

2

AGENTES RACIONALES

Dado nuestro señalado interés en la construcción de agentes racionales, en este capítulo delimitaremos el concepto de agente, de manera que podamos utilizarlo para estructurar el resto de este texto. De lo anterior, se sigue que este curso es acerca de la representación de conocimiento y el razonamiento en función del estudio y síntesis de agentes inteligentes. La Sección 2.1 presenta una definición de agente racional con elementos computacionales. La Sección 2.2 introduce una posible caracterización del comportamiento flexible y autónomo que los agentes racionales deben observar, con especial énfasis en la autonomía. La Sección 2.3 introduce el concepto de medio ambiente y su relación con los agentes racionales en él situados. La Sección 2.4 presenta una arquitectura abstracta de agente y su adaptación a diferentes tipos de agentes reportados en la literatura. Finalmente, la Sección 2.5 propone una serie de lecturas suplementarias y ejercicios. Es de esperar que la aproximación informal a estos conceptos facilite este primer encuentro, así como su posterior formalización y estudio a fondo.

Organización

2.1 AGENTES COMPUTACIONALES

Históricamente, fuera de la IA, el término agente ha sido usado con dos acepciones. Primero, a partir de Aristóteles [4] 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 [72]. Como veremos, ambas nociones de agencia han influenciado considerablemente su forma computacional.

Agencia y filosofía

Agencia y derecho

En el contexto de la computación, Wooldridge [111], 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 2.1). 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 especializados, no especializados, programadores y diseñadores.

Agencia y computación

Pero, **¿Qué es un agente racional artificial?** Como veremos, la respuesta no es sencilla, ni unívoca. Franklin y Graesser [42] 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 iniciar esta sección. Una definición consensual de agente [112, 94] en este contexto, puede ser:

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.

Definición consensual

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 2.2, las **acciones** del agente modifican el medio ambiente; mientras que los cambios en el medio ambiente son **percibidos** por el agente. Eventualmente, las ac-

Ventajas de la definición

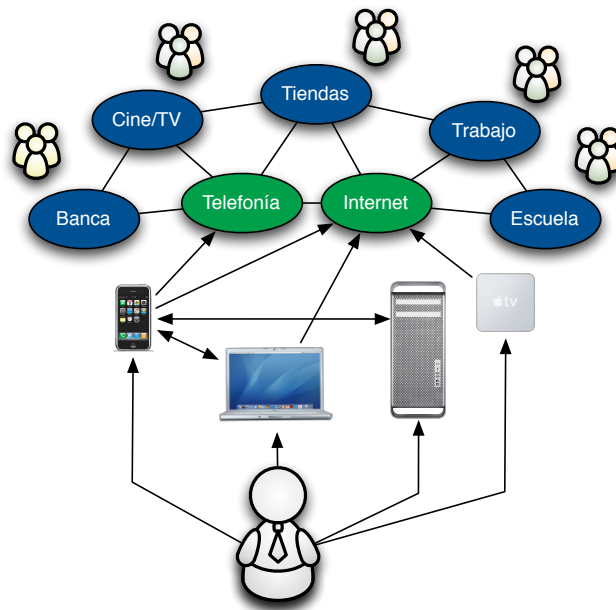


Figura 2.1: Entornos computacionales ubicuos, distribuidos, e interconectados, de amplia disponibilidad, demandan la delegación de tareas en agentes inteligentes. Adaptado de Wooldridge [111].

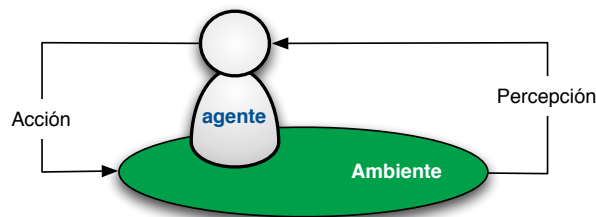


Figura 2.2: Abstracción de un agente a partir de su interacción persistente con el medio ambiente, para la consecución de sus metas.

ciones del agente lo llevan a la consecución de sus objetivos. Russell y Subramanian [95] encuentran que tal abstracción presenta tres ventajas:

1. Nos permite observar las facultades cognitivas de los agentes al servicio de encontrar cómo hacer lo correcto.
2. Permite considerar diferentes tipos de agente, incluyendo aquellos que no se supone tengan tales facultades cognitivas.
3. Permite considerar diferentes especificaciones sobre los subsistemas que componen los agentes.

Sin embargo, por si misma la definición consensual de agente resulta demasiado **general**. Consideren el Ejemplo 2.1 de un programa UNIX que difícilmente puede ser concebido como un agente, pero se ajusta a la definición consensual del término.

Problemas de la definición

Ejemplo 2.1. La Figura 2.3 muestra a *xbiff*, una pequeña interfaz gráfica para *biff*, un sistema de notificaciones de correo electrónico para UNIX. Cuando un mensaje llega al buzón del usuario, *biff* emite una alerta. Por su parte *xbiff* usa la alerta para desplegar el ícono que se muestra en la ventana superior izquierda.

Esta pequeña aplicación se las arregla para identificar a su usuario, encontrar su buzón de correo electrónico, buscar mensajes nuevos y comunicar al usuario

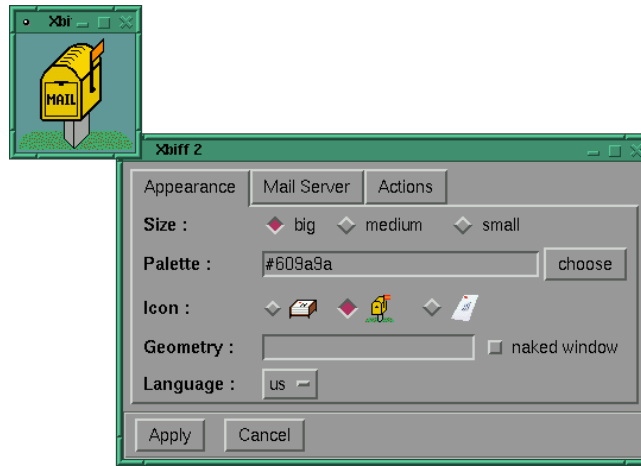


Figura 2.3: ¿Es la aplicación xbiiff un agente?

la presencia de éstos. Es decir, xbiiff es un sistema de cómputo que actúa persistentemente en su entorno 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 xbiiff no sea muy listo, pero **hace lo correcto** y su funcionamiento es claro, independientemente de la especificación e implementación subyacentes. ¿Es xbiiff un agente?

Primero, observen que hacer lo correcto nos provee de una aproximación indirecta a la racionalidad, que además resulta funcional. Se dice que un agente es **racional** si hace lo correcto. Esto reduce el problema de establecer cuando un agente es racional, al de definir cómo y cuando evaluar el éxito del agente. El término **medida de desempeño** se refiere al criterio usado para determinar el éxito 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 usamos ese estándar para medir el desempeño del agente. Bajo ninguna circunstancia, el agente puede manipular ¹ tal estándar. El Ejemplo 2.2 nos muestra una medida de desempeño típica de los concursos de robótica, que cumple con las características mencionadas.

Desempeño

Ejemplo 2.2. En el 3^{er} Torneo Nacional de Robots Limpiadores 2006 (Xalapa, Ver., México), el desempeño del agente fue establecido por un estándar externo al agente, especificado mediante la fórmula de evaluación diseñada por los organizadores:

$$PT = \sum_i (CL_i \times DA_i) \times (TMax / TOper)$$

donde: PT es el puntaje total. El segundo término evalúa la persistencia del agente: TMax es el tiempo máximo de la prueba y TOper es el tiempo que el robot se mantuvo en operación. Si el robot trabaja durante toda la prueba no es penalizado, si no la puede terminar, si lo es. Si el robot no arranca, este factor es cero, y por tanto el puntaje total del agente es cero. El primer término evalúa la capacidad limpiadora del agente. Para ello el piso a limpiar, rectangular, se divide conceptualmente en celdas. CL_i es la i-ésima celda limpiada con éxito, o si se prefiere toma valor 1 si la celda se limpio, y 0 en caso contrario. DA_i es el valor de dificultad de acceso a la i-ésima celda limpiada. El robot que limpia más celdas de mayor dificultad y funciona más tiempo es el ganador. La función está acotada entre cero y un máximo, por lo que existe un desempeño óptimo.

Segundo, observen que la racionalidad de un agente se define en relación con el **éxito esperado**, dadas sus circunstancias. Esto es, no podemos exigir a un agente

¹ Un agente que manipula la medida de desempeño, suele llegar a razonamientos del tipo –Al fin y al cabo, ¡Quién diablos quiere una maestría!

que tome en cuenta lo que no puede percibir, o haga lo que sus efectores no pueden hacer. Es decir, además de la **medida de desempeño**, otros factores presentes al evaluar la racionalidad de un agente incluyen:

Evaluación de la racionalidad

- 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.
- Las **competencias** del agente, entendidas como el repertorio de acciones que éste puede llevar a cabo en su medio ambiente.

Con los elementos anteriores, podemos definir un **agente racional ideal**, como 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. 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 el diseño de un agente ideal. El ejemplo 2.3 muestra la función ideal de `xbiff`.

Agente ideal

Función de comportamiento

Ejemplo 2.3. *En el caso de `xbiff` la función ideal es muy sencilla. Si `biff` avisa que llegó un mensaje nuevo, desplegar el icono de mensaje nuevo; en cualquier otro caso, desplegar el icono de mensajes leídos.*

Y es aquí donde los problemas de `xbiff` para ser calificado como un agente comienzan. ¿Porqué si esta aplicación cumple con la definición consensual de agente y tiene una función ideal, difícilmente se le ve como un agente? Primero, es muy difícil que un programa con una función de comportamiento ideal tan simple, sea concebido como un agente. De hecho, lo normal es que no sea factible enumerar todas las entradas de la función de comportamiento ideal de un agente, es decir, el número de posibles secuencias de percepción a considerar, suele ser muy grande. De hecho, un problema central en la programación de agentes es la búsqueda de mecanismos que aproximen las funciones ideales que no pueden definirse por extensión. A esto se le conoce como el problema de **selección de acción**. Precisamente, como dice Maes [67], el problema de ¿Cómo hacer lo correcto? Y es que, segundo, el repertorio de acciones de un agente suele mayor que el de `xbiff`, por lo que una selección es necesaria. En particular, el comportamiento de `xbiff` difícilmente puede verse como flexible y autónomo.

Selección de acción

2.2 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 [41] argumenta que para ser percibido como inteligente, un agente debe exhibir cierto tipo de comportamiento, denominado más tarde por Wooldridge y Jennings [112], **comportamiento flexible y autónomo**. Este tipo de 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; así como explotar las oportunidades que se presentan para conseguir sus metas.
- **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.

Una caracterización más detallada de la autonomía es presentada por Covrigaru y Lindsay [28]. 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 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 la 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. Ninguna de sus funciones, acciones o decisiones, está totalmente gobernada por un agente externo.
7. Una vez en operación, el agente no necesita ser programado nuevamente por un agente externo.

El sexto punto requiere una defensa rápida, puesto que la dimensión social de los agentes no será abordada aquí, sino en el curso de Sistemas Multi Agentes ². En este punto se puede adivinar una tensión entre autonomía y sociabilidad ¿Para qué necesita un agente autónomo socializar con otros 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 [90] 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 solo ambos conceptos no están reñidos, sino que no puede haber autonomía sin sociabilidad. En el contexto de la IA, es Newell [75] quien ofrece argumentos similares.

Autonomía y sociabilidad

Covrigaru y Lindsay argumentan además, que ser autónomo depende no solo 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.

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 [112] llaman la **noción débil** de agente,

Agencia débil y fuerte

² <https://www.uv.mx/personal/aguerra/sma>

como contrapunto 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.

2.3 MEDIO AMBIENTE

Por medio ambiente, entendemos el espacio donde un agente, o un grupo de ellos, se encuentra **situado**. Brooks [19] 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 [38], 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.

En este texto asumimos una tercera vía propuesta por Ricci [79, 92], 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 con iniciativa, llamados **artefactos**. Como es de esperar, los agentes pueden usar estos artefactos para sensar y actuar en el medio ambiente. Agentes y artefactos pueden organizarse en **espacios de trabajo**, para reflejar la topología del ambiente y proveer una noción de localidad ó sitio. Esto constituye un meta-modelo que puede aproximar ambientes tanto reales, como virtuales. La Figura 2.4 ilustra esto, junto con un componente social propio de los Sistemas Multi Agente: Las **organizaciones**.

Real o virtual

Artefactos

Espacio de trabajo

Organización

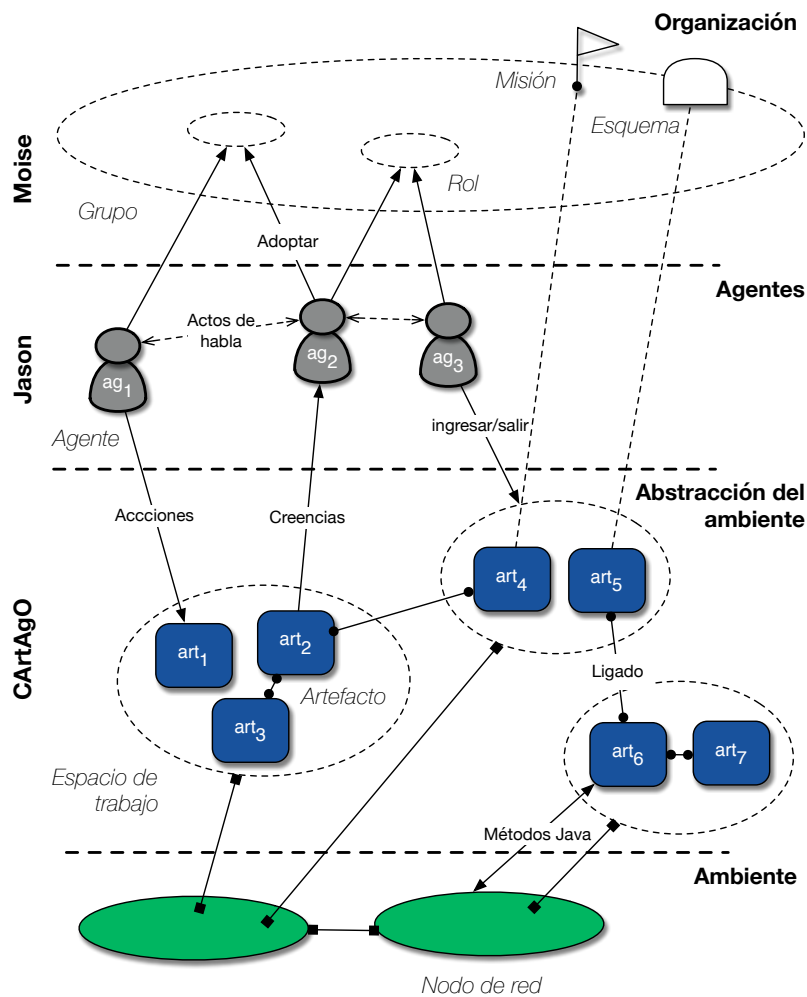


Figura 2.4: Vista general del esquema de Agentes y Artefactos, extendido con organizaciones: JaCaMo.

El concepto de espacio de trabajo se asemeja al de **entorno de trabajo**, propuesto por Russell y Norvig [94] 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 2.4. *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. 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; de cara al agente los artefactos actualizan su representación del ambiente y proveen una serie de operaciones que el agente puede ejecutar al usar sus artefactos.

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 2.1: Ejemplos de agentes y su descripción PEAS según Russell y Norvig [94].

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. El Cuadro 2.1 muestra otros ejemplos de descripciones PEAS. Ahora bien, los elementos que componen este tipo de descripciones nos permiten concebir 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 se sitúan 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**.

*Observable**Eficazmente observable**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 interactúa 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 consideradas 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 **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 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 está 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

Ambiente	Observ.	Determ.	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 2.2: Ejemplos de ambientes estudiados en IA y sus propiedades según Russell y Norvig [94].

Esta categorización sugiere que es posible encontrar diferentes clases de entornos de trabajo. Russell y Norvig [94] presentan algunos ejemplos de ambientes bien

estudiados en Inteligencia Artificial y sus propiedades (ver Cuadro 2.2). 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.

2.4 ARQUITECTURAS DE AGENTE

Es posible formalizar la definición abstracta de agente que hemos presentado (pág. 11), en la forma de una **arquitectura abstracta** de agentes [111]. Primero asumiremos que el ambiente puede caracterizarse por medio de un conjunto finito de **estados** discretos posibles, definido como:

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

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:

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

Ahora, supongamos que el ambiente se encuentra en algún estado inicial e_0 y que el agente comienza su actividad eligiendo una acción α_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:

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

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.

Para modelar el efecto de una acción en el ambiente, usamos una **función de transición** entre estados ³ que mapea una corrida terminando en acción a un conjunto de posibles estados del ambiente:

$$\tau : R^{Ac} \rightarrow \wp(E)$$

En esta forma, el ambiente se asume como sensible a su historia, esto es, su próximo estado no está determinado exclusivamente por la acción ejecutada por el agente en el estado actual. Las acciones ejecutadas por el agente en el pasado, también afectan esta transición. Esta definición introduce un grado de indeterminismo en el ambiente, y por lo tanto, incertidumbre sobre el resultado de una acción ejecutada en cierto estado. Como vimos, considerar que el ambiente es no determinista, es importante para entender las limitaciones de los agentes.

Si $\tau(r) = \emptyset$ para todo $r \in R^{Ac}$, entonces no hay posibles estados sucesores para r y se dice que el sistema ha **terminado** su corrida. Asumimos que todas las corridas terminan eventualmente.

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 τ es la función de transición de estados.

Arquitectura
abstracta
Estados

Acciones

Corrida

Corridas posibles

Función de transición

Condición de paro

Ambiente

Los **agentes** se modelan como funciones que mapean corridas que terminan en un estado del ambiente, a acciones: Agente

$$Ag : R^E \rightarrow Ac$$

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 $(e_0, \alpha_0, e_1, \alpha_1, e_2, \dots)$ representa una corrida del agente Ag en el ambiente $Env = \langle E, e_0, \tau \rangle$ si y solo si e_0 es el estado inicial del ambiente Env ; $\alpha_0 = Ag(e_0)$; y para $u > 0$:

Sistema agente

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

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, la forma más sencilla de implementar un programa de agente es a través de su **mapeo ideal**. El Algoritmo 2.1 muestra la función que implementa un agente de este tipo. Programa de agente
Agente mapeo ideal

Algoritmo 2.1 Programa de agente basado en mapeo ideal

```

1: function AGENTE-MAPEO-IDEAL( $p$ )           ▷  $p$  es una percepción del ambiente.
2:    $push(p, percepts)$                        ▷  $percepts$  es una cola de percepciones.
3:    $acción \leftarrow tabla(percepts)$         ▷  $tabla : percepts \rightarrow acciones$  predefinida.
4:   return acción
5: end function

```

Aunque ésta es también la forma más sencilla de fracasar en la construcción de un agente inteligente. Aún para agentes en ambientes controlados como el 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.

Algoritmo 2.2 Programa de ambiente

```

1: procedure AMBIENTE( $e, \tau, ags, fin$ )         ▷  $e$  es el estado inicial del ambiente.
2:   repeat
3:     for all  $ag \in ags$  do                 ▷  $ags$  es un conjunto de agentes.
4:        $p(ag) \leftarrow percibir(ag, e)$ 
5:     end for
6:     for all  $ag \in ags$  do
7:        $acción(ag) \leftarrow ag(p(ag))$ 
8:     end for
9:      $e \leftarrow \tau(\bigcup_{ag \in ags} acción(ag))$    ▷  $\tau$  es una función de transición del ambiente.
10:  until  $fin(e)$                              ▷  $fin$  es un predicado de fin de corrida.
11: end procedure

```

Podemos también implementar un programa básico de ambiente que ilustre la relación entre éste y los agentes situados en él. El Algoritmo 2.2 muestra el programa del **ambiente**. La historicidad del ambiente queda oculta en las percepciones del agente. Programa del ambiente

La función $percibir/2$ mapea estados del ambiente a alguna entrada perceptiva. Los agentes con formas robóticas, pueden implementar $percibir/2$ usando hardware, por ejemplo, cámaras de vídeo, sensores infrarrojos, sonares, etc. Los agentes software pueden implementar esta función usando comandos del sistema operativo. El Ejemplo 2.5 muestra diferentes opciones de percepción para el agente `xbiff`.

³ Véase Fagin y col. [39], p. 154, pero aquí τ es aplicada a un solo agente.

Ejemplo 2.5. 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 ⁴.

Ahora bien, sea *Per* un conjunto no vacío de percepciones, la función de *percibir*/2 se define como el mapeo del conjunto de estados del ambiente *E* al conjunto de percepciones posibles *Per*:

$$\text{percibir} : E \rightarrow \text{Per}$$

El Ejemplo 2.6 muestra la diferencia entre percepción y estados posibles del ambiente, en el caso del agente *xbiff*.

Ejemplo 2.6. Para el agente de estilo *xbiff*, podemos definir es conjunto de percepciones posibles como $\text{Per} = \{\text{mensajesNuevos}, \text{noMensajesNuevos}\}$, mientras que los estados posibles del ambiente en *E* son más complejos, incluyendo como mencioné, el *login* del usuario, el número de cada tipo de mensaje en el buzón, etc.

Levitin [64], 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 función *acción*/1 se define entonces como el mapeo entre conjuntos de percepciones *Per* y el conjunto de acciones posibles *Acc* del agente:

$$\text{acción} : \text{Per} \rightarrow \text{Acc}$$

Un agente puede definirse ahora como $\text{Ag} = \langle \text{percibir}, \text{acción} \rangle$. Usando esta extensión, 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 $\text{percibir}(e_1) = \text{percibir}(e_2)$. 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**.

Ejemplo 2.7. 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 $e, e' \in E$, $\text{percibir}(e) = \text{percibir}(e')$ 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 solo 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.

2.4.1 Agentes reactivos

Consideren ahora nuestro primer programa de agente concreto: Un agente reacti-

⁴ Todas estas etiquetas para los mensajes de correo electrónico se encuentran especificadas por el estándar de formato para mensajes de texto en ARPA Internet (RFC# 822).

Percepción

Acción

Agente

Omnisciencia

Agente reactivo

vo [19], o reflex [94], selecciona sus acciones con base en su percepción actual del ambiente, ignorando el resto de su historia perceptiva. La Figura 2.5 ilustra la interacción de estos agentes con su ambiente. El Ejemplo 2.8 nos muestra la formulación del programa `xbiff` como si fuese un agente reactivo.

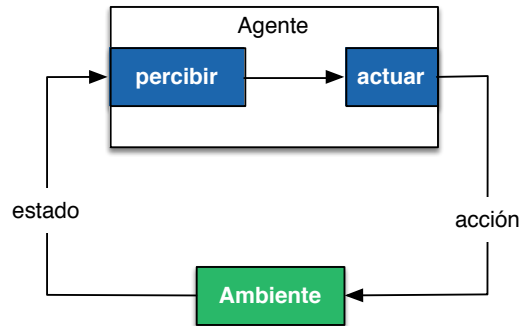


Figura 2.5: Abstracción de un agente reactivo.

Ejemplo 2.8. Los estados posibles del ambiente, las acciones y la función de selección de acción para `xbiff`:

$$E = \{\text{correoNuevo}\}$$

$$Ac = \{\text{despliegaIconoCorreoNuevo}, \text{despliegaIconoCorreoViejo}\}$$

$$Ag(e) = \begin{cases} \text{despliegaIconoCorreoNuevo} & \text{Si } e = \text{correoNuevo} \\ \text{despliegaIconoCorreoViejo} & \text{En otro caso} \end{cases}$$

La manera más sencilla de implementar este tipo de programa de agente es mediante reglas **condición-acción**. El Algoritmo 2.3 muestra la forma general de un programa de agente reactivo.

Algoritmo 2.3 Programa de agente reactivo o reflex

```

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

De manera que la formulación de `xbiff` puede transformarse, utilizando reglas condición-acción, en:

1. If $p = \text{correoNuevo}$ Then *despliegaIconoCorreoNuevo*.
2. If *true* Then *despliegaIconoCorreoViejo*.

Aunque hay otras maneras de implementar agentes reactivos, todas comparten una limitación formal: producen un comportamiento racional, solo si la decisión correcta puede obtenerse a partir de la percepción actual del agente. Esto es, su comportamiento es correcto si, y solo si, el ambiente es accesible o efectivamente accesible.

2.4.2 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 **acción** para mapear estados internos a acciones posibles:

$$\text{acción} : I \rightarrow Ac$$

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

$$\text{siguiente} : I \times Per \rightarrow I$$

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

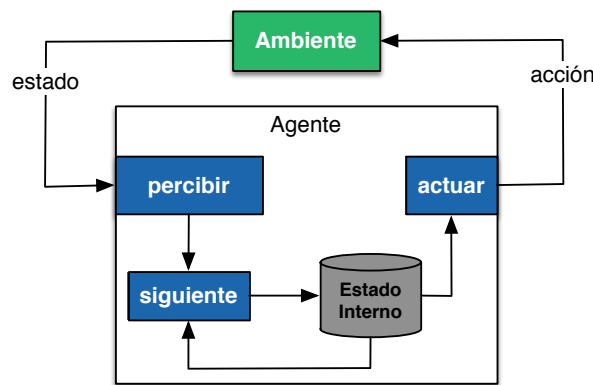


Figura 2.6: Agente con un estado interno.

La Figura 2.6 nos muestra un agente con estado interno, cuyo algoritmo se muestra en el Cuadro 2.4. 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.

Algoritmo 2.4 Programa de agente con estado

```

1: function AGENTECONESTADO( $e$ )
2:    $p \leftarrow \text{percibir}(e)$ 
3:    $\text{estado} \leftarrow \text{siguiente}(\text{estado}, p)$ 
4:    $\text{regla} \leftarrow \text{selecciónAcción}(\text{estado}, \text{reglas})$ 
5:    $\text{acción} \leftarrow \text{AcciónRegla}(\text{regla})$ 
6:   return  $\text{acción}$ 
7: end function

```

2.4.3 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 [76] del ambiente y el comportamiento deseado. El agente manipulará sintácticamente esta representación para decidir como 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.

El concepto de agente como un demostrador de teoremas es extremadamente seductor. 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 ϕ puede considerarse como una **especificación** del comportamiento racional de un agente. Esta idea es centra en la concepción de **lenguajes de programación orientados a agentes**.

Especificación

La manera tradicional de implementar un agente que cumpla con la especificación de la teoría ϕ 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 ϕ se concibe como una especificación ejecutable directamente por el agente.

Programación Orientada a Agentes

Refinamiento

Sea L el conjunto de fórmulas bien formadas en una lógica, esto es, todas las expresiones sintácticamente correctas de la lógica en cuestión. Normalmente se asume una lógica de primer orden clásica, pero puede tratarse de una proposicional; incluso en el área de SMA, es común utilizar complejas lógicas multi-modales. El conjunto de **bases de conocimiento** en L se define como $D = \wp(L)$, es decir, el conjunto de todos los conjuntos de fbf en L . Los elementos de D se denotan Δ, Δ_1, \dots . 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** ρ . Escribimos $\Delta \vdash_{\rho} \psi$ si la fbf ψ puede ser probada en Δ . Definimos la función *siguiente*/2 del agente como:

Bases de Conocimiento

Reglas de Inferencia

$$\textit{siguiente} : D \times \textit{Per} \rightarrow D$$

Algoritmo 2.5 Programa de agente lógico

```

1: function SELECCIÓN-ACCIÓN( $\Delta : D, Ac$ ) ▷  $Ac$  Acciones.
2:   for all  $a \in Ac$  do
3:     if  $\Delta \vdash_{\rho} \textit{ejecuta}(a)$  then ▷  $\rho$  predefinida,  $\Delta$  base de conocimientos.
4:       return  $a$ 
5:     end if
6:   end for
7:   for all  $a \in Ac$  do
8:     if  $\Delta \not\vdash_{\rho} \neg \textit{ejecuta}(a)$  then
9:       return  $a$ 
10:    end if
11:  end for
12:  return null
13: end function

```

El Algoritmo 2.5 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 que valide $\textit{ejecuta}(a)$. 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

2.4.4 Agentes basados en 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 [74]. 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 E que incluye los estados meta o cumple con determinadas propiedades especificadas en algún lenguaje lógico.

Metas

Espacio de Estados

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 [69, 74].

*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:

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

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 solo 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 Ψ . Podemos expresar que un agente Ag tiene éxito en el ambiente de tareas $\langle Env, \Psi \rangle$ de dos maneras diferentes: i) $\forall r \in R(Ag, Env)$ 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 Ψ ; ii) $\exists r \in R(Ag, Env)$ 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 Ψ .

Observen que Ψ 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 $Ag = R^E \rightarrow Ac$, 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 $r \in R$. Ψ 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 [28] 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).

Metas y autonomía

Ejemplo 2.9. *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.*

2.4.5 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 desempeño, por ej., utilidades. Una **utilidad** es un valor numérico que denota la bondad de 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 $u : E \rightarrow \mathfrak{R}$ la cual asocia valores reales a cada estado del ambiente. Para poder especificar una

Utilidad

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.

Ejemplo 2.10. *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:*

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

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 $k \in \mathbb{R}$ 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

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

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

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

Esta ecuación es idéntica a la noción de maximizar la utilidad esperada, tal y como lo expresan Von Neumann y Morgenstern [108] 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 [100], 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 [95], 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. Por otra parte, los formalismos asociados a esta aproximación a los agentes racionales, son más ajenos a los propósitos de este curso.

2.5 LECTURAS Y EJERCICIOS SUGERIDOS

La formalización del concepto de agente está basada en el texto de Wooldridge [110], aunque una fuente más accesible a este material se encuentra en su libro introductorio a los Sistemas Multiagente [111]. La clasificación de los agentes y sus programas está basada en el texto de Russell y Norvig [94]. El artículo de Covrigaru y Lindsay [28] 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 Von Neumann y Morgenstern [108], donde queda de manifiesto una de las principales ventajas de este enfoque: su formalización. El concepto de racionalidad acotada fue introducido por Simon [100] 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 [95].

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 [94] para una panorámica del tema de los agentes racionales con respecto a la IA. Huns y Singh [53] ofrecen una colección de artículos fundamentales en el área, algunos de los cuales serán revisados en las tareas del curso. Weiß [109] estructuró el primer texto completo y coherente sobre el estudio de los SMA. Wooldridge [111] 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 [99] se sitúa un marco de referencia económico, más próximo a la Teoría de Juegos, pero con un componente algorítmico importante. Un excelente complemento para este texto.

Ejercicio 2.1. *Piense un poco en su tema de tesis y describa un agente y su medio ambiente en ese contexto. Base su descripción en el esquema PEAS.*

Ejercicio 2.2. *¿Existe una medida de desempeño para su agente? ¿Puede formularla?*

Ejercicio 2.3. *¿Es posible agrupar las percepciones y acciones de su agente en artefactos? Describa un ejemplo.*

Ejercicio 2.4. *¿Qué arquitectura elegiría para su agente? Justifique la respuesta.*