | estocástico | no | no | no | mono |
| Análisis imágenes | si | si | si | semi | no | mono |
| Manejo de autos | parcial | estocástico | no | no | no | multi |
| Diagnóstico | parcial | estocástico | no | no | no | mono |

**Cuadro 1.2:** Ejemplos de ambientes estudiados en Inteligencia Artificial y sus propiedades según Russell y Norvig [161].

## 1.4 ARQUITECTURAS DE AGENTE

Es posible formalizar la definición abstracta de agente que hemos presentado (pág. 4), en la forma de una **arquitectura abstracta** de agentes [186]. Primero asumiremos que el ambiente puede caracterizarse por medio de un conjunto finito de **estados** discretos posibles, definido como:

*Arquitectura abstracta*

*Estados*

$$E = \{e, e', \dots\} \tag{1.3}$$

Como hemos visto, no todos los ambientes son discretos, en el sentido estricto del término. Sin embargo, asumir que el ambiente es discreto facilitará esta primera aproximación formal al concepto de agente. Por otra parte, este supuesto se justifica por el hecho de que aún cuando el ambiente no fuese discreto, éste puede ser modelado como un ambiente discreto con cierto nivel deseable de precisión.

Definamos la **competencia** o **capacidad** de un agente, como el conjunto finito de **acciones** que éste puede ejecutar: *Acciones*

$$Ac = \{\alpha, \alpha', \dots\} \tag{1.4}$$

Ahora, supongamos que el ambiente se encuentra en algún estado inicial $e_0$ y que el agente comienza su actividad eligiendo una acción $\alpha_0$ en ese estado. Como resultado de esta acción, el ambiente puede responder con cierto número de estados posibles. Eventualmente, uno de estos estados posibles resulta de esta interacción, digamos $e_1$. Inmerso en este nuevo estado del ambiente, el agente selecciona otra acción a ejecutar, y así continua. Una **corrida** $r$ de un agente en un ambiente es, por lo tanto, una secuencia finita de estados y acciones intercalados: *Corrida*

$$r = e_0 \xrightarrow{\alpha_0} e_1 \xrightarrow{\alpha_1} e_2 \xrightarrow{\alpha_2} e_3 \xrightarrow{\alpha_3} \cdots \xrightarrow{\alpha_{u-1}} e_u \tag{1.5}$$

Sea $R$ el conjunto de todas las **posibles corridas finitas** sobre $E$ y $Ac$. Definimos $R^{Ac}$ como el subconjunto de las corridas que terminan en una acción y $R^E$ como el subconjunto de las corridas que terminan en un estado del ambiente. *Corridas posibles*

Para modelar el efecto de una acción en el ambiente, usamos una **función de transición** entre estados, similar a la propuesta por Fagin y col. [59], pero aquí $\tau$ es aplicada a un solo agente, estableciendo un mapeo entre las corridas del agente terminadas en acción y el conjunto de posibles estados del ambiente: *Función de transición*

$$\tau : R^{Ac} \to \wp(E) \tag{1.6}$$

El ambiente se asume sensible a la **historia**, su próximo estado no está determinado exclusivamente por la última acción ejecutada por el agente. Las acciones ejecutadas por el agente en el pasado, también afectan esta transición. Esta función también introduce un grado de **indeterminismo** en el ambiente, y por lo tanto, incertidumbre sobre el resultado de una acción ejecutada en cierto estado. Considerar que el ambiente es no determinista, es importante para entender las limitaciones de los agentes. *Memoria histórica* *Indeterminismo*

Si $\tau(r) = \emptyset$ para todo $r \in R^{Ac}$, entonces no existe ningún estado sucesor para la corrida $r$ del agente. En este caso, se dice que el sistema ha **terminado** su corrida. Asumimos que todas las corridas terminan eventualmente. *Condición de paro*

Un **ambiente** se define como $Env = \langle E, e_0, \tau \rangle$ donde $E$ es el conjunto de los posibles estados del ambiente, $e_0 \in E$ es un estado inicial y $\tau$ es la función de transición de estados. *Ambiente*

Los **agentes** se modelan como funciones que establecen un mapeo entre *Agente*

las corridas que terminan en un estado del ambiente y las acciones en la competencia del agente:

$$Ag : R^E \rightarrow Ac \tag{1.7}$$

Observen que mientras los ambientes se asumen no deterministas, asumimos que los agentes son deterministas.

Un **sistema agente** es una tupla conformada por un agente y un ambiente. El conjunto de posibles corridas del agente $Ag$ en el ambiente $Env$ se denota como $R(Ag, Env)$. Por simplicidad, consideraremos por ahora solo corridas finitas. Una secuencia de la forma:

*Sistema agente*

$$(e_0, \alpha_0, e_1, \alpha_1, e_2, \dots)$$

representa una corrida del agente $Ag$ en el ambiente $Env = \langle E, e_0, \tau \rangle$ si y sólo si $e_0$ es el estado inicial del ambiente $Env$; $\alpha_0 = Ag(e_0)$; y para $u > 0$:

$$e_u \in \tau((e_0, \alpha_0, \dots, \alpha_{u-1})) \quad \text{y} \quad \alpha_u = Ag((e_0, \alpha_0, \dots, e_u))$$

Puesto que nuestra tarea es implementar **programas de agente**, podemos usar la formalización propuesta para definir un programa de agente que acepte percepciones de su ambiente y regrese acciones sobre éste. Como habíamos mencionado (Página 6), la forma más sencilla de implementar un programa de agente es a través de su **función ideal**. El Algoritmo 1.1 muestra la implementación de un agente basado en su función ideal.

*Programa de agente*

*Agente función ideal*

**Algoritmo 1.1** Programa de agente basado en su función ideal

```
1: function AGENTEIDEAL(Per)          ▷ Per ∈ E es una percepción.
2:     R^E ← push(Per, R^Ac)          ▷ R^E, R^Ac = ∅ inicialmente.
3:     Acc ← Ag(R^E)                  ▷ Ag : R^E → Ac es la función ideal.
4:     R^Ac ← push(Acc, R^E)
5:     return Acc
6: end function
```

Aunque ésta es también la forma más sencilla de fracasar en la construcción de un agente inteligente, aún para agentes en ambientes observables, deterministas, estáticos y discretos. Consideren el caso del ajedrez, la tabla del mapeo alcanza ordenes de $10^{150}$ entradas. Si consideramos que el número de átomos en el universo observable es menor que $10^{80}$, entenderemos las razones del fracaso ineludible de esta estrategia: Los requerimientos de **almacenamiento** para $R^E$ son irreales; El **costo computacional** de $Ag$ está precisamente en función de $|R^E|$.

*Limitaciones de la función ideal*

Podemos definir un programa básico de ambiente, para ilustrar la relación entre éste y los agentes situados en él. El Algoritmo 1.2 muestra el programa del **ambiente**. El caso monoagente se da cuando $|Ags| = 1$.

*Programa del ambiente*

La función *percibir*/2 establece un mapeo entre un estado del ambiente y los sensores de un agente, para computar una **percepción**. Sea $Per$ un conjunto no vacío de percepciones, la función se define como sigue:

*Percepciones*

$$percibir : Ag \times E \rightarrow Per$$

Los agentes con formas robóticas, pueden implementar esta función usando hardware, por ejemplo: Cámaras de vídeo, sensores infrarrojos, sonares,

**Algoritmo 1.2** Programa de ambiente

```
1: procedure AMBIENTE(e, τ, Ags, fin)                ▷ e ∈ E estado inicial.
2:     repeat
3:         for all Ag ∈ Ags do
4:             actuar(Ag) ← agenteIdeal(percibir(Ag, e))
5:         end for
6:         e ← τ(⋃_{Ag∈Ags} actuar(Ag))      ▷ τ función de transición del ambiente.
7:     until fin(e)                          ▷ fin es un predicado de fin de corrida.
8: end procedure
```

etc. Los agentes software pueden implementar es función usando comandos del sistema operativo. El Ejemplo 1.6 muestra diferentes opciones de percepción para el agente `xbiff`.

**Ejemplo 1.6.** *Un agente del estilo de `xbiff` puede usar el comando de UNIX `whoami` para determinar la identidad de su usuario y encontrar así su buzón de correo. Luego puede usar el comando `grep` para contar el número total de mensajes, el número de mensajes leídos, y el número de mensajes abiertos pero no leídos* [5].

Observen que las percepciones y los estados del mundo no son lo mismo. El Ejemplo 1.7 muestra la diferencia entre percepción y estados posibles del ambiente, en el caso del agente `xbiff`.

**Ejemplo 1.7.** *Para el agente de estilo `xbiff`, podemos definir es conjunto de percepciones posibles como $Per = check \in \{true, false\}$, tal que $Per = true$ si hay un mensaje nuevo en el buzón (Ver Ecuación 1.2); Mientras que los estados posibles del ambiente E son más complejos, incluyendo como se menciona en el ejemplo anterior, el `login` del usuario, el número de cada tipo de mensaje en el buzón, etc.*

En este sentido, Levitin [117], en su libro sobre nuestro cerebro y la música, ilustra esta diferencia. La palabra tono se refiere a la representación mental que un organismo tiene de la frecuencia fundamental de un sonido. Un fenómeno enteramente psicológico relacionado con la frecuencia de las moléculas de aire vibrando a nuestro alrededor. Por psicológico entendemos que es resultado de una serie de procesos físico químicos en el agente que percibe el tono; que dan lugar a esa representación interna enteramente subjetiva. Isaac Newton fue el primero en darse cuenta que el color se percibe de manera similar –la luz no tiene color, ésta es una construcción de nuestro cerebro. El filósofo Georges Berkeley planteo esta curiosa pregunta: Si un árbol cae en el bosque y nadie está ahí para escucharlo caer ¿Produjo algún sonido? La respuesta es no.

*Acciones*

La función *actuar*/2 se define entonces como el mapeo entre conjuntos de percepciones *Per* y el conjunto de acciones posibles *Acc* del agente:

$$actuar : Ag \times Per \rightarrow Acc$$

*Agente*

Si el agente es conocido, o estamos en un caso donde solo hay un agente en el sistema, la variable *Ag* puede obviarse en las funciones *percibir* y *actuar*. De forma que ahora podemos definir un **agente** como la tupla

5 Todas estas etiquetas para los mensajes de correo electrónico se encuentran especificadas por el el estándar de formato para mensajes de texto en ARPA Internet (RFC# 822).

$Ag = \langle percibir, actuar \rangle$.

Usando estas definiciones, es posible expresar interesantes propiedades sobre la percepción de los agentes. Supongamos dos diferentes estados del mismo ambiente, $e_1 \in E$ y $e_2 \in E$, tal que $e_1 \neq e_2$ pero $percibir(e_1) = percibir(e_2)$. Entonces tenemos dos estados diferentes del ambiente que mapean a la misma percepción, y desde el punto de vista del agente $e_1$ y $e_2$ son **indistinguibles**. Lo anterior se ilustra ene el Ejemplo 1.8.

**Ejemplo 1.8.** *Observe que la definición de Per para el agente estilo* `xbiff`*, oculta al agente la identidad del usuario! Todo lo que el agente "cree" es que hay un buzón con o sin mensajes nuevos. Si son mis mensajes o los mensajes del usuario que entró más tarde al sistema, resulta indistinguible para nuestro agente.*

Dados dos estados del ambiente $e, e' \in E$, $percibir(e) = percibir(e')$ será abreviado como $e \sim e'$. Observen que "$\sim$" es una relación de equivalencia sobre $E$. El ambiente es accesible para el agente, si y sólo si $|E| = | \sim |$ y entonces se dice que el agente es **omnisciente**. Si $| \sim | = 1$ entonces se dice que el agente no tiene capacidad de percepción, es decir, el ambiente es percibido por el agente como si tuviera un estado único, porque todos los estados de éste son idénticos desde la perspectiva del agente.

*Omnisciencia*

### 1.4.1 Agentes reactivos

Consideren ahora nuestro primer programa de agente concreto: Un **agente reactivo** [26], o reflex [161], selecciona sus acciones con base en su percepción actual del ambiente, ignorando el resto de su historia perceptiva. La Figura 1.6 ilustra la interacción de estos agentes con su ambiente.

*Agente reactivo*

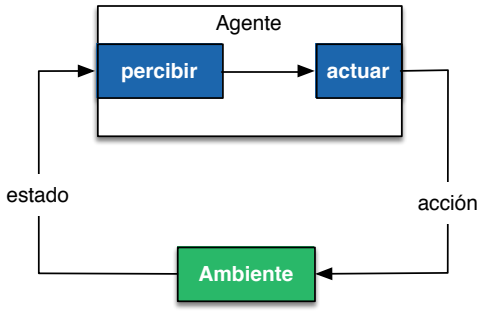


**Figura 1.6: Abstracción de un agente reactivo.**

El Ejemplo 1.9 nos muestra la formulación del programa `xbiff` como si fuese un agente reactivo.

**Ejemplo 1.9.** *Los estados posibles del ambiente, las acciones y la función de selección de acción para* `xbiff`*:*

$$percibir \leftarrow check() \in \{true, false\}$$

$$actuar = \begin{cases} set() & Si\ percibir = true \\ unset & En\ cualquier\ otro\ caso \end{cases}$$

La manera más sencilla de implementar este tipo de programa de agente es mediante reglas **condición-acción**. El agente `xbiff` puede escribirse usando dos reglas:

1. If $percibir = true$ then $set()$
2. If $true$ then $unset()$.

El Algoritmo 1.3 muestra la forma general de un programa de agente reactivo. Aunque hay otras maneras de implementar agentes reactivos, todas comparten una limitación formal: producen un comportamiento racional, sólo si la decisión correcta puede obtenerse a partir de la percepción actual del agente. Esto es, solo se puede demostrar que su comportamiento **correcto**, si y sólo si, su medio ambiente es accesible o **efectivamente accesible**.

*Limitaciones*

**Algoritmo 1.3** Programa de agente reactivo o reflex

```
1: function AGENTE-REACTIVO(e)
2:     estado ← percibir(e)
3:     regla ← selecciónAcción(estado, reglas)      ▷ reglas condición-acción.
4:     acción ← cons(regla)                         ▷ cons, el consecuente de la regla.
5:     return acción
6: end function
```

### 1.4.2 Agentes con estado

La forma más eficiente de enfrentar un ambiente **inaccesible** es llevando un registro de lo percibido, de forma que el agente tenga acceso mediante este registro, a lo que en cierto momento ya no puede percibir. Esto es, el agente debe mantener una forma de **estado interno** que depende de su historia de percepciones y por lo tanto refleja al menos algunos de los aspectos del ambiente no observados en la percepción actual del agente. Sea $I$ el conjunto de estados internos posibles de un agente. Redefinimos la función **actuar** para mapear estados internos a acciones posibles:

*Estado interno*

*Acción*

$$actuar : I \rightarrow Ac$$

Una nueva función $siguiente/2$, mapea estados internos y percepciones a estados internos. Se usa para actualizar el estado interno del agente:

$$siguiente : I \times Per \rightarrow I$$

Un agente inicia con algún estado interno $i_0 \in I$, observa entonces el estado del ambiente $e$ generando así un percepto $p \leftarrow percibir(e)$. El estado interno del agente se actualiza, $i_1 \leftarrow siguiente(i_0, p)$. La acción seleccionada por el agente es entonces ejecutada y el agente repite este ciclo.

El Algoritmo 1.4 nos muestra el programa de un **agente con estado** (Figura 1.7). La diferencia con el agente puramente reactivo, es que la función $siguiente/2$ modifica el estado interno del agente interpretando su entrada perceptiva, usando el conocimiento que el agente tenía sobre el ambiente.

*Agente con estado*

**Algoritmo 1.4** Programa de agente con estado

```
1: function AGENTE-CON-ESTADO(e)
2:     p ← percibir(e)
3:     estado ← siguiente(estado, p)
4:     regla ← selecciónAcción(estado, reglas)
5:     acción ← cons(regla)
6:     return acción
7: end function
```

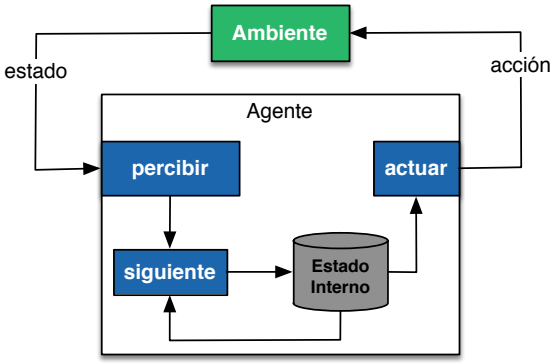


**Figura 1.7: Agente con un estado interno.**

### 1.4.3 Agentes lógicos

La aproximación tradicional de la IA a los sistemas inteligentes, nos dice que el comportamiento racional puede obtenerse a partir de una **representación simbólica** del ambiente y el comportamiento deseado. El agente manipulará sintácticamente esta representación para actuar. Llevada al extremo, esta aproximación nos lleva a formular el estado de un agente como un conjunto fórmulas lógicas; y la selección de acción como demostración de teoremas o deducción lógica.

*Representación simbólica*

La idea de un agente como un demostrador de teoremas es seductora. Supongamos que disponemos de una teoría que establece cómo debe comportarse un agente racional. Tal teoría debe explicar cómo un agente genera sus metas, cómo explota su comportamiento orientado por las metas y su reactividad, etc. Lo importante es que esta teoría $\phi$ puede considerarse como una **especificación** del comportamiento racional de un agente.

*Especificación*

La manera tradicional de implementar un agente que cumpla con la especificación de la teoría $\phi$ es **refinar** paso a paso la teoría, concretando sus conceptos en estructuras de datos y procedimientos computacionales ejecutables. Sin embargo, en el enfoque basado en demostración de teoremas, tal refinamiento no se lleva a cabo. La teoría $\phi$ se concibe como una especificación que el agente puede ejecutar directamente.

*Refinamiento*

Sea $L$ el conjunto de fórmulas bien formadas en una lógica. El conjunto de **bases de conocimiento** en $L$ se define como $D = \wp(L)$, es decir, el conjunto de conjuntos de fbf en $L$. Los elementos de $D$ se denotan $\Delta, \Delta_1, \ldots$ El estado interno del agente es siempre un miembro de $D$. El proceso de decisión del agente se especifica mediante un conjunto de **reglas de inferencia** $\rho$.

*Bases de Conocimiento*

*Reglas de Inferencia*

Escribimos $\Delta \vdash_\rho \psi$ si la fbf $\psi$ puede ser derivada a partir de $\Delta$, siguiendo las reglas en $\rho$. Definimos la función *siguiente*/2 como:

$$siguiente : D \times Per \rightarrow D$$

El Algoritmo 1.5 muestra una función de selección de acción para un agente lógico. La primera parte de esta función (líneas 2 a 6) iteran sobre las acciones del agente para encontrar una acción $a$, tal que $actuar(a)$ se siga logicamente de la base de conocimientos $\Delta$, que conforma el estado interno del agente; usando las reglas de inferencia en $\rho$. Si tal acción no se encuentra, se busca una acción $a$ (líneas 7 a 11) que al menos sea **consistente** con el estado interno del agente. Si no se encuentra ninguna acción, se regresa *null* para indicar el fallo.

*Consistencia*

**Algoritmo 1.5** Programa de agente lógico

```
1:  function Selección-Acción(Δ : D, Acc)        ▷ Δ base de conocimiento.
2:      for all a ∈ Acc do                        ▷ Acc acciones.
3:          if Δ ⊢_ρ actuar(a) then               ▷ ρ reglas de inferencia.
4:              return a
5:          end if
6:      end for
7:      for all  a ∈ Acc do
8:          if Δ ⊬_ρ ¬actuar(a) then
9:              return a
10:         end if
11:     end for
12:     return null
13: end function
```

## 1.5 METAS Y UTILIDADES

Implicita en la arquitectura de los agentes que hemos presentado, está la intención de hacer lo correcto. La cuestión, tal y como lo plantean Covrigaru y Lindsay [44] es ¿Cómo decirle a un agente lo que debe hacer, si decirle cómo? (Ver la discusión sobre autonomía y flexibilidad en la Sección 1.2). Las metas y el concepto de utilidad han sido utilizados con este propósito, bajo tal restricción.

### 1.5.1 Agentes basados en metas

*Metas*

En la tradición de la IA, las **metas** describen situaciones que son deseables para un agente, y son definidas como cuerpos de conocimiento acerca de los estados del medio ambiente que el agente desea ver realizados [134]. Esta concepción de las metas está relacionada con el concepto de **espacio de estados de un problema** compuesto por un estado inicial del ambiente, $e_0 \in E$; por un conjunto de operadores o acciones que el agente puede ejecutar para cambiar de estado; un espacio de estados alcanzables a partir del estado inicial $e_0$ al aplicar cualquier secuencia de operadores; y un test para verificar si el estado alcanzado es una meta, por ej., pertenece a un subconjunto de

*Espacio de Estados*

$E$ que incluye los estados meta o cumple con determinadas propiedades especificadas en algún lenguaje lógico.

Implícita en la arquitectura del agente, está su "intención" de ejecutar las acciones que el "cree" le garantizan satisfacer cualquiera de sus metas. Esto se conoce en filosofía como **silogismo práctico** y constituye la definición de **racionalidad** usada en IA [124, 134].

*Silogismo práctico*

*Racionalidad*

Es posible especificar las metas que el agente debe satisfacer usando una especificación basada en predicados. Un predicado en este contexto es definido como la función:

$$\Psi : R \to \{0, 1\}$$

esto es, una función de las corridas del agente a 1 (verdadero) o 0 (falso). Se considerará que una corrida $r \in R$ satisface la especificación si y sólo si $\Psi(r) = 1$. Un **ambiente de tareas** se define entonces como el par $\langle Env, \Psi \rangle$. Dado un ambiente de tareas:

*Ambiente de tareas*

$$R_\Psi(Ag, Env) = \{r | r \in R(Ag, Env) \wedge \Psi(r)\}$$

denota el conjunto de todas las corridas del agente $Ag$ en el ambiente $Env$ que satisfacen la tarea especificada por $\Psi$. Podemos expresar que un agente $Ag$ tiene **éxito** en el ambiente de tareas $\langle Env, \Psi \rangle$ de dos maneras diferentes:

*Éxito*

1. $\forall r \in R(Ag, Env)$ tenemos que $\Psi(r)$, lo que puede verse como una especificación **pesimista** de éxito, puesto que el agente tiene éxito únicamente si todas sus corridas satisfacen $\Psi$;
2. $\exists r \in R(Ag, Env)$ tal que $\Psi(r)$, lo cual es una versión **optimista** de la definición de éxito, puesto que especifica que el agente tiene éxito si al menos una de sus corridas satisface $\Psi$.

Observen que $\Psi$ opera sobre corridas del agente, de forma que si queremos especificar un test de meta, como se define en el espacio de estados de un problema, algunos cambios son necesarios. Recuerde que el ambiente se define como $Env = \langle E, e_0, \tau \rangle$ donde $E$ es el conjunto finito de estados discretos del ambiente, y que el agente es definido como $Ag = R^E \to Ac$, un mapeo de corridas que terminan en un estado del ambiente a acciones. El conjunto $E$ corresponde al espacio de estados en la definición del espacio de estados del problema. Es posible identificar un subconjunto $G \subset E$ de estados finales, tal que $\Psi(r)$ es verdadera solo si uno o más $e \in G$ ocurren en $r \in R$. $\Psi$ especifica la prueba de meta y el conjunto de acciones $Ac$ define los operadores.

Una vez que un agente alcanza un estado que satisface el test de meta, se dice que ésta ha sido satisfecha. Las metas definidas de esta forma, constituyen un subconjunto de las metas que un agente puede enfrentar. Como ya lo comentamos, Covrigaru y Lindsay [44] concluyen que un sistema se percibe más fácilmente como autónomo si tiene un conjunto de metas de alto nivel (aquellas que no son submetas de otra meta) y algunas de ellas son homeostáticas (la satisfacción de la meta no termina al alcanzar el estado que la define, el sistema verifica continuamente si ese estado ha sido abandonado, para buscar satisfacerlo nuevamente).

**Ejemplo 1.10.** *La meta de* `xbiff`*, comunicar que hay mensajes en el buzón, es homeostática. Cada vez que el estado del medio ambiente le indica al agente la presencia de mensajes, este ejecutará la acción de cambiar su icono. Sin embargo,* `xbiff` *tiene una única meta.*

### 1.5.2 Agentes basados en funciones de utilidad

Otra forma de comunicarle al agente que hacer, sin decirle como hacerlo, es definiendo sus metas indirectamente, por medio de alguna medida de **utilidad**, es decir, un valor numérico que denota la bondad, o lo deseable que resulta un estado del ambiente. Implícitamente, un agente tiene la intención de alcanzar aquellos estados que **maximizan** su utilidad a largo término. Por lo tanto, la especificación de una tarea en este enfoque corresponde simplemente a una función $u : E \to \Re$ la cual asocia valores reales a cada estado del ambiente. Para poder especificar una visión a largo plazo del desempeño del agente, algunas veces las utilidades no se asignan a los estados del ambiente, sino a las corridas del agente.

*Utilidad*

**Ejemplo 1.11.** *Pensemos en un agente de correo electrónico más complejo que el programa* `xbiff`*, uno que filtre el* spam *en el correo recibido. La utilidad para una corrida r de este agente, puede definirse como:*

$$u(r) = \frac{SpamFiltrado(r)}{SpamRecibido(r)} \tag{1.8}$$

*Esto da una medida normalizada del desempeño del agente en el rango de 0 (el agente no logró filtrar ningún mensaje spam) a 1 (el agente filtró todo el spam recibido) durante la corrida r.*

Asumiendo que la función de utilidad $u$ tiene algún límite superior en los valores que asigna, por ej., existe un $k \in \Re$ tal que $\forall r \in R.u(r) \leq k$, entonces es posible hablar de **agentes óptimos**, agentes que maximizan la utilidad esperada. Definamos $P(r|Ag, Env)$ como la probabilidad de que una corrida $r$ ocurra cuando el agente $Ag$ está situado en el ambiente $Env$, y sea $R$ el conjunto de todas las posibles corridas de $Ag$ en $Env$. Es evidente que:

*Agentes óptimos*

$$\sum_{r \in R(Ag,Env)} P(r|Ag, Env) = 1 \tag{1.9}$$

Entonces el agente óptimo $Ag_{opt}$ entre el conjunto de agentes posibles $Ags$ en el ambiente $Env$ está definido como:

$$Ag_{opt} = \arg \max_{Ag \in Ags} \sum_{r \in R(Ag,Env)} u(r)P(r|Ag, Env) \tag{1.10}$$

Esta ecuación es idéntica a la noción de maximizar la utilidad esperada, tal y como lo expresan **VonNeumann1947** en su libro sobre Teoría de la Decisión. La utilidad esperada se toma conforme a las creencias del agente, por lo que la racionalidad perfecta no requiere omnisciencia. De cualquier forma, como señala Simon [173], los agentes enfrentan limitaciones temporales y tienen capacidades limitadas de deliberación, por lo que propone el estudio de una **racionalidad acotada**.

*Racionalidad acotada*

Russell y Subramanian [162], introducen el concepto de **agente óptimo acotado**, donde el conjunto de agentes posibles $Ags$ es remplazado por un subconjunto de él, identificado como $Ags_m$, que representa el conjunto de agentes que pueden ser implementados en una máquina $m$ con características específicas, por ej., memoria, almacenamiento, velocidad de procesador, etc., de forma que las restricciones de optimización se imponen en el programa del agente y no en las funciones del agente, sus deliberaciones o su comportamiento.

*Agente óptimo acotado*

Esta concepción de agente racional nos dice las propiedades del agente deseable $Ag_{opt}$, pero no nos dice cómo implementar tal agente. En la práctica normalmente resulta difícil derivar la función de utilidad $u$ apropiada para un agente. Para una descripción más detallada del uso de funciones de utilidad en la programación de los agentes, véase el Capítulo 3 de este texto.

## 1.6 LECTURAS Y EJERCICIOS SUGERIDOS

La formalización del concepto de agente está basada en el texto de Wooldridge [185], aunque una fuente más accesible a este material se encuentra en su libro introductorio a los Sistemas Multiagente [186]. La clasificación de los agentes y sus programas está basada en el texto de Russell y Norvig [161]. El artículo de Covrigaru y Lindsay [44] ofrece una discusión profunda sobre la autonomía en los agentes artificiales. El antecedente directo al uso de funciones de utilidad en agentes se encuentra en el trabajo de **VonNeumann1947**, donde queda de manifiesto una de las principales ventajas de este enfoque: su formalización. El concepto de racionalidad acotada fue introducido por Simon [173] siendo uno de sus temas preferidos. Una discusión más reciente sobre el tema de la racionalidad acotada, puede encontrarse el artículo de Russell y Subramanian [162].

Actualmente contamos con referencias bibliográficas coherentes y completas sobre el tema de la programación de agentes y los SMA. Se sugiere una lectura rápida de los capítulos uno y dos del libro de Russell y Norvig [161] para una panorámica del tema de los agentes racionales con respecto a la IA. Huns y Singh [105] ofrecen una colección de artículos fundamentales en el área, algunos de los cuales serán revisados a lo largo del curso. Weiß [183] estructuró el primer texto completo y coherente sobre el estudio de los SMA. Wooldridge [186] nos ofrece un texto de introducción básico a los SMA, imprescindible de leer a lo largo del curso, por su cobertura y complejidad moderada. El texto de Shoham [169] se situa un marco de referencia económico, más próximo a la Teoría de Juegos, pero con un componente algorítmico importante. Un excelente complemento para este texto.

### Ejercicios sugeridos

**Ejercicio 1.1.** *En el Concurso Nacional de Robots Limpiadores 2005 (Xalapa, Ver., México), los agentes participantes debían limpiar el piso de una sala, dividida virtualmente en celdas de diferente dificultad de acceso. La Figura 1.8 ilustra este escenario –Las celdas bajo el mobiliario son de más difícil acceso que las celdas sin mobiliario*

*encima. La basura se reparte aleatoriamente al inicio de la prueba. Defina una medida de desempeño normalizada, que favorezca a los agentes que limpian más celdas de difícil acceso. Asumiendo que hay un tiempo máximo para llevar a cabo la tarea, incluya en la medida de desempeño una forma de premiar a los agentes más rápidos (o castigar a los más lentos).*

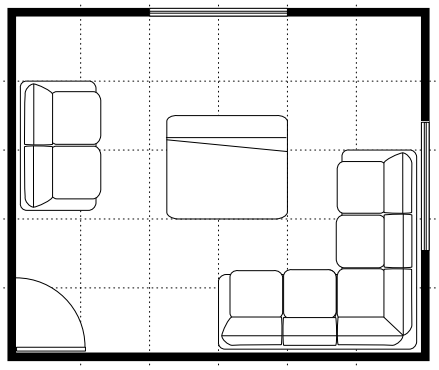

**Figura 1.8**: Una habitación a ser limpiada por robots, las celdas tienen diferentes dificultades de acceso.

**Ejercicio 1.2.** *Elabore una descripción PEAS del entorno de trabajo que define el Concurso Nacional de Robots Limpiadores 2005, descrito en el ejercicio anterior.*

**Ejercicio 1.3.** *¿Qué artefactos es posible definir para encapsular los actuadores y sensores del agente en este entorno de trabajo?*

**Ejercicio 1.4.** *¿Qué propiedades tiene este entorno de trabajo?*

**Ejercicio 1.5.** *Defina un sistema agente para este entorno de trabajo y ejemplifique una corrida del sistema.*

**Ejercicio 1.6.** *Defina un agente reactivo para este entorno de trabajo, incluyendo su formulación en términos de un conjunto de reglas condición-acción.*

**Ejercicio 1.7.** *¿Que preferiría usar para guiar el comportamiento de su robot, metas o funciones de utilidad? Justifique su respuesta.*